Style-based quantum generative adversarial networks for Monte Carlo events

Carlos Bravo-Prieto[1,2], Julien Baglio[3], Marco Cè[3], Anthony Francis[4,3], Dorota M. Grabowska[3], and Stefano Carrazza[5,3,1]

[1]Quantum Research Centre, Technology Innovation Institute, Abu Dhabi, UAE

[2]Departament de Física Quàntica i Astrofísica and Institut de Ciències del Cosmos (ICCUB), Universitat de Barcelona, Barcelona, Spain.

[3]Theoretical Physics Department, CERN, CH-1211 Geneva 23, Switzerland.

[4]Institute of Physics, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan.

[5]TIF Lab, Dipartimento di Fisica, Università degli Studi di Milano and INFN Sezione di Milano, Milan, Italy.

We propose and assess an alternative quantum generator architecture in the context of generative adversarial learning for Monte Carlo event generation, used to simulate particle physics processes at the Large Hadron Collider (LHC). We validate this methodology by implementing the quantum network on artificial data generated from known underlying distributions. The network is then applied to Monte Carlo-generated datasets of specific LHC scattering processes. The new quantum generator architecture leads to a generalization of the state-of-the-art implementations, achieving smaller Kullback-Leibler divergences even with shallow-depth networks. Moreover, the quantum generator successfully learns the underlying distribution functions even if trained with small training sample sets; this is particularly interesting for data augmentation applications. We deploy this novel methodology on two different quantum hardware architectures, trapped-ion and superconducting technologies, to test its hardware-independent viability.

1 Introduction

Quantum computing is a new paradigm whereby quantum phenomena are harnessed to perform computations. The current availability of noisy intermediate-scale quantum (NISQ) computers [1], and recent advances towards quantum computational supremacy [2, 3], have led to a growing interest in these devices to perform computational tasks faster than classical machines. Among many of the near-term applications [4, 5], the field of Quantum Machine Learning (QML) [6, 7] is held as one promising approach to make use of NISQ computers.

Early work in QML was mostly focused on speeding up linear algebra subroutines [8–11], widely used in classical machine learning, by leveraging the Harrow-Hassidim-Lloyd algorithm [12]. This approach is promising, though its utility depends on the existence of large-scale quantum computers with low gate errors and enough qubits to perform quantum error correction. More recent proposals focus on defining a quantum neural network (QNN), or parameterized quantum circuit [13–16], which then can be trained to implement a function class [17–19]; these proposals can be implemented on current NISQ-era devices. For example, several QNNs have been proposed for pattern classification [20–23] or data compression [24–27]. This QML approach to quantum computing is a research topic that can be adapted, improved, and tested on many research problems in disparate scientific fields. Motivated by this idea, we propose to investigate the possibility of using QNNs for generative modeling [28–30]. More specifically, we explore the uses of QNNs for the generation of Monte Carlo events through quantum generative adversarial networks (qGANs) [31, 32].

The generative adversarial framework employs two competing networks, the generator and the discriminator, that are trained alternatingly [33]. The generator produces candidates while the discriminator evaluates them. The objective of the discriminator is to distinguish the real samples from the generated ones. That is, the discriminator plays the role of the generator's adversary, and therefore, their competition is a zero-sum two-player game. By substituting either the discriminator, the generator, or both with quantum systems, the scheme can be translated to quantum computing [31].

In recent months, the spreading interest in QML has led to different qGAN implementations [34–40]. Our contribution here can be summarized in three distinct aspects. (1) Previous proposals employed toy data for their qGAN training. In contrast, we test our model using data for a quantum scattering process. In particular, we first train and validate our qGAN model with artificial data from known underlying probability density functions. Then, in order to test our model in a realistic set-up, we use as training sets simulated Monte Carlo events for particle

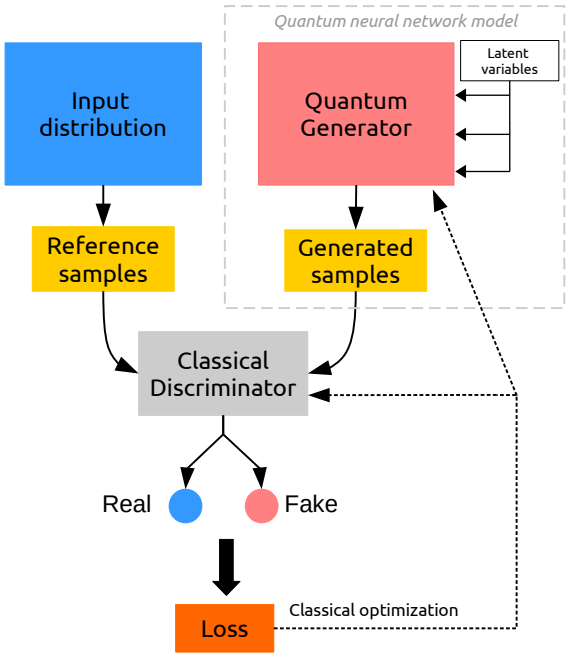

Figure 1: Schematic steps involved in the style-qGAN training. Reference samples from a known input distribution are initially prepared. Simultaneously, we inject the latent variables over the whole quantum generator to produce the fake generated samples. Then, both set of samples are used to train the classical discriminator. The quality of the training is measured by an appropriate loss function, which is then also used to update the quantum generator and the classical discriminator. This procedure is repeated following an adversarial approach.

physics processes at the Large Hadron Collider (LHC) at CERN. (2) We propose an alternative quantum generator architecture. Traditionally, the prior noise distribution, or latent variables in the language of generative models, is provided to the quantum generator through its first quantum gates. We instead repeatedly encode the latent variables over all the quantum network. This allows us to obtain a generalized version of the state-of-the-art qGAN implementations; when working with a realistic dataset, we achieve significantly smaller Kullback-Leibler (KL) divergences. A similar concept was introduced in the classical context [41], coined as style-based generative adversarial network (GAN), which was proven to be useful in facial recognition tasks. In the classical approach, however, a preprocessing of the latent variables is done. Although our approach do not incorporate this preprocessing step of the latent variables, given the analogy of repeatedly encoding them into the network, from now on we refer to our qGAN model as style-qGAN. (3) We validate and assess our style-qGAN in quantum hardware. Specifically, we successfully implement our model in two different quantum architectures, namely ion traps and superconducting qubits.

It is important to highlight that several research groups from the high-energy physics (HEP) community are investigating potential applications of quantum technologies in HEP applications and obtaining interesting results [42–47]. The study presented in this manuscript should be considered both as an improvement over the state-of-the-art from the algorithmic point of view, and as a proof-of-concept for a future quantum hardware deployment, providing a robust and reproducible starting point for future investigations. In particular, the introduction of GAN models in HEP Monte Carlo simulation has been discussed extensively in the last years, see Refs. [48–54]. In this work, we consider the possibility to use a qGAN model in a data augmentation context, where the model is trained with a small amount of input samples and it learns how to sample the underlying distribution.

The paper is structured as follows. In Sec. 2 we define our style-qGAN model. In Sec. 3 we validate the style-qGAN model using toy data. Then, in Sec. 4 we train the style-qGAN generator on simulated LHC events. Finally, in Sec. 5 we test our final model on real quantum hardware, in Sec. 6 we compare our model to previous qGAN architectures, and in Sec. 7 we present our conclusion and future development directions. In addition, we include in Appendix A a noise simulation of the algorithm using the simplified noise model provided by IBM Q for the target machine used in Sec. 5.

2 Implementation

2.1 Workflow design

The classical implementation of a GAN model [33] involves at least three components: the discriminator model, the generator model, and the adversarial training procedure. In this work we consider a hybrid quantum-classical system, where the generator model has a quantum representation through a QNN while the discriminator is a classical neural network model. This choice is motivated by the practical positive implications of a hardware-based generative model, in particular the possibility to obtain performance improvements in a real quantum device. The idea of using a quantum device for the generation of samples is very attractive because the complicated aspects of density modeling and sampling are delegated to a hardware architecture.

There are alternative approaches where both models could be represented by a QNN [31, 37–40]. However, after testing some prototype architectures, we have observed faster convergence when using a classical discriminator.

In Figure 1 we schematically show the steps involved in the style-qGAN presented here. The procedure starts from the preparation of reference samples from a known distribution function that we would like to encode in the quantum generator model. At

the same time, we define a QNN model where we inject stochastic noise in all quantum gates of the network. The generator model is then used to extract fake generated samples that, after the training procedure, should match the quality of the known input distribution. Lastly, both sets of samples are used to train the discriminator model. The quality of the training is measured by an appropriate loss function which is monitored and optimized classically by a minimization algorithm based on the adversarial approach. The training process consists of simultaneous stochastic gradient descent for both models which, after reaching convergence, delivers a quantum generator model with realistic sampling.

In the following paragraphs, we first introduce the optimization procedure and the quantum generator network, and validate the procedure by using reference samples from known distribution functions to train the model on a quantum simulator. We then train our style-qGAN model with Monte Carlo-generated LHC events using again a quantum simulator. Finally, the best-trained model is deployed on real quantum hardware devices based on superconducting and trapped-ion technologies.

All calculations involving quantum circuit simulation are performed using `Qibo` v0.1.6 [55, 56] on classical hardware. For this particular project, we have used the `tensorflow` [57] backend which provides the possibility to use gradient descent optimizers during the training step. The style-qGAN model is publicly available through the `Qibo` framework and the code repository in [58].

2.2 Optimization procedure

Our style-qGAN comprises of a QNN for the generator $G(\boldsymbol{\phi_g}, \boldsymbol{z})$ and a classical network for the discriminator $D(\boldsymbol{\phi_d}, \boldsymbol{x})$, where $\boldsymbol{\phi_g}$ and $\boldsymbol{\phi_d}$ are the parameters of the corresponding networks. The quantum generator transforms samples from a prior standard Gaussian noise distribution $\boldsymbol{z} \sim p_{\text{prior}}(\boldsymbol{z})$, also called latent vector, into samples generated by $G(\boldsymbol{\phi_g})$, thus mapping $p_{\text{prior}}(\boldsymbol{z})$ to a different distribution p_{fake} of generated data. On the other hand, the discriminator takes as input samples $\boldsymbol{x}$ and tries to distinguish between fake data from the generator and real data from the reference input distribution p_{real}. The training corresponds to an adversarial game, where we alternate between improving the discriminator to distinguish fake and real data, and the generator to cheat the discriminator with new fake data.

In this work, we consider the binary cross-entropy for the optimization objective. The generator's loss function can be defined as

$$\mathcal{L}_G(\boldsymbol{\phi_g}, \boldsymbol{\phi_d}) = -\mathbb{E}_{\boldsymbol{z}\sim p_{\text{prior}}(\boldsymbol{z})}[\log D(\boldsymbol{\phi_d}, G(\boldsymbol{\phi_g}, \boldsymbol{z}))]\,, \tag{1}$$

while the discriminator's loss function can be defined as

$$\begin{aligned}\mathcal{L}_D(\boldsymbol{\phi_g}, \boldsymbol{\phi_d}) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{real}}(\boldsymbol{x})}[\log D(\boldsymbol{\phi_d}, \boldsymbol{x})] \\ + \mathbb{E}_{\boldsymbol{z}\sim p_{\text{prior}}(\boldsymbol{z})}[\log(1 - D(\boldsymbol{\phi_d}, G(\boldsymbol{\phi_g}, \boldsymbol{z})))]\,.\end{aligned} \tag{2}$$

Notice that the adversarial training corresponds to a minimax two-player game,

$$\min_{\boldsymbol{\phi_g}} \mathcal{L}_G(\boldsymbol{\phi_g}, \boldsymbol{\phi_d})\,, \tag{3}$$

$$\max_{\boldsymbol{\phi_d}} \mathcal{L}_D(\boldsymbol{\phi_g}, \boldsymbol{\phi_d})\,, \tag{4}$$

where the optimum uniquely corresponds to the Nash equilibrium between the loss functions.

The optimization of the parameters $\boldsymbol{\phi_g}$ and $\boldsymbol{\phi_d}$ is done alternatingly by updating the quantum generator and classical discriminator. The optimizer used to update the steps in this work is the ADADELTA [59], which is a stochastic gradient descent method that monotonically decreases its learning rate. The starting learning rates utilized are 0.1 for the classical discriminator and 0.5 for the quantum generator.

2.3 Quantum generator architecture

The goal of the quantum generator is to implement a quantum feature map to encode the latent variables and then produce the fake samples. Let $\mathcal{H}$ be a Hilbert space. The quantum feature map $\Psi_{\phi_g} : \boldsymbol{z} \to \mathcal{H}$ encodes the latent variables into a quantum state $\left|\Psi_{\phi_g}(\boldsymbol{z})\right\rangle$. This action is equivalent to applying a quantum circuit $\mathcal{U}_{\phi_g}(\boldsymbol{z})$ to the initial state $|0\rangle^n$, where n is the number of qubits. In particular, we employ the architecture shown in Figure 2. We consider a layered QNN, where each layer is composed of a set of entangling gates U_{ent} preceded by two alternating R_y and R_z single-qubit rotations. After implementing the layered network, a final layer of R_y gates is applied. Note that U_{ent} is specific to each example and $R_j(\theta_k) = e^{-i\theta_k\sigma_j/2}$, where σ_j are the Pauli operators. The number of layers can be modified to tune the capacity of the quantum generator.

The action of the quantum feature map is dictated by each qubit rotation. That is, each gate is parameterized by the set of trainable parameters $\boldsymbol{\phi_g}$ and, most importantly, the latent vector $\boldsymbol{z}$. We encode them by using a linear function as

$$R^{l,m}_{y,z}\left(\boldsymbol{\phi_g}, \boldsymbol{z}\right) = R_{y,z}\left(\phi_g^{(l)} z^{(m)} + \phi_g^{(l-1)}\right)\,, \tag{5}$$

where $l = \{2, 4, 6, \ldots\}$ takes different even values on each rotation up to the total number of trainable parameters, and $m = \{1, 2, \ldots, D_{\text{latent}}\}$, being D_{latent} the latent dimension to be specified for each implementation. The encoding in Eq. 5 is analogous to that used in classical neural networks. Namely, $\phi_g^{(l)}$ and $\phi_g^{(l-1)}$ play the role of the weights and biases, respectively, while the rotation gate plays the role of the non-linear activation function.

The novelty of the quantum generator architecture used for the style-qGAN model is the embedding of the latent variables into every quantum gate of the network, in contrast to previous qGAN proposals which introduced them only once at the beginning of the network. Our style-based generator contains sequences of layers with latent variable encoding and trainable parameters, much in the spirit of recent developments in the QML field. For instance, for univariate functions $f(z)$, Ref. [17] shows that large enough QNNs with sequences of quantum gates with data encoding and trainable parameters are universal function approximators. In contrast, encoding data only once restricts the function class that can be learned [60]. Although much less is known for multivariate functions $f(\boldsymbol{z})$, Ref. [18] shows that two particular schemes with repetition blocks of data encoding are also universal function approximators. Hence, the style-based approach is the intuitive step for qGAN implementations toward quantum feature maps that may be classically intractable and, as we numerically explore in Sections 3 and 4, also reliable function approximators even with shallow QNNs.

In the following, let us show that the style-based quantum generator generalises the standard quantum generator approach.

Theorem 1. *Let be a standard quantum generator $\mathcal{U}_{\phi_g^*}(\boldsymbol{z})$ as a unitary operation where the latent variables $\boldsymbol{z}$ are encoded once in the circuit, and $\mathcal{S}_\Psi^* = \{\mathcal{U}_{\phi_g^*}(\boldsymbol{z})\}$ be its set of achievable quantum feature maps for any set of parameters $\boldsymbol{\phi_g^*}$. Equivalently, let $\mathcal{S}_\Psi = \{\mathcal{U}_{\phi_g}(\boldsymbol{z})\}$ be the set of achievable quantum feature maps by a style-based quantum generator for any set of parameters $\boldsymbol{\phi_g}$. Then, the following relation holds*

$$\mathcal{S}_\Psi^* \subset \mathcal{S}_\Psi \, .$$

Moreover, for any set of parameters $\boldsymbol{\phi_g^*}$ there exists a set of parameters $\boldsymbol{\phi_g}$ such that

$$\mathcal{S}_\Psi^* = \mathcal{S}_\Psi \, .$$

Proof. Assume any standard quantum generator $\mathcal{U}_{\phi_g^*}(\boldsymbol{z})$ where the latent variables $\boldsymbol{z}$ are encoded once in the circuit. Then, $\mathcal{U}_{\phi_g^*}(\boldsymbol{z})$ can be split as the action of an encoder of the latent variables $E_{\phi_g^{*1}}(\boldsymbol{z})$ and a variational circuit $V_{\phi_g^{*2}}$ as $\mathcal{U}_{\phi_g^*}(\boldsymbol{z}) = V_{\phi_g^{*2}} \, E_{\phi_g^{*1}}(\boldsymbol{z})$, where $\boldsymbol{\phi_g^*} = (\boldsymbol{\phi_g^{*1}}, \boldsymbol{\phi_g^{*2}})$. In contrast, the style-based approach utilises the same circuit architecture $\mathcal{U}_{\phi_g}(\boldsymbol{z})$ but repeatedly performs a linear encoding as in Eq. 5. Equivalently, the action of the style-based quantum generator can be seen as $\mathcal{U}_{\phi_g}(\boldsymbol{z}) = V_{\phi_g^2}(\boldsymbol{z}) \, E_{\phi_g^1}(\boldsymbol{z})$, where $\boldsymbol{\phi_g} = (\boldsymbol{\phi_g^1}, \boldsymbol{\phi_g^2})$.

By a proper tuning of the parameters $\boldsymbol{\phi_g^2}$ in $V_{\phi_g^2}(\boldsymbol{z})$, that is, by setting $\phi_g^{(l)}$ to zero and $\phi_g^{(l-1)}$ equal to each component of $\boldsymbol{\phi_g^{*2}}$ in Eq. 5, we obtain $V_{\phi_g^2}(\boldsymbol{z}) = V_{\phi_g^{*2}}$. Similarly, by setting $\boldsymbol{\phi_g^1} = \boldsymbol{\phi_g^{*1}}$, we obtain $E_{\phi_g^1}(\boldsymbol{z}) = E_{\phi_g^{*1}}(\boldsymbol{z})$. Hence, we recover the standard quantum generator from the style-based approach, namely, $\mathcal{U}_{\phi_g^*}(\boldsymbol{z}) = \mathcal{U}_{\phi_g}(\boldsymbol{z})$ and therefore $\mathcal{S}_\Psi^* = \mathcal{S}_\Psi$. Given the extra degrees of freedom in the style-based approach, it also holds that $\mathcal{S}_\Psi^* \subset \mathcal{S}_\Psi$ for any set of parameters. □

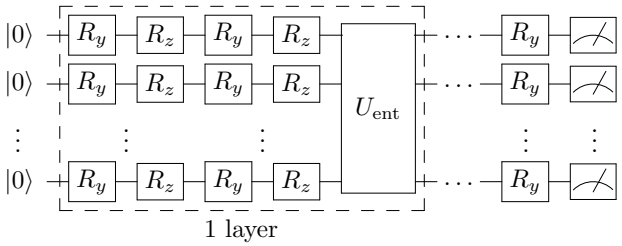

Figure 2: Quantum neural network employed for the qGAN model. As indicated by the dashed box, each layer is composed of a set of entangling gates U_{ent}, to be specified for each example, preceded by two alternating R_y and R_z single-qubit rotations. After implementing the layered network, a final layer of R_y gates is applied.

From the previous result, the following observation can be made. The training of a style-qGAN might be initialized with the optimal parameters of a pretrained qGAN model. This way, the style-qGAN improves the performance of the latter, given the extra degrees of freedom, yet using the same quantum circuit. This is particularly interesting for the NISQ era, where quantum resources are limited. In the worst case scenario, the style-qGAN remains with the same optimal parameters, and therefore, with an equal performance.

Recall that the quantum generator's task is creating fake samples to fool the classical discriminator. The fake samples are prepared by acting with the parameterized QNN on the initial n-qubit state $|0\rangle^{\otimes n}$, and then measuring in the computational basis. For our implementations, each qubit delivers one sample component. That is, the sample $\boldsymbol{x} \in \mathbb{R}^n$ is generated as

$$\boldsymbol{x} = \left(-\left\langle \sigma_z^1 \right\rangle, -\left\langle \sigma_z^2 \right\rangle, \ldots, -\left\langle \sigma_z^n \right\rangle\right) \, , \tag{6}$$

with

$$\left\langle \sigma_z^i \right\rangle = \left\langle \Psi_{\phi_g}(\boldsymbol{z}) \right| \sigma_z^i \left| \Psi_{\phi_g}(\boldsymbol{z}) \right\rangle \, , \tag{7}$$

where $\left|\Psi_{\phi_g}(\boldsymbol{z})\right\rangle$ is the output state from the quantum generator. Notice, however, that for other models, more sophisticated ways of generating fake samples could be more convenient to implement. For instance, one could generate a sample component by computing expectation values involving several qubits. Finally, let us briefly comment that we used a deep convolutional neural network for the discriminator. More details about the classical discriminator implementation can be found in the code [58].

3 Validation examples

In this section, we show examples of style-qGAN models obtained for known prior distribution functions in

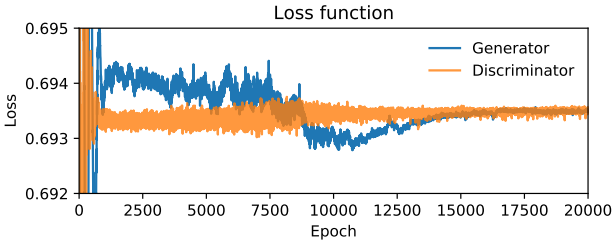

Figure 3: Example of loss function convergence. After an initial warm-up phase, the loss function of both models converges. This indicates that the style-qGAN has been successfully trained.

one and three dimensions. The results presented here have been obtained after a systematic process of fine-tuning and manual hyper-optimization of the training and quantum generator model.

3.1 1D Gamma distribution

In order to test the framework proposed above, we consider the sampling of a 1D gamma distribution with probability density function given by

$$p_\gamma(x,\alpha,\beta) = x^{\alpha-1}\frac{e^{-x/\beta}}{\beta^\alpha \Gamma(\alpha)}\,, \tag{8}$$

where Γ is the Gamma function. In this example we take $p_\gamma(x,1,1)$ as the input distribution and train a style-qGAN with 1 qubit, 1 latent dimension and 1 layer using 10^4 samples from the input distribution. The total number of trainable parameters is 10. We perform a linear pre-processing of the data to fit the samples within $x \in [-1,1]$. We undo this transformation after the training. Note that this single-qubit validation example is useful to test the quality of the style-qGAN model, as it has been shown that a single qubit is enough to approximate univariate functions [19]. In Figure 3 we show the evolution of the loss function for the generator and discriminator models in terms of the number of epochs. We observe the typical behavior of GAN training and a convergence region after 15000 epochs. The style-qGAN is trained with batch sizes of 128 samples.

A necessary property of this framework is that the style-qGAN model learns the underlying distribution from a small data set. To demonstrate this, we train a style-qGAN model with a set number of reference samples and then use it to generate two sample sets of different size. In particular, we train the style-qGAN with 10^4 reference samples and then use it to generate sets of 10^4 and 10^5 samples.

The top of Figure 4 shows the smaller sample distribution generated by the style-qGAN model in blue and a sampling of the reference distribution of the same size in red. This enables a comparison also using the Kullback-Leibler divergence (KL) [61]. In both cases, the 10^4 samples have been transformed

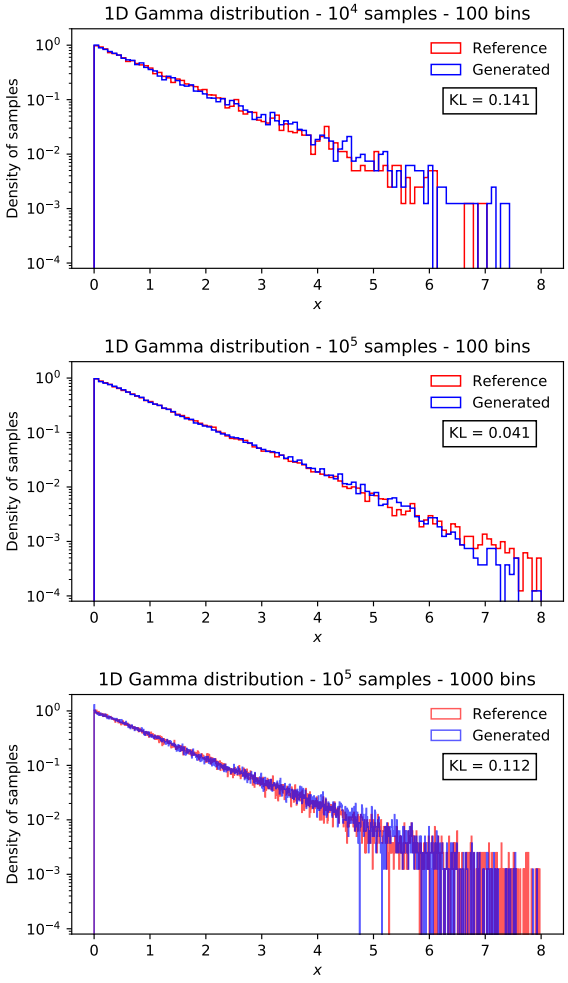

Figure 4: Examples of 1D gamma distribution sampling for the reference underlying distribution (red) and a style-qGAN model (blue) that has been trained with 10^4 reference samples. The top plot compares 10^4 generated samples using 100 bins. The middle plot compares 10^5 generated samples using 100 bins. Finally, the bottom plot compares 10^5 samples using 1000 bins, which enables a direct comparison of the KL divergences. We observe a good level of agreement between both distributions with low values of the KL distance, despite the model being trained on a small training set in addition to an almost constant KL in the data augmentation regime.

into histograms with 100 bins linearly spaced on the x-axis of the figure. We observe that the distributions are statistically similar even for this high-density binning choice. The KL divergence of two displayed distributions is 0.141, which entails a high degree of similarity.

Going further in the middle of Figure 4 we show the same results as in the top on the larger set containing 10^5 samples. Again, we use for comparison a re-sampling of the reference distribution at the same size as the generated set and show both distributions on a grid with 100 linearly spaced bins. Having generated an order of magnitude more samples than the training set we observe that the style-qGAN model

performs well. Both distributions are visually very close to each other and the KL divergence of 0.041 signals a high degree of similarity.

In order to compare the two KL divergences, note that they are computed on discrete histograms. Therefore, for a direct comparison, the number of bins for the larger sample set has to be increased proportionally to the increase in generated sample size, i.e. to compare with 100 bins for the 10^4 sample size we need to set 1000 bins for the 10^5 sample size. In this case, for a style-qGAN that provides an equally good description of the underlying distribution function, the KL divergence will stay constant or decrease. Here, with this change in binning, we find the KL divergence is 0.112. The results are shown in the bottom panel of Figure 4. This behavior confirms that the style-qGAN model is able to learn the underlying distribution function even if trained with a small training sample set. Such a feature is particularly interesting in the context of data augmentation applications [62, 63], where few samples are available, nonetheless the style-qGAN model can generalize and learn the underlying distribution with satisfactory outcome.

3.2 3D correlated Gaussian distribution

The previous test shows that a style-qGAN model implemented on a single qubit can be trained and produce acceptable results. However, this particular set-up does not include entanglement between qubits. In order to study the impact of the entanglement term U_{ent} in the considered QNN, we select as an underlying distribution a 3D correlated Gaussian distribution centered at $\overline{\mu} = (0, 0, 0)$ with covariance matrix

$$\sigma = \begin{pmatrix} 0.5 & 0.1 & 0.25 \\ 0.1 & 0.5 & 0.1 \\ 0.25 & 0.1 & 0.5 \end{pmatrix}. \tag{9}$$

For this specific set-up, we consider a 3-qubit model with 3 latent dimensions and 1 layer. The U_{ent} consists of two controlled-R_y gates acting sequentially on the 3 qubits. The total number of trainable parameters is 34. As in the previous example, we perform a linear pre-processing of the data to fit the samples within $x \in [-1, 1]$, and then we undo this transformation after the training. In Table 1 we summarize the style-qGAN configurations obtained for both examples discussed in this section.

Following the same training procedure employed in Section 3.1 and again using 10^4 reference samples to train the style-qGAN model, we test how well our model samples this specific 3D correlated Gaussian distribution. The results are shown in Figure 5. In the top row, we compare the one-dimensional cumulative projections of samples generated by the style-qGAN model with the reference input distribution function for 10^5 samples. We again use a grid of 100 linearly spaced bins per dimension in order to highlight small differences between the prior reference distribution and the artificial samples. For this example, we also observe that the distributions are statistically similar as the corresponding KL distances are quite small and close to each other. In the second row, we show 10^5 samples produced by the style-qGAN model in two-dimensional projections.

	1D gamma	**3D Gaussian**
Qubits	1	3
D_{latent}	1	3
Layers	1	1
Epochs	3×10^4	1.3×10^4
Training set	10^4	10^4
Batch size	128	128
Parameters	10	34
U_{ent}	None	2 sequential CR_y

Table 1: Summary of the style-qGAN set-up for the 1D gamma distribution and the 3D correlated Gaussian distribution.

To further study the features of the style-qGAN model in the third row of plots in Figure 5 we show the two-dimensional projections of the ratio between samples generated from the prior reference distribution and the style-qGAN model. In this way, we can visualize how well the model learns not only the distributions but also the correlations between the dimensions of the problem. A ratio of one, given by a white coloring of the corresponding bin in the figure, would imply the reference and generated samples are identical. Note that we aim to generate unseen samples, not an identical copy of the reference set. However, at the same time, the model should not diverge significantly, depicted by deep blue and red in the figure, nor occupy space in the grey area of the figure. We observe a good level of agreement, in particular in those regions where the sampling frequency is higher. The largest deviations are seen at the edges of the distributions, where the sampling frequency is lower. These deviations are evidence of the limitations in our model, common to the GAN method; however, their severe appearance is an artifact of visualization due to data augmentation.

To better quantify how well the correlations have been learned, we study the covariance matrices defined by the reference and the generated samples. The summed eigenvalues of the reference and generated covariance matrices give a means to estimate the similarity between the learned underlying correlations. We find agreement between the reference and generated eigenvalues to the better of 10% for style-qGAN set-ups with equal and more than 3 latent dimensions. Recall that the latent variables are introduced in every gate of the circuit, including the entangling ones U_{ent}. With $D_{\text{latent}} < 3$ we observe significant deviations of factors of one order of magnitude while

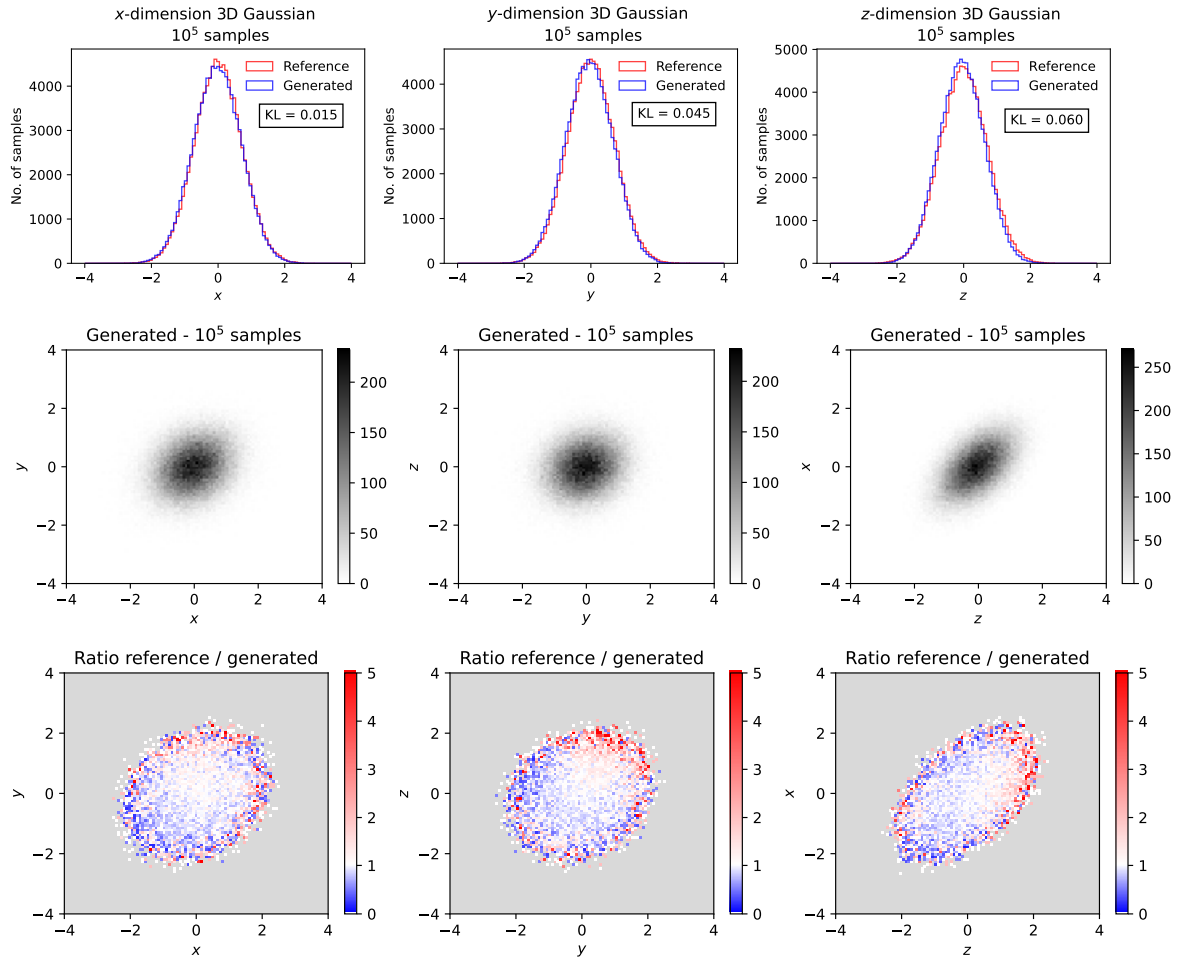

Figure 5: Marginal samples distributions for each dimension x, y, z of the 3D correlated Gaussian distribution for the style-qGAN model trained with 10^4 samples (top row), together with the corresponding two-dimensional sampling projections (middle row) and the ratio to the reference underlying prior distribution (bottom row). The style-qGAN generator model learns the correlations and provides acceptable samples when compared to the reference distribution. Note that we choose a grey background for the plots at the bottom row to more clearly highlight a ratio of one between reference and generated samples, indicated by white.

for $D_{\text{latent}} \geq 3$ no further significant improvement is seen. The same holds for increasing the number of layers in the style-qGAN model. This suggests that the number of latent dimensions introduced is a key hyperparameter once the number of layers allows a sufficient complexity. However, training success also depends on the convergence of the GAN parameters through optimization. This means that, in practice, having more layers and parameters than the minimal set might be a better choice.

Since the eigenvalues are known also exactly through Eq. 9 we furthermore can compare the performance of the style-qGAN with increased generation sample size. We find that the style-qGAN with 3 latent dimensions and 1 layer (shown here) generates sets that reproduce the exact eigenvalues of the input covariance matrix to better than $\lesssim 6\%$ for 10^3, $\lesssim 1.3\%$ for 5×10^3 and $\lesssim 0.8\%$ for 2×10^4 samples.

This analysis demonstrates a key property of a functioning GAN model – that the larger set of generated samples more closely agrees with the reference input distribution function. The observation that our style-qGAN fulfils this property confirms its viability as a functioning quantum implementation of the generative adversarial network idea for multi-dimensional correlated data.

4 Generating LHC events

After the validation of the style-qGAN model presented in the previous section, let us consider a training dataset from HEP. One of the big challenges involving Monte Carlo (MC) event generation for modelling physics processes in HEP is the large number of data points required in order to compare predictions of physical observables to experimental data.

In this context, we have generated 10^5 MC events for $pp \to t\bar{t}$ production at LHC with $\sqrt{s} = 13$ TeV with MadGraph (`MG5_aMC` [64, 65]) at leading order

in the strong coupling constant. From this simulated events we sample the Mandelstam variables (s, t) and the rapidity. Here, s and t are understood as the local partonic variables, $s = (p_1 + p_2)^2$, $t = (p_1 - p_3)^2$, where p_1 and p_2 are the four-momenta of the incoming quarks within the proton that collide to produce a top quark with four-momentum p_3 and an anti-top quark with four-momentum p_4. Note that all momenta are given in the center-of-mass frame.

We consider a 3-qubit model with 5 latent dimensions and 2 layers. Again, U_{ent} consists of two controlled-R_y gates acting sequentially on the 3 qubits. The total number of trainable parameters is 62. The style-qGAN model has been trained on 10^4 samples. See Table 2 for more details. In this case, we perform a linear preprocessing of the data to fit the samples within $x \in [-1, 1]$ after a power transform [66] from the Python package `Scikit Learn` [67]. As previously, we undo this transformation after the training.

Following the same training procedure employed in the previous section, in the top row of Figure 6 we compare the one-dimensional cumulative projections of samples generated by the style-qGAN model with the reference input distribution function for 10^5 samples. We use a grid of 100 linearly spaced bins for y and 100 log-spaced bins for s and t. For this example, the distributions are again statistically similar, with the corresponding KL distance being small and close to each other. In the second row of Figure 6 we show 10^5 samples produced by the style-qGAN model in two-dimensional projections.

The bottom row of plots in Figure 6 shows the ratio between samples generated from the prior original MC distribution and the style-qGAN model. Again, even for this physically-realistic model, we observe a remarkable level of agreement, especially in those regions where the sampling frequency is higher. Most importantly, we observe that the style-qGAN learns the correlations between the three dimensions.

Applying the same reasoning as in the previous section we compute the eigenvalues of the covariance matrices derived from the reference and generated data sets. To this extent we use the larger sized reference data set calculated previously using MadGraph (`MG5_aMC`). We find that the summed eigenvalues of the covariance matrices derived from samples generated by the shown style-qGAN with 5 latent dimensions and 2 layers agree with the corresponding reference to $\sim 9 - 13\%$ for 10^3, $\sim 8 - 15\%$ for 5×10^3 and $\sim 7 - 14\%$ for 2×10^4 samples. Here the quoted range originates from comparing different samples of the reference data. Furthermore, we suppress effects from the inverse transformation that converts the generated sample and instead focus on the learning capability of the style-qGAN model by estimating the covariances on the transformed reference data sets. It should be stressed that this test is slightly different from the one in the previous section since the exact eigenvalues are not known. As a result, the sampling error of the reference enters and an agreement at the previous level should not be expected as too close of an agreement would indicate the model is overfitted. However, our model exhibits the expected and necessary behavior, even when applied to realistic data.

	$pp \to t\bar{t}$ **LHC events**
Qubits	3
D_{latent}	5
Layers	2
Epochs	3×10^4
Training set	10^4
Batch size	128
Parameters	62
U_{ent}	2 sequential CR_y

Table 2: Summary of the style-qGAN set-up for the LHC events distribution.

5 Sampling from quantum hardware

In order to benchmark our style-qGAN model on real quantum hardware, we performed several runs on two different types of architectures. This allows us to qualitatively assess the impact of decoherence and noise, issues that are typical for NISQ computers, and to check whether the model can already give good results without waiting for error-corrected machines. The first quantum architecture we used is based on superconducting transmon qubits as provided by IBM Q quantum computers [1]. The second is based on trapped ion technology as provided by IonQ quantum computers [2] and accessible to us using cloud resources from Amazon Web Services (AWS).

Implementing our style-qGAN onto real quantum hardware introduces a new parameter into the model: the number of shots done for each calculation. Specifically, we now perform a quantum experiment each time we measure the three-qubit state, and we collect the results after a set number of experiments (shots) have been carried out. These then build up expectation values which are the generated samples. In this work we typically perform a number of 1000 shots per sample.

Prior to running on actual quantum hardware, we performed noise simulations using the IBM Q simplified noise model, which provides an approximation of the properties of real device backends, and enables us to test how well the results presented in Section 4 would be preserved in the noisy environment. Results

[1] IBM's roadmap for scaling quantum technology, Sept. 2020.

[2] Scaling IonQ's Quantum Computers: The Roadmap, Dec. 2020.

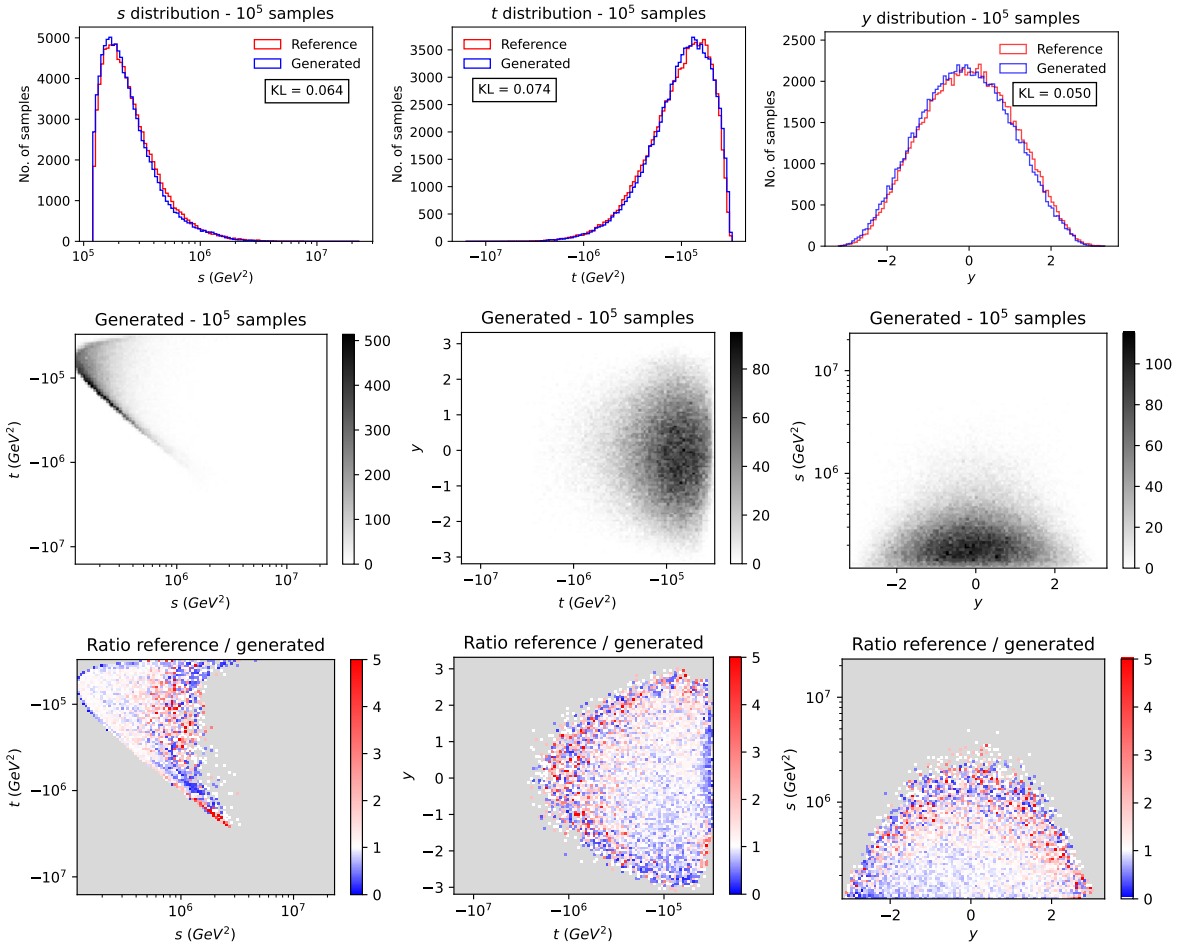

Figure 6: Marginal samples distributions for the physical observables s, t, y in $pp \to t\bar{t}$ production at the LHC for the style-qGAN model trained with 10^4 samples (top row), together with the corresponding two-dimensional sampling projections (middle row) and the ratio to the reference underlying prior MC distribution (bottom row). The style-qGAN generator model learns the correlations and provides acceptable samples when compared to the reference distribution. Note that we choose a grey background for the plots at the bottom row to more clearly highlight a ratio of one between reference and generated samples, indicated by white.

are provided in Appendix A and show that the impact of the noise is expected to be visible to a degree. We leave noise mitigation to further work. For the noise simulation as well as the actual runs on IBM Q quantum devices, we have selected in particular the `ibmq_santiago` 5-qubit Falcon r4L quantum processor. For our circuit, we need only three qubits with at least one directly connected to the other two. We use a translation layer written in `Qiskit` [68] to implement the circuit in Figure 2 and automatically select the three qubits out of the five that have the best noise properties. Note that this also allows us to test the impact of potential interference between qubits, as IonQ qubits are fully connected while those of IBM Q are not.

We present in Figure 7 examples of samples that have been generated using the `ibmq_santiago` machine on IBM Q. We use a 3-qubit model with 5 latent dimensions and 1 layer and for which the hyperparameters are the same as the ones used in Section 4 and trained on 10^4 samples. In contrast to the previous Sec. 4, for this implementation in the quantum hardware we reduced the number of layers to one. This means that we have trained a different style-qGAN with only one layer and then deployed the model to the quantum architecture. This change is motivated by the desire to diminish the effect of noise by reducing the depth of the circuit. Note, the analysis presented in Appendix A shows little deviation between the one- and two-layer result ratios, further strengthening this choice. To compute each fake sample, we have performed 1000 shots on the quantum circuits. In the top row of Figure 7, we compare the one-dimensional cumulative projections of samples generated by the style-qGAN model with the reference input distribution functions for 10^5 samples. The binning choice is equivalent to that used in Figure 6. In the middle row, we display the generation of 10^5 samples in two-dimensional projections. In the bottom row of plots in Figure 7, we show again the

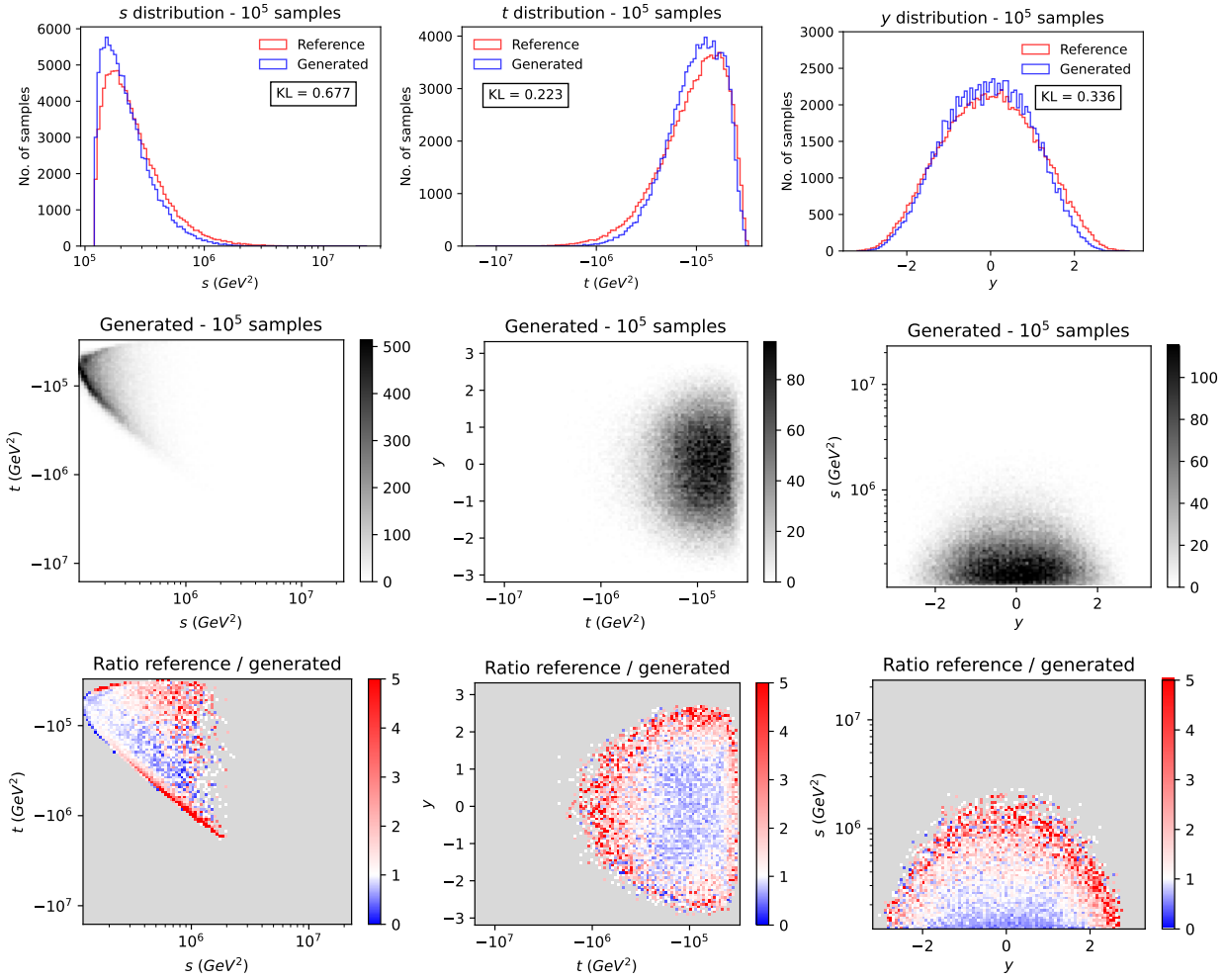

Figure 7: Marginal samples distributions for the physical observables s, t, y in $pp \to t\bar{t}$ production at the LHC using the style-qGAN generator model trained with 10^4 samples on `ibmq_santiago` (top row), together with the corresponding two-dimensional sampling projections (middle row) and the ratio to the reference underlying prior MC distribution (bottom row). Note that we choose a grey background for the plots at the bottom row to more clearly highlight a ratio of one between reference and generated samples, indicated by white.

ratio between the reference samples, generated using the MC event generator, and the samples generated by the style-qGAN on the `ibmq_santiago` quantum hardware. As expected, the agreement is worse than in Figure 6 because of the noise and reduced capacity of the quantum generator, nevertheless the results are reasonable. The style-qGAN generator model deployed in this NISQ hardware still manages to capture the correlations and provides reasonably good samples when compared to the reference distribution. The KL distances reported in the top row of plots are still relatively small, at most one order of magnitude larger than the KL distances reported in Figure 6.

During the current NISQ-era, the different quantum hardware architectures are not standardized and can have limits on the potential applications of the machines. As part of the implementation of our model onto quantum hardware, we were also able to study how the style-qGAN performs across different platforms. The aim is to understand whether and to what extent the style-qGAN's performance is hardware-dependent and also its potential hardware transferability. In view of this study of different quantum technologies, we have also performed a run with 10^3 samples only, on IonQ machines and separately on IBM Q machines. We have selected this fairly small amount of samples mainly due to external constraints on IonQ machine access on AWS. Note that the purpose of these tests is not to compare the two different hardware technologies, but instead to test whether the style-qGAN model works well on different quantum architectures. We use again a translation layer, written in Python with the `Braket SDK` from Amazon, between our circuit and the quantum hardware, and we have also performed around 1000 shots for the measurement of the generated samples. We stress again that although the amount of samples is quite low, the purpose of this test is to assess how the algorithm performs on different quantum technologies using the same amount of samples, not to obtain fine-

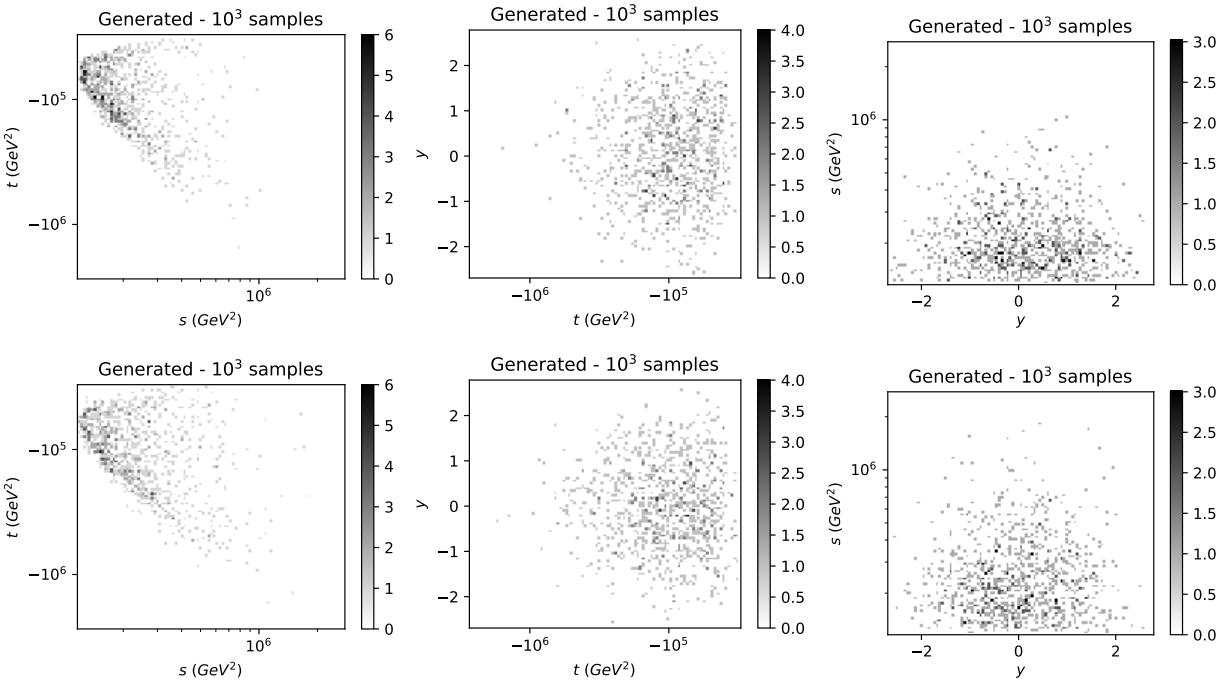

Figure 8: Example of two-dimensional sampling projections for $pp \to t\bar{t}$ production using the style-qGAN generator model on `ibmq_santiago` (top row) and IonQ (bottom row) trained with 10^4 samples.

grained results.

We show in Figure 8 the two-dimensional projections using IBM Q `ibmq_santiago` machine (upper row) and IonQ machine (lower row). It is clear that the sampling is sparser than in Figure 7, due to the lower number of samples; nonetheless, the style-qGAN captures the underlying distribution and correlations. This is particularly visible on the left plots for the $t - s$ correlation. The comparison between the upper row and the lower row also indicates that both architectures obtain similar results. This demonstrates that the style-qGAN can give good results on two different quantum hardware architectures.

	style-qGAN	**qGAN [39]**
Qubits	3	3
D_{latent}	5	5
Layers	2	2
Epochs	3×10^4	3×10^4
Training set	10^4	10^4
Batch size	128	128
Parameters	62	36
U_{ent}	2 CR_y gates	2 CR_y gates
KL s	0.064	0.338
KL t	0.074	11.171
KL y	0.050	0.049

Table 3: Summary of the style-qGAN and qGAN configuration for the training setup in Section 4. The bottom rows show the KL divergences for (s, t, y) samples obtained with both models after training.

6 Comparison with previous models

In order to quantify the quality of the generalized style-qGAN model described in the previous sections, we implement and compare the results obtained with the standard qGAN architecture presented in Refs. [34–36] and adapted to the continuous real sampling space in Ref. [39].

In Table 3 we summarize the differences between both architectures in the context of LHC event generation presented in Section 4. In order to provide a fair comparison we have tested both models keeping identical the number of qubits, layers, latent space dimension, training epochs, batch size and entangling gates. With this setup, the style-qGAN model provides 62 parameters to be trained while the qGAN model 36 parameters. This is an artifact of the linear encoding in Eq. 5 requiring two trainable parameters in each rotation, instead of one. Note that in order to increase the number of parameters of the qGAN model we have to include extra layers and thus modify radically its architecture. At the bottom of Table 3 we compute the KL divergences for (s, t, y) samples obtained with both models after training using the simulation setup presented in Section 4. While the KL divergence for Gaussian-like distributions, such as y, are similar between both models, there is a remarkable difference in terms of quality for non-Gaussian distributions, (s, t). This translates the lack of flexibility of the original qGAN which has been addressed in its generalization through the style-qGAN procedure.

7 Conclusion

In this work, we explore the use of quantum neural networks (QNNs) for Monte Carlo event generation, specifically for scattering processes at the Large Hadron Collider (LHC). We focus specifically on quantum generative adversarial networks (qGANs), which employ two competing networks, the generator and the discriminator, that are trained alternatingly. Here we propose a novel quantum generator model that does not follow the traditional path where the prior noise distribution is provided to the quantum generator through its first quantum gates. We instead choose to embed it on every single-qubit and entangling gate of the network. This allows for a generalization of the state-of-the-art qGAN implementations, achieving smaller Kullback-Leibler divergences even with shallow-depth networks. As a similar concept has been utilized in the classical context, coined as style-GANs, we choose to call our novel architecture a style-qGAN. A future interesting development of this work could be introducing additional features from the classical style-GAN, such as the preprocessing of the latent variables.

As this is a new quantum generative architecture, the body of this work focused on validating and assessing our methodology on various data sets and hardware architectures. In particular, we not only trained our model on toy data, namely 1D gamma and 3D correlated Gaussian distributions, but also on data for realistic quantum processes at the LHC, generated via MadGraph. For both toy data and realistic data, we saw strong evidence that the style-qGAN model could be used for data augmentation, as it was able to reproduce known reference distributions from small sample sets. Additionally, we deployed the models on two different quantum hardware architectures – superconducting qubits (IBM Q) and trapped ions (IonQ). Despite working with a small sample set, we observed that the style-qGAN works well, at least qualitatively, on different hardware architectures. This points to its hardware-independent viability.

The results presented here should be considered both as an improvement over the state-of-the-art from the algorithmic point of view, and as a proof-of-concept for a future quantum hardware deployment, providing a robust and reproducible starting point for future investigations. Nevertheless, this is a first attempt to bridge the power of quantum machine learning algorithms into the complexity of Monte Carlo simulation in HEP. We wish that the approach presented here will inspire new HEP applications that may benefit from quantum computing.

Code availability: The code is available in Ref. [58].

Acknowledgments

C.B.-P. acknowledges Stavros Efthymiou for useful discussions about the realization of the manuscript's code. S.C. thanks Marco Zaro and Luigi Favaro for discussions about classical GAN models applied to Monte Carlo events. This project is supported by CERN's Quantum Technology Initiative. M.C. is supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement number 843134. S.C. is supported by the European Research Council under the European Union's Horizon 2020 research and innovation Programme (grant agreement number 740006). The authors acknowledge the support of the CloudBank EU as a pilot brokering cloud service at CERN, allowing for access to Amazon Web Services in order to run our algorithm on IonQ hardware. The authors also acknowledge the use of IBM Quantum services for this work.

References

[1] J. Preskill, Quantum **2**, 79 (2018).

[2] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, *et al.*, Nature **574**, 505 (2019).

[3] H.-S. Zhong, H. Wang, Y.-H. Deng, M.-C. Chen, L.-C. Peng, Y.-H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu, *et al.*, Science **370**, 1460 (2020).

[4] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, Nature Reviews Physics **3**, 625–644 (2021).

[5] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, Reviews of Modern Physics **94**, 015004 (2022).

[6] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature **549**, 195 (2017).

[7] M. Schuld and F. Petruccione, *Supervised learning with quantum computers*, Vol. 17 (Springer, 2018).

[8] N. Wiebe, D. Braun, and S. Lloyd, Physical Review Letters **109**, 050505 (2012).

[9] S. Lloyd, M. Mohseni, and P. Rebentrost, arXiv preprint arXiv:1307.0411 (2013).

[10] P. Rebentrost, M. Mohseni, and S. Lloyd, Physical Review Letters **113**, 130503 (2014).

[11] I. Kerenidis and A. Prakash, Physical Review A **101**, 022316 (2020).

[12] A. W. Harrow, A. Hassidim, and S. Lloyd, Physical Review Letters **103**, 150502 (2009).

[13] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Quantum Science and Technology **4**, 043001 (2019).

[14] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, Advanced Quantum Technologies **2**, 1900070 (2019).

[15] C. Bravo-Prieto, J. Lumbreras-Zarapico, L. Tagliacozzo, and J. I. Latorre, Quantum **4**, 272 (2020).

[16] M. Larocca, N. Ju, D. García-Martín, P. J. Coles, and M. Cerezo, arXiv preprint arXiv:2109.11676 (2021).

[17] M. Schuld, R. Sweke, and J. J. Meyer, Physical Review A **103**, 032430 (2021).

[18] T. Goto, Q. H. Tran, and K. Nakajima, Physical Review Letters **127**, 090506 (2021).

[19] A. Pérez-Salinas, D. López-Núñez, A. García-Sáez, P. Forn-Díaz, and J. I. Latorre, Physical Review A **104**, 012405 (2021).

[20] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Nature **567**, 209 (2019).

[21] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, Physical Review A **101**, 032308 (2020).

[22] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, Quantum **4**, 226 (2020).

[23] T. Dutta, A. Pérez-Salinas, J. P. S. Cheng, J. I. Latorre, and M. Mukherjee, Physical Review A **106**, 012411 (2022).

[24] J. Romero, J. P. Olson, and A. Aspuru-Guzik, Quantum Science and Technology **2**, 045001 (2017).

[25] A. Pepper, N. Tischler, and G. J. Pryde, Physical Review Letters **122**, 060501 (2019).

[26] C. Bravo-Prieto, Machine Learning: Science and Technology **2**, 035028 (2021).

[27] C. Cao and X. Wang, Physical Review Applied **15**, 054012 (2021).

[28] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, npj Quantum Information **5**, 1 (2019).

[29] K. E. Hamilton, E. F. Dumitrescu, and R. C. Pooser, Physical Review A **99**, 062323 (2019).

[30] B. Coyle, D. Mills, V. Danos, and E. Kashefi, npj Quantum Information **6**, 1 (2020).

[31] P.-L. Dallaire-Demers and N. Killoran, Physical Review A **98**, 012324 (2018).

[32] S. Lloyd and C. Weedbrook, Physical Review Letters **121**, 040502 (2018).

[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Communications of the ACM **63**, 139–144 (2020).

[34] C. Zoufal, A. Lucchi, and S. Woerner, npj Quantum Information **5**, 1 (2019).

[35] J. Zeng, Y. Wu, J.-G. Liu, L. Wang, and J. Hu, Physical Review A **99**, 052306 (2019).

[36] H. Situ, Z. He, Y. Wang, L. Li, and S. Zheng, Information Sciences **538**, 193 (2020).

[37] L. Hu, S.-H. Wu, W. Cai, Y. Ma, X. Mu, Y. Xu, H. Wang, Y. Song, D.-L. Deng, C.-L. Zou, *et al.*, Science advances **5**, eaav2761 (2019).

[38] M. Benedetti, E. Grant, L. Wossnig, and S. Severini, New Journal of Physics **21**, 043023 (2019).

[39] J. Romero and A. Aspuru-Guzik, Advanced Quantum Technologies **4**, 2000003 (2021).

[40] M. Y. Niu, A. Zlokapa, M. Broughton, S. Boixo, M. Mohseni, V. Smelyanskyi, and H. Neven, Physical Review Letters **128**, 220505 (2022).

[41] T. Karras, S. Laine, and T. Aila, IEEE Transactions on Pattern Analysis and Machine Intelligence **43**, 4217 (2021).

[42] A. Pérez-Salinas, J. Cruz-Martinez, A. A. Alhajri, and S. Carrazza, Physical Review D **103**, 034027 (2021).

[43] W. Guan, G. Perdue, A. Pesah, M. Schuld, K. Terashi, S. Vallecorsa, and J.-R. Vlimant, Machine Learning: Science and Technology **2**, 011003 (2021).

[44] S. Y. Chang, S. Vallecorsa, E. F. Combarro, and F. Carminati, arXiv preprint arXiv:2101.11132 (2021).

[45] S. Y. Chang, S. Herbert, S. Vallecorsa, E. F. Combarro, and R. Duncan, EPJ Web of Conferences **251**, 03050 (2021).

[46] V. Belis, S. González-Castillo, C. Reissel, S. Vallecorsa, E. F. Combarro, G. Dissertori, and F. Reiter, EPJ Web of Conferences **251**, 03070 (2021).

[47] G. R. Khattak, S. Vallecorsa, F. Carminati, and G. M. Khan, The European Physical Journal C **82**, 1 (2022).

[48] P. Baldi, L. Blecher, A. Butter, J. Collado, J. N. Howard, F. Keilbach, T. Plehn, G. Kasieczka, and D. Whiteson, arXiv preprint arXiv:2012.11944 (2021).

[49] M. Backes, A. Butter, T. Plehn, and R. Winterhalder, SciPost Physics **10**, 89 (2021).

[50] A. Butter and T. Plehn, in *Artificial Intelligence For High Energy Physics* (World Scientific, 2022) pp. 191–240.

[51] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman, and T. Plehn, SciPost Physics **10**, 139 (2021).

[52] A. Butter, T. Plehn, and R. Winterhalder, SciPost Physics Core **3**, 9 (2020).

[53] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, and R. Winterhalder, SciPost Physics **8**, 70 (2020).

[54] A. Butter, T. Plehn, and R. Winterhalder, SciPost Physics **7**, 75 (2019).

[55] S. Efthymiou, S. Ramos-Calderer, C. Bravo-Prieto, A. Pérez-Salinas, D. García-Martín, A. Garcia-Saez, J. I. Latorre, and S. Carrazza, Quantum Science and Technology **7**, 015018 (2021).

[56] S. Efthymiou, S. Carrazza, S. Ramos, bpcarlos, AdrianPerezSalinas, D. García-Martín, Paul, J. Serrano, and atomicprinter, qiboteam/qibo: Qibo 0.1.6-rc1 (2021).
[57] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from tensorflow.org.
[58] afrancis heplat, C. Bravo-Prieto, S. Carrazza, M. Cè, J. Baglio, and d-m grabowska, Qtith/style-qgan: v1.0.0 (2021).
[59] M. D. Zeiler, arXiv preprint arXiv:1212.5701 (2012).
[60] M. Ostaszewski, E. Grant, and M. Benedetti, Quantum **5**, 391 (2021).
[61] S. Kullback and R. A. Leibler, The Annals of Mathematical Statistics **22**, 79 (1951).
[62] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)* (2018) pp. 289–293.
[63] F. H. K. dos Santos Tanaka and C. Aranha, arXiv preprint arXiv:1904.09135 (2019).
[64] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, Journal of High Energy Physics **07**, 079 (2014).
[65] R. Frederix, S. Frixione, V. Hirschi, D. Pagani, H. S. Shao, and M. Zaro, Journal of High Energy Physics **07**, 185 (2018).
[66] I.-K. Yeo and R. A. Johnson, Biometrika **87**, 954 (2000).
[67] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Journal of Machine Learning Research **12**, 2825–2830 (2011).
[68] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hernández, J. Carballo-Franquis, A. Chen, C.-F. Chen, *et al.*, Qiskit: An Open-source Framework for Quantum Computing (2019).

A Noise simulation on IBM Q device

We have performed a noise simulation of our style-qGAN on an IBM Q device, taking as a device baseline the `ibmq_santiago` 5-qubit Falcon r4L quantum processor that we have used for our runs on real IBM Q hardware.

The noise model takes into account the readout error probability of each qubit (mean value of the probability of reading $|1\rangle$ while being in the state $|0\rangle$, and the probability of reading $|0\rangle$ while being in the state $|1\rangle$), the relaxation time constants of each qubit (relaxation time and dephasing time), the gate error probability of each basis gate, and the gate length (timing of the gate) of each basis gate. The values are taken from the calibration information of the selected device for the noise simulation. Note that this calibration is performed at regular intervals. The generation of 10^5 samples on the actual machine took about one week, implying that the calibration parameters may have varied significantly during the full run.

We show in Figure 9 the result of our noise simulation. The KL distances displayed in the top row are comparable to the KL distances reported in Figure 7. We also compare our noise simulation to the run on actual IBM Q hardware, the latter being reported in Section 5; this is shown in the bottom row. The plots show a significant amount of white points, signaling that the noise simulation seems to capture most of the errors induced by running on actual quantum hardware and that errors beyond the parameters reported in the previous paragraph are subdominant.

We also compare our noise simulation to the noiseless simulation. The results shown in Figure 10 indicate that while the noise has an impact, as expected, there are still many points close to the ratio of one; therefore the style-qGAN still performs fairly well in a noisy environment. This has led us to believe that running on actual quantum hardware gives good results.

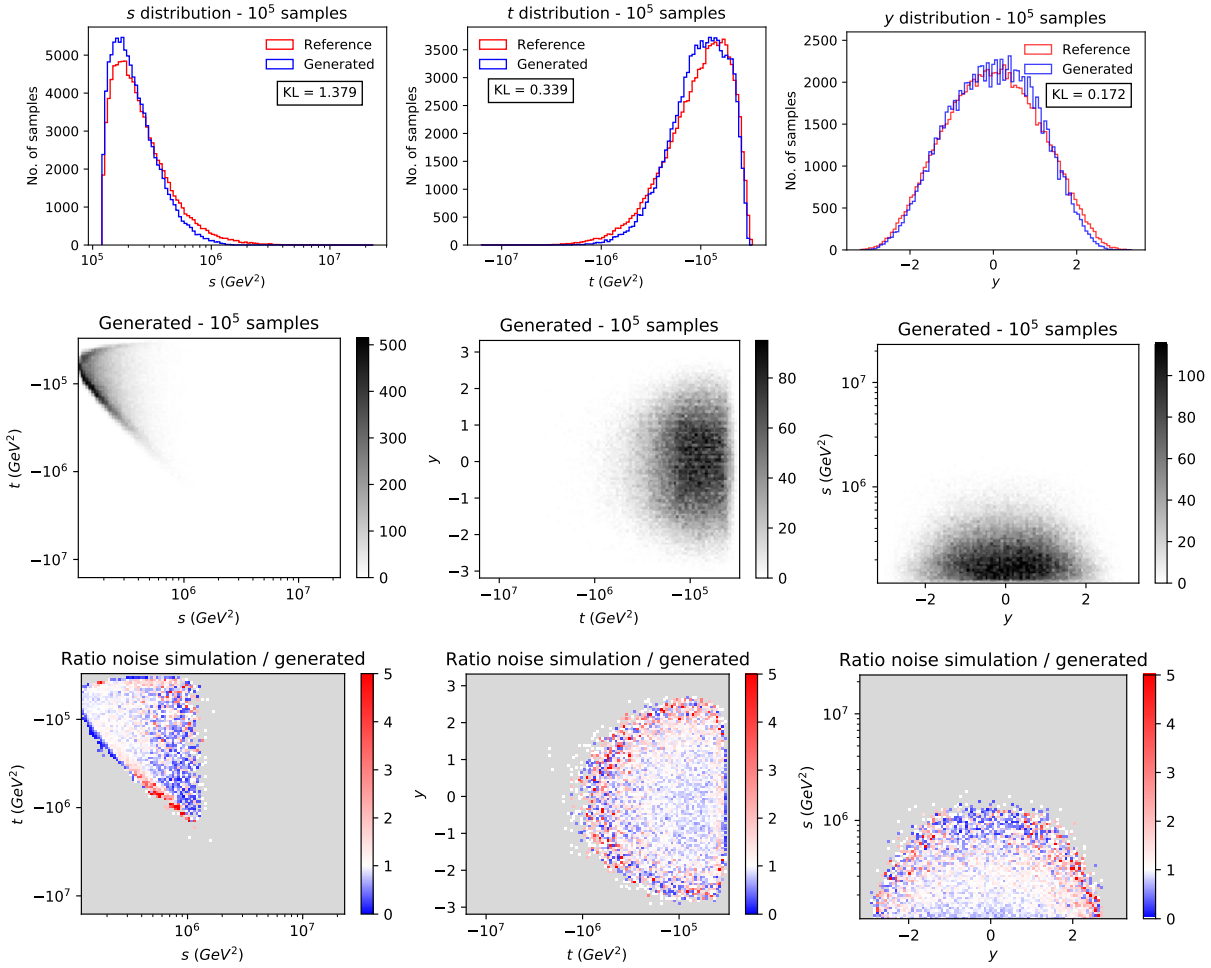

Figure 9: Marginal samples distributions for the physical observables s, t, y in $pp \to t\bar{t}$ production at the LHC using the style-qGAN generator model in a noise simulation of `ibmq_santiago` device (top row), trained with 10^4 samples (top row), together with the corresponding two-dimensional sampling projections (middle row) and the ratio to the reference underlying prior MC distribution (bottom row). Note that we choose a grey background for the plots at the bottom row to highlight when the reference and generated samples are identical.

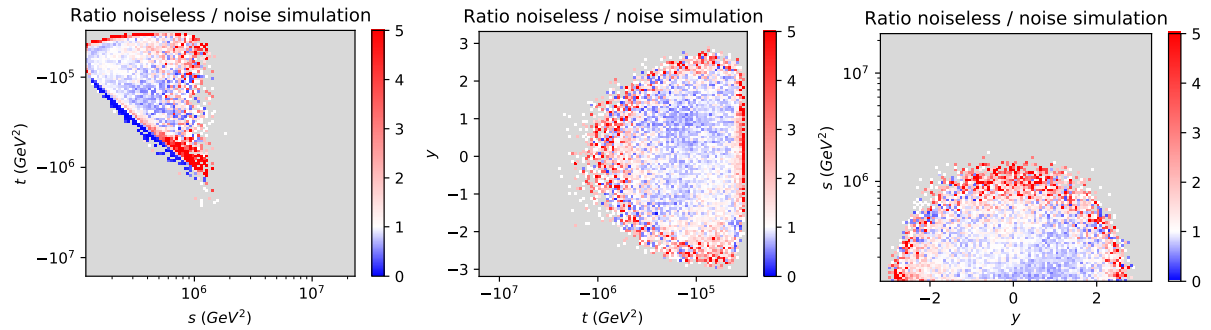

Figure 10: Ratio of two-dimensional sampling projections using the noise simulation of `ibmq_santiago` device to the corresponding noiseless simulation.