Regularizing Quantum Circuit Born Machines through enforcing sparsity

Ignacio Fernández Graña

May 24, 2021

Abstract

Quantum generative models have been proposed to exhibit stronger expressibility than their classical counterparts. Several approaches towards implementing these models are thought to be useful in the NISQ era, such as Quantum Generative Adversarial Networks or Quantum Circuit Born Machines (QCBM). In this work we implement and study a QCBM. The model is tested with two different target distributions, the first bein quantum data generated by a shallow QCBM and secondly with the standard Bars and Stripes dataset. We specifically focus on the effects of regularization in the performance of the model generalizing to unseen data.

[The code of this work is available `https://github.com/inafergra/Quantum-circuit-Born-machines`.]

Contents

1 Introduction

Current state-of-the-art quantum computers offer a very limited number of qubits, with reduced connectivity and quality. For this reason, a lot of research is being now done to build algorithms that can run in the quantum processors that will be available in the next few years. These are known as *Noisy Intermediate Scale Quantum* (NISQ) algorithms. Subject to strict hardware constraints, these algorithms should have a limited depth and use as little gates as possible, to avoid decoherence. Some of the fields where NISQ quantum algorithms are thought to be useful are optimization, quantum chemistry, finance or quantum machine learning, among others. Specifically, *Quantum Machine Learning* (QML) extends classical Machine Learning methods via the integration of quantum algorithms.

In this work we focus on a specific subclass of Machine Learning algorithms, *probabilistic generative models*. Given a set of samples drawn from a target distribution $\pi(\mathbf{x})$, the goal of is to learn to generate samples that are close to the unknown target distribution. The distribution generated by the model $p(\mathbf{x})$ should approach to $\pi(\mathbf{x})$ througout the learning process. By learning $\pi(\mathbf{x})$ the model should then be able to generalize to new unseen samples $\mathbf{x} \sim \pi(\mathbf{x})$ from the target distribution. Some examples of classical generative models are Naive Models or GANs [1].

Quantum generative models have gained a lot of attention recently, as they are thought to have a stronger expressibility than their classical counterparts [2]. A number of quantum generative models have been proposed during the last few years, as for example QGANS [3] or Quantum Circuit Born Machines. In this work we focus on the latter.

Quantum Circuit Born Machines (QCBMs) are a subclass of Parametrized Quantum Circuits (PQCs), where the parameters (usually angles of rotation) determining the circuit are optimized through a classical loop. QCBMs take

advantage of the inherent probabilistic nature of Quantum Mechanics. Born Machines can generate samples from a probability distribution via the measurement of the wavefunction at the end of the circuit. Specifically, given a final statevector $|\psi\rangle$, a measurement produces a sample $\mathbf{x} \sim p(\mathbf{x}) = |\langle \mathbf{x} \mid \psi \rangle|^2$. The generated distribution is therefore governed by *Born's measurement rule*. QCBMs are thought to be able to efficiently simulate probability distributions with sampling complexities that are intractable using classical computers [4], due to their stronger expressibility.

QCBMs are a class of *implicit* generative models, as one has no access to the statevector of a real quantum circuit. Implicit models do not specify an explicit parametrization of the target distribution, unlike *explicit* generative models, but rather they define a process that, after training, aims to draw samples from the underlying data distribution. In the case of QCBMs, this process is of course the PQC itself.

During the learning process of a QCBM, a function denominated *loss function* is minimized. The loss function is a measure of how well the model fits the the data. In this work we make use of the MMD loss, which is guarantee to converge to zero if and only if the generated distribution exactly matches the target distribution. One then needs to classically minimize the loss function. It is convenient to optimize this loss function through some *gradient-based* learning algorithm, as the gradient of the MMD loss can be efficiently computed by measuring expectation values in a quantum circuit, via the shift parameter rule. In this work we use the gradient-based algorithm use L-BFGS-B [5]. An important performance metric used used throughout this work is the Kullback-Leibler divergence, which defines a measure of *distance* between distributions.

One of the major problems arising in the learning process of a machine learning model is the so-called *overfitting*. Overfitting is a sign that the model is not correctly capturing the underlying true distribution, but instead it is memorizing the data that it uses for training. There are several ways to go around this issue. Most of the times, overfitting is a sign that the complexity of the model is not adequate for the problem it is trying to tackle. Overfitting can also due to a non sufficiently large data set to train the model. This affects the ability of the model to generalize to unseen data, which is vital for real-case application of many models. One way to regularize the model, and the one we will focus on in this work, is to enforce sparsity in the model [6].

The numerical simulations carried out in this work have been done with the Cirq framework for quantum computing [7].

The rest of the work is structures as follows. In section 2, the design of the QCBM we train is explained, and the learning process of the model is discussed in detail. In section 3 the numerical results are presented. We conclude the work in section 4.

2 The model

In this section we present the specific circuit design of the QCBM. A generic QCBM consists of a Parametrized Quantum Circuit (PQC) that generates samples from a specific distribution via measurement of the resulting quantum state. A classical optimization loop is then used to minimize the loss function and update the parameters accordingly, with the goal of learning the target distribution.

2.1 The ansatz

The QCBM proposed in this work consists of repeating blocks, each of them following the same structure. Each block has a rotation layer followed by a entangler layer. The rotation layer consists of R_x followed by R_y rotations in all the qubits, each rotation with a different trainable parameter. The entangler layer consists of CNOT gates between succesive qubits in the circuit, with no learnable parameters. The last qubit is also entangled with the first one with a CNOT. The QCBM is the formed by repeating these blocks a number of times, called *depth*.

2.2 The learning process

The learning process in a QCBM is and hybrid procedure, where the parameters $\vec{\theta}$ of the (quantum) model are optimized in a classical loop. Many classical optimization schemes are available. As it was shown in [8], *gradient-based* optimization methods are more accurate and efficient than gradient-free optimization for training QCBMs, specially in the presence of noise. One can efficiently compute the gradient of the loss function with respect to $\vec{\theta}$ via the *shift parameter rule*, by just measuring expectation values in the circuit. In this work we make use of the L-BFSG-B optimization, which is known to work well in noiseless settings. This is adequate as we will not consider sampling noise in this work.

The function that is minimized during the learning process is known as *loss function*. It serves as a measure of how well the model fits the data. In this work we will work with *Maximum Mean Discrepancy* loss:

$$\mathcal{L} = \underset{x\sim p_\theta, y\sim p_\theta}{E}[K(x,y)] - 2\underset{x\sim p_\theta, y\sim \pi}{E}[K(x,y)] + \underset{x\sim \pi, \sim \pi}{E}[K(x,y)] \tag{1}$$

Here, $E()$ indicates expectation value and $K(x,y) = \phi(x)^T\phi(y)$ is the kernel function. The function ϕ maps the original Hilbert space to a higher-dimensional space. By expressing the above loss function only in terms of scalar products $\phi(x)^T\phi(y)$, one avoids the need to explicitly construct the mapping ϕ. This is known as the *kernel trick*. We use a gaussian kernel $K(x,y) = \frac{1}{c}\sum_{i=1}^{c} \exp\left(-\frac{1}{2\sigma_i}|x-y|^2\right)$, with $c = 2$ and $\sigma_{1,2} = 0.25, 4$.

The gradient of this loss can be exactly computed via the shift parameter rule as a function of expectation values in the circuit under shifted parameters of the circuits:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \underset{x\sim p_{\theta^+}, y\sim p_\theta}{E}[K(x,y)] - \underset{x\sim p_{\theta^-}, y\sim p_\theta}{E}[K(x,y)] - \underset{x\sim p_{\theta^+}, y\sim \pi}{E}[K(x,y)] + \underset{x\sim p_{\theta^-}, y\sim \pi}{E}[K(x,y)] \tag{2}$$

where $p_{\theta^+}(x)$ and $p_{\theta^+}(x)$ are the output probabilities of the QCBM with the i-th parameter shifted, $\vec{\theta}^{\pm} = \vec{\theta} \pm \frac{\pi}{2}\vec{e}_i$, where $\vec{e}_i$ is a unit vector in the i-th direction. The number of measurements of the quantum circuit used to determine the expectation values in equations 1 and 7 are called *shots*.

In a real quantum computer one does not have access to the actual statevector of the circuit. Therefore, to measure the expectation values, one has to repeatedly prepare the final wavefunction and measure it. The number of times N that one samples from the distribution is popularly called *shots*. Nevertheless, since we are simulating our QCBM in a classical computer, we can exactly compute the statevector and therefore compute the expectation values exactly. Indeed , in this work we will mainly work with the statevectors themselves, as our focus is on the study of the regularization of the model and not on the effects of sampling noise.

An important performance metric used in this work is the Kullback-Leibler divergence. The KL-divergence between the true distribution π and the generated distribution p_θ is defined as

$$KL\left(\pi \| p_\theta\right) = \sum_x \pi(x) \ln\left[\pi(x)/p_\theta(x)\right] \tag{3}$$

and indicates a measure of distance between the two given distributions.

2.3 Regularization and sparsity

Many regularization techniques have been proposed to prevent Machine Learning models to overfit. In general lines, regularization discourages the model to become too complex or flexible while fitting the training data, with the goal of having a better generalization ability to unseen data.

We say a model is *sparse* if a number of parameters in the model, $\vec{\theta}$, are 0 or close to zero. Enforcing sparsity therefore means to make the model learn how to fit the target data with the additional constraint of keeping the norm of $\vec{\theta}$ small. There exists several ways to enforce sparsity during training. One of them is the so-called *dropout*, where a certain percentage of the parameters are randomly set to zero after some time during training. One can also enforce sparsity the model by adding an additional term to the loss function. This additional term represents some notion of norm of the vector parameter $\vec{\theta}$. Some common norms used in ML are the $L1$ norm (*Lasso* regularization) and the $L2$ norm (*Ridge* regularization), defined as:

$$||\vec{\theta}||_1 = \sum_i |\theta_i| \ , \qquad ||\vec{\theta}||_2 = \left(\sum_i |\theta_i|^2\right)^{\frac{1}{2}} \tag{4}$$

where θ_i are the components of the vector $\vec{\theta}$. There are some differences between choosing the L1 and L2. In this work we use the L2 norm as it has friendlier derivative, since the L1 norm involve absolute values

The additional regularization term in the loss function is thereforeis given by:

$$\mathcal{L}_2 = \underset{x\sim p_\theta, y\sim p_\theta}{E}[K(x,y)] - 2\underset{x\sim p_\theta, y\sim \pi}{E}[K(x,y)] + \underset{x\sim \pi, \sim \pi}{E}[K(x,y)] + \lambda\ ||\vec{\theta}||_2. \tag{5}$$

where λ is an hyperparameter of the model that controls how much the model is regularized. The optimal value of this hyperparameter is usually dependent on the specific problem we are considering, and is determined through an hyperparameter search.

By adding this term, we are forcing the model to minimize the usual MMD loss *while* minimizing the norm of the parameter vector. This will, effectively, prevent the model to overfit the training data, as the miniminum of the loss function now is not exactly the configuration that minimizes the MMD loss with the training data.

The gradient of the loss will also change. In the case of the L2 norm, the gradient is given by:

$$\frac{\partial ||\vec{\theta}||_2}{\partial \theta_i} = \frac{|\theta_i|}{\sqrt{\sum_j |\theta_j|^2}} \tag{6}$$

so the regularized gradient of the loss is given by:

$$\frac{\partial \mathcal{L}_\in}{\partial \theta_i} = \underset{x\sim p_{\theta^+}, y\sim p_\theta}{E}[K(x,y)] - \underset{x\sim p_{\theta^-}, y\sim p_\theta}{E}[K(x,y)] - \underset{x\sim p_{\theta^+}, y\sim \pi}{E}[K(x,y)] + \underset{x\sim p_{\theta^-}, y\sim \pi}{E}[K(x,y)] + \lambda \frac{|\theta_i|}{\sqrt{\sum_j |\theta_j|^2}} \tag{7}$$

3 Numerical results

In this section we present the results of simulating the learning of the QCBM in a classical computer, and the numerical results are presented. The proposed QCBM has been tested with two different target distributions, and the effects of enforcing sparsity in the model is benchmarked. Firstly we make the model learn the probability distribution generated by a shallow QCBM. Next we try the model in the standard *Bars and Stripes* (BAS) dataset, frequently used in ML.

3.1 Shallow QCBM

The first target distribution we use is one generated by another QCBM. This first QCBM will be *shallow*, meaning that it will consist of a reduced number of blocks, in this case just 3 blocks. These blocks have the structure presented in section 2.1. The model we train to simulate this target distribution is a *deep* QCBM. This deep QCBM is made of 15 blocks, also following the same design.

To study the effects of sparsity as a regularization technique, we need to test the model with a set of samples which the model has not been trained with. This set is called the *test set*, in contraposition with the *training set* that the model uses to learn. In this case, both the training set and the test set consist of 100 random samples drawn from the target distribution output by the shallow QCBM. The generated distribution by the deep QCBM is exactly computed with the amplitudes of the statevector, this is, with $N = \infty$ shots. Note that, since both the training set and test set are drawn randomly from the target distribution, the test set is equally likely to deviate from the true distribution than the training set.

To choose the regularization hyperparameter λ in equation we performed a sweep search. Figure 1a shows the KL-divergence between the test set and the generated distribution for different values of λ, where $\lambda = 0$ represents the unregularized model. The KL-divergence in this figure is averaged over 5 different trainings of the QCBM with different test and training sets. This is done in order to avoid 'getting lucky' in one run by sampling an optimal test/training set from the target distribution. We must focus on the KL-divergence of the test set, as the goal of regularization is to improve the performance in the test set. We can see that for values of λ between 0 and $8.8 \cdot 10^{-3}$ the tendency of the KL-divergence is decreasing. After $\lambda = 8.8 \cdot 10^{-3}$, the KL-divergence keeps growing for increasing λ. This is not shown in the figure but it was checked it in the numerical experiments. The KL-divergence for the training set is also shown in figure 1b. As expected, it increases for every value of $\lambda > 0$. Indeed, we can see a correlation between the dips and peaks of the KL-divergence in the training and the test set. In figure 2 we show a comparison between the generated distributions in the unregularized and regularized settings.

Nevertheless, the tendency of the KL-divergence is clearly non-linear with varying λ. This makes sense as the MMD loss function is also highly non-linear, and therefore adding an additional term will change the training landscape completely. Due to this highly non-linear behaviour, it is hard to give some general guidelines on how to choose the regularization parameter. Another reason for this high non-linearity is that the model we are using is rather small, meaning that it does not have a lot of trainable parameters. Changing the value of a single parameter will give a very large and unpredictable variation in the final result. This is the reason why λ has to very small as well, also since smaller models tend to overfit less.

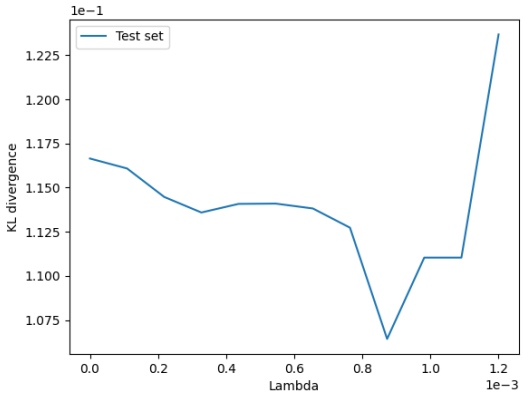
3.2 Bars and stripes

Next, we make use of the *Bars and Stripes* (BAS) dataset [9, 8] consisting of vertical bars and horizontal stripes in $n \times m$ pixel 2D images. For a $n \times m$ pixel grid, there are only $2^n + 2^m - 2$ valid configurations out of the $2^{n \cdot m}$ total possibilities. For a 2×2, this translates into 6 valid configurations out of the total 16 configurations. One needs nm qubits to represent a $n \times m$ grid on the quantum circuit.

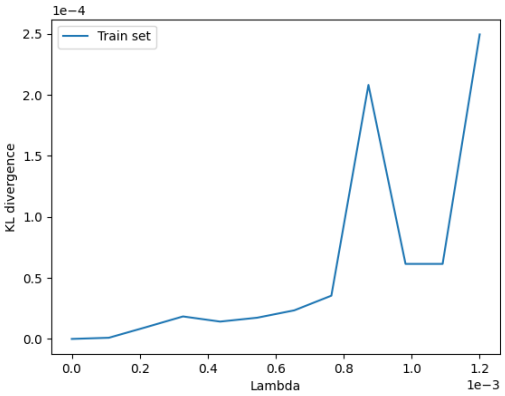
We first start with a 3×3 pixel grid. This grid can be thought as a binary array where each element can have a value 0 or 1, and a bar (stripe) is formed if a whole column (row) has the same value. Valid configurations are therefore only present bars or stripes. Each of the configurations can be assigned a number by flattening the binary array and converting the resulting binary number to decimal. Note that we can always have access to the true distribution of a BAS data set. This is just a distribution where valid configurations have equal non-zero probabilities and the non-valid configurations have 0 probability.

We first want to check whether the same deep QCBM with depth 15 used in the previous section is expressive enough to capture the distribution of the 2×2 BAS dataset. To this end, we use an unregularized model and train it with the true distribution. This means we are *not* sampling from the target distribution, but feeding the *true* target distribution to the model. Therefore, it does not make sense to talk about training and test sets here. Indeed, the model can easily memorize the target distribution, as it achieves a KL-divergence of . Figure 5 shows a comparison between the true and the generated distribution. Note that regularization techniques do not make sense in this case, as no generalization of the data is taking place: the training set is the true target distribution we want the model to learn.

Of course, for real use-application, one does not have access to the true target distribution, and one needs to sample from it to learn. The model should then be able to generate data according to the distribution underlying these samples. Therefore the goal is to learn how to generalize to data which the model has not been trained with. To test the generalization capabilities of the model, we use a test set and a training set, similarly to the previous section. Both the test set and the training set are made of 300 random samples drawn from the 2×2 BAS dataset. We compare the KL-divergence between the distribution generated by the trained model and the test set by varying the regularization

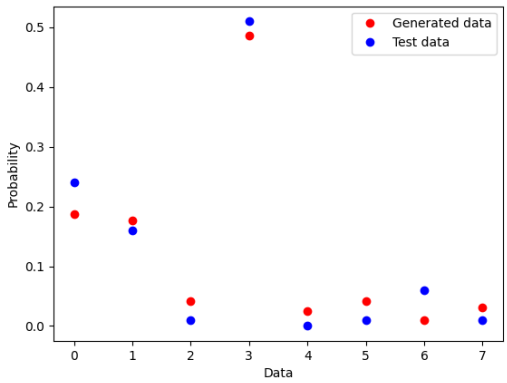


(a) Test set.

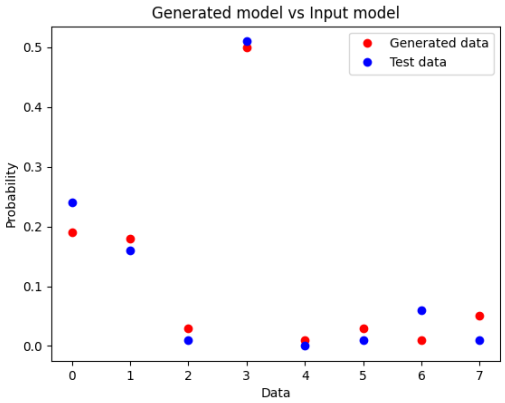


(b) Training set.

Figure 1: Comparison of the value of the KL-divergence between the test/training set of the shallow QCBM and the generated distribution by the deep QCBM for different values of λ. The values of the KL-divergence shown here are taking as an average over 5 runs. Note the different scales on the y-axis of both figures. The generated distribution was sampled with $N = \infty$ shots from the deep QCBM and the train and set distribution were sampled with $N = 100$ shots from the shallow QCBM. The minimum KL-divergence in the test set is reached at $\lambda = 8.8 \cdot 10^{-3}$, which is taken as the optimal value.



(a) Generated data by the unregularized model with $\lambda = 0$. The KL-divergence is 0.1169.



(b) Generated data by the regularized model with $\lambda = 8.8 \cdot 10^{-3}$. The KL-divergence is 0.1065

Figure 2: Comparison of the distribution generated by the test data set and the generated distribution by the deep QCBM, both in the unregularized case where $\lambda = 0$ (a) and the regularized case (b) where $\lambda = 8.8 \cdot 10^{-3}$. The test set is drawn from the shallow QCBM. The fact that the KL-divergence is lower for the regularized model showcases a successful application of regularization. At this optimal λ there is a decrease of 8.6% in the test KL-divergence.

(a) Valid (b) Valid (c) Non valid (d) Non valid

Figure 3: Examples of valid and non valid configurations in the 2×2 BAS dataset.

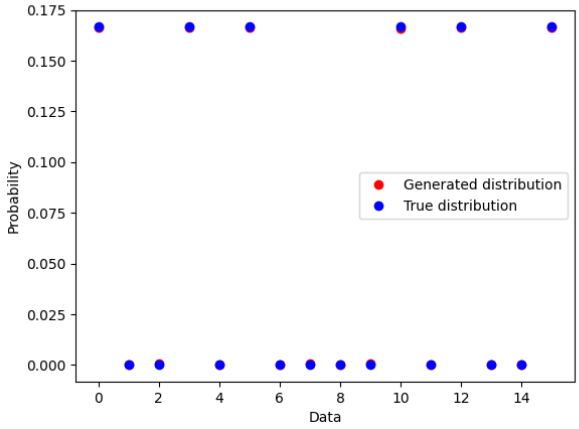


Figure 4: Comparison between the true distribution of the 2×2 BAS dataset, in blue, and the generated distribution by the unregularized model, in red. The KL-divergence is 0.021.

pararameter λ in figure 5, where the values of the KL-divergence are averaged over 5 runs of the model. The minimum value is obtained at $\lambda = 2.6 \cdot 10^{-3}$. Similarly to the previous section, we can see that the landscape is very non-linear. This makes getting conclusive results about the effects of enforcing sparsity a difficult task. In figure 6 we show a comparison between the generated distributions in the unregularized and regularized settings.

3.3 Is regularization necessary?

From the results in both of the studied data sets, we have seen that indeed, for certain values of the parametrization parameter λ, the model decreased the KL-divergence with the test set with respect to the unregularized model. Nevertheless, the QCBM studied in this work has a relatively low number of parameters, and as can be seen in both datasets, the effects of regularization are highly-non lineal. In fact, it could even seem like the apparent increase in performance in the test set we observed is not due to the regularization itself, but that it is rather related to the intrinsic randomness of quantum circuits. In both data sets the maximum decrease of the KL-divergence between the model's output distribution and the target distribution was just below a 10%, and it was achieved with a very low value of λ, in the order of 10^{-3}. With a larger model with more parameters, a model is more likely to overfit the data, as it has a stronger expressibility. It would therefore be interesting, as future outlook, if the effects of regularization are enhanced in bigger models, thus offering a more solid proof in favour of regularization. The caveat being, of course, that classically simulating big quantum circuits quickly becomes computationally intractable.

Finally, the risk of overfitting is also related to the amount of available data for training. In generative models, one gets the data by sampling from the target distribution. If the samples drawn from this distribution are noiseless, i.e., they perfectly represent the underlying distribution, then the model would be able to learn this target distribution by just sampling a sufficiently big number of data examples, avoiding the need of regularization. The number of needed samples will of course depend on the sampling complexity of the target distribution. On the other hand, if we do not have access to good quality data, or if the number of samples is not enough, then regularization becomes a must, as the drawn samples are not a good representation of the target distribution.

4 Conclusions

In this work we have implemented and studied a class of quantum generative model called Quantum Circuit Born Machines. The goal of this algorithm is to learn to generated a given target distribution. To benchmark the performance of the model two different target distributions were used. First, the QCBM was trained to learn a distribution generated by another shallower QCBM, and then the model was trained to generate the 2×2 BAS dataset. The models where shown to be expressible enough to accurately generate samples from this datasets provided access to the true distribution. Besides, we studied whether if a regularized model has an improved generalization ability. The numerical results show regularization being beneficial in both datasets, after a thorough hyperparameter search for the optimal value of the regularization parameter λ. Nevertheless, due to the model and dataset sizes, one can not extract general conclusions about regularization for these models.

As a future outlook, training a bigger model is likely to give more conclusive results about regularization effects. A finer optimization of the hyperparameters would also probably yield more clear results. On another note, other regularization techniques, such as L1 regularization or dropout, could be studied and implemented, complementing the results of this work.

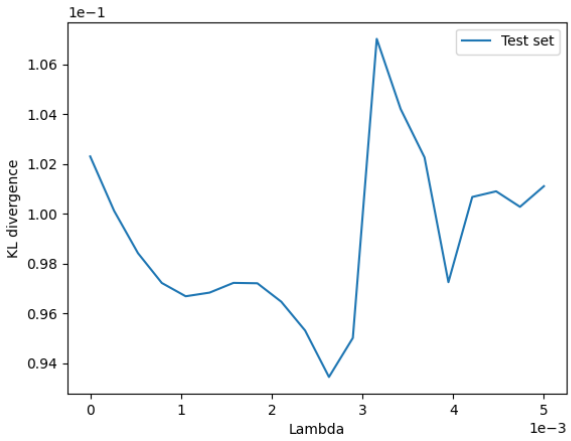


Figure 5: Comparison of the value of the KL-divergence between the test set and the generated distribution by the deep QCBM for different values of λ. The generated distribution was sampled with $N = \infty$ shots from the deep QCBM and the train and set distribution consist of 300 random samples from the 2×2 BAS dataset. The minimum KL-divergence in the test set is reached at $\lambda = 2.6 \cdot 10^{-3}$, which is taken as the optimal value. At this optimal λ there is a decrease of 9.6% in the test KL-divergence.

References

[1] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[2] A. P. Lund, Michael J. Bremner, and T. C. Ralph. "Quantum sampling problems, BosonSampling and quantum supremacy". In: *npj Quantum Information* 3.1 (Apr. 2017), p. 15. ISSN: 2056-6387. DOI: 10.1038/s41534-017-0018-2. URL: https://doi.org/10.1038/s41534-017-0018-2.

[3] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. "Quantum Generative Adversarial Networks for learning and loading random distributions". In: *npj Quantum Information* 5.1 (Nov. 2019). ISSN: 2056-6387. DOI: 10.1038/s41534-019-0223-2. URL: http://dx.doi.org/10.1038/s41534-019-0223-2.

[4] Sergio Boixo et al. "Characterizing quantum supremacy in near-term devices". In: *Nature Physics* 14.6 (June 2018), pp. 595–600. ISSN: 1745-2481. DOI: 10.1038/s41567-018-0124-x. URL: https://doi.org/10.1038/s41567-018-0124-x.

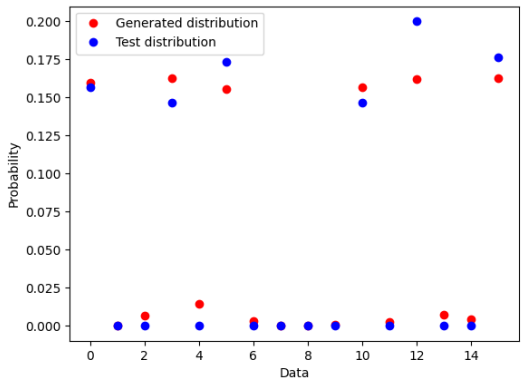
[5] Jinfeng Zeng et al. "Learning and inference on generative adversarial quantum circuits". In: *Physical Review A* 99.5 (May 2019). ISSN: 2469-9934. DOI: 10.1103/physreva.99.052306. URL: http://dx.doi.org/10.1103/PhysRevA.99.052306.

[6] Casey Meehan, Kamalika Chaudhuri, and Sanjoy Dasgupta. *A Non-Parametric Test to Detect Data-Copying in Generative Models*. 2020. arXiv: 2004.05675 [cs.LG].

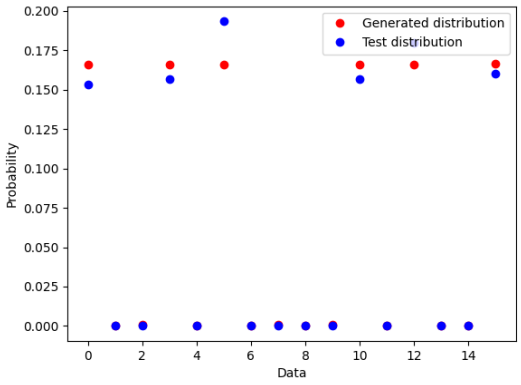
[7] Cirq Developers. *Cirq*. Version v0.11.0. See full list of authors on Github: https://github .com/quantumlib/Cirq/graphs/c May 2021. DOI: 10.5281/zenodo.4750446. URL: https://doi.org/10.5281/zenodo.4750446.

[8] Jin-Guo Liu and Lei Wang. "Differentiable learning of quantum circuit Born machines". In: *Physical Review A* 98.6 (Dec. 2018). ISSN: 2469-9934. DOI: 10.1103/physreva.98.062324. URL: http://dx.doi.org/10.1103/PhysRevA.98.062324.

[9] Zhao-Yu Han et al. "Unsupervised Generative Modeling Using Matrix Product States". In: *Phys. Rev. X* 8 (3 July 2018), p. 031012. DOI: 10.1103/PhysRevX.8.031012. URL: https://link.aps.org/doi/10.1103/PhysRevX.8.031012.



(a) Generated data by the unregularized model with $\lambda = 0$. The KL-divergence is 0.1023.



(b) Generated data by the regularized model with $\lambda = 2.6 \cdot 10^{-3}$. The KL-divergence is 0.0938

Figure 6: Comparison of the test set distribution and the generated distribution by the deep QCBM, both in the unregularized case where $\lambda = 0$ (a) and the regularized case (b) where $\lambda = 2.6 \cdot 10^{-3}$, for the 2×2 BAS dataset. The values of the KL-divergence shown here are taking as an average over 5 runs. The fact that the KL-divergence is lower for the regularized model showcases a successful application of regularization.