QGAN Paper Review

Content Summary

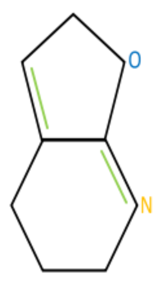
The Problem:

- The problem is to search the space of molecules to find drug candidates for diseases: molecules that obey chemical and physical properties and bind to the receptor of a target disease. This process is very difficult, usually taking several years and billions of dollars in effort. To generate such molecules we can utilize GANs, however...
- Molecules exist in a space of the **order of magnitude** 10^{60} **different possibilities** making it very difficult for classical machine learning techniques to search this space.
  Proposed Solution:
- The paper proposes a QGAN-HG, quantum GAN where the generator is part quantum, combining aspects of quantum advantage and classical machine learnings ability to model.

Modeling Task

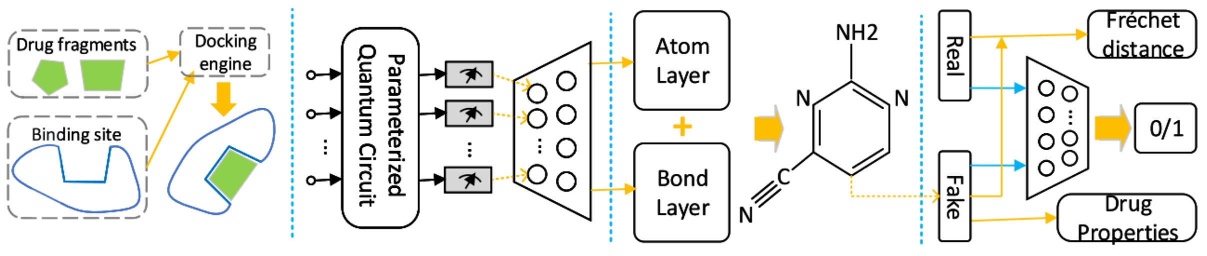
Molecule Representation:

- Atoms are represented by a vector and bonds are represented by a matrix (square matrix of atom dimension).
  - Each entry refers to some properties of the atom or bond making it equivalently a 9x5 atom vector or 9x9x5 bond matrix where each 5 element long entry is some one hot description of a property of the atom or bond.



$$B = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 2 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{bmatrix}$$

$$A = [1 \quad 3 \quad 1 \quad 2 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]$$



(a) (b) (c) (d)

Model Structure:

1. The generator is composed of a parameterized quantum circuit followed by a classical neural network. It generates a candidate drug composed of an atom vector and a bonds matrix which represents the entire molecule.
   1. The quantum part generates a distribution of descriptions for a hypothetical molecule and the classical part takes each feature probability and realizes that into its atom and bond information.
2. The discriminator learns the difference between actual molecules and generated molecules, and the generator learns to fool the discriminator hence more accurately approximating the distribution of real molecules.

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

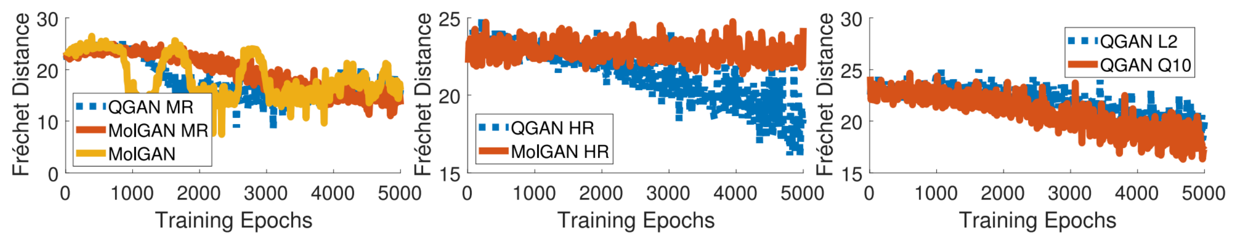
The paper optimizes the cross-entropy (typical for GANs), the discriminator tries to minimize the surprise of data being real or fake, while the generator tries to maximize the surprise of the discriminator to the result.

Metrics:

1. Fréchet Distance (FD): Measures divergence between real and generated molecules.
2. Valid Score: molecule validity.

3. Drug specific metrics: drug-likeness (QED), solubility (logP), and synthesizability (SA)

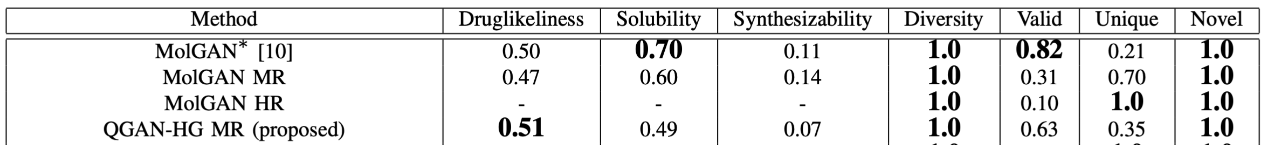
Results Summary



"Results show that classical GAN with 85.07% reduced parameters cannot properly learn molecule distribution, however our QGAN-HG with only 15 extra quantum gate parameters can learn molecular task at par with original GAN"

- Good results reached with QGAN-HG and patched variation: 128 batch size FD mean is (12.3342, 0.7057)
    - "Generated molecules are considered realistic enough once the preset FD cutoff point of 12.5 is reached during the learning process."
    - Results for MolGAN and MolGAN (MR) FD Score not shown, so we can't compare them

Statement cannot be completely verified, but the original MolGAN does significantly outperform all variants of the QGAN-HG.



Method	Druglikeliness	Solubility	Synthesizability	Diversity	Valid	Unique	Novel
MolGAN* [10]	0.50	**0.70**	0.11	**1.0**	**0.82**	0.21	**1.0**
MolGAN MR	0.47	0.60	0.14	**1.0**	0.31	0.70	**1.0**
MolGAN HR	-	-	-	**1.0**	0.10	**1.0**	**1.0**
QGAN-HG MR (proposed)	**0.51**	0.49	0.07	**1.0**	0.63	0.35	**1.0**

MolGAN shown here is retrained (not original)

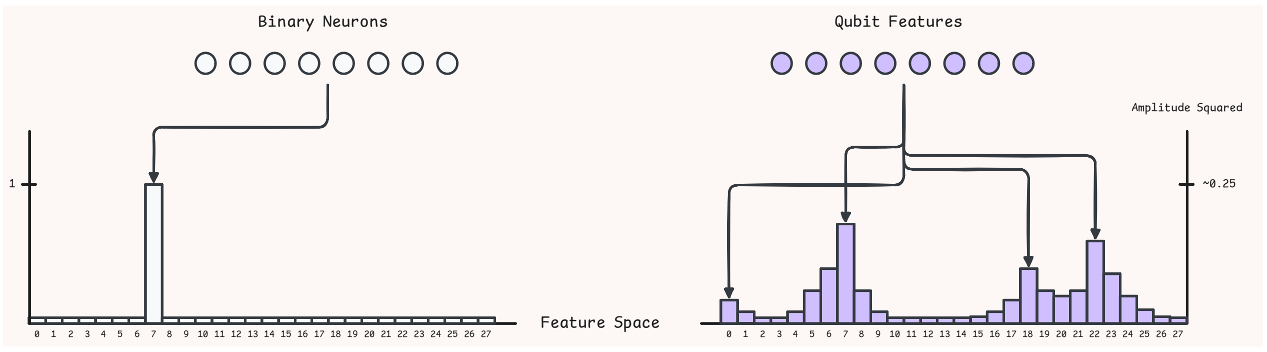
The QGAN-HG seems to outperforms the MolGAN on a few metrics while the MolGAN also outperforms the QGAN-HG on others.

I analyze these results more in depth further down.

Quantum Parallelism

The paper claims that using a quantum circuit in the generator reduces the amount of parameters needed to get good performance significantly. Whats the theory behind this reduction?

Qubit Superpositions



The quantum circuit part of the GAN works with qubit features. Qubits can be in a superposition of 0 or 1 with some phase as compared to normal bits which either encode a 0 or 1.

When we compose a feature with bits using binary 0s and 1s they form a description/feature vector. If we have n bits we get 2^n possible descriptions, and each binary vector can only represent 1 description at a time of course. With qubits however, every description has some amplitude and phase (Look at image above).

We don't just consider one state but a superposition of all states, this representation contains exponentially more information with the same number of qubits as bits.

$$\text{bit} = b_0 2^0 + b_1 2^1 + b_2 2^2 + b_3 2^3 + b_4 2^4 + b_5 2^5 + b_6 2^6 + b_7 2^7 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

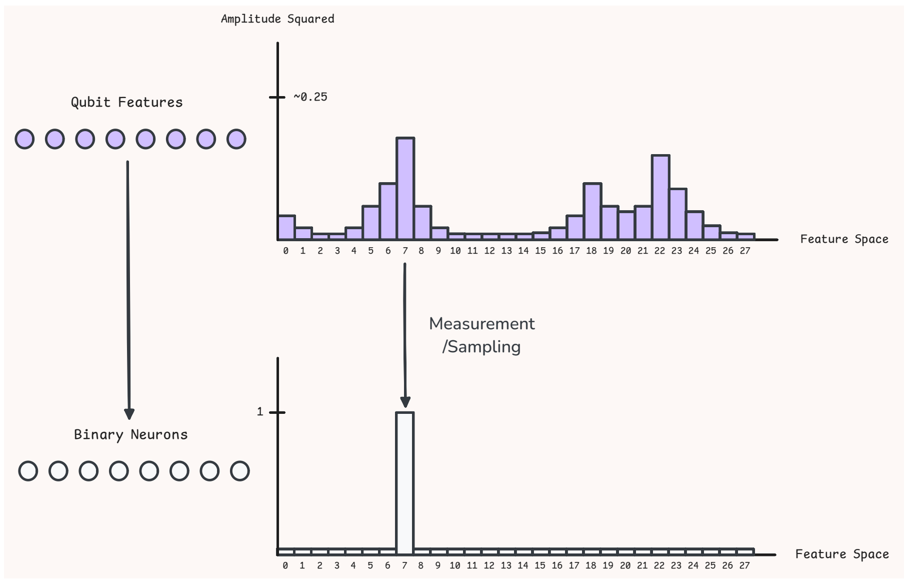
You can represent this bit vector as a single number but I write it as a vector here to demonstrate that each entry corresponds with a property.

$$\text{qubit} = |\psi\rangle = r_0 e^{i\phi_0}|0\rangle + r_1 e^{i\phi_1}|1\rangle + r_2 e^{i\phi_2}|2\rangle + \cdots + r_{255} e^{i\phi_{255}}|255\rangle = \begin{bmatrix} \frac{1}{16} \\ -\frac{1}{16} \\ \frac{i}{16} \\ -\frac{i}{16} \\ \vdots \\ \frac{1}{16} \\ -\frac{1}{16} \\ \frac{i}{16} \\ -\frac{i}{16} \end{bmatrix}$$

Qubit Measurement

There is one major problem with this, its that we can only read it through an interaction/operation called measurement that reduces our 2^n dimensional vector to an n dimensional vector, from a qubit feature to a binary feature.

- A dramatic reduction in information, the worst being that in order to measure again we need to go through the entire process needed to create the qubit state again since we can't clone qubit states.



My interpretation of this is that the final qubit vector state that we intend to observe is a probability distribution over the space of descriptions. And measurement is sampling from this distribution. This means that meaningful final states will have a few likely descriptions (low entropy, but this isn't enforced at all).

Quantum Entanglement

The big computational advantage comes from the parallelism of entangling qubits. Whatever operation we do on the qubits is an operation on all features in the superposition.

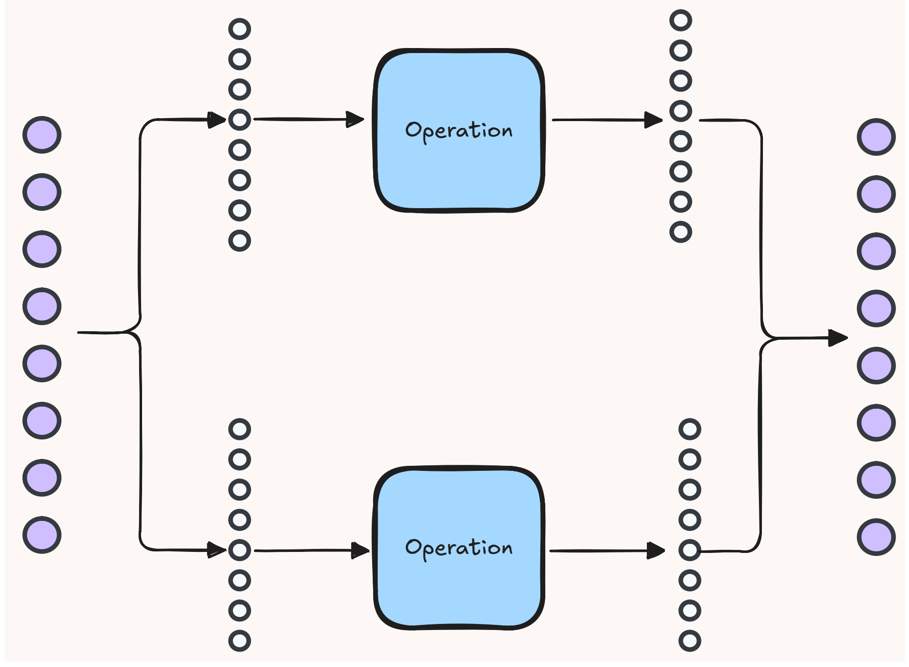
For example if we have the qubit state of:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$$

Then applying a CNOT gate:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

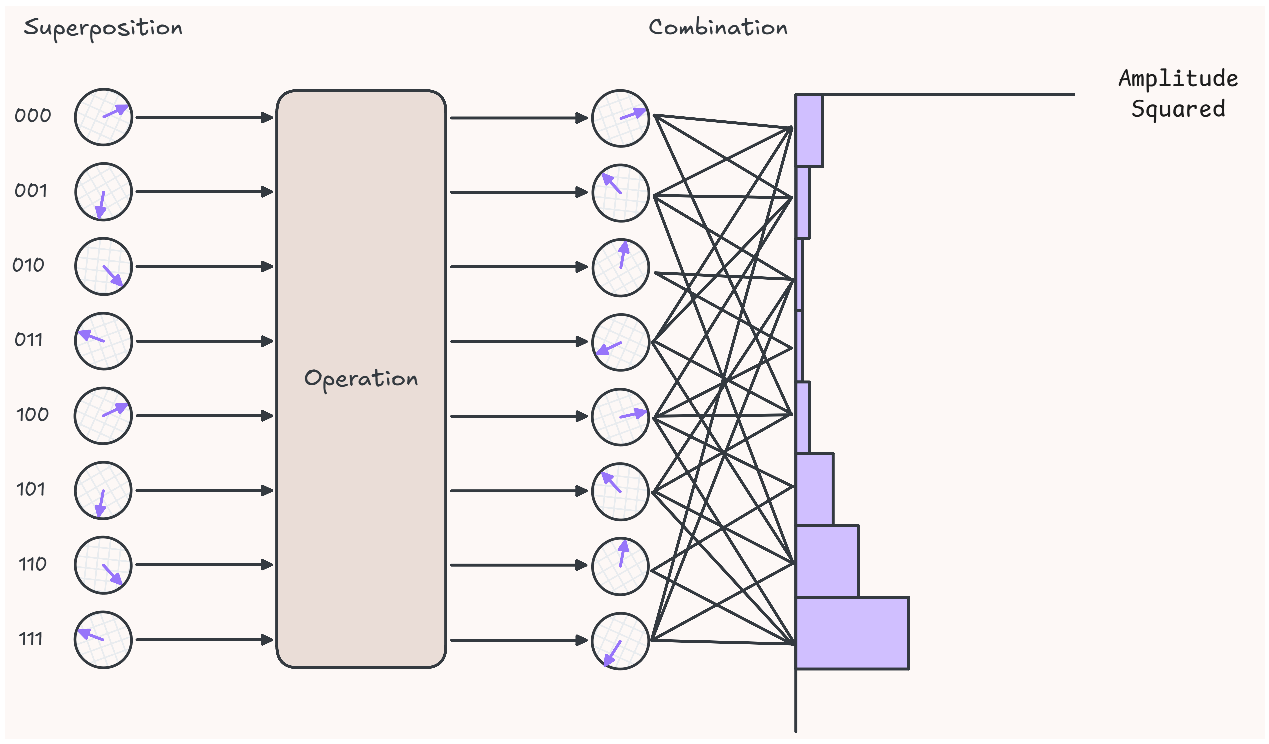
Its simple but powerful, we manipulated both states at once even though only did one operation.



Now we can use this power by going into a superposition of many descriptions and doing some operation on all of them at once.

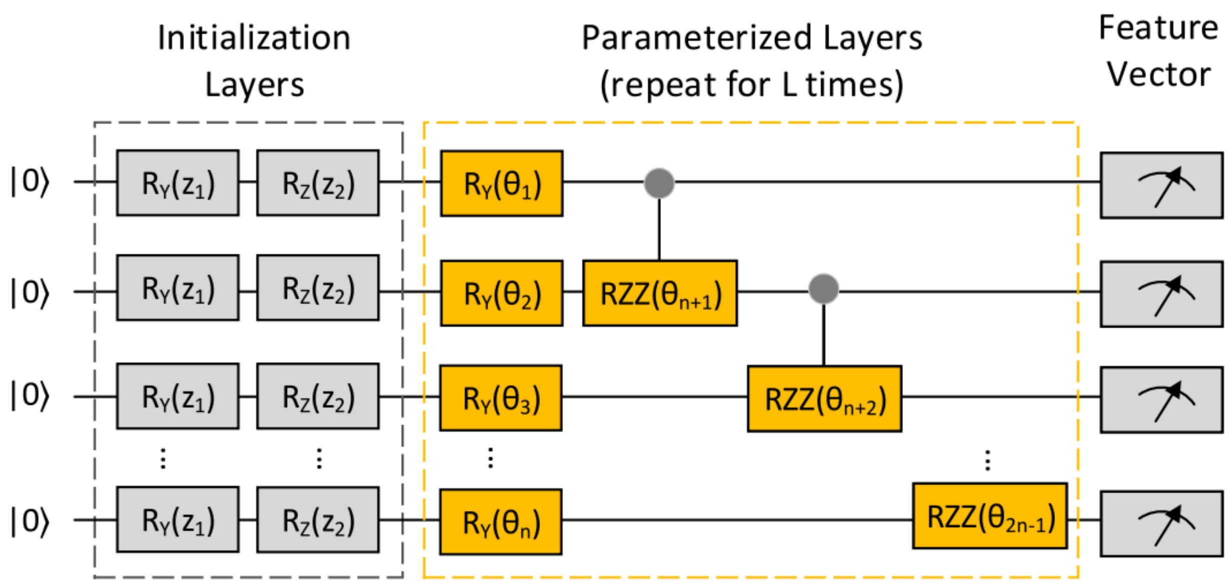
The question from here is how do we use this information, we don't want to just sample from the distribution here since that would be the same as if we just randomly did one of the operations.

Instead we do some sort of combining operation. Then we have actually gained some knowledge from every path.



The next question is how should we implement this idea.

Variational Quantum Circuit



In the paper's generator we start with the quantum circuit. Everything else is a minimalistic MolGAN.

Sampling

This paper generates random samples by generating a random superposition instead of just sampling from gaussian noise. (the details here aren't super important)

- We randomly select $z_1 = \sin^{-1}(X)$, X is a random variable for a uniform distribution from -1 to 1. (Similarly for z_2)
- Then we compute $R_Z(z_2)R_Y(z_1)|0\rangle$ for each qubit.

$$|\psi\rangle = (\alpha e^{-i\phi/2}|0\rangle + \beta e^{i\phi/2}|1\rangle) \otimes (\alpha e^{-i\phi/2}|0\rangle + \beta e^{i\phi/2}|1\rangle) \otimes (\alpha e^{-i\phi/2}|0\rangle + \beta e^{i\phi/2}|1\rangle)$$

Parameterized Layers

In machine learning we try to learn a matrix, a transformation of the feature vector in order to learn something or get conclusions. Here we can do the same thing but using qubit features, learning a unitary matrix instead of a weight matrix.

A complex eigenvalue transformation can be thought of in terms of a rotation and a scaling operation:

$$U(\theta) = R(\theta_1)D(\theta_2)$$

Since we are including complex numbers and only want unitary operations we can decompose this into:

$$U(\theta) = R_Y(\theta_1)D_\phi(\theta_2)$$

Where D is a diagonal matrix of complex unit vectors (phase shift factors) and R is a rotation matrix. This idea is very similar to the idea proposed before where each description undergoes an independent operation and then they are combined.

The paper chooses to use R_y gates to perform the rotation/combination part.

Rotation/Combination/Weighted Sum

Let's understand the $R_Y(\theta_k)$ Gates responsible for information combination. For a single qubit, the rotation gate performs a weighted sum to compute the the new state:

$$R_y(\theta_k) = \begin{bmatrix} \cos(\frac{1}{2}\theta_k) & -\sin(\frac{1}{2}\theta_k) \\ \sin(\frac{1}{2}\theta_k) & \cos(\frac{1}{2}\theta_k) \end{bmatrix}$$

$$\psi_k = (\cos / \sin)(\frac{1}{2}\theta_k)\psi_1 \pm (\cos / \sin)(\frac{1}{2}\theta_k)\psi_2$$

theta corresponds to θ on the bloch sphere (why the factor of 1/2 is there), and the rotation gate signifies a rotation of the amplitude around the 0, 1 states

For many qubits potentially entangled, each with potentially different amplitude rotation angles:

$$\psi_k = (\cos(\frac{1}{2}\theta_1)\sin(\frac{1}{2}\theta_2)\cos(\frac{1}{2}\theta_3)\cdots)\psi_1 \pm \cdots$$

For n qubits the weight is a product of n cosine and sine functions. The weighted sum will contain 2^n elements (goes through all 2^n combinations of sin and cos), and there are 2^n weighted sums being computed corresponding with the $2^n \times 2^n$ tensored together rotation matrices.

There is one more important distinction between this weighted sum and the classical neural network sum, since we are using complex numbers this is really a sum of a bunch of 2D vectors in the complex plane so depending on the phase results can interfere or construct in hopefully meaningful ways. (instead of just interfering constructively and deconstructively on the real number line)

RZZ Operation

Now to the operation that controls what we actually do in the big superposition. For this paper we do an application of the RZZ gate for every adjacent qubit.

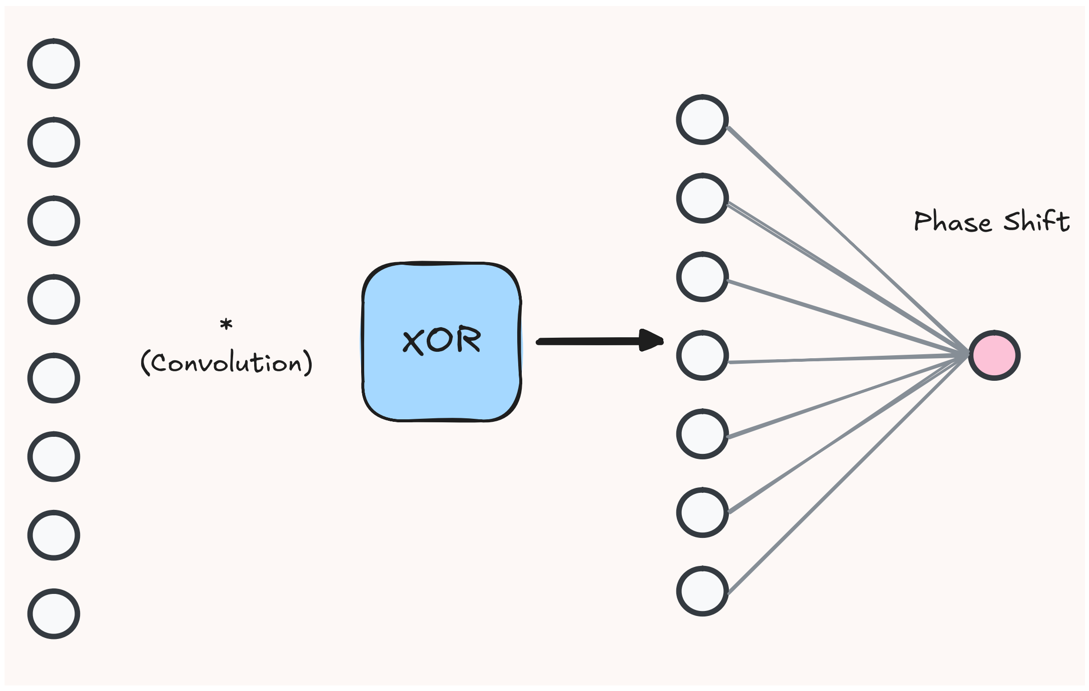
$$RZZ(\theta_k) = \begin{bmatrix} e^{i\frac{1}{2}\theta_k} & 0 & 0 & 0 \\ 0 & e^{-i\frac{1}{2}\theta_k} & 0 & 0 \\ 0 & 0 & e^{-i\frac{1}{2}\theta_k} & 0 \\ 0 & 0 & 0 & e^{i\frac{1}{2}\theta_k} \end{bmatrix}$$

This will 2 important things, first it will entangle each qubit together making the quantum state really contain 2^n points of information. Second it represents the diagonal matrix which applies phase shifts to each description in quantum state.

What this does is apply a XOR like gate to each adjacent bit pair in each element of the superposition. For each XOR that results in 1, we subtract $\theta_k/2$ from the phase, when the XOR results in 0 we add $\theta_k/2$ to the phase. Then we do this same process for each pair corresponding with some θ_k.

This is the same as convoluting the each binary number with XOR gate (where the xor gate results in -1/2 and 1/2 instead of 1 and 0).

$$\Delta\phi = (|n\rangle * \text{XOR}) \cdot \vec{\theta}, \text{n is a binary number in the superposition like 010100.}$$



Mistakes

The RZZ gates, as applied in the paper, do not influence the output in single-layer models due to their positioning right before the measurement step since it only modifies the phase (tested in code). I believe the intention was to have it applied in the reverse order, like the idea behind decomposing this into a phase shift then rotation or parallel operation then combination.

Second, for multiple layers the new layers don't get new parameters. Parameter sharing across layers may undermine the network's representational capacity, contrary to the intent of multi-layer architectures.

The Schrödinger Equation and The Parameter Shift Rule

For neural networks it's very easy to find the derivative of the loss function with respect to a parameter. We simply need to know how the parameter effects its layer's activation, and how its activation effects the loss function and apply the chain rule.

For variational quantum circuits all we need to do is find how the parameter θ_k effects the expected value function (the activation value in this case), and if we know that we can apply the chain rule to get the derivative of the loss function with respect to the parameter.

To do that we first need to understand what the parameter actually means. The quantum state vector obeys the shrodinger equation, in order to change a quantum state we have time evolve it:

$$\frac{d}{dt}\vec{\psi} = \frac{-i}{2}H\vec{\psi}$$

The factor of 1/2 is used to align some ideas together, when t becomes θ laters It will correspond with the bloch sphere angle (and its also convenient later)

Where H is a hermitian matrix we are free to choose. Now in order to introduce quantum gates which do valid operations on quantum states we choose different H matrices.

As an example lets say that we want to use the Y hermitian matrix on the first qubit:

$$\frac{d}{dt}\vec{\psi} = \frac{-i}{2}(Y \otimes I \otimes I \cdots)\vec{\psi}$$

Then we can solve the shrodinger equation and get $\vec{\psi}$ in terms of t. How? We use the same procedure we use for any system of differential equations, finding the eigenvalues and eigenvectors. Let's say for simplicity (and because its an assumption used for the parameter shift rule) that the eigenvalues of the hermitian matrices we choose are ± 1, then (since tensoring multiplies eigenvalues together) the solution to the shrodinger equation is:

$$\vec{\psi}(t) = c_1\vec{\psi}_1 e^{\frac{1}{2}it} + c_2\vec{\psi}_2 e^{-\frac{1}{2}it}$$

The thing that makes the shrodinger equation special compared to other system of differential equations is that we don't have an axis that dominates or disappears (corresponding with eigenvalues with real parts), instead each eigenvector spins in the complex plane retaining a constant length.

Now noticing that this is periodic we can use θ instead of t. Choosing some value of θ gives us a time evolution unitary matrix which updates our quantum state:

$$\vec{\psi}(\theta) = U(\theta)\vec{\psi}_0$$

Now we have a way of applying a parameterized transformation to a quantum state. Notice that if we apply transformation after this or before this particular time evolution we will still get the same form:

$$\vec{\psi}(\theta) = c_1 U(\vec{\psi}_1) e^{\frac{1}{2}i\theta} + c_2 U(\vec{\psi}_2) e^{-\frac{1}{2}i\theta}$$

and so in general if we treat all other parameters as constant and focus on one parameter in the quantum circuit (anywhere in the quantum circuit) the final quantum state can always be expressed in this form:

$$\vec{\psi}(\theta) = c_1\vec{\psi}_1 e^{\frac{1}{2}i\theta} + c_2\vec{\psi}_2 e^{-\frac{1}{2}i\theta}$$

If you want the general form with all parameters you need to go through all combinations of $\pm\frac{1}{2}\theta_k$ summed in the exponent with a corresponding coefficient and eigenvector.

Now how does this final quantum state influence the activation value, the expected value function:

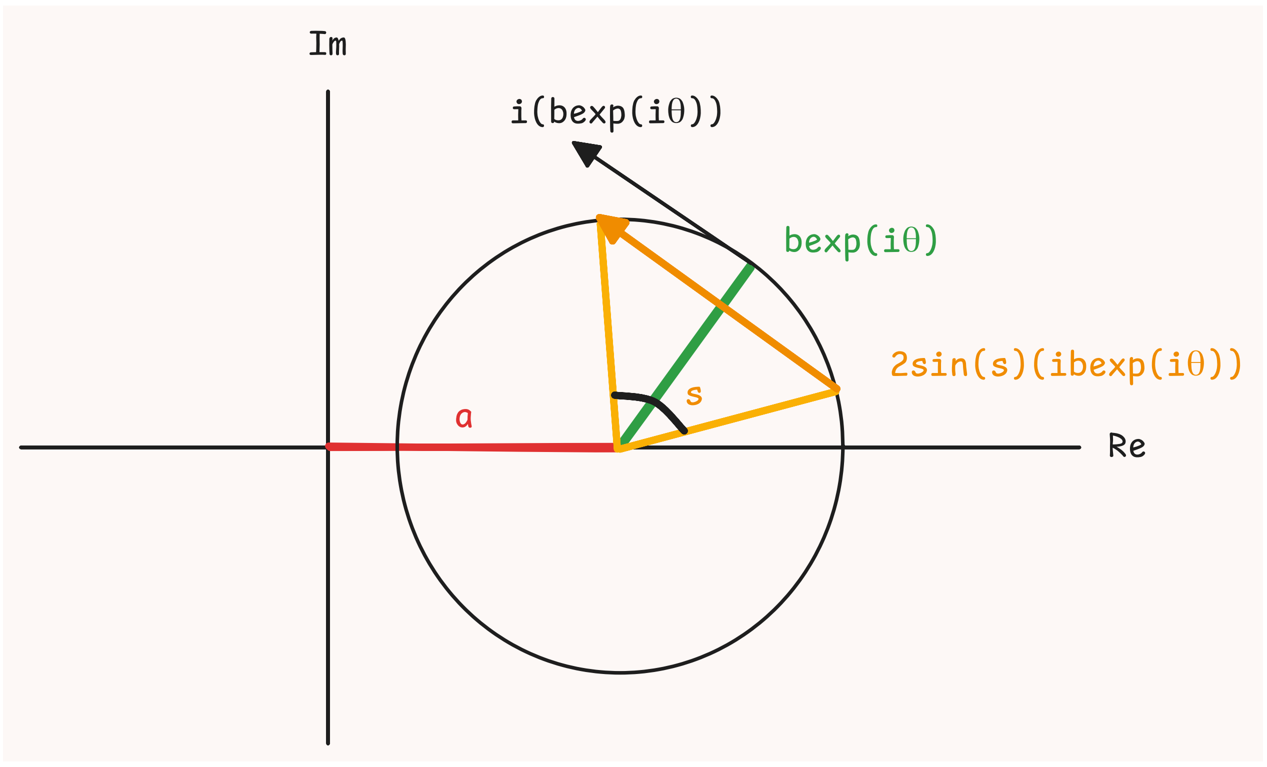
$$f_i(\theta) = \vec{\psi}^\dagger(\theta) B_i \vec{\psi}(\theta)$$

All we need to do is expand this out (skipping an expanding, multiplying out step here):

$$f_i(\theta) = a + \Re(be^{i\theta}) = a + |b|\cos(\theta + \arg(b))$$

The constants a and b depend on x and the other parameters. a is real and b is complex

Now that we have a form for f in terms of θ, can we find the derivative using a single evaluation of θ (like we can with neural networks). The answer is no (unless we know the final quantum state which is only possible in simulation), we need to plug in 2 different θ values in order to find the derivative:



The Parameter Shift Trick

$$f(\theta + s) - f(\theta - s) = 2\sin(s)\frac{df}{d\theta}(\theta)$$

Repeat this trick for each parameter to get the entire gradient.

Extra Thoughts

The final form of the activation function is troubling for me. Sure you can train this network but changing one parameter changes the amount of impact another parameter has (by changing b not a) which is not the case in normal neural networks and I think this makes it much less trainable. I don't know for sure but It seems like the loss landscape would be extremely bumpy making a lot of small pits for the gradient to end up in since changing one parameter effects the scale the other parameters see.

b will be a sum of exponentials in terms of each θ_k excluding the one we are varying, the full list will be of length 2^{n-1} having combinations of excluding and including each exponent.
If you think about it the more parameters are included the parameters with a high magnitude b value will get increasingly disrupted making each parameter less significant in the final result, and whenever a parameter tries to gain more influence the other ones will kick it down. b will be proportional to $1/2^n$ I imagine this is the equivalent of the vanishing gradient problem for vqcs (or maybe my logic here is flawed, Im not sure).

Paper Analysis

The paper claims that we can highly reduce the number of parameters by introducing a quantum circuit while still getting results on par with a model that contains more parameters.

We've seen conceptually how this would happen, VQCs allow us to do massive matrix multiplications that would otherwise be unfeasible to do quickly on classical computers with the caveat that we must downscale the information at the end and only get the expected value of each feature, but this is fine as long as the information is meaningful*.

I'll analyze how valid this claim is by reviewing the results, running the code myself, keeping track of additional information, and making justified modifications.

Paper's Results

Models to Compare:

1. The original MolGan and reduced+retrained variations.
2. The QGAN Medium Reduced (15%, 51,000 parameters) with 8 qubits.

Im choosing to exclude the highly reduced cases since they don't have enough parameters (2% molgan) to actually learn molecule representations so I don't think it makes sense to include it in the comparison since a logical comparison would involve at least one learning actually valid molecules.

Metrics to Analyze:

1. The Valid Score
    1. The original molGAN paper uses this metric and its score of 98.1 (and higher in some variations) to demonstrate the success of the model. This metric is critical to model performance.
2. The drug specific metrics: Druglikeliness, Solubility, Synthesizability
    1. It seems like druglikeliness fluctuates around 0.5 closely, its not very reliable as a metric unless it deviates to around 0.55 or 0.44.
    2. The original MolGAN paper (and also the partially retrained molgan in this paper) have a solubility score of around 0.7 which is the higher end, and on the lower end 0.5 to 0.4.
    3. From this paper it seems like synthesizeability is a good metric to judge how good this model is the original MolGAN has scores from 0.7 to 0.95 (while the partially retrained one in this paper has a score of only 0.11). It's also important to the problem statement and models that score high in this category are what we actually want.
3. FD Score
    1. Introduced in this paper and models with a FD score of 12.5 or below are said to be accurate.

Comparision:
MolGAN (MR) has a valid score of 0.31 while the QGANs proposed are around 0.6, the FD scores are similar and the other metrics fluctuate, for example the the MolGAN (MR) does substantially better on solubility while on synthesizability some models are better while others are worse.

From this it seems like the QGAN is at least better in molecule validity, but I couldn't find the training logs for the MolGAN (MR) to analyze further than this.

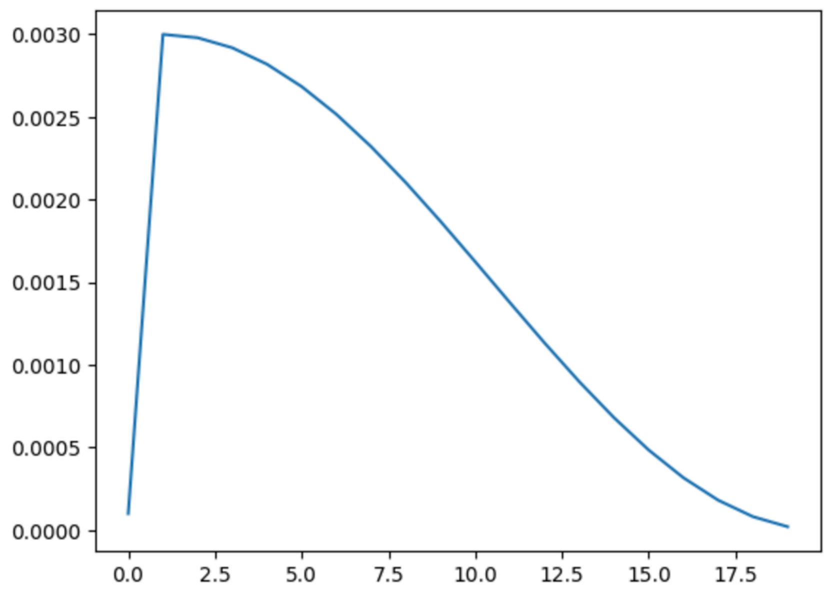
I did an initial rerun using the code they provided and the medium reduced model actually reached the valid score of 0.6. So let's rerun the code multiple times and see if the qgan actually performs better or if it just got lucky (because of how notoriously difficult GANs can be sometimes).

Experiment Reruns

To see how significant the results are, whether the quantum circuit helps substantially, I reran the experiment multiple times tracking extra diagnostics and making modifications to the code as I saw needed:

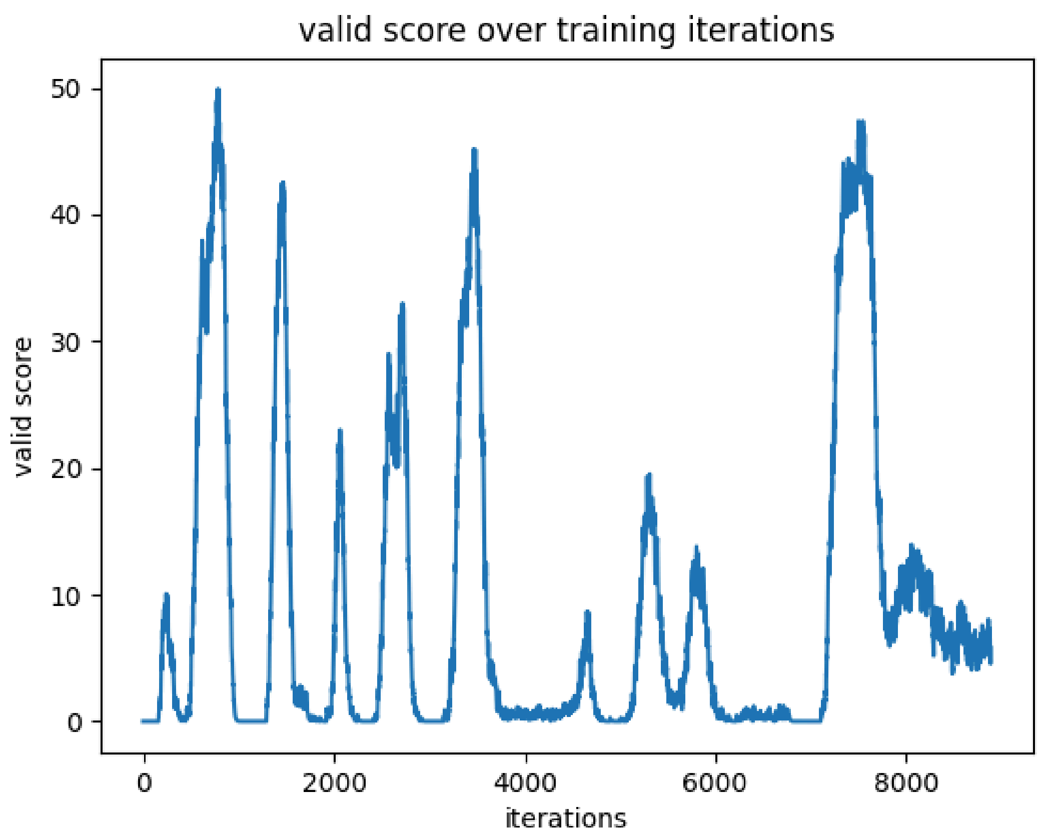
Modifications:

1. Swapping the quantum circuit layer order (since right now the RZZ layer at the end doesn't change the outputs)
2. Including gradient clipping (the discriminators gradient seems to keep on spiking leading to model collapse) with max norm being 0.5.
3. Change learning rate strategy, use linear warmup+cosine annealing. Use a learning rate of 1e-3 like the original molgan paper did (instead of 1e-4 used in this paper), used a separate Adam optimizer for quantum circuit with learning rate 5e-3 (to make the contributions from the quantum circuit more impactful).
4. Different parameter initialization based on the number of quantum circuit parameters, qubit number increased to 10.

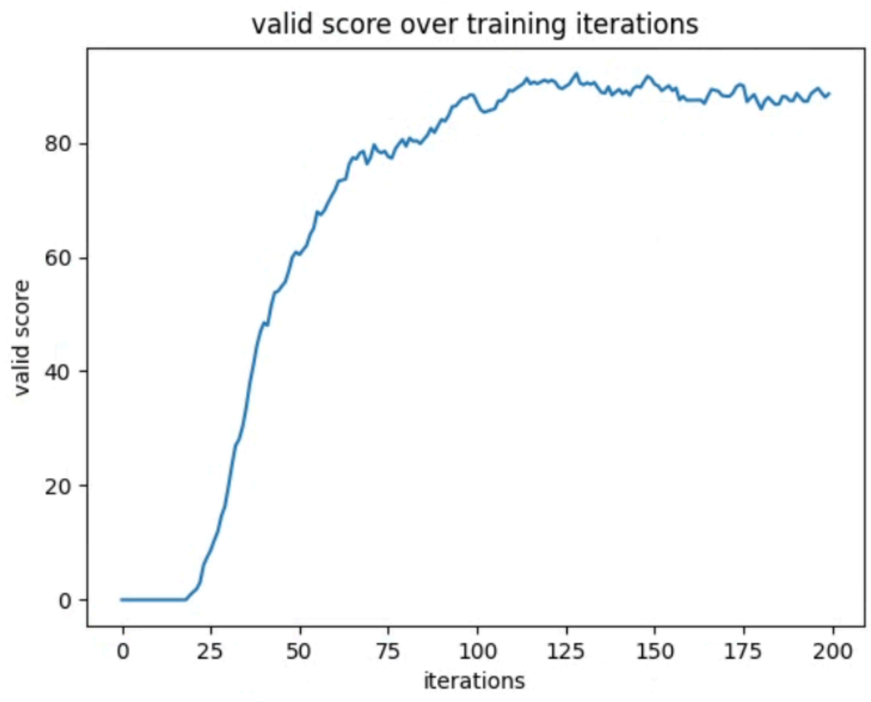


Linear Warmup+Cosine Annealing learning rate schedule

Stability Problem:
A lot of the modifications I got was from testing different configs and tracking gradient magnitudes. There was a problem with the qgan model collapsing extremely often (under the original version being rerunned). It doesn't collapse as often with the changes but every few attempts it does still collapse about halfway through and never recovers.



The problem I believe was mostly from the parameter initialization, they were randomly initialized (0,2pi) before but now I have it so that its variance decreases inversely proportional to the number of parameters.



This is an example of an experiment rerun with the modifications (where the model didn't collapse).

Rerun Results

For this I trained each model, the MolGAN (mr) and the QGAN-HG (mr) 10 qubits, 8 times. The paper uses the model with the least FD Score to compare, I've used the same thing here but used the average of the lowest 10 for more consistency.

Models\Metrics	**FD Score**	**Valid Score**	**Druglikeliness**	**Solubility**	**Synthesizability**
MolGAN (MR)	11.42 ($\pm$ 1.08)	46.71 ($\pm$ 13.18)	0.47 ($\pm$ 0.00)	0.66 ($\pm$ 0.03)	0.12($\pm$ 0.05)
QGAN-HG (MR) 10Q	10.71 ($\pm$ 1.62)	**76.64 ($\pm$ 9.74)**	0.49 ($\pm$ 0.01)	0.67 ($\pm$ 0.05)	0.10 ($\pm$ 0.06)

All the results are close except the valid score which the proposed model performs significantly better on. Although the ranges for the 2 overlap, the FD score is also noticeably better and under 12.5 is said to be an accurate model.

Conclusions

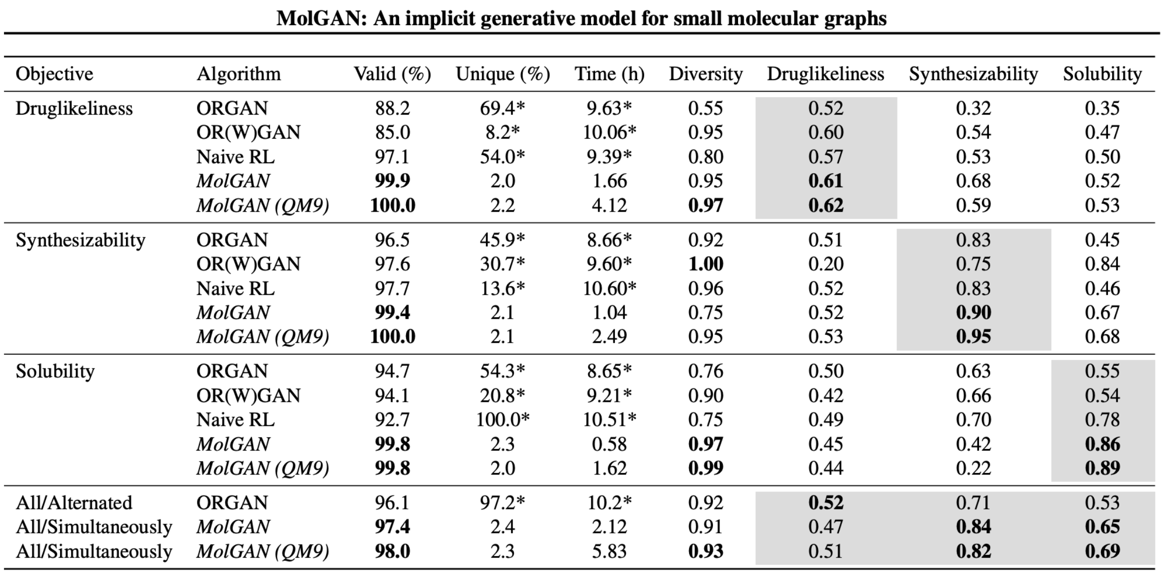
The quantum circuit included in the generator to create a hybrid quantum classical generator increases molecule validity substantially. The model however, also causes more instability, this was mitigated for the most part from some modifications, but model collapse occurs noticeably more often (I also experimented with many learning rates). The question of scalability is left unanswered, it could be that more qubits will not scale well and could even cause more instability.

Let's go over the conclusion of the paper. This is the main statement:

> "Results show that classical GAN with 85.07% reduced parameters cannot properly learn molecule distribution, however our QGAN-HG with only 15 extra quantum gate parameters can learn molecular task at par with original GAN"

This claim is overstated.

1. The MolGAN (MR) can learn the molecule distribution under the qualifier that it has an FD score of less than 12.5 consistently. On valid score the QGAN (MR) 10 qubits does outperform it significantly, however its a stretch to say that the MolGAN (MR) "cannot properly learn" when it achieves very similar results on every other metric.
2. The statement that it can learn on par with the original GAN is not entirely accurate, the metric that sets the QGAN apart from the MR MolGAN is valid score, and in that sense it nearly maxes it out, but clearly there is a limit to that metric, once models get good enough its not the best comparator.
   1. For example, an important metric to compare is synthesizability which ranges significantly higher than the QGAN (max 0.19 vs ~0.8 and higher) and was the big criteria that decides which model we would want to use practically.



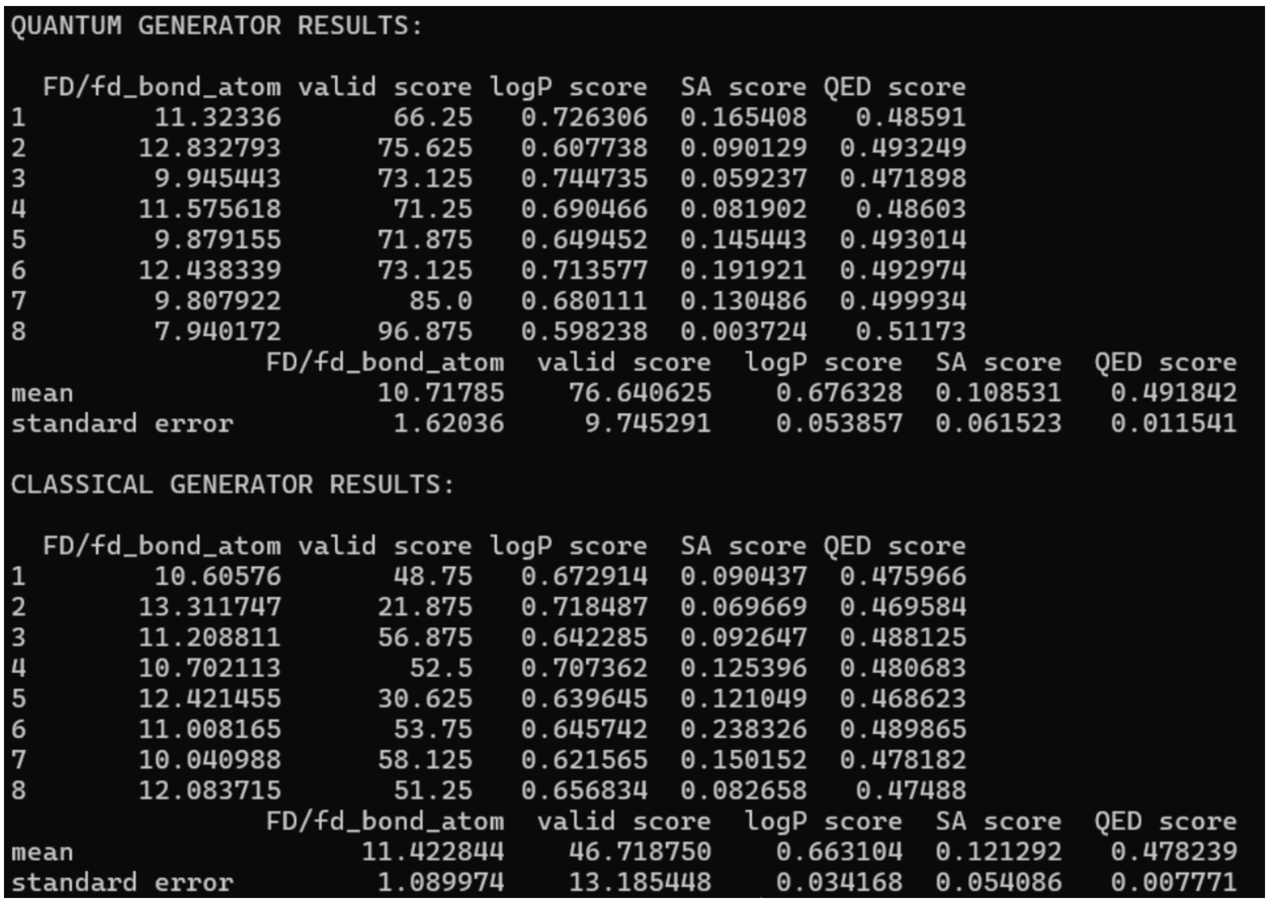
MolGAN: An implicit generative model for small molecular graphs

Objective	Algorithm	Valid (%)	Unique (%)	Time (h)	Diversity	Druglikeliness	Synthesizability	Solubility
Druglikeliness	ORGAN	88.2	69.4*	9.63*	0.55	0.52	0.32	0.35
	OR(W)GAN	85.0	8.2*	10.06*	0.95	0.60	0.54	0.47
	Naive RL	97.1	54.0*	9.39*	0.80	0.57	0.53	0.50
	MolGAN	**99.9**	2.0	1.66	0.95	**0.61**	0.68	0.52
	MolGAN (QM9)	**100.0**	2.2	4.12	**0.97**	**0.62**	0.59	0.53
Synthesizability	ORGAN	96.5	45.9*	8.66*	0.92	0.51	0.83	0.45
	OR(W)GAN	97.6	30.7*	9.60*	**1.00**	0.20	0.75	0.84
	Naive RL	97.7	13.6*	10.60*	0.96	0.52	0.83	0.46
	MolGAN	**99.4**	2.1	1.04	0.75	0.52	**0.90**	0.67
	MolGAN (QM9)	**100.0**	2.1	2.49	0.95	0.53	**0.95**	0.68
Solubility	ORGAN	94.7	54.3*	8.65*	0.76	0.50	0.63	0.55
	OR(W)GAN	94.1	20.8*	9.21*	0.90	0.42	0.66	0.54
	Naive RL	92.7	100.0*	10.51*	0.75	0.49	0.70	0.78
	MolGAN	**99.8**	2.3	0.58	**0.97**	0.45	0.42	**0.86**
	MolGAN (QM9)	**99.8**	2.0	1.62	**0.99**	0.44	0.22	**0.89**
All/Alternated	ORGAN	96.1	97.2*	10.2*	0.92	**0.52**	0.71	0.53
All/Simultaneously	*MolGAN*	**97.4**	2.4	2.12	0.91	0.47	**0.84**	**0.65**
All/Simultaneously	*MolGAN (QM9)*	**98.0**	2.3	5.83	**0.93**	0.51	**0.82**	**0.69**

Table to reference for how the orignal MolGAN performed.

Raw Results

Using lowest 10 FD:



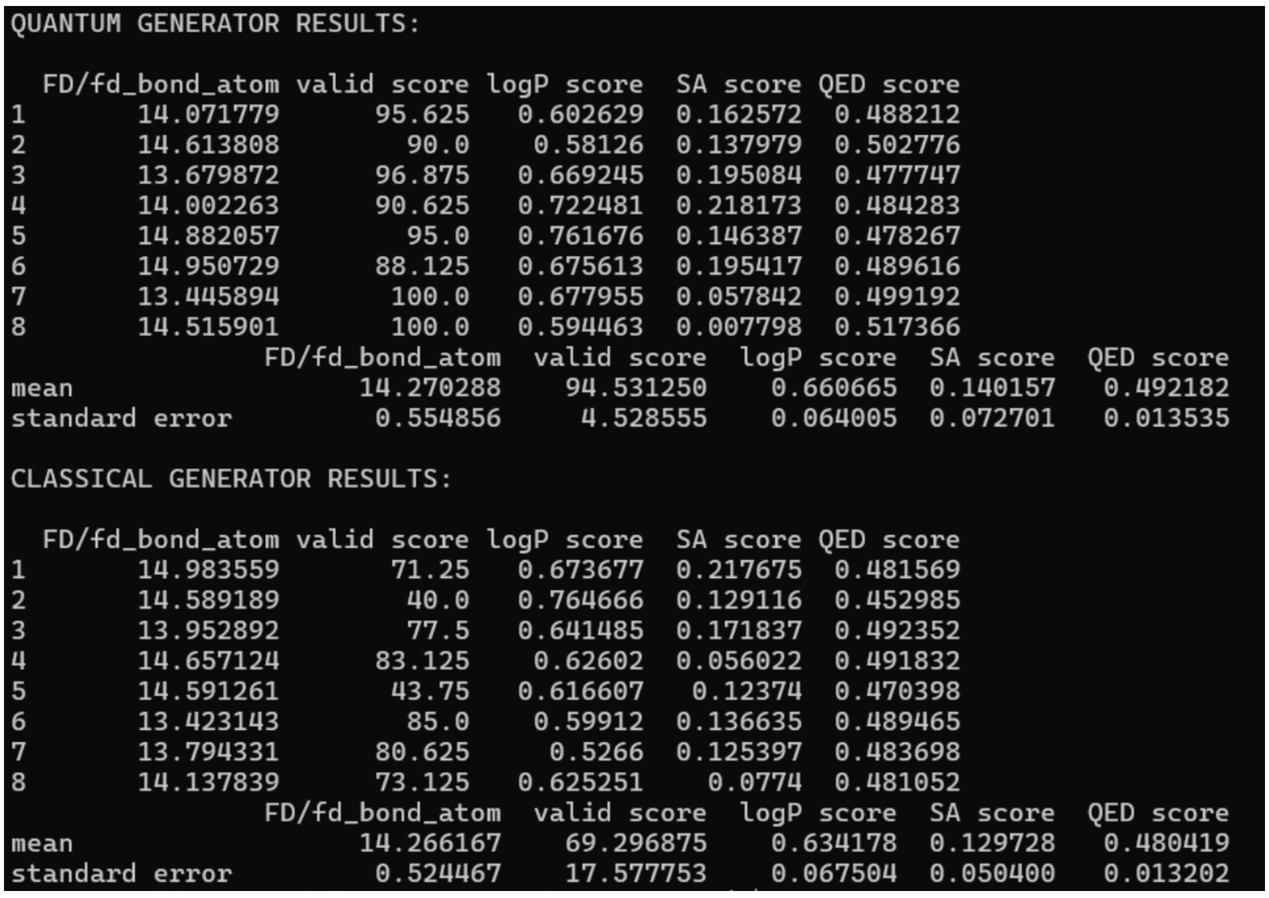
```
QUANTUM GENERATOR RESULTS:

  FD/fd_bond_atom valid score logP score  SA score QED score
1        11.32336       66.25   0.726306  0.165408   0.48591
2       12.832793      75.625   0.607738  0.090129  0.493249
3        9.945443      73.125   0.744735  0.059237  0.471898
4       11.575618       71.25   0.690466  0.081902   0.48603
5        9.879155      71.875   0.649452  0.145443  0.493014
6       12.438339      73.125   0.713577  0.191921  0.492974
7        9.807922        85.0   0.680111  0.130486  0.499934
8        7.940172      96.875   0.598238  0.003724   0.51173
                    FD/fd_bond_atom  valid score  logP score  SA score  QED score
mean                       10.71785    76.640625    0.676328  0.108531   0.491842
standard error              1.62036     9.745291    0.053857  0.061523   0.011541

CLASSICAL GENERATOR RESULTS:

  FD/fd_bond_atom valid score logP score  SA score QED score
1        10.60576       48.75   0.672914  0.090437  0.475966
2       13.311747      21.875   0.718487  0.069669  0.469584
3       11.208811      56.875   0.642285  0.092647  0.488125
4       10.702113        52.5   0.707362  0.125396  0.480683
5       12.421455      30.625   0.639645  0.121049  0.468623
6       11.008165       53.75   0.645742  0.238326  0.489865
7       10.040988      58.125   0.621565  0.150152  0.478182
8       12.083715       51.25   0.656834  0.082658   0.47488
                    FD/fd_bond_atom  valid score  logP score  SA score  QED score
mean                      11.422844    46.718750    0.663104  0.121292   0.478239
standard error             1.089974    13.185448    0.034168  0.054086   0.007771
```

Using highest valid 10:



```
QUANTUM GENERATOR RESULTS:

  FD/fd_bond_atom valid score logP score  SA score QED score
1       14.071779      95.625   0.602629  0.162572  0.488212
2       14.613808        90.0    0.58126  0.137979  0.502776
3       13.679872      96.875   0.669245  0.195084  0.477747
4       14.002263      90.625   0.722481  0.218173  0.484283
5       14.882057        95.0   0.761676  0.146387  0.478267
6       14.950729      88.125   0.675613  0.195417  0.489616
7       13.445894       100.0   0.677955  0.057842  0.499192
8       14.515901       100.0   0.594463  0.007798  0.517366
                    FD/fd_bond_atom  valid score  logP score  SA score  QED score
mean                      14.270288    94.531250    0.660665  0.140157   0.492182
standard error             0.554856     4.528555    0.064005  0.072701   0.013535

CLASSICAL GENERATOR RESULTS:

  FD/fd_bond_atom valid score logP score  SA score QED score
1       14.983559       71.25   0.673677  0.217675  0.481569
2       14.589189        40.0   0.764666  0.129116  0.452985
3       13.952892        77.5   0.641485  0.171837  0.492352
4       14.657124      83.125    0.62602  0.056022  0.491832
5       14.591261       43.75   0.616607   0.12374  0.470398
6       13.423143        85.0    0.59912  0.136635  0.489465
7       13.794331      80.625     0.5266  0.125397  0.483698
8       14.137839      73.125   0.625251    0.0774  0.481052
                    FD/fd_bond_atom  valid score  logP score  SA score  QED score
mean                      14.266167    69.296875    0.634178  0.129728   0.480419
standard error             0.524467    17.577753    0.067504  0.050400   0.013202
```

References

1. https://arxiv.org/pdf/2101.03438 (QGAN-HG Paper)
2. https://arxiv.org/abs/1805.11973 (MolGAN Paper)