Quantum Wasserstein Generative Adversarial Networks

Zohim Chandani

Supervisors: Dr Steven Herbert, Seyon Sivarajah

Cambridge Quantum Computing, Cambridge, UK

Abstract

Huge efforts by the global scientific community has been poured into devising quantum algorithms using their classical counterparts as inspiration. An example of this was recently published by Shouvanik et. al. [1] where the advantages of the Wasserstein metric in classical generative adversarial network (GAN) training was exploited and transferred to the quantum domain. Below, I employ the Wasserstein-GAN algorithm and utilise it to reproduce random 1, 2, 4 and 8 qubit statevectors. I outline the benefits of this in quantum chemistry calculations where gate count can be significantly reduced to help derive results from current NISQ machines.

1. Generative Adversarial Networks

The Generative Adversarial Network (GAN) algorithm was proposed by Goodfellow et. al. [2] to tackle problems in generative modelling within unsupervised learning. In the GAN architecture framework, the generative source is pitted against an adversary whose job it is to distinguish the data samples it is presented with as coming from either the real or the generated distribution. Conceptually, the generator can be thought of in analogy to a counterfeiter generating fake currency samples and the discriminator analogous to the police trying to detect counterfeit currency. Both parties try to update their strategies in each iteration of training until convergence to Nash equilibrium [3] is achieved.

Mathematically, the goal of a GAN is to train a generator network, $G(z;\theta_g)$, which produces samples from the data distribution, $p_{data}(x)$, by transforming vectors of noise z as $x = G(z;\theta_g)$. The generator network, G, is a differentiable function represented by a deep neural network (DNN) with tunable parameters θ_g. A second DNN, $D(z;\theta_d)$, represents the discriminator which is trained to output a single scalar, $D(x)$, where $D(x)$ is the probability that the data sample x is from the real source rather than the generated distribution. Hence, a well trained discriminator will maximise $D(x)$. We simultaneously train the generator to minimise $1-D(G(z))$. Overall, D and G play a minimax game with the value function $V(D,G)$:

$$\min_{\text{G}} \max_{\text{D}} V(D,G) = \mathbb{E}_{x\sim p_{data(x)}}[D(x)] + \mathbb{E}_{z\sim p_z}[1-D(G(z))] \tag{1}$$

where $\mathbb{E}$ is the expected value.

A schematic of the GAN architecture is shown in Figure 1. Both models are trained using gradient descent and backpropagation. The generator never sees the real data sample but learns to improve its strategy of amalgamating noise and producing a data sample via the gradient signals it gets from the discriminator in each iteration. Nash equilibrium is reached when the discriminator is unable to differentiate between input samples, i.e. $D(x) = \frac{1}{2}$. Moreover, after training is completed, the discriminator network can be discarded given that the generator has no more updates to its strategy. For proofs on optimality of GAN training and convergence criteria satisfaction see [2]. Progress on results generated by GAN training from 2014 onwards is shown in Figure 2.

It is clear from Figure 2 that GANs can produce excellent samples, however, training GANs requires finding Nash equilibrium of a non-convex game with continuous high dimensional parameters. Using gradient descent techniques to find Nash equilibrium may cause algorithm convergence to fail. Followup work by Goodfellow in 2016 [5] introduces new techniques motivated by a heuristic understanding of the non-convergence problem which leads to improved learning performance and sample generation.

1.1. Wasserstein Metric

Machine learning problems are posed in such a manner that the solution lies at the minimum of a loss landscape which is found via updating parameters through gradient descent. In GAN training, this amounts to

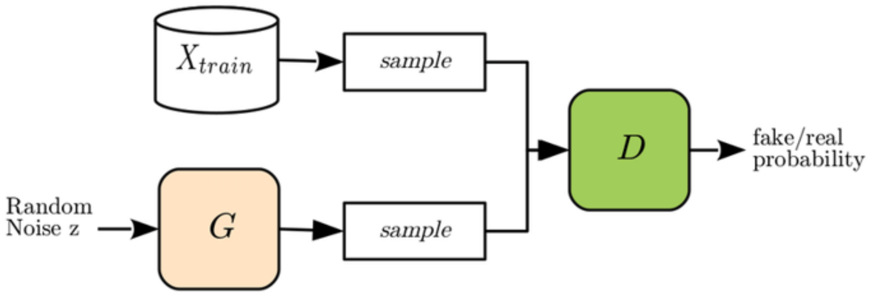


Figure 1: Generative Adversarial Network. Figure reproduced from [4].

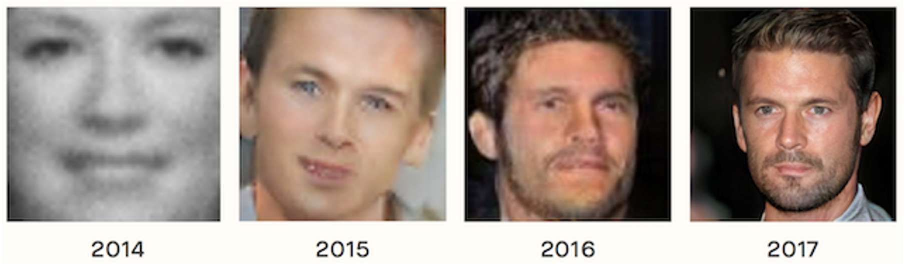


Figure 2: Improvements in GAN training to generate facial images - Goodfellow et al. (2014), Radford et al. (2015), Liu and Tuzel (2016), and Karras et al. (2017).

iteratively minimising the distance between the real data distribution, $\mathbb{P}_r$, and the parameterised distribution, $\mathbb{P}_\theta$. Many such distance measures exist, namely the Kullback-Leibler (KL) divergence, $KL(\mathbb{P}_r||\mathbb{P}_\theta)$, total variation (TV) distance, $\delta(\mathbb{P}_r||\mathbb{P}_\theta)$ and the Jenson-Shannon (JS) divergence, $JS(\mathbb{P}_r||\mathbb{P}_\theta)$, each with their advantages and drawbacks.

The question of a suitable distance metric between the real and the parameterised distribution was studied by Arjovsky et. al. [6]. We denote $\rho(\mathbb{P}_r, \mathbb{P}_\theta)$ where ρ is the distance measure under study. A fundamental feature required for such distance measures is the convergence of probability distributions. Namely, a sequence of distributions $(\mathbb{P}_t)_{t\in\mathbb{N}}$ converges if and only if there is a distribution $\mathbb{P}_\infty$ such that $\rho(\mathbb{P}_r, \mathbb{P}_\infty)$ tends to zero which depends on how one defines ρ.

For situations with distributions supported by low dimensional manifolds, the parameterised model and the true distribution usually have non-negligible intersection and thus the KL distance is infinite. This is usually remedied by adding a small noise term to the model distribution to cover all examples, however, for image generation models, the noise degrades the quality of samples and makes them blurry [7]. Other issues of convergence and discontinuous loss functions arise for JS, KL, reverse, KL and TV metrics which are studied in Example 1 in [6]. The conclusion of this study allows the authors to introduce the Wasserstein metric

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = inf\ E_{(x,y)\sim\gamma}[||x-y||] \tag{2}$$

where x and y are data points in the probability distributions $\mathbb{P}_r$ and $\mathbb{P}_\theta$ and $\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_\theta)$. This ensures that γ belongs to the transport plan Π, and we pick a plan which minimises the cost. The authors prove the advantages of $W(\mathbb{P}_r, \mathbb{P}_\theta)$ as a loss metric compared to other measures because of its continuity and smoothness and introduce it in GAN training to propose the Wasserstein-GAN (WGAN) algorithm. Moreover, the authors emphasise that none of their presented results were cherry-picked examples and there was no observed evidence of mode collapse (generator produces only a limited variety of plausible samples) [8].

Another benefit of the WGAN is that it allows one to train the discriminator till optimality. Once the discriminator is at ideal performance, it simply provides a loss to the generator which is trained like any other DNN. This ensures that there is no requirement to balance the discriminator and generator capacity properly. The better the discriminator, the higher the quality of gradients we use to train the generator [6].

2. Quantum GANs

Quantum Generative Adversarial Networks (QGANs) replace DNNs and introduce parameterised quantum circuits (PQCs) to the GAN architecture where the goal is to generate quantum data samples [9]. The ability of quantum information processors to represent vectors in N-dimensional spaces using $logN$ qubits, and to perform manipulations of sparse low-rank matrices in time $O(poly(logN))$ implies that QGANs exhibit a potential exponential advantage over classical GANs when the object of the game is to reproduce the statistics of measurements made on high-dimensional data sets. The theoretical arguments for the structure and motivation of QGANs presented in [9] was accompanied by an experimental implementation approach by Demers et. al. [10]. For a simple tutorial example, see [11].

The quantum adversarial game is set up as follows. First the discriminator tries to improve its strategy with the generator parameters fixed. As long as the generator is producing data which is different from the statistics of the real sample, the discriminator can always adjust its strategy to discriminate true from fake data with probability $> 1/2$. During generator training with discriminator parameters fixed, the generator can always decrease the success probability of the discriminator by moving in a direction which decreases the relative entropy between the true and the generated data. For the purposes of this investigation, the data in this case amounts to generating arbitrary statevectors. Neither the generator nor the discriminator perform quantum tomography; optimisation drives them to the optimal result. Recent work has also explored the use of QGANs to facilitate efficient learning and loading of generic probability distributions into quantum states [12].

An important point mentioned in [10] is that of bias being induced in the discriminator. If the real or generated data is systematically fed into the discriminator, than the discriminator might learn to alternate its response and achieve superiority. To ensure that the discriminator doesn't rely on the statistics of the choice of the source, the choice of input data must be made by the toss of a fair coin.

Hybrid QGANs were also introduced in [9] where a combination of DNNs with PQCs are employed [13] however I shall only focus on the fully quantum mechanical case in my pursuit.

2.1. Quantizing Wasserstein Distance

Inspired by the success of using the Wasserstein distance as a sensible loss metric in the classical GAN training, Shouvanik et. al. [1] proposed a mathematical derivation to transfer the integral formulation to the quantum realm by utilising expectation value measurements of hermitian operators on quantum states.

A metric is defined such that it holds the three axioms of identity of indiscernibles, symmetry and the triangle inequality. These axioms also imply the non-negativity condition: $d(x, y) \geq 0 \; \forall \; x, y \in X$ where X is a set. However, the authors in [1] prove that the quantised version of the Wasserstein distance is a semimetric which need not hold the triangle inequality axiom. The final cost function to be employed in the minimax game is given by:

$$L = \min_{\mathrm{G}} \max_{\phi \psi} \mathbb{E}_{|R\rangle}[\psi] - \mathbb{E}_{|F\rangle}[\phi] - \mathbb{E}_{|R\rangle \bigotimes |F\rangle}[\zeta] \tag{3}$$

where $\mathbb{E}_{|R\rangle}[\psi] = Tr(|R\rangle \psi)$ which is the expectation value of measuring the hermitian ψ on state the statevector $|R\rangle$ and ψ is a product of paulis. ζ refers to the entropic regularising hermitian which is added to further smoothen the loss function in light of classical results [14]. The algorithm used to calculate the Wasserstein metric between two quantum states is as follows:

Algorithm 1: QWGAN algorithm. All experiments use the Scipy optimiser: Powell with discriminator and generator weights bound between 0-2 and 0-2π. Other hyperparameters are maximum number of iterations = 2000, maximum function evaluations = number of qubits $\times$ 3, function tolerance = 10^{-10} and argument tolerance = 10^{-3}.

1. The real circuit generates a real state - $|R\rangle$
2. The generator circuit generates a fake state - $|F\rangle$
3. The discriminator performs measurements of expectation values on $|R\rangle$ and $|F\rangle$
4. For the 1-qubit case, the discriminator first creates the hermitian operator

$$\phi = \alpha_0 X_0 + \alpha_1 Y_0 + \alpha_2 Z_0 + \alpha_3 I_0$$

   where X_0 is the X expectation value on qubit 0 and α's are the coefficients. The expectation value $\langle\phi\rangle = \langle F|\,\phi\,|F\rangle$ is measured.

5. The discriminator then creates the hermitian ψ where

$$\psi = \beta_0 X_0 + \beta_1 Y_0 + \beta_2 Z_0 + \beta_3 I_0$$

   and the expectation value $\langle\psi\rangle = \langle R|\,\psi\,|R\rangle$ is measured.

6. The regulariser term is calculated using $|R\rangle$ and $|F\rangle$, details of which can be found in the supplemented code [15].
7. The parameters to be tuned to **maximise** the loss for the discriminator are the $\alpha's$ and $\beta's$.
8. The parameters to be tuned to **minimise** the loss for the generator are the input parameters to the generator circuit i.e. the rotation angles.

```
while Fidelity < 0.99 or Wasserstein loss ≈ 0 do
    Maximise the Wasserstein loss for the discriminator
    Update the discriminator parameters
    Minimise the Wasserstein loss for the generator
    Update the generator parameters
    Calculate the updated loss metric with the new parameters
    Calculate the updated fidelity
end
```

9. This can be generalised to the n-qubit case where $\phi = \sum_{i=0}^{4n-1} \alpha_i A_i$ and $\psi = \sum_{j=0}^{4n-1} \beta_j B_j$ where A_i and B_j are tensor products of Pauli matrices.

The astute reader may notice that the while loop condition for $Fidelity < 0.99$ is not permissible on actual quantum devices. Fidelity calculations require measurement of statevectors which leads to collapse of the wave function. The adjacent condition of Wasserstein loss $\approx$ 0 should hence be used for an experimental implementation. Below I present results in both the fidelity and the Wasserstein loss domain to validate the numerical training performed.

An important point to note is one where the Wasserstein loss $\approx$ 0 and not = 0 when optimal fidelity is achieved. This is because when deriving the generative form of the problem, the Golden-Thompson approximation is used which introduces a error term in the loss function. Furthermore, the regulariser in equation C.5 in [1] is relaxed to improve training efficiency which induces a further error term. This is why some of the plots shown in the results section below have Wasserstein loss $\approx$ 0 when convergence is achieved.

It is important to note that the circuit architecture of the PQC used to generate $|R\rangle$ and $|F\rangle$ is arbitrary and an area of active research [16]. The aim is to add a variety of enough gates to the PQC so that it is rich enough to explore the 2^ndimensional Hilbert space of the problem. However a careful balance between richness and number of gates must be struck as more gates induces more errors which are prevalent in the current NISQ era [17]. Careful thought must also be given to the native gate set of the device if an experimental implementation is to be performed. The authors of [1] use parameterised X, Y and Z rotation gates along with a ladder of XX, YY and ZZ gates. I have also included CNOT gates to this mix but one can use other heuristics which have been proven to work for similar problems.

3. Implementing QWGANs in $t\,|ket\rangle$

The tket framework is a software platform for the development and execution of gate-level quantum computation, providing state-of-the-art performance in circuit compilation [18]. The toolset is designed to aid platform-agnostic software and extract the most out of the available NISQ devices of today.

For machine learning problems, tket can be used along with Scipy optimisers, some of which accept bounds as constraints of the problem [19]. Unbounded optimisation can also be performed as the gate rotation angles are just multiples of 2π. However, if the output weights after each iteration keep on increasing without sign of convergence, this hints at the problem of the loss gliding away from the minimum of the loss landscape.

The choice of optimiser employed was Powell which utilises bi-directional search along each search vector in turn using techniques such as Golden-section [20] or Brent [21]. Hence the function to be minimised need not be differentiable as no derivatives are taken. The reader may recall the enforced criteria, in Section 1, for D and G to be differentiable functions. This condition still holds here however, there is no enforced criteria to employ a gradient based optimiser (Note: Scipy v $<$ 1.4.0 does not allow bounds for Powell).

In the QWGAN framework, the structure of the generator is a PQC whereas the discriminator performs expectation value measurements. Hence, there is no reason for the choice of optimiser and hyperparameters to be the same for both models. One must be guided by previous experience, intuition and examine the outputs after each step of training. One can also try all possible combinations of optimisers and hyperparamaters but the search space of possible solutions grows very quickly.

A particular result I noticed in various runs is where the optimiser gets stuck between two loss values. This 'ringing' behaviour was avoided by tuning the optimiser until the loss values were shoved out of their saddle node behaviour and varied in a more fluid manner. Intuitively, this could also have been solved by utilising a momentum based optimiser which would force the converging values to get out of the rut of being stuck at a saddle point. This view is further supported by the authors in [1] who utilise a self developed, momentum based, optimiser for their results. However, an important result quoted in [6] is that the WGAN training becomes unstable at times when momentum based optimisers such as Adam are used on the discriminator or when one uses high learning rates. These contradicting views are left open for further research at this stage.

3.1. Other implementation methods

A frivolous attempt to implement QWGANs in PennyLane was made whilst facing frustrations with Scipy optimisers. PennyLane is a cross-platform Python library for differentiable programming of quantum computers [22]. It allows access to quantum devices and simulators from key industry players and provides functionality to construct hybrid quantum models connected to classical machine learning (ML) libraries.

PennyLane operates on the concept of a quantum node (QNode) which wraps the standard definition of a quantum circuit along with the device it is executed on into one function. The QNode is designed to be differentiable and allows one to then utilise ML libraries for quantum machine learning problems.

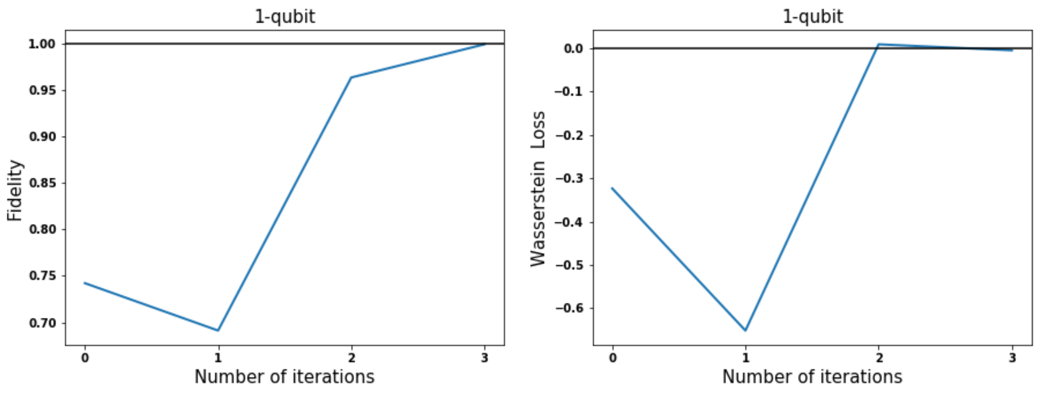
TensorFlow (TF) [23] and Pytorch [24] extend the standard numpy definition of arrays to differentiable tensors. If either is utilised, one must then cast all variables and constants as tensors. Standard numpy commands must also be substituted with their appropriate equivalents. However, one can avoid this hassle by noting that a wrapped version of numpy is provided by PennyLane which, when imported, automatically converts all numpy commands to tensors on which gradients can be evaluated.

Although gradient calculations using PennyLane with TF were faster than using Scipy optimisers, I did not observe loss function convergence. The explosion in loss may be due to QWGAN training becoming unstable when a momentum based optimiser was used as stated in [6]. However my attempts to remedy this were menial and should not be taken as a final conclusion.

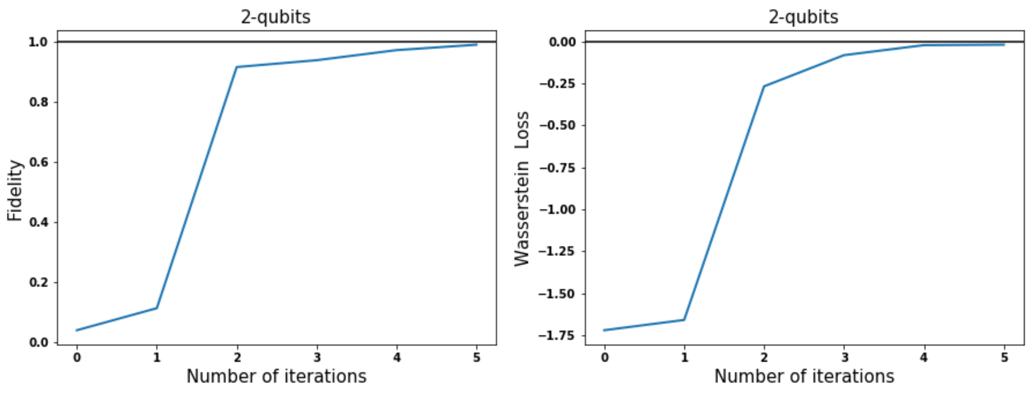
Some time was also spent to experiment with TensorFlow Quantum [25] which is a quantum machine learning library combining the power of TF with Cirq. A final conclusion of its functionality and usefulness cannot be provided by me at this stage as I myself am a novice with TF.

4. Results

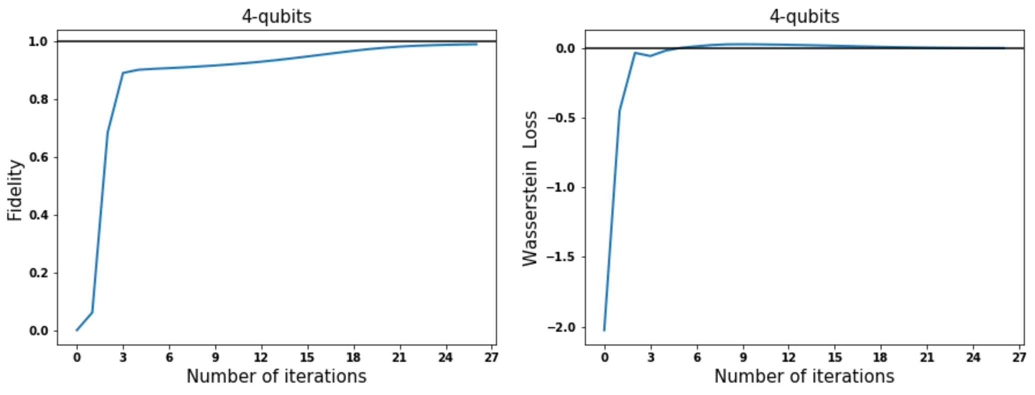
Below I present results for learning 1, 2, 4 and 8 qubit pure states. Each circuit was initialised with a random $|R\rangle$ and $|F\rangle$ and the parameters were updated in each iteration to tune the generator to achieve minimum Wasserstein loss which in turn corresponds to optimal fidelity ($|\langle F|R\rangle|^2$).



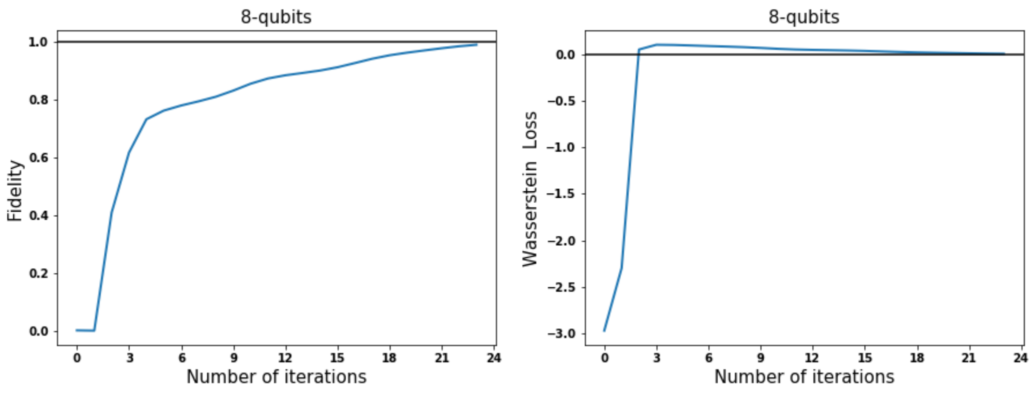
(a)



(b)



(c)



(d)

Figure 3: Numerical results for QWGAN performance.

5. Analysis

A striking feature from the results presented above is the nonlinear behaviour of the Wasserstein loss as a function of fidelity. Figure 3d in particular makes this evident where the Wasserstein loss becomes positive from iteration 3 onwards and then settles slowly to zero. Moreover, there is no evident connection between the relative increase in fidelity and its corresponding variation in loss. The answer to this lies in understanding the mathematics behind the quantisation of the Wasserstein distance.

The approximations made whilst calculating the regulariser term, which is explained in Section 2 above, means that at optimal fidelity the Wasserstein loss ≈ 0. This behaviour is noticeable in Figure 3a and 3b as the loss settles to a value just below zero. In practice, the effect these approximations yield are relatively minuscule variations and the positive results they produce are obvious to decipher.

The negative nature of the metric requires further investigation. Formally, a metric between two distributions is defined to be positive semi-definite, however, it is clear that the loss approaches zero from below. The answer to this lies in the transformation of the integral formulation of the metric to the quantum case and the derivation which precedes this result. An attempt to understand the mathematical derivation in [1] must be conducted.

It is important to bear in mind that the runtime for statevector simulations for the 1-qubit case was around 30 seconds whereas the 8-qubit QWGAN converged in 2 hours. Hence careful thought must be given to the application of the algorithm. If the QWGAN is employed for circuit approximation in the current NISQ era, then the trade-off between training time and reduction in gate count is one which should be grasped. However, since the premise of quantum computing is to provide a computational speedup, the time scaling with qubit count for an actual experiment will be much better.

6. Conclusions & Future Work

An immediate extension to this work is to utilise the QWGAN in quantum chemistry and condensed matter theory calculations. Recent work by Runger et. al. [26] demonstrates implementation of dynamical mean field theory calculations on a quantum computer which suffers from noisy state preparation. This could be mitigated by QWGAN which could prepare the required state with less overall gate count and hence better fidelity. This application was also proven by [1] where numerical calculations to produce a 6-qubit state using 11900 gates was approximated with 52 gates.

An experimental implementation of QGANs in a superconducting quantum circuit was made by Hu et. al. [27] and more recently by Huang et. al. [28]. These can be studied in detail and adopted to implement the QWGAN. This would test its performance on a noisy device and provide conclusions to its practical uses in the current NISQ era. This would be an extension to [1] in which they perform simulations with Gaussian sampling noise with varying standard deviations and reproduce positive results to generate a 4-qubit states with optimal fidelity.

Although loss metrics in GAN training have been studied extensively [6] with the Wasserstein metric triumphing, this needs to be confirmed in the QGAN case to hail the studied QWGAN as victorious. Only then can the community proceed with delving deeper into solving more sophisticated quantum machine learning problems using the Wasserstein metric as a measure of loss.

The presented results were produced by bounding the discriminator weights between $0-2$, however, there is no such enforced requirement for this constraint. The success of these results with this loosely hanging constraint needs to be solidified.

A frequently occurring question in quantum machine learning is the avoidance of barren plateaus where the gradient along any direction of the loss function is ≈ 0. Mclean et. al. [29] show that for a wide class of PQCs, the probability that the gradient along any direction is non-zero is exponentially small as a function of the number of qubits. Thus random parameter initialisation may not be wise for variational quantum algorithms. An initialisation strategy to tackle the barren plateaus problem was given by Grant et. al. [30] which involves initialising the PQC with random values in the initial layers followed by shallow layers which each evaluate to the identity. This limits the effective depth of the circuit used to calculate the first parameter update and hence ensures that the training proceeds without being stuck in a plateau right at the start. Notebook examples to demonstrate this are given by TensorFlow Quantum [31] and PennyLane [32].

Possible solutions to escape barren plateaus has also been investigated by Huembeli and Dauphin [33]. In this work, the authors introduce a method to compute the Hessian of the loss function of a variational quantum circuit and to use it to escape the flat regions of the loss landscape. This result is particularly important for certain Scipy optimisers which use the Hessian for minimisation.

A relevant result by Sweke et. al. [34] to keep in mind in the context of hybrid quantum-classical optimisation, in which the loss function and its gradient can be expressed as functions of expectation values is that exact gradient descent is not possible. All optimisation methods which require estimations of expectation values from a finite number of measurements should hence be considered within the framework of stochastic gradient descent. This conclusion is crucial to analyse in detail if one is to proceed in this realm of research.

7. Acknowledgements

I would like to acknowledge the constant guidance and support given to me by Dr Steven Herbert and Seyon Sivarajah. Your overarching knowledge of machine learning and quantum computing combined with your experience in the field has guided me every step of the way. And lastly to the wider CQC team for dealing with all other technical questions and making this internship an enjoyable experience.

References

[1] Shouvanik Chakrabarti, Huang Yiming, Tongyang Li, Soheil Feizi, and Xiaodi Wu. Quantum wasserstein generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 6781–6792, 2019.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. pages 2672–2680, 2014.

[3] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[4] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models, 2017.

[5] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[7] Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*, 2016.

[8] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318, 2017.

[9] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *Physical review letters*, 121(4):040502, 2018.

[10] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1), Jul 2018.

[11] PennyLane. Quantum GANs. https://pennylane.ai/qml/demos/, 2020.

[12] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):1–9, 2019.

[13] Jinfeng Zeng, Yufeng Wu, Jin-Guo Liu, Lei Wang, and Jiangping Hu. Learning and inference on generative adversarial quantum circuits. *Physical Review A*, 99(5):052306, 2019.

[14] Christian Clason, Dirk A Lorenz, Hinrich Mahler, and Benedikt Wirth. Entropic regularization of continuous optimal transport problems. *Journal of Mathematical Analysis and Applications*, page 124432, 2020.

[15] Zohim Chandani. QWGANs in tket . https://github.com/zohimchandani/qWGANs-, 2020.

[16] Tyson Jones and Simon C Benjamin. Quantum compilation and circuit optimisation via energy dissipation, 2018.

[17] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[18] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. t|ket: A retargetable compiler for nisq devices. *Quantum Science and Technology*, Apr 2020.

[19] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[20] Yen-Ching Chang. N-dimension golden section search: Its variants and limitations. In *2009 2nd International Conference on Biomedical Engineering and Informatics*, pages 1–6. IEEE, 2009.

[21] Gautam Wilkins and Ming Gu. A modified brent's method for finding zeros of functions. *Numerische Mathematik*, 123(1):177–188, 2013.

[22] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Carsten Blank, Keri McKiernan, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

[23] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[25] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J Martinez, Jae Hyeon Yoo, Sergei V Isakov, Philip Massey, Murphy Yuezhen Niu, Ramin Halavati, Evan Peters, et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

[26] I. Rungger, N. Fitzpatrick, H. Chen, C. H. Alderete, H. Apel, A. Cowtan, A. Patterson, D. Munoz Ramo, Y. Zhu, N. H. Nguyen, E. Grant, S. Chretien, L. Wossnig, N. M. Linke, and R. Duncan. Dynamical mean field theory algorithm and experiment on quantum computers, 2019.

[27] Ling Hu, Shu-Hao Wu, Weizhou Cai, Yuwei Ma, Xianghao Mu, Yuan Xu, Haiyan Wang, Yipu Song, Dong-Ling Deng, Chang-Ling Zou, et al. Quantum generative adversarial learning in a superconducting quantum circuit. *Science advances*, 5(1):eaav2761, 2019.

[28] Kaixuan Huang, Zheng-An Wang, Chao Song, Kai Xu, Hekang Li, Zhen Wang, Qiujiang Guo, Zixuan Song, Zhi-Bo Liu, Dongning Zheng, Dong-Ling Deng, H. Wang, Jian-Guo Tian, and Heng Fan. Realizing a quantum generative adversarial network using a programmable superconducting processor, 2020.

[29] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1), Nov 2018.

[30] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[31] TensorFlow Quantum. Barren Plateaus. https://www.tensorflow.org/quantum/tutorials/barren$_{p}lateaus1_{s}ummary$, 2020.

[32] PennyLane. Barren plateaus in quantum neural networks. https://pennylane.ai/qml/demos/tutorial$_{b}arren_{p}lateaus.html$, 2020.

[33] Patrick Huembeli and Alexandre Dauphin. Characterizing the loss landscape of variational quantum circuits. *arXiv preprint arXiv:2008.02785*, 2020.

[34] Ryan Sweke, Frederik Wilde, Johannes Jakob Meyer, Maria Schuld, Paul K Fährmann, Barthélémy Meynard-Piganeau, and Jens Eisert. Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum*, 4:314, 2020.