Efficient and Optimized Small Organic Molecular Graph Generation Pathway Using a Quantum Generative Adversarial Network

Max Cui
University of Toronto
Toronto, CA
max.mcui@gmail.com

Linda Chang
Aspiring Scholars Directed Research Program
Fremont, USA
lindachang189@gmail.com

Adelina Chau
University of California, Berkeley
Berkeley, USA
chauadelina06@gmail.com

Hasset Mekuria
University of California, Berkeley
Berkeley, USA
ihasset2019@gmail.com

Leena Adwankar
Aspiring Scholars Directed Research Program
Fremont, USA
leena.adwankar@asdrp.org

Sriaditya Pendyala
Aspiring Scholars Directed Research Program
Fremont, USA
quarrylaneschool.sriaditya@gmail.com

Dr. Larry McMahan
Aspiring Scholars Directed Research Program
Fremont, USA
larry.mcmahan@asdrp.org

***Abstract*—Contemporary drug discovery and development processes require billions of dollars and lengthy amounts of time, which is why researchers are utilizing computational chemistry methods like machine learning to speed up molecular synthesis pathways. Generative Adversarial Networks, in particular, have shown immense potential for accelerating the drug discovery pipeline. However, these models consume a significant amount of computational resources, as they must perform adversarial training on expansive datasets to accurately sample the vast chemical space. Aiming for more efficient runtimes and rigorous analyses of high-dimensional molecular data with quantum computing, we previously developed a hybrid quantum-classical generative adversarial network (QGAN) to generate potential small organic drug candidates. Later, inspired by the classical NetGAN, we upgraded QGAN to a hybrid quantum-classical graph generative adversarial network (QNetGAN) which generated graph structures via random walks. This model was more efficient and had a higher accuracy rate (47.0%) than QGAN (2.3%). Most recently, we developed QNetGAN v2, an upgraded version of QNetGAN that features graph convolution. This model was able to generate 273 out of 300 structurally valid molecules that satisfy the octet rule and Lipinski's Rule of Five, yielding an 91.0% success rate and a training time of 35 minutes. This was made possible by integrating several post-processing algorithms into the original QNetGAN model, including the octet rule satisfaction validator, hydrogen addition algorithm, formal charge calculation, and molecular geometry (.XYZ file coordinate) calculation. In the future, we plan to use the QNetGAN v2 pathway to generate molecules with a wider range of elements and add functionality to calculate important molecular properties, ideally developing a third and more capable version of the QNetGAN molecular generation pathway that is able to outperform the accuracy of classical NetGAN.**

***Index Terms*—Quantum Graph Generative Adversarial Network, molecular generation, graph convolution, QNetGAN**

I. Introduction

Drug discovery processes have historically been very costly and lengthy, requiring upwards of two billion dollars and between five to ten years, though applications of machine learning algorithms have been able to greatly reduce these expenses [1] [2]. The most promising algorithms currently in use in computational drug discovery are variants of generative adversarial networks (GANs), which, at the most basic level contains two neural networks (the generator and the discriminator) that compete against each other in a zero-sum game until neither has a significant advantage over the other [3]. De Cao and Kipf first proposed using GANs for drug discovery with the development of the MolGAN, generating small molecular graphs with 9 atoms or less [4]. In 2021, Tsujimoto et. al. developed L-MolGAN which increased the maximum generated molecular graph size from 9 atoms to 20 atoms [5]. However, because of the massive size of the chemical space—which is on the order of 10^{60}—these GANs require large datasets to train efficiently on high-dimensional data like molecules. This, combined with the need for adversarial training—due to their architecture consisting of two competing sub-networks—causes GAN models to consume large amounts of computational resources [6]. As a result, quantum computing is leveraged to improve the computational efficiency and accuracy of these models; quantum GANs utilize superposition and entanglement for exponentially more data storage and efficient parallel computation, providing them with a notable advantage over classical ones.

In late 2021, Li et. al. proposed a hybrid quantum-classical generative adversarial network architecture with a quantum generator for drug discovery [7]. Recently, inspired by the work of Bojchevski, Shchur, and Zugner in their design of the classical NetGAN created by, Cui et. al previously developed a hybrid quantum-classical graph-based generative adversarial network for improving molecular synthesis via random-walks [8] [9]. The classical NetGAN model generates graph structures via biased second-order random walks, inherently preserving the topological properties—that is, already-present relationships between vertices and edges—of the real-world data it is modeled on [9]. Additionally, because real-world graphs are generally sparse, meaning most of the adjacency matrix is filled with zeros, by only generating nonzero entries of adjacency matrices, NetGAN is able to provide notable speedups compared to similar models [9]. Furthermore, NetGAN incorporates Long Short-Term Memory (LSTM) cells and gradient penalty drawn from the Wasserstein GAN to expedite model training and prevent mode collapse, one of the largest problems affecting GANs [9] [10].

The QNetGAN model developed by Cui et. al was demonstrated to have decreased training time and memory usage than its classical counterpart, but the model accuracy was significantly less—47% versus 93% [8]. This was due to the suboptimal design of the QNetGAN molecule verification system, which failed to account for the fact that molecules can and often do contain double and triple bonds, and that sometimes hydrogen atoms attached to heavy atoms can be implicitly specific. The geometry of the generated molecules were also not completely optimized, and factors such as electron-electron repulsion and bond angles were not considered.

Our newest model, QNetGAN v2, builds upon the work of the original QNetGAN by implementing and integrating several molecular post-processing algorithms to improve model accuracy. Specifically, an octet-rule/hydrogen-addition algorithm accounts for implicit hydrogens and ensures balance between hydrogen and non-hydrogen atoms, while a formal charge algorithm finds the most stable atom configurations for structural isomers and the geometry optimization algorithm calculates the 3D spatial coordinates of the generated molecular graphs. The QNetGAN v2 architecture also differs slightly from the original QNetGAN by including graph convolution layers, as these layers are specifically designed for extracting features from graph-structured data like molecules. Hence, combining this with the Long Short-Term Memory Cells (LSTMs) from the original QNetGAN allows QNetGAN v2 to be highly optimized for operations on sequential, graph-based data.

QNetGAN v2 has proven quite successful, demonstrating a 91.0% success rate with 273 out of 300 total generated molecules having valid chemical formulae, with all atoms constituting the molecule satisfying the octet rule, as verified by the octet rule algorithm. In addition, these 273 molecules were able to exist in a 3-dimensional space, with optimal bond angles and Valence Shell Electron-Pair Repulsion (VSEPR) geometries, which were successfully calculated with the molecular geometry post-processing algorithm. Therefore, QNetGAN v2 has proven to be a significant improvement from the original QNetGAN.

II. Dataset and Materials

As with the original QNetGAN, we trained QNetGAN v2 on the graph version of the QM9 dataset available from PyTorch Geometric. This dataset contains about 133885 small, stable organic molecules, consisting solely of carbon, oxygen, nitrogen, fluorine and hydrogen, in a graphical representation containing a node feature vector and an adjacency matrix describing which atoms are bonded to each other.

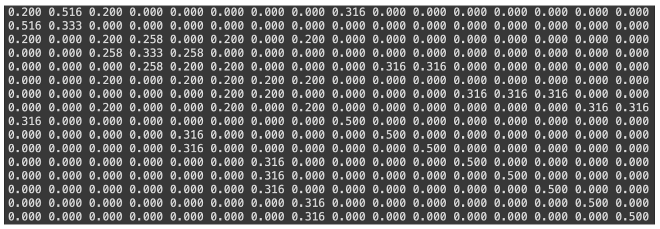
All code was written on and executed on a server provided by the Aspiring Scholars Directed Research Program (ASDRP), using Python (version 3.10.10). QNetGAN v2 was implemented using the libraries PyTorch (version 2.2.1+cu121), PyTorch Geometric (version 2.5.1), NetworkX (version 3.1), RDKit (version 2022.09.5), NumPy (version 1.26.4), and Pennylane (version 0.30.0). To visualize our generated molecules, we used Avogadro2.

III. Methods

A. *QNetGAN v2 Architecture*

QNetGAN v2's internal architecture is closely modeled off of that of the original QNetGAN, with several modifications. To that effect, the model consists of four main components: the dataset loader, the graph convolution and LSTM-based generator, the classical discriminator, and the molecular post-processing algorithms.

The QM9 dataset from PyTorch Geometric was stored in the dataset loader, which provided the discriminator with examples of real, chemically valid molecular graphs. The dataset loader served as a helper algorithm by normalizing the data and splitting the input molecular graphs into training and testing edges via the `train_test_split_edges` utility. The dataset loader also calculated the normalized adjacency matrix of the molecular graphs, as they were not provided in the original dataset. The normalized adjacency matrix is calculated as $D^{-0.5}AD^{-0.5}$, where A is the adjacency matrix with self-loops added and D is the degree matrix [11]. An example is shown in Fig. 1.


```
0.200 0.516 0.200 0.000 0.000 0.000 0.000 0.000 0.316 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.516 0.333 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.200 0.000 0.200 0.258 0.000 0.200 0.000 0.200 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.258 0.333 0.258 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.258 0.200 0.200 0.000 0.000 0.000 0.316 0.316 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.200 0.000 0.200 0.200 0.200 0.200 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.200 0.200 0.000 0.000 0.000 0.000 0.316 0.316 0.316 0.000 0.000
0.000 0.000 0.200 0.000 0.000 0.200 0.000 0.200 0.000 0.000 0.000 0.000 0.000 0.000 0.316 0.316
0.316 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.500 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.316 0.000 0.000 0.000 0.000 0.500 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.316 0.000 0.000 0.000 0.000 0.000 0.500 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.316 0.000 0.000 0.000 0.000 0.500 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.316 0.000 0.000 0.000 0.000 0.000 0.500 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.316 0.000 0.000 0.000 0.000 0.000 0.000 0.500 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.316 0.000 0.000 0.000 0.000 0.000 0.000 0.500 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.316 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.500
```
Fig. 1. Example of a Normalized Adjacency Matrix

QNetGAN v2's generator is almost identical to the original QNetGAN's generator, as both models feature Quantum LSTMs, with different parameterized quantum circuits (PQCs) serving as the forget, input, update and output gates of a classical LSTM. These PQCs use `StronglyEntanglerLayers` and `AmplitudeEmbeddingLayers` from Pennylane to map the qubits from their initial $|0\rangle$ state to their final state (Fig. 2). The expected value of the PauliZ observable on each of the qubits is then calculated to measure the circuit.

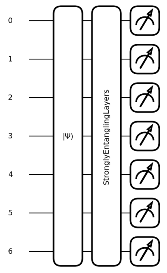


Fig. 2. The PQC

This also encodes the features of the input molecular graph into the rotation angles of the qubits. However, a difference between the two versions is that QNetGAN v2 replaces the regular convolution/PyTorch `Dense` layers found in the original QNetGAN with PyTorch Geometric `GCNConv` layers, which are specifically designed for graphs and hence better suited to our graph generation needs. The output of the generator is an adjacency matrix, which is inputted into the discriminator for judging.

The classical discriminator performs the task of classifying the output adjacency matrices as real or fake, based on the examples it was trained from the QM9 dataset. The internal architecture of the discriminator is essentially analogous to that of the original QNetGAN, except it also contains `GCNConv` layers, which allow the discriminator to better capture the properties and features of the input graphs. The output of the discriminator is a scalar quantity indicating how real or fake the input adjacency matrix was.

Lastly, the generated adjacency matrices are verified and converted into a molecular graph representation using NetworkX and RDKit, allowing the molecular geometry and 3D coordinates to be calculated via the post-processing algorithms.

B. Molecular Geometry Post-processing Algorithms

As mentioned in [8], the molecules generated by the original QNetGAN contained numerous chemically undesirable properties, such as incomplete octets and unfeasible bond angles and lengths. As such, we developed a set of post-processing algorithms that improved their chemical validity. The accuracy of these algorithms were tested by comparing the output processed molecules output from our algorithms with the original molecules from the QM9 dataset. Each algorithm demonstrated 100% accuracy.

1) Octet Rule: One important rule in chemistry is that atoms are most stable with their outer shell filled with eight electrons. Many molecules generated by the original QNetGAN had molecules containing atoms that did not fit this description, so we designed an octet rule validator for this purpose [8]. This algorithm traverses through the molecule using a depth-first search (DFS) and returns `true` if all atoms satisfy the octet rule, or `false` if one or more atoms do not.

2) Hydrogen Addition: After detecting that a molecule does not satisfy the octet rule, the quickest fix is to add bonds to hydrogens in the places where atoms are missing. This algorithm traverses the molecule using DFS, adding hydrogen atoms and augmenting the adjacency matrix where necessary to create a molecule that satisfies the octet rule. An example of this algorithm in action can be seen in Fig. 3 below.

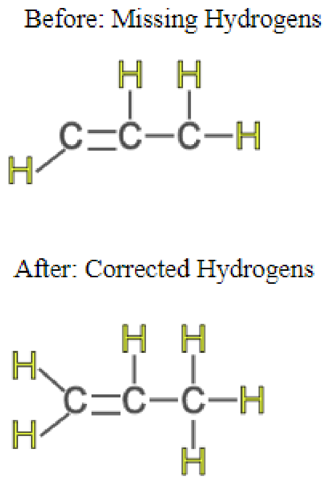


Fig. 3. Hydrogen Addition: Before and After

3) Formal Charge: Some molecules have structural isomers, which are molecules with an identical atom composition as another, but with different orientations or arrangements of atoms. There may exist several valid atom configurations that all satisfy the octet rule, so formal charge is used to determine which isomer(s) are more chemically stable. To optimize the geometry in this case, we implemented a formal charge calculation algorithm, which reflects the charge assigned to an atom in a molecule assuming all chemical bonds are shared equally [12]. Formal charge is calculated as

$$\text{Valence electrons} - \text{Nonbonding electrons} - \frac{\text{Bonding electrons}}{2} \tag{1}$$

The ultimate goal is to find the atom configuration that minimizes the formal charge on all atoms—in other words, as close to 0 as possible. In the case that not all atoms can have a formal charge of 0, the extra electrons should be placed onto the most electronegative atom(s) in the molecule.

4) Molecular Geometry Calculation: The molecular coordinates generated by the original QNetGAN were all 2-dimensional with the z-coordinates equal to 0.0 [8]; hence, reconfiguration of .XYZ files was necessitated in order to create chemically feasible structures. The first step of this algorithm is to identify cyclic molecules, as the process for calculating coordinates for cyclic and noncyclic molecules are very different [13]. We used a simple algorithm based on Floyd's Tortoise and Hare algorithm (Fig. 4) to divide a molecule into cyclic and noncyclic portions. We then applied the appropriate coordinate calculation method based on these results. [14].

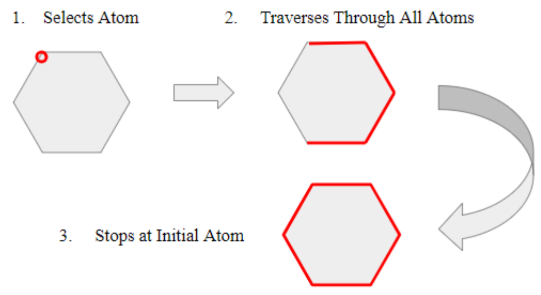


Fig. 4. Cycle Detection Algorithm

After we identified the number of bonds and unbonded electrons associated with each non-hydrogen atom and assigned each a molecular geometry description according to Valence Shell Electron Pair Repulsion (VSEPR) theory (e.g. tetrahedral, trigonal planar, linear, bent) [15]. Cyclic molecules were assigned cycle sizes rather than VSEPR descriptions. We then created specific functions for each geometry to assign coordinates using trigonometric functions, well-known average bond angles for each VSEPR structure and bond lengths derived from a database [15].

For molecules with multiple central atoms, we utilized rotation matrices and Euler angles in conjunction with another DFS to build the molecule. Since the scope of this project only includes small molecules with under 20 atoms, it is computationally efficient to use many depth-first searches rather than optimizing the number of these searches. This generation calls the molecular geometry functions to determine displacement coordinates for atoms bonded to a specific central atom, and then rotates them appropriately to match bonds already formed on that atom using the matrix multiplication of the displacement coordinates and rotation vector. The angles of rotation are determined using Rodrigues's rotation formula [17].

C. QNetGAN v2 Training Workflow

As their name suggests, GANs are trained adversarially, with the training loop alternating between the generator and discriminator as these two models compete to gain an advantage over the other—our QNetGAN v2 is no different [3].

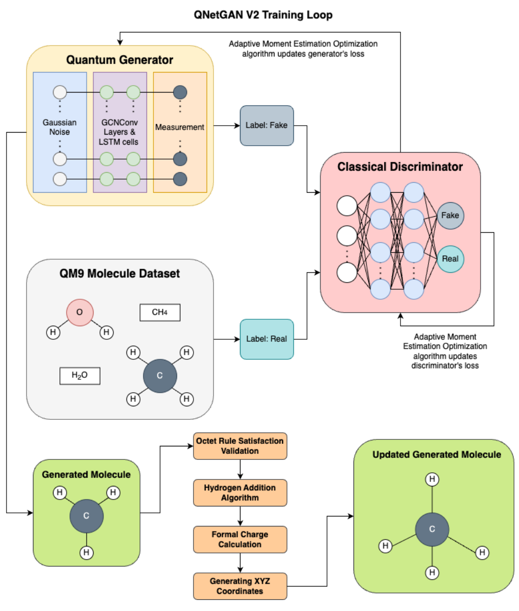

QNetGAN V2 Training Loop
Adaptive Moment Estimation Optimization algorithm updates generator's loss
Quantum Generator
Gaussian Noise
GCNConv Layers & LSTM cells
Measurement
Label: Fake
Classical Discriminator
Fake
Real
QM9 Molecule Dataset
O
H
CH4
H2O
C
Label: Real
Adaptive Moment Estimation Optimization algorithm updates discriminator's loss
Generated Molecule
Octet Rule Satisfaction Validation
Hydrogen Addition Algorithm
Formal Charge Calculation
Generating XYZ Coordinates
Updated Generated Molecule

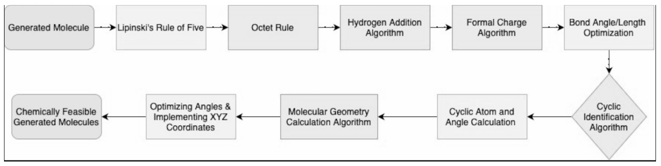
Fig. 5. QNetGAN v2 Training Schematic

In the first step of the training loop, shown above, an example of a real, molecular graph is inputted to the discriminator to get a feeling for what constitutes a chemically feasible or chemically infeasible molecular graph (Fig. 5). Meanwhile, the parameters of the generator—namely the angles of the rotation gates in the `AmplitudeEmbeddings` in the LSTM Cells —are initialized with random noise from a Gaussian distribution. A similar square matrix of Gaussian noise (representing an adjacency matrix) serves as the initial input to the generator. This matrix is passed through the PyTorch Geometric `GCNConv` layers, which performs graph convolution; the `edge_index` of the training data inputted to the discriminator is also passed through these graph convolution layers for consistency between both models when comparing results. This matrix is then passed through the LSTM cells before being sent to the discriminator for inference.

The discriminator determines the validity of the generated adjacency matrix and calculates both the generator's loss and its own loss, using the Adaptive Moment Estimation optimization algorithm to update the parameters of both models. Motivated by the Wasserstein GAN, a gradient penalty function was also implemented to enforce the Lipschitz constraint, hence affecting smoother training and mitigating mode collapse [10].

To optimize the training, learning rate schedulers were also implemented to automatically adjust the learning rate between training epochs [16]. This is an important part of hyperparameter tuning, as the learning rate controls the size of the step the optimizer takes to reach the loss function's minima. A large learning rate helps the model learn quickly, but can cause it to overshoot the minima; a small learning rate causes the model to converge slowly and potentially get stuck in a plateau [16]. Specifically, the generator used a combination of the `CosineAnnealingWarmRestarts` and `PolynomialLR` with degree 2, while the discriminator used a `CosineAnnealingLR` and `PolynomialLR`, also with degree 2.

The training loop terminates when the loss of both models stops changing significantly; at this point, QNetGAN v2 has converged. After convergence, we sample several adjacency matrices from the generator, using a method similar to the reparameterization trick used in variational autoencoders [11]. Because the entries of these matrices are not integers—they represent the probability of an edge between two vertices, rather than the existence of an edge between two vertices —a utility function is used to realize a graph from these probabilistic adjacency matrices. Finally, these matrices are converted to RDKit molecules, where they are then passed through the molecular post-processing algorithms, which determine if the molecules can exist in the real world and if so, calculate their 3D geometry and coordinates (Fig. 6).



Generated Molecule
Lipinski's Rule of Five
Octet Rule
Hydrogen Addition Algorithm
Formal Charge Algorithm
Bond Angle/Length Optimization
Cyclic Identification Algorithm
Cyclic Atom and Angle Calculation
Molecular Geometry Calculation Algorithm
Optimizing Angles & Implementing XYZ Coordinates
Chemically Feasible Generated Molecules

Fig. 6. Molecular Geometry Post-processing Pipeline

IV. Results

We trained QNetGAN v2 on the first 140000 examples from the QM9 dataset from PyTorch Geometric in batches of 200, before sampling 300 fake molecules from the generator after convergence. The drug-likeness and chemical validity molecules were verified using the octet rule post-processing algorithm, as well as Lipinski's Rule of Five, a set of criteria which determines if a molecule is a suitable orally active drug. The conditions that make up Lipinski's Rule of Five are as follows: no more than 5 hydrogen bond donors, no more than 10 hydrogen bond acceptors, molecular mass less than 500Da, and an octanol-water partition coefficient less than 5. Formal charge and molecular geometry calculations were utilized to determine whether the molecule could exist in a stable configuration in 3-dimensional space. Based on the aforementioned criteria, 273 of the 300 generated molecules were chemically valid, resulting in a 91.0% success rate. Examples of these molecules can be seen in Fig. 7.

From Table I, we can see that in terms of memory usage, QNetGAN v2 was slightly more efficient than the original QNetGAN, using up to 3.1GB of RAM (versus 3.4GB for QNetGAN).

TABLE I
Comparison of Accuracy and Efficiency

Model	Accuracy %	Training Time	Memory Usage
NetGAN	93.3+%	50 to 65min	8.2GB
QNetGAN	47.0%	10 to 20min	3.4GB
QNetGAN v2	91.0%	35 to 45min	$\leq$ 3.1GB

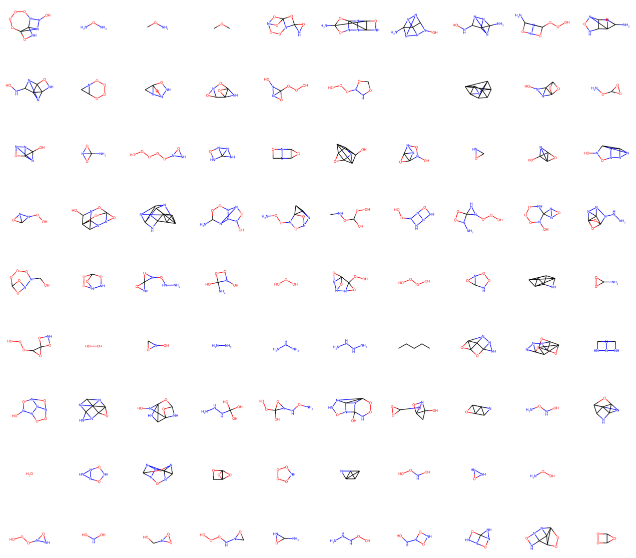


Fig. 7. Examples of Valid Molecules

In terms of computational efficiency, with GPU acceleration, QNetGAN v2 took up to 45 minutes to reach convergence. When considering the seconds per epoch training speed of both models, QNetGAN v2 took around 12 to 15 seconds per epoch, while the original QNetGAN took about 9 seconds per epoch [8]. Hence we see that QNetGAN v2 was around 67% slower than QNetGAN, which is because the model was trained on a larger portion (10% vs 1%) of the dataset. If we take this into account, we find that QNetGAN v2 was able to learn the patterns in the real-world molecular graph data much quicker than the original QNetGAN.

However, it is important to note that the model was run on a quantum simulator, due to lack of free access to powerful-enough devices; hence the computational resource usage and accuracy of the model are not representative of running the model on an actual quantum computer. Specifically, the simulator needs to keep copies of the qubits' state to perform expensive matrix multiplications to replicate the quantum gate operations, whereas these operations are native to quantum computers—the operations are much more efficient. Furthermore, the simulator assumed all qubits were ideal (no noise or decoherence), and that quantum gate fidelity (accuracy of the physical gate a specific quantum hardware offers compared to the ideal gate) was maximal. On a real quantum computer, qubits are extremely sensitive to their surroundings and can decohere quite quickly, and state and gate fidelity is rarely 100%, which introduces error into the computation. In addition, the accuracy of performing multi-qubit operations is highly dependent on the layout of the individual qubits on the quantum computer and the speed of the quantum processor, which will negatively affect the accuracy of QNetGAN v2.

Nevertheless, the 91.0% accuracy rate is a strong indicator of QNetGAN v2's true potential, although we expect that this percentage will decrease when the molecular geometry algorithm gets integrated into the model pipeline.

V. Discussion and Future Work

While the 91.0% success rate certainly is promising, it is important to remember that the model was run on a simulator, which assumes everything is ideal. As this is not the case in the real world, we aim to run the model on a noisy quantum simulator and possibly an actual quantum device to get a better understanding of the computational resource usage and efficiency. We would also be able to scale the model to larger, more complex molecule datasets with a wider range of atom types. To be able to run QNetGAN v2 on a NISQ device, the model must be converted to a fully-quantum architecture, which we are in the process of doing with QNetGAN v3, the next version of our model. As such, are exploring equivariant graph embeddings for encoding adjacency matrices into qubit states, quantum dropout layers, and quantum graph convolution layers [18] [19] [20]. Of course, this will also require implementation of error mitigation and error correction techniques to make our model more robust. However, our group has encountered difficulty with getting access to quantum resources that are able to run the QNetGAN v2 model without being financially overbearing. We will continue to work to get funding for this endeavor.

As for the post-processing algorithms, we are also working on refining the cycle detection algorithm to detect multiple cycles and label the atoms in theses cycles to work with molecules with multiple cyclo components. We are also looking to augment the geometry calculation algorithm so that it is able to adjust atom coordinates based on electron-electron repulsion and other molecular interactions; current geometries assume no interactions between electrons. We expect this to be completed within the next 3 months, due to the complex linear algebra and advanced chemistry knowledge this required. Upon completion, we will integrate the algorithm into the QNetGAN v2 pipeline to add another layer of validation to generated molecules.

Furthermore, as seen in Fig 7, most molecules only contain single bonds. In the real world, many organic molecules have double, triple, and aromatic bonds, so we are also working on implementing this functionality. This would require a reinterpretation of the entries of probabilistic adjacency matrices, which we are currently researching.

Our ultimate goal is to develop a fully-quantum QNetGAN model with over 95% accuracy in generating chemically valid molecules, and calculate various pharmaceutical properties useful to drug researchers.

Acknowledgment

We would like to thank our research advisor, Dr. Larry McMahan, for sharing his knowledge and expertise in guiding us through this complex project and offering insights, and Mr. Neil McMahan for helping us troubleshoot problems with ASDRP's computational resources. Next, we'd like to thank the previous members of the QGAN and QNetGAN research groups for contributing their teamwork and diverse knowledge. Without them, QNetGAN v2 and the previous models would not exist. Finally, we'd like to thank ASDRP for providing us with the opportunity to conduct this innovative research.

References

[1] G. A. Van Norman, "Drugs, Devices, and the FDA: Part 1," JACC: Basic to Translational Science, vol. 1, no. 3, pp. 170–179, Apr. 2016, doi: 10.1016/j.jacbts.2016.03.002.
[2] C. H. Wong, K. W. Siah, and A. W. Lo, "Estimation of clinical trial success rates and related parameters," Biostatistics, vol. 20, no. 2, pp. 273–286, Jan. 2018, doi: 10.1093/biostatistics/kxx069.
[3] I. J. Goodfellow et al., "Generative Adversarial Networks," arXiv.org, Jun. 10, 2014. https://doi.org/10.48550/arXiv.1406.2661
[4] D. C. Nicola and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," arXiv.org, May 30, 2018. https://doi.org/10.48550/arXiv.1805.11973
[5] Y. Tsujimoto, S. Hiwa, Y. Nakamura, Y. Oe, and T. Hiroyasu, "L-MolGAN: An improved implicit generative model for large molecular graphs," ChemRxiv, 2021. doi:10.26434/chemrxiv.14569545.v3
[6] D. Saxena and J. Cao, "Generative Adversarial Networks (GANs)," ACM Computing Surveys, vol. 54, no. 3, pp. 1–42, May 2021, doi: 10.1145/3446374.
[7] J. Li, M. Alam, C. M. Sha, J. Wang, N. V. Dokholyan, and S. Ghosh, "Invited: Drug Discovery Approaches using Quantum Machine Learning," 2021 58th ACM/IEEE Design Automation Conference (DAC), Dec. 2021, Published, doi: 10.1109/dac18074.2021.9586268.
[8] M. Cui, A. Chau, M. Pan, V. Vaiyakarnam and L. McMahan, "Molecular Geometry Generation Processes Through Hybrid Quantum-Classical Generative Adversarial Networks and Python-Based Self-Consistent Field Molecular Calculations," 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), Bellevue, WA, USA, 2023, pp. 312-313, doi: 10.1109/QCE57702.2023.10258.
[9] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating Graphs via Random Walks," arXiv.org, Mar. 02, 2018. https://arxiv.org/abs/1803.00816v2
[10] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," arXiv.org, Jan. 26, 2017. https://doi.org/10.48550/arXiv.1701.07875
[11] T. N. Kipf and M. Welling, "Variational Graph Auto-Encoders," arXiv.org, Nov. 21, 2016. https://arxiv.org/abs/1611.07308
[12] Libretexts, "Formal Charge," Chemistry LibreTexts, Jan. 30, 2023. https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Physical_Properties_of_Matter/Atomic_and_Molecular_Properties/Formal_Charges/Formal_Charge
[13] Jens. Sadowski and Johann. Gasteiger, "From atoms and bonds to three-dimensional atomic coordinates: automatic model builders," Chemical Reviews, vol. 93, no. 7, pp. 2567–2581, Nov. 1993, doi: 10.1021/cr00023a012.
[14] GeeksforGeeks, "Floyd s Cycle Finding Algorithm," GeeksforGeeks, May 10, 2023. https://www.geeksforgeeks.org/floyds-cycle-finding-algorithm/
[15] "VSEPR Theory," Germanna Academic Center for Excellence, Mar. 2022. https://germanna.edu/sites/default/files/2022-03/VSEPR%20Theory.pdf
[16] K. Li, "How to Choose a Learning Rate Scheduler for Neural Networks," neptune.ai, Aug. 09, 2023. https://neptune.ai/blog/how-to-choose-a-learning-rate-scheduler
[17] Simsangcheol, "Rodrigues' rotation formula - Simsangcheol - Medium," Medium, Apr. 13, 2023. https://medium.com/@sim30217/rodrigues-rotation-formula-47489db49050
[18] M. Schuld, "An equivariant graph embedding," PennyLane Demos, Jan. 01, 2024. https://pennylane.ai/qml/demos/tutorial_equivariant_graph_embedding/
[19] F. Scala, "Dropout for Quantum Neural Networks," PennyLane Demos, Mar. 12, 2024. https://pennylane.ai/qml/demos/tutorial_quantum_dropout/
[20] J. Zheng, Q. Gao, and Y. Lv, "Quantum Graph Convolutional Neural Networks," arXiv.org, Jul. 07, 2021. https://arxiv.org/abs/2107.03257
[21] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial Autoencoders," arXiv.org, Nov. 18, 2015. https://arxiv.org/abs/1511.05644