Quantum Generative Flow Networks

Eli J. Laird
Department of Computer Science
Southern Methodist University
Dallas, Texas
ejlaird@smu.edu

***Abstract*—The convergence of quantum and classical computing, particularly in the realm of quantum machine learning, has gained substantial attention. Hybrid quantum algorithms, merging the strengths of both computing paradigms, offer a promising avenue, especially in the era of NISQ (noisy intermediate-scale quantum) computing. Leveraging these hybrid approaches allows researchers to employ quantum algorithms on existing quantum processors while harnessing classical methods to handle complex computational challenges. A core component of these hybrid systems is the Parameterized Quantum Circuit (PQC). These circuits, defined by parameterized unitary operations, possess exceptional expressiveness, effectively modeling intricate distributions. This study introduces a novel application of Parameterized Quantum Circuits in Generative Flow Networks (GFlowNet-PQC) for object generation tasks. The GFlowNet-PQC model demonstrates efficiency and quality on par with classical deep neural network-based approaches. Despite longer training times, the quantum model achieves comparable results with significantly fewer parameters (151), emphasizing its potential in scenarios prioritizing parameter efficiency. The code for this implementation is available in the following repo: https://github.com/elilaird/quantum-gflownets.**

***Index Terms*—quantum-classical systems, generative flow networks, parameterized quantum circuits**

I. Introduction

The convergence of quantum and classical systems has emerged as a focal point in the realm of quantum computing, particularly in the context of quantum machine learning. Hybrid quantum algorithms, adeptly combining the strengths of quantum and classical computing, have become instrumental in achieving enhanced computational performance. This synergy proves especially crucial in the current NISQ (noisy intermediate-scale quantum) era, where fully error-corrected large-scale quantum computers remain elusive. Hybrid approaches empower researchers to deploy quantum algorithms on existing quantum processors, strategically utilizing classical methods to address computationally challenging aspects, such as parameter optimization.

A prominent archetype of hybrid quantum-classical systems is the Parameterized Quantum Circuit (PQC) [1], [2]. These circuits feature unitary operations that incorporate parameters to define rotations, typically constructed from single-qubit rotation gates. The flexibility of PQCs to adjust rotation degrees through parameter control endows them with remarkable expressiveness [3], enabling effective modeling of intricate distributions. While these circuits transform quantum states using parameters, the optimization of these parameters takes place on classical computers. Analogous to the optimization of parameters in deep neural networks using task-specific loss functions for classification, PQCs have demonstrated significant success across a spectrum of machine learning applications, encompassing supervised learning [1], [4], [5], generative modeling [1], [2], [6], and reinforcement learning [7]–[9].

One notable domain where PQCs have outperformed classical counterparts is in reinforcement learning. For instance, [7] introduced a Softmax-PQC as a policy network, showcasing comparable performance in various and superior results in select reinforcement learning environments compared to classical deep neural network-based models. Additionally, [8] and [9] expanded the applications of PQCs to include Q-learning, noting substantial improvements in agent efficiency within RL environments.

This paper sets out to adapt the reinforcement learning applications of parameterized quantum circuits to Generative Flow Networks (GFlowNets). GFlowNets, inspired by reinforcement learning methodologies for sequential object generation [10]–[12], iteratively construct objects by sampling actions or building blocks, mirroring the interaction of a reinforcement agent within an environment. However, unlike RL agents that sample actions to maximize rewards, GFlowNets sample actions with probabilities proportional to rewards. This distinction positions GFlowNets to harness the capability of PQCs in modeling complex action environments, extending the advantages witnessed in the reinforcement learning applications of PQCs.

Our key contributions in this paper are as follows:

(i) Introduced a Generative Flow Network implemented through a Parameterized Quantum Circuit (GFlowNet-PQC).
(ii) Implemented the proposed model in a simple object generation task, with quantum noise included and excluded.
(iii) Analyzed the efficiency and quality of the model's object generation capabilities in comparison to a classical deep neural network-based approach.

II. Related Work

A. Parameterized Quantum Circuits

Parameterized Quantum Circuits (PQCs) constitute a category of quantum circuits characterized by tunable unitary operations, providing a flexible framework for implementing

quantum algorithms. These circuits serve as a foundational element in hybrid quantum-classical systems, where classical systems play a pivotal role in optimizing the parameters defining the unitary rotation operators. In the context of machine learning, PQCs have demonstrated extensive utility, particularly in applications such as supervised learning [1], [4], [5], generative modeling [1], [2], [6], and reinforcement learning [7]–[9].

B. Generative Flow Networks

Generative Flow Networks (GFlowNets) constitute a category of generative models that operate sequentially, generating objects by iteratively sampling a policy to append new features to an incomplete object [10] [12]. What sets GFlowNets apart is their distinctive approach of generating objects in accordance with a probability distribution proportional to a reward function. This characteristic allows for the creation of diverse object distributions. In contrast to reinforcement learning policies that aim to maximize rewards, GFlowNets focus on learning to sample high-reward instances from the modes of a distribution.

A key principle underlying GFlowNets is the maintenance of equilibrium in probabilities (flows) entering and leaving a state. Bengio et al. showcased in their work [12] that when a policy adheres to this condition, referred to as the Flow Matching Condition, the objects it generates exhibit probabilities proportional to the reward. GFlowNets achieve this condition through training with Trajectory Balance Loss [11], which ensures a balance between the incoming and outgoing flows within specific states.

III. Proposed Approach

A. Circuit Design

The design of the parameterized circuit in this paper aligns with established practices in PQC-based reinforcement learning, as seen in prior works such as [7]. We define a parameterized unitary operator $U(s, \theta)$ that takes an n-qubit state and trainable parameters θ. Building on the approach in [5], [7], [13], we partition the circuit into L layers, where each layer consists of a variational layer and an encoding layer, as illustrated in Figure 1.

The variational layer incorporates single-qubit R_z, R_x, and R_y rotations, as well as Controlled-Z gates. The gates within the variational layer are parameterized by a learnable matrix Θ, which governs the rotations. On the other hand, the encoding layer comprises single-qubit R_x rotations and is responsible for encoding the classical reinforcement learning state into a quantum state. While the variational layer is parameterized by the Θ matrix, the encoding layer uses a learnable scaling matrix Λ in accordance with [7]. These alternating layers closely resemble the layers of a deep neural network.

B. Representation of the Policy

1) General Softmax Policy: Given the inherently probabilistic nature of quantum states, projective measurements into these states serve as our GFlowNet policy. To represent the various actions our policy can produce, we partition the Hilbert space into A subspaces, each associated with a corresponding measurement P_a. This measurement P_a projects the quantum state into the subspace representing the action a.

Leveraging the projections from the measurement P_a, we can approximate the expected state $\langle S \rangle$ by measuring the state T times. The expected state $\langle S \rangle$ serves as the action sampled from the policy. In the context of learning a policy in reinforcement learning and GFlowNets, the distribution of the policy is conventionally adjusted during training to transition from an exploratory strategy to an exploitation strategy.

To facilitate this transition, [7] introduced Softmax-PQC, which involves passing the expected state through a softmax function parameterized by β, as depicted in Equation 1. By updating the β parameter, the model can shift from an exploration strategy (large β) to an exploitation strategy (small β).

$$\pi_{\theta,\lambda}(a|s) = \frac{e^{\beta \langle S_a \rangle_{s,\theta}}}{\sum_{a'} e^{\beta \langle S_{a'} \rangle_{s,\theta}}} \tag{1}$$

2) Forward and Backward Policies: GFlowNets employ two distinct policies for action sampling: a forward policy P_f and a backward policy P_b. The forward policy reflects the probability of generating the state S_{t+1} given state S_t, denoted as $P_B(S_{t+1}|S_t)$. Conversely, the backward policy represents the probability of sampling a parent state S_t given the child state S_{t+1}, expressed as $P_B(S_t|S_{t+1})$. In simpler terms, the forward policy is the constructive policy responsible for building the object at S_{t+1} by taking action a, while the backward policy deconstructs the object by removing action a from the object at S_{t+1}.

To model the forward and backward policies, we employ unnormalized action logits, departing from the normalized softmax probabilities utilized in [7]. Subsequently, these logits undergo processing through a linear layer with A x 2 output neurons, where the first A neurons represent the forward policy logits and the last A neurons represent the backward policy logits, as illustrated in Equation 2.

$$\begin{aligned} Logits &= Linear(e^{\beta \langle S_a \rangle_{s,\theta}}) \\ P_F(S_{t+1}|S_t) &= Logits[0:A] \\ P_B(S_t|S_{t+1}) &= Logits[A:A*2] \end{aligned} \tag{2}$$

C. Reward

GFlowNets adopt a strategy of sampling actions with probabilities proportional to the associated reward. In the context of a GFlowNet, the reward serves to shape the modes of a probability distribution over actions. Unlike traditional reinforcement learning settings where the objective is to maximize the overall reward, GFlowNets focus on sampling from the modes of the distribution.

One distinctive feature of GFlowNets is their ability to sample non-maximum but high-reward actions, fostering a diversification in actions. The reward in a GFlowNet is task-specific, tailored to the objectives of the particular application.

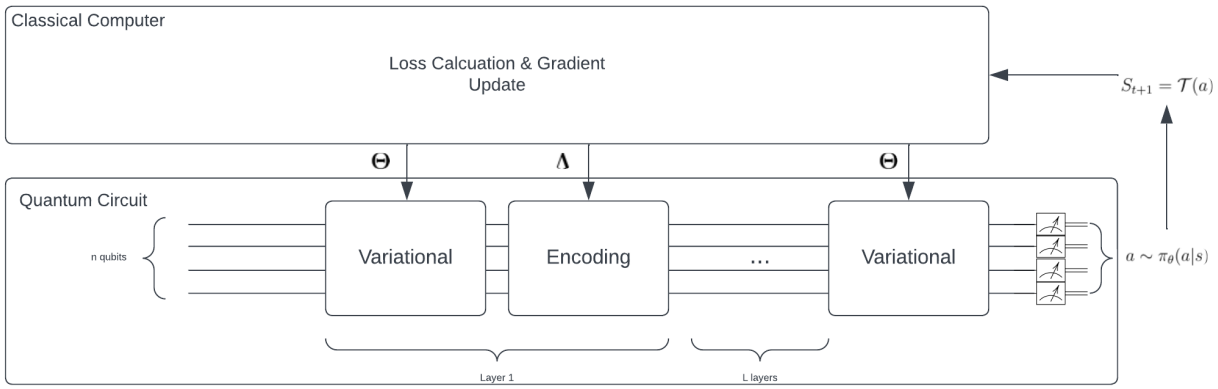

Fig. 1. Parameterized Quantum Circuit with alternating variational and encoding layers and classical computer (top) for loss and gradient calculations. Variational layers are parameterized by the learnable $\mathbf{\Theta}$ matrix and encoding layers are parameterized by the scaling matrix $\mathbf{\Lambda}$.

In this paper, our exploration involves employing a GFlowNet to iteratively construct objects belonging to a specific class. In this scenario, the reward is based on the state S_t belonging to a particular class with values shown in Table I.

D. Trajectory Balance Loss

In training the policy as delineated in Section III-B, we employ the Trajectory Balance Loss function introduced in [11] for optimizing GFlowNets. This loss function is designed to incentivize the policy to sample actions with a probability proportional to the reward. The computation of the Trajectory Balance Loss is carried out for a completed object at state S after its construction through the sampling of the policy $\pi_{\theta,\lambda}(a|s)$.

We define the sequence of intermediate objects at S_t as a trajectory τ. The loss for a state S constructed over a trajectory of τ samples can be computed using the following equation:

$$L_{TB}(S) = \left(\log \frac{Z \prod_t^\tau P_F(S_{t+1}|S_t)}{R(S_t) \prod_t^\tau P_B(S_t|S_{t+1})}\right)^2 \tag{3}$$

, where P_F and P_B are the forward and backward policies, Z is a trainable scaling parameter, and $R(\cdot)$ is the reward function.

E. Training Procedure

We execute the training process, as defined in Section III-B, employing Algorithm 1. This training algorithm involves a transition function $\mathcal{T}(a)$ that produces a new state S_{t+1} given a sampled action a, the reward R, and the policy $\pi_{\theta,\lambda}$. The training process commences by initializing the state to a predefined value S_0, tailored to the specific application. For instance, in generating a smiley face, the initial state would be an empty circle. The initial state is then added to the trajectory τ, which serves to track the intermediate states S_t throughout the process.

Subsequently, the generation of a complete object S takes place by sampling actions from the policy $\pi_{\theta,\lambda}$, executing these actions using the transition function $\mathcal{T}$, and recording intermediate states in the trajectory τ until a predetermined maximum number of actions is reached. Upon reaching the maximum, the trajectory balance loss is calculated, considering the reward $R(S_t)$ for each state within the completed trajectory.

Algorithm 1 GFlowNet-PQC Policy Training Loop

Input: policy $\pi_{\theta,\lambda}$, transition function $\mathcal{T}$, $EPOCHS$, $R(\cdot)$, $MAX\_ACTIONS$

for epoch in $EPOCHS$ **do**
  $actions\_taken \leftarrow 0$
  S_0 ▷ Initialize state
  $\tau \leftarrow \emptyset + \{S_0\}$ ▷ Initialize trajectory
  repeat
    $a \leftarrow \pi_{\theta,\lambda}$ ▷ Sample action
    $S_{t+1} \leftarrow \mathcal{T}(a)$ ▷ Take action
    $\tau \leftarrow \tau + \{S_{t+1}\}$ ▷ Append S_{t+1} to trajectory
    $actions\_taken++$
  until $actions > MAX\_ACTIONS$
  $\theta \leftarrow \theta - \eta \nabla Loss_{TB}(R(S), log_Z, \tau)$ ▷ Update θ
end for

IV. Evaluation

To assess the performance of the GFlowNet-PQC policy in object generation, we apply the policy to a simple object generation task involving the creation of smiley faces. This task is used in the popular GFlowNet tutorial [14]. In our experiments, we compare the quantum GFlowNet with a classical deep neural network-based approach for generating these objects. While the primary focus of this paper is to demonstrate the feasibility of generating compositional objects with a quantum GFlowNet, we conduct a comparative analysis

with the classical version, evaluating efficiency and the ability to generate the specified class.

Efficiency is assessed by measuring the total time required for training over *50,000* epochs and the time needed to generate a single object. The choice of *50,000* epochs aligns with the classical GFlowNet's convergence benchmark as outlined in [14].

The quality of a GFlowNet is defined by its capacity to generate objects of a specific class with a probability proportional to the predefined reward distribution. Given the limited number of possible actions for the smiley face generation task, we can precisely define the reward distribution by setting rewards for smiley faces and frowny faces accordingly. The quality metric quantifies the GFlowNet's ability to adhere to this distribution, measured as the percentage of objects generated for a particular class over a sample of 100 objects, as illustrated in Equation 4. Furthermore, we gauge the percentage of valid faces generated, where a valid face exhibits no conflicting features, such as simultaneous smiling and frowning.

$$Quality\ Ratio = \frac{\sum_i^N f(O_i) == C}{N} \tag{4}$$

, where $f(\cdot)$ predicts the class of the object and C is the specified class to generate.

A. Smiley-Face Generation

This experiment aims to assess the capability of the proposed GFlowNet-PQC in generating simple objects characterized by a limited number of possible actions. In this context, a complete object refers to a graphical face, which can be categorized as either 'smiley' or 'frowny'. Each face is generated with the following action space: {smile, frown, left eyebrow down, right eyebrow down, left eyebrow up, right eyebrow up}. Figure 2 provides a visual representation of a smiley and frowny face as illustrative examples.

Fig. 2. Left: example of a 'smiley' face. Right: example of an 'frowny' face from [14].

B. Experiment Design

The experimental setup outlined in the previous section will be executed using the TensorFlow (TF) [15] and TensorFlow Quantum (TFQ) [16] Python packages. Leveraging TensorFlow enables the encapsulation of Parametrized Quantum Circuits (PQC) and GFlowNets within custom Keras models, facilitating straightforward gradient tracking and updates. For the classical GFlowNets used as a comparison, PyTorch [17] will be employed.

The parameterized quantum circuits, implemented in TFQ, will undergo simulation using both a noiseless simulator and a noisy simulator. The first simulator, an ideal simulator embedded within the *ControlledPQC* TFQ class, abstains from modeling noise within the quantum circuits. In contrast, the second simulator, integrated into the *NoisyControlledPQC* TFQ class, introduces noise to the simulator measurements via Monte Carlo Sampling. Both classes leverage the Cirq simulator, integrated into TFQ, which automatically executes simulations when invoked through the Keras wrapper.

The rewards, as detailed in Table I, are configured to align with a 2:1 ratio between the *smiley* and *frowny* classes, adhering to the approach presented in [14]. Originally, we designated rewards as 2 and 1 for *smiley* and *frowny* faces, respectively. However, the rewards were subsequently scaled by a factor of 2 to amplify the reward signal while preserving the established ratio.

Class	**Reward**
Smiley	4
Frowny	2
Invalid	0
Target Quality	0.66

TABLE I
REWARD VALUES FOR THE *smiley* AND *frowny* CLASSES WITH THE TARGET REWARD DISTRIBUTION BEING A 2:1 OF SMILEY FACES TO FROWNY FACES. ZERO REWARD IS GIVEN FOR INVALID FACES, SUCH AS FACES WITH BOTH FROWNS AND SMILES.

In tackling the smiley face generation task, the quantum GFlowNets used 6 qubits, allocating one for each possible action. The GFlowNet-PQC structure comprised 6 layers, where the initial layer is a variational layer parameterized by Θ. Subsequent layers consist of another variational layer and an encoding layer parameterized by Λ. Each variational layer entailed 3 θ parameters per qubit, responsible for the R_x, R_y, and R_z rotations. Meanwhile, each encoding layer required a singular scaling λ parameter per qubit.

Conversely, the classical GFlowNet adopted a simple two-layer multilayer perceptron design with 512 hidden neurons. As outlined in [11], [14], both GFlowNet variants learned a *log Z* parameter to facilitate trajectory balance loss optimization. A comprehensive summary of all parameters is provided in Table II.

Parameter	**Quantum**	**Classical**
Lambda (Λ)	30	X
Theta (Θ)	108	X
Linear (Input)	X	3072
Linear (Hidden)	X	512
Linear (Hidden)	X	6144
Linear (Output)	12	12
Log Z	1	1
Total Parameters	**151**	**9741**

TABLE II
NUMBER OF PARAMETERS FOR EACH LAYER IN BOTH QUANTUM GFLOWNETS AND THE CLASSICAL GFLOWNET USED FOR THE SMILEY FACE GENERATION TASK.

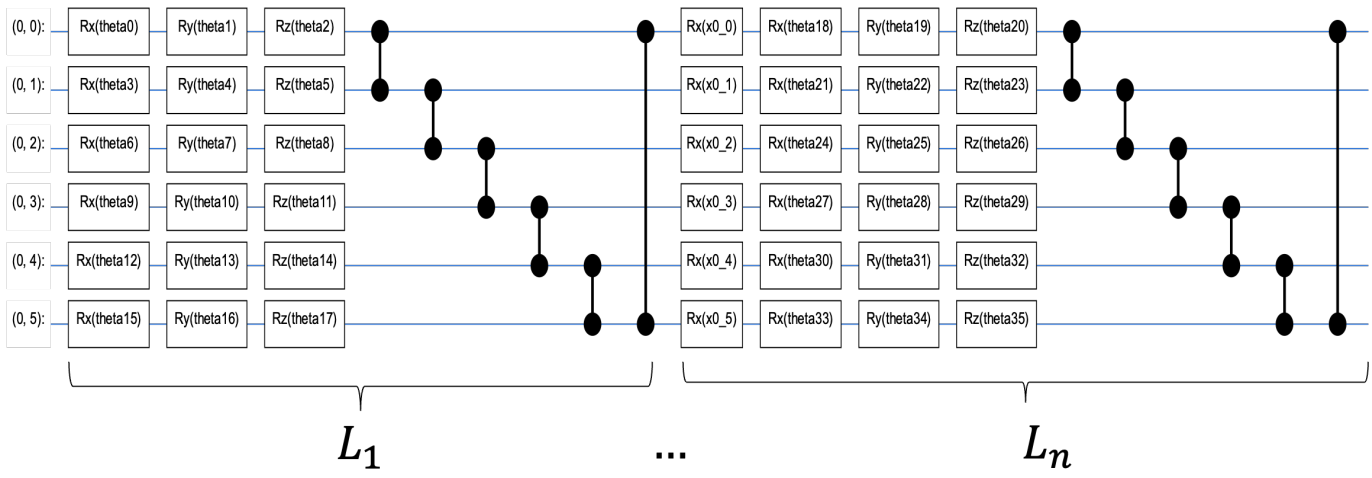

Fig. 3. In depth circuit diagram for the Parameterized Quantum Circuit used in the smiley face generation task.

V. Results

A. Smiley Face Generation Quantitative Results

The quantitative outcomes from the smiley face experiment are delineated in Tables III and V-A. Examining the efficiency results in Table III, we observe a noteworthy discrepancy in training durations between the GFlowNet-PQC and the classical GFlowNet over *50,000* epochs.

The classical GFlowNet completed training in a mere 4 minutes, contrasting with the significantly longer durations of 6 hours and 52 minutes for the noisy GFlowNet-PQC and 51 minutes for the noiseless GFlowNet-PQC. Delving into the average time required to generate a single object, excluding logistical times for gradient calculations, etc., we find that the classical approach demonstrated efficiency with an average generation time of *0.00268* seconds. In comparison, the noiseless GFlowNet-PQC and the noisy GFlowNet-PQC exhibited longer average generation times of *0.05748* seconds and *0.08018* seconds, respectively.

It's important to note that the recorded average generation time for the noisy GFlowNet-PQC may not accurately represent the training time divided by the total epochs. This discrepancy arises because the state of the Monte-Carlo sampling within the *NoisyControlledPQC* class resets every ten samples and was not captured by the Python timing commands for reasons unknown to the author. If we were to include this correction to the total time, the average generation time for the noisy GFlowNet-PQC would be *0.49558* seconds, encompassing the gradient calculations.

The results pertaining to object generation quality are detailed in Table V-A. Our target reward distribution, as outlined in Table I, dictates a 2:1 smiley-to-frowny face ratio, translating to a 66% representation of smiley faces within a set of 100 samples.

The classical GFlowNet yielded 72 smiley faces out of 100, deviating by +6 faces from the desired 66, while the noiseless GFlowNet-PQC generated 61 smiley faces out of 100, showing a deviation of -5 faces. Notably, the noisy GFlowNet-PQC generated the maximum possible 100 smiley faces out of 100, indicating an approach that maximizes the reward for smiley faces without adhering to the target distribution.

Model	Training Time (hh:mm:ss)	Avg. Generation Time (s)	Epochs
Quantum Noisy	6:52:59	0.08018	50k
Quantum Noiseless	00:51:58	0.05748	50k
Classical	00:04:09	0.00268	50k

TABLE III
EFFICIENCY TIMES FOR CLASSICAL AND QUANTUM GFLOWNET-PQCS FOR THE SMILEY FACE GENERATION EXPERIMENT. ALL TIMES IN THIS TABLE *exclude* GRADIENT CALCULATIONS AND OTHER LOGISTICAL LOGGING OPERATIONS.

In terms of quality, the evaluation of performance improvements between quantum and classical approaches remains inconclusive, particularly when comparing the classical GFlowNet to the noiseless GFlowNet-PQC. Avenues for future exploration include investigating the impact of introducing noise to the quantum GFlowNet and its ability to align with a specified reward distribution.

It is important to highlight that all three GFlowNets, whether classical or quantum, successfully generated 100% valid faces out of the 100 sampled, underscoring the efficacy of the models in maintaining validity throughout the generation process.

Model	Quality Ratio	Valid Faces (%)
Quantum Noisy	1.0	100
Quantum Noiseless	0.61	100
Classical	0.72	100

TABLE IV
QUALITY METRICS FOR QUANTUM AND CLASSICAL GFLOWNETS FOR THE SMILEY FACE GENERATION EXPERIMENT.

In Figures 4, 5, and 6, we present the convergence behavior of the trajectory loss for the classical GFlowNet, noiseless

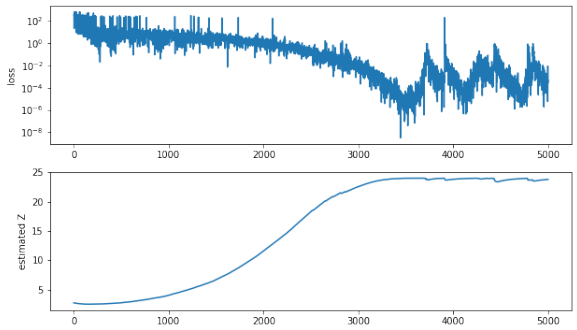

Fig. 4. Average training loss (top) and Z parameter (bottom) for training of the classical GFlowNet model over every 10th epoch.

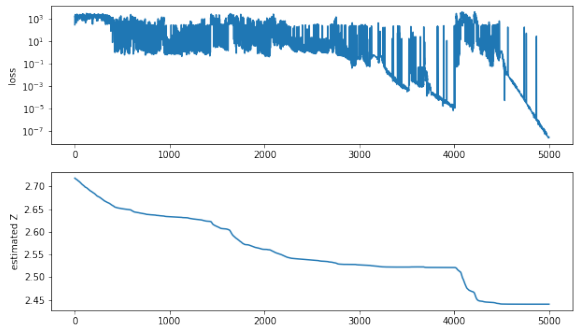

Fig. 5. Average training loss (top) and Z parameter (bottom) for training of the noiseless GFlowNet-PQC model over every 10th epoch.

GFlowNet-PQC, and noisy GFlowNet-PQC models, respectively. Despite all models eventually reaching convergence, a noticeable distinction emerges in the descent patterns. Figure 4 illustrates a gradual decline in the trajectory loss for the classical GFlowNet, whereas Figures 5 and 6 depict sharp decreases in the quantum models.

It is noteworthy that the quantum GFlowNet-PQCs required approximately 30,000 to 40,000 epochs to achieve their pronounced reduction in loss. In contrast, the classical GFlowNet exhibited a gradual decrease until approximately epoch 35,000, where it oscillated around 10^{-5}. An intriguing observation pertains to the optimization of the Z parameter. The classical GFlowNet demonstrated an increasing trend in the Z until converging to a value around 24, while the quantum GFlowNet-PQCs exhibited a decreasing trend, converging to values around 2.45.

Given that the Z parameter serves as the partition coefficient for the reward distribution, as described in [11], it represents the logarithm of the total reward over the possible state space. For the smiley face task, which encompasses four possible smiley face terminal states and four possible frowny face states, the total reward is 24. This true value aligns with the Z values learned by the classical GFlowNet. However, the quantum GFlowNet-PQCs learn a much smaller partition parameter, yet still effectively generate objects following the desired reward distribution. This discrepancy raises intriguing questions regarding the relative values of log Z, warranting further exploration in future research on GFlowNets.

B. Parameter Efficiency of Quantum vs Classical GFlowNets

In preceding sections, we emphasized the capacity of quantum GFlowNet-PQCs to minimize a trajectory loss function, enabling the generation of objects with quality comparable to their classical counterparts. A noteworthy distinction between the quantum and classical approaches lies in the number of trainable parameters involved. Specifically, both the noisy and noiseless versions of quantum GFlowNet-PQCs require a mere *151* trainable parameters, as opposed to the *9741* parameters demanded by the classical GFlowNet, as illustrated in Table II. This substantial contrast in parameter count underscores a key advantage of employing PQCs over classical neural networks, particularly in operational scenarios where a minimal parameter footprint is crucial, despite the considerably longer training time required for quantum models.

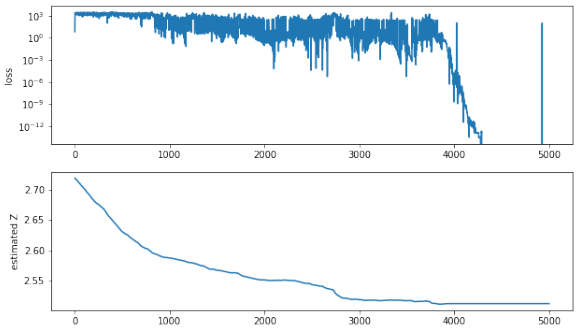

Fig. 6. Average training loss (top) and Z parameter (bottom) for training of the noisy GFlowNet-PQC model over every 10th epoch.

C. Smiley Face Generation Qualitative Results

In this section, we visually explore the smiley face generation capabilities of GFlowNets. Figure 7 illustrates the distribution of faces generated by the noiseless GFlowNet-PQC. The figure displays a distribution of both smiley and frowny faces, closely aligning with the ratio documented in Table V-A. Notably, there is a consistent trend observed in the GFlowNet-PQC, where it consistently samples the *right eyebrow up-left eyebrow down* combination of actions for both frowny and smiley faces. While this trend is likely a random side effect of optimization, a more targeted approach to reward distribution could be employed to encourage a more diverse range of actions concerning eyebrow orientation. However, as our rewards are defined solely in relation to smiles and frowns, this behavior was not a primary focus of our investigation. Additional figures for the noisy GFlowNet-PQC and classical GFlowNet models are included in the Appendix A.

Fig. 7. Qualitative results for the Smiley Face generation example using the noiseless GFlowNet. This figure shows 64 sampled faces of the trained noiseless GFlowNet-PQC.

VI. Conclusion

This work introduces a novel application of Parameterized Quantum Circuits (PQCs) within the domain of Generative Flow Networks (GFlowNets). The proposed GFlowNet-PQC model exhibits promising results in the generation of simple objects, such as smiley faces, showcasing its efficiency and quality in comparison to classical deep neural network-based approaches.

In terms of efficiency, despite the longer training times of quantum GFlowNet-PQCs, their ability to achieve comparable results with significantly fewer trainable parameters (151 parameters) highlights their potential advantage in scenarios where parameter efficiency is paramount.

The quality of object generation by GFlowNet-PQC is demonstrated by adhering to a specified reward distribution, with the noisy version achieving 100% success in generating smiley faces. While the noiseless version and the classical approach exhibit slight deviations from the target distribution, their ability to consistently generate valid faces underscores their efficacy.

The exploration of training dynamics reveals intriguing patterns, particularly in the optimization of the partition parameter Z. The quantum GFlowNet-PQCs converge to smaller Z values, challenging conventional expectations. This prompts further investigation into the implications and dynamics of log Z values in the context of GFlowNets.

Overall, this study establishes the feasibility and potential advantages of employing PQCs in GFlowNets for object generation tasks, paving the way for further research into quantum-classical hybrid models and their applications in machine learning.

References

[1] M. Benedetti, E. Lloyd, S. H. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:189999815

[2] D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot, C. Brecque, L. Egan, O. Perdomo, and C. Monroe, "Training of quantum circuits on a hybrid quantum computer," *Science Advances*, vol. 5, no. 10, p. eaaw9918, 2019. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.aaw9918

[3] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, "Expressive power of parametrized quantum circuits," *Physical Review Research*, vol. 2, no. 3, jul 2020. [Online]. Available: https://doi.org/10.1103/2Fphysrevresearch.2.033125

[4] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, mar 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-0980-2

[5] M. Schuld, R. Sweke, and J. J. Meyer, "Effect of data encoding on the expressive power of variational quantum-machine-learning models," *Physical Review A*, vol. 103, no. 3, mar 2021. [Online]. Available: https://doi.org/10.1103/2Fphysreva.103.032430

[6] J.-G. Liu and L. Wang, "Differentiable learning of quantum circuit born machines," *Phys. Rev. A*, vol. 98, p. 062324, Dec 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.98.062324

[7] S. Jerbi, C. Gyurik, S. Marshall, H. J. Briegel, and V. Dunjko, "Parametrized quantum policies for reinforcement learning," in *Neural Information Processing Systems*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:244843259

[8] A. Skolik, S. Jerbi, and V. Dunjko, "Quantum agents in the gym: a variational quantum algorithm for deep q-learning," *Quantum*, vol. 6, p. 720, may 2022. [Online]. Available: https://doi.org/10.22331/q-2022-05-24-720

[9] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, "Variational quantum circuits for deep reinforcement learning," *IEEE Access*, vol. 8, pp. 141 007–141 024, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:195767325

[10] E. Bengio, M. Jain, M. Korablyov, D. Precup, and Y. Bengio, "Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation," Nov. 2021. [Online]. Available: http://arxiv.org/abs/2106.04399

[11] N. Malkin, M. Jain, E. Bengio, C. Sun, and Y. Bengio, "Trajectory balance: Improved credit assignment in GFlowNets," Oct. 2022, arXiv:2201.13259 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2201.13259

[12] Y. Bengio, S. Lahlou, T. Deleu, E. J. Hu, M. Tiwari, and E. Bengio, "GFlowNet Foundations," Aug. 2022, arXiv:2111.09266 [cs, stat].

[13] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, "Data re-uploading for a universal quantum classifier," *Quantum*, vol. 4, p. 226, feb 2020. [Online]. Available: https://doi.org/10.22331/2Fq-2020-02-06-226

[14] B. Emmanuel, "GFlowNet Tutorial." [Online]. Available: https://colab.research.google.com/drive/1fUMwgu2OhYpQagpzU5mhe9_Esib3Q2VR

[15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[16] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, R. Halavati, M. Y. Niu, A. Zlokapa, E. Peters, O. Lockwood, A. Skolik, S. Jerbi, V. Dunjko, M. Leib, M. Streif, D. V. Dollen, H. Chen, S. Cao, R. Wiersema, H.-Y. Huang, J. R. McClean, R. Babbush, S. Boixo, D. Bacon, A. K. Ho, H. Neven, and M. Mohseni, "Tensorflow quantum: A software framework for quantum machine learning," 2021.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

APPENDIX A
ADDITIONAL FIGURES

Fig. 8. Qualitative results for the Smiley Face generation example using the classical GFlowNet. This figure shows 64 sampled faces of the trained classical GFlowNet.

Fig. 9. Qualitative results for the Smiley Face generation example using the noisy GFlowNet-PQC. This figure shows 64 sampled faces of the trained noisy GFlowNet-PQC.