The Pennsylvania State University
The J. Jeffrey and Ann Marie Fox Graduate School

ADDRESSING SCIENTIFICALLY COMPLEX PROBLEMS WITH DOMAIN AWARE CLASSICAL AND QUANTUM MACHINE LEARNING ALGORITHMS

A Dissertation in
Computer Science and Engineering
by
Collin Beaudoin

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2025

The dissertation of Collin Beaudoin was reviewed and approved by the following:

Swaroop Ghosh
Professor of Electrical Engineering and Computer Science
Dissertation Advisor
Chair of Committee

Mahmut Taylan Kandemir
Professor of Computer Science and Engineering

Abhronil Sengupta
Associate Professor of Electrical Engineering

Joseph Najem
Assistant Professor of Mechanical Engineering

Chitaranjan Das
Department Head of Computer Science and Engineering

Abstract

Quantum computing and material discovery show great potential for advances in science and technology. However, progress in these domains is often hindered by the high computational cost and complexity of required methods. This thesis demonstrates that leveraging domain-aware classical and quantum machine learning (QML) can reduce resource requirements and accelerate discovery.

In the context of improving quantum algorithm implementation, we address the challenge of high error rates in quantum circuits. We explore quantum circuit synthesis, demonstrating that generative graph-based models can produce functionally equivalent circuits with reduced gate counts and depth, enhancing efficiency and reliability. To overcome the limitations of human-designed quantum circuits, we propose the first graph diffusion model for quantum circuit generation. This novel approach enables the design of new, functional quantum circuits, representing a significant step toward the development of automated quantum architecture.

In search of practical applications of quantum algorithms, we explore the field of molecular design. Previous efforts in molecular property prediction, a critical step in molecular synthesis, have focused on scaling model size and complexity to achieve performance gains. This approach has led to marginal improvements in accuracy, accompanied by a near-exponential growth in parameter counts. This thesis examines how innovations in classical and hybrid quantum-classical natural language processing (NLP) and model architecture can achieve state-of-the-art performance with significantly fewer trainable parameters. We begin by exploring hybrid QML models. A comparative analysis between classical Long Short-Term Memory (LSTM) networks and hybrid quantum-classical LSTMs (QLSTMs) reveals that, while current noisy intermediate-scale quantum (NISQ) devices underperform on standard classical benchmarks, they show promise in identifying molecular substructures with higher fidelity than their classical counterparts. Additionally, we analyze and compare the performance of large language models (LLMs), graph neural networks (GNNs), and recurrent neural networks (RNNs). Our findings suggest that smaller models can outperform more resource-intensive counterparts with task-specific model design, informed architecture selection, and problem-aware data curation. Furthermore, we evaluate advanced linguistic techniques and data augmentation on the performance of classical and hybrid QML. Augmentation enables significant improvements, showing the importance of data pre-processing in molecular property prediction.

This thesis establishes a practical framework for quantum circuit optimization/design and advancing molecular property prediction.

THE CONTENT OF THIS THESIS IS BASED ON RESEARCH THAT HAS APPEARED, OR IS FORTHCOMING, IN PEER-REVIEWED PUBLICATIONS ACROSS VARIOUS VENUES. THE PRIMARY ASSOCIATED WORKS ARE AS FOLLOWS:

[1] BEAUDOIN, C., KUNDU, S., TOPALOGLU, R. O., & GHOSH, S. (2022). Quantum machine learning for material synthesis and hardware security. In Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (pp. 1-7).

[2] BEAUDOIN, C., PHALAK, K., & GHOSH, S. (2024). Predicting Side Effect of Drug Molecules Using Recurrent Neural Networks. IEEE Transactions on Emerging Topics in Computational Intelligence.

[3] BEAUDOIN, C. & GHOSH, S. (2025). Evaluating Effects of Augmented SELFIES for Molecular Understanding Using QK-LSTM. *preprint arXiv:2504.20789*.

[4] BEAUDOIN, C., PHALAK, K., & GHOSH, S. (2024). AltGraph: Redesigning Quantum Circuits Using Generative Graph Models for Efficient Optimization. In Proceedings of the Great Lakes Symposium on VLSI 2024 (pp. 44-49).

[5] BEAUDOIN, C. & GHOSH, S. (2025). "Q-Fusion: Diffusing Quantum Circuits", *preprint arXiv:2504.20794*.

We live in an impatient age. We demand results. We find a long and laborious process very irksome, and are constantly seeking for a short cut. But there is no easy method of securing discipline. It is axiomatic that there is no royal road to learning. The effort for discipline must be intensive, and to a considerable degree it must be lifelong.

— Calvin Coolidge 1924.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I want to thank my advisor, Dr. Swaroop Ghosh. His support, guidance, and belief in me throughout my thesis have meant a lot. Dr. Ghosh is a fantastic mentor and one of the kindest and most helpful individuals I have known. His patience, encouragement, and mentorship have shaped my academic and personal growth. I am truly fortunate to have had him as my advisor.

My thanks go to Dr. Zachary Kurmas and Dr. Christian Trefftz for encouraging my interest in quantum computing and for supporting my academic path. Their belief in me opened the door to Penn State and helped set the course of my research. I also wish to thank the Computer Science and Computer Engineering Department faculty at Grand Valley State University. Their mentorship laid the essential groundwork for everything that followed.

To my friends, Dan McAlary, Thomas Gomez, Adam Freeman, Tess Riggs, Michael Kane, Anthony Gualtieri, Emily Thimmesch, Megan Hoffman, and the Roses, thank you for your friendship and support along the way. I am also thankful to the Whitebread family and the Happy Valley Barbarians for making me feel at home in State College. Your support has meant so much to me. To my labmates, thank you for your collaboration, support, and friendship.

I also thank my parents, Todd and Christina. Your sacrifices, love, and support have been the foundation of my success. I am grateful to my grandfather for his wisdom and to my late grandparents, whose hard work made this journey possible.

To my wife, Samantha, thank you. Your love, encouragement, and belief in me have carried me through everything. I am grateful to share this journey with you.

Thank you to everyone not mentioned by name who has played a role in this journey. I leave you with a quote from my favorite author, J.R.R. Tolkien, which has guided me many times:

“Go back?" he thought. "No good at all! Go sideways? Impossible! Go forward? Only thing to do! On we go!”

This material was supported in part by the National Science Foundation (NSF) under grants CNS-1722557, CCF-1718474, OIA-2040667, DGE-1723687, and DGE-1821766, as well as through generous support from Intel. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of these agencies. I would also like to thank my co-authors and my dissertation committee, Dr. Mahmut Taylan Kandemir, Dr. Abhronil Sengupta, and Dr. Joseph Najem, for their invaluable feedback, insights, and support throughout this process.

I would like to thank my wife, Samantha Beaudoin. Her kindness, support, and patience made this possible. I would also like to thank my friends and family for their constant support which brought me to where I am today.

Chapter 1 Introduction

1.1 Overview of Quantum Computing

1.1.1 Qubits

Qubits are the fundamental units in a quantum computer. In contrast to classical bits, which can exist only in two discrete states, 0 or 1, a qubit can be in a superposition of both states. This uniquely quantum property arises from the principles of quantum mechanics and enables qubits to encode a probabilistic spectrum of information, unlike their classical counterparts. As a result, qubits lie at the heart of the parallelism that gives quantum computation its theoretical power.

Mathematically, the state of a single qubit is described by a complex combination of two basis states, denoted by $|0\rangle$ and $|1\rangle$. A state of a qubit can be expressed as

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad \text{where} \quad \alpha, \beta \in \mathbb{C}, \quad \text{and} \quad |\alpha|^2 + |\beta|^2 = 1.$$

The squared magnitudes $|\alpha|^2$ and $|\beta|^2$ represent the probabilities of measuring the qubit in the states $|0\rangle$ and $|1\rangle$, respectively, assuming measurement in the computational basis. The basis states themselves correspond to the classical binary values, represented as

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Bloch spheres are used for geometric intuition, a unit sphere in three-dimensional space on which each point on the surface represents a unique state of a qubit. Any qubit state can be

written in spherical coordinates as

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle,$$

Where the parameters define a point on the surface of the Bloch sphere, this representation provides a visualization tool for understanding single qubit operations as rotations about the axes of the sphere. It also offers insight into the role of quantum coherence and relative phase in quantum algorithms and interference effects.

The implementation of qubits varies based on platform-specific goals. Examples include the discrete energy levels of superconducting circuits, the spin states of trapped ions, the polarization modes of single photons, or localized electron spin states in diamond nitrogen-vacancy (NV) centers. Each architecture presents a unique set of trade-offs regarding coherence times, gate fidelities, scalability, and operational temperature requirements. Engineering reliable qubits that balance these factors remains one of the central challenges in building scalable quantum computers.

1.1.2 Quantum Gates

All quantum gates act on Hilbert spaces of dimension $2^n \times 2^n$, where n is the number of qubits in the system, to manipulate the qubits within a quantum circuit. These quantum gates are analogous to classical logic gates. However, unlike classical gates, which operate on binary values, quantum gates manipulate the probability amplitudes of quantum states. Quantum gates are represented by unitary matrices, U, which satisfy the condition $U^\dagger U = I$, where $U^\dagger$ denotes the conjugate transpose of U, and I is the identity matrix. This unitary condition ensures that the total probability is preserved.

Each gate implements a unitary transformation acting on the state of one or more qubits. These operations must respect the linearity of quantum mechanics and are typically realized through controlled interactions with external fields. The physical implementation of quantum gates varies depending on the underlying hardware. For instance, microwave pulses induce rotations in superconducting qubits, while lasers are employed in trapped ions and semiconductor quantum dots. In photonic qubits, beam splitters and phase shifters enact gate operations, whereas radio-frequency pulses control spin states in Nuclear Magnetic Resonance (NMR) systems. Gate operation times vary across technologies, from sub-nanosecond scales in photonic and superconducting systems to milliseconds or longer in NMR and ion-trap implementations.

1.1.2.1 Single Qubit Gates

Single-qubit gates operate on individual qubits and can be visualized as rotations on the Bloch sphere. These gates form the foundation of universal quantum computation. The most fundamental is the Hadamard gate:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The Hadamard gate creates equal superpositions from basis states and maps between the computational and diagonal bases. It is key in algorithms like the Quantum Fourier Transform and Grover's search. General single-qubit gates are described as rotation around the Bloch sphere axes:

$$R_X(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}, \quad R_Y(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix},$$

$$R_Z(\theta) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$$

Where a unitary operation can be expressed as:

$$U(\theta, \phi, \lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}.$$

1.1.2.2 Multi-Qubit Gates

To achieve universal quantum computation, single qubit gates must be combined with multi-qubit gates, which enable entanglement and conditional control between qubits. The most basic of these is the Controlled-NOT (CNOT) gate:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

At the three-qubit level, the Toffoli gate (or CCNOT) performs a NOT on the target qubit

only when both control qubits are in the $|1\rangle$ state. Its matrix is given by:

$$\text{Toffoli} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

1.1.2.3 Universality and Circuit Construction

Together, single-qubit rotations and a suitable two-qubit gate (such as CNOT) form a universal gate set capable of approximating any unitary operation on an arbitrary number of qubits to arbitrary precision. This foundational result underlies the design of quantum circuits and quantum programming languages. Quantum circuits composed of these gates can simulate the dynamics of any quantum system within the confines of the no-cloning theorem and decoherence limits.

1.1.3 Quantum Circuits

A quantum circuit is a model for quantum computation. It represents the sequential application of quantum gates to a set of qubits to manipulate their quantum states and implement algorithms. The structure of a quantum circuit can be broadly decomposed into three distinct stages: state initialization, unitary evolution, and measurement.

A quantum system is usually initialized in the basis state $|0\rangle$ for all individual qubits. The subsequent transformation involves the application of quantum gates represented by a unitary matrix. These gates may act on individual qubits or multiple qubits, enabling phenomena such as superposition and entanglement, which are central to quantum computational advantage. The overall transformation enacted by a circuit is described by the composition of these gates, which corresponds to matrix multiplication in linear algebra. Finally, the quantum state is projected onto the computational basis in the measurement phase. This projection collapses the quantum state, yielding classical outcomes with probabilities determined by the squared magnitudes of the state's complex amplitudes following the Born rule.

To illustrate, consider a single-qubit quantum circuit where a Hadamard gate H is applied

to an initial state $|0\rangle$. The Hadamard gate and the basis state are given by:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \text{and} \quad |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Applying H to $|0\rangle$ yields the state:

$$|\psi\rangle = H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}.$$

This resulting state is a balanced superposition of the $|0\rangle$ and $|1\rangle$ states. On the Bloch sphere, this transformation rotates the state vector from the north pole (corresponding to $|0\rangle$) to a point on the equator along the X-axis. The representation in Fig. 1.1 visually emphasizes how quantum gates perform rotations in Hilbert space.

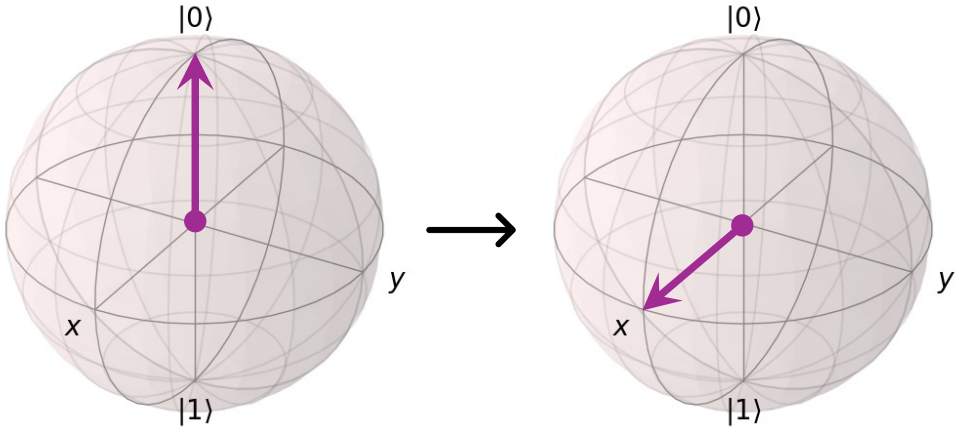


Figure 1.1: Hadamard gate applied to $|0\rangle$ making the qubit a superposition state $(|0\rangle + |1\rangle)/\sqrt{2}$. The action is visualized as a $\pi/2$ rotation about the Y+Z axis on the Bloch sphere.

Although quantum circuits are often described using high-level abstractions, executing them on real quantum hardware necessitates translating these circuits into a form compatible with the hardware's native gate set. For instance, multi-qubit gates such as the Toffoli (CCNOT) must be decomposed into gates like the CNOT, Hadamard, and phase gates, depending on the device architecture. This transformation process, known as transpilation, involves gate decomposition and device-specific mapping to account for qubit connectivity and gate fidelity.

1.1.4 Full Stack of Quantum Computing on Superconducting Qubits

Logical quantum circuits represent the idealized form of quantum algorithms, abstracted from any specific hardware constraints. However, to execute these circuits on real quantum hardware, they must be transformed to align with the physical and architectural limitations of the target platform. This conversion process is illustrated in Figure 1.2, using superconducting qubits, the most widely adopted quantum hardware platform at the time of writing, as a representative example.

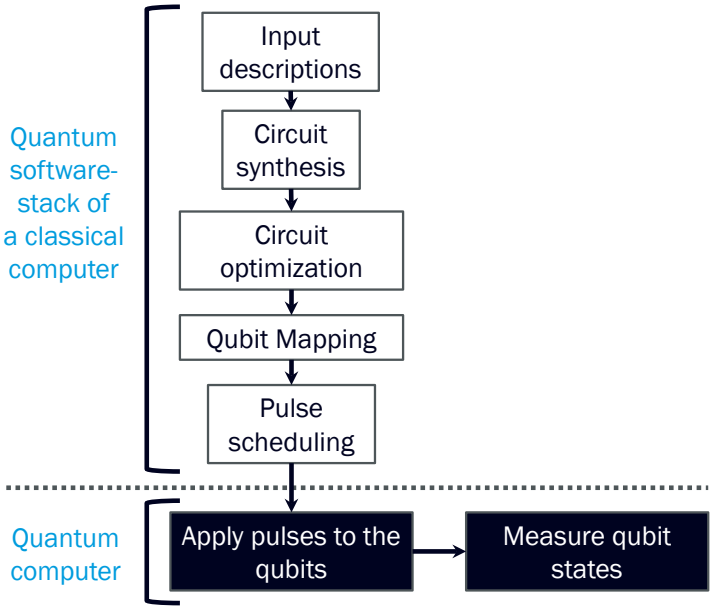


Figure 1.2: The general process of superconducting hardware for conversion between a logical circuit and a physical circuit.

The compilation pipeline begins with a hardware-agnostic quantum program, often written in an intermediate representation such as OpenQASM. This program is passed to a transpiler, which also takes as input a description of the target hardware, including the native gate set and qubit connectivity (coupling map). The transpiler first performs gate synthesis, converting any unsupported logical operations into sequences of hardware-native gates. Following this, the circuit undergoes optimization, where redundant or canceling operations are eliminated, and adjacent rotations are merged to reduce circuit depth and error accumulation.

The next phase, known as qubit mapping, assigns the logical qubits from the circuit to a subset of the physical qubits available on the device. Due to variations in qubit fidelity and connectivity, this mapping is a critical optimization step; choosing an optimal layout can significantly increase the likelihood of successful execution. However, due to hardware

connectivity constraints, some two-qubit gates may not be directly executable after mapping. To resolve this, SWAP gates are inserted into the circuit, allowing logical qubit states to be exchanged between physically connected qubits. These SWAP operations are themselves decomposed into native gates and may undergo further optimization.

Once the circuit conforms entirely to the hardware's native gate set and connectivity constraints, it is translated into microwave pulse sequences. These physical control signals drive quantum gate operations on superconducting qubits. These pulses are orchestrated by a control system that schedules and delivers the correct signals to each qubit at precise times, as specified by the transpilation output.

Despite these elaborate transformations, all quantum hardware remains subject to noise and imperfections. Both qubit coherence and gate fidelity impose practical limitations, meaning that even fully transpiled and optimized circuits are still susceptible to errors during execution.

1.1.5 Errors in Quantum Computing

Quantum information processing is sensitive to errors due to the nature of quantum states and their susceptibility to disturbances from the surrounding environment. Unlike classical bits, which are robust to minor fluctuations, qubits can suffer significant degradation from even minimal noise. These disturbances, from control imperfections, thermal fluctuations, electromagnetic interference, or couplings to uncontrolled degrees of freedom, are collectively called quantum noise. The resulting errors can distort the stored quantum information.

At the most fundamental level, quantum errors can be categorized into three archetypal types: bit-flip errors, phase-flip errors, and their combination, the bit-phase-flip error. These can be represented using the Pauli operators X, Z, and $Y = iXZ$, respectively. Bit-flip errors, induced by the X operator, change $|0\rangle \leftrightarrow |1\rangle$, analogous to classical bit errors. Phase-flip errors, induced by Z, leave the basis states unchanged but reverse the relative phase of a superposition: $Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle$. In realistic quantum systems, these errors may occur in combination and at varying rates, depending on the physical implementation and environmental interactions.

1.1.5.1 Gate Error

Gate errors occur when quantum gates are implemented imperfectly. Theoretically, a gate corresponds to a unitary operation U, applied with perfect precision. However, the implemented gate may deviate from the ideal due to imperfections in control electronics, signal distortions, or calibration errors, resulting in a perturbed operation $\tilde{U}$. As a result, the state transformation

becomes $|\psi\rangle \mapsto \tilde{U}|\psi\rangle$, which may deviate from the desired outcome.

1.1.5.2 Decoherence Error

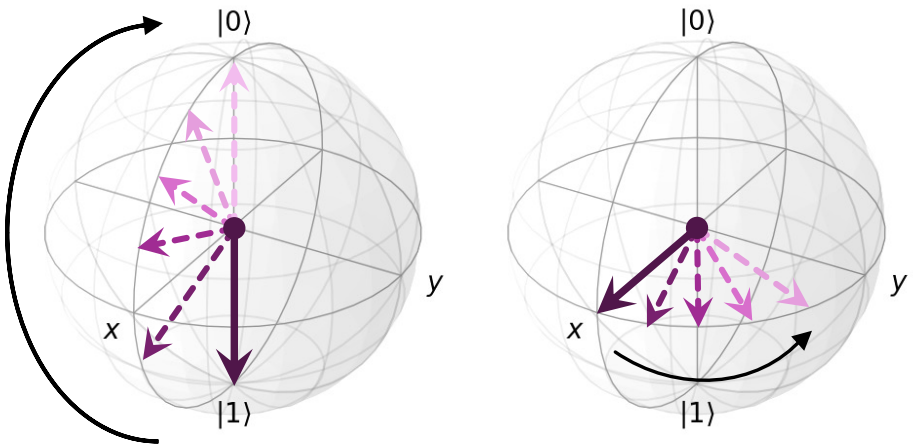


Figure 1.3: Relaxation (*left*) and dephasing (*right*) errors illustrated on the Bloch sphere; solid arrows represent the original pure states, and dashed arrows indicate their trajectory & transformation under noise. The combined effect of these processes leads to depolarizing errors.

Decoherence refers to the process by which qubits lose their quantum nature and transition into classical. It is driven by interactions/entanglement with the system and its surrounding environment, erasing its quantum information. Decoherence is typically modeled using two principal mechanisms:

Relaxation: the spontaneous decay of the excited state $|1\rangle$ to the ground state $|0\rangle$, often due to energy dissipation into the environment.

Dephasing: the randomization of the relative phase between $|0\rangle$ and $|1\rangle$, without energy exchange.

Together, these processes cause the qubit to contract toward the sphere's center, representing the transition from a pure quantum state to a maximally mixed state. A visualization of this is in Fig. 1.3. This evolution can often be approximated by a depolarizing channel, which captures the uniform loss of quantum information.

1.1.5.3 Crosstalk Error

Crosstalk refers to unintended interactions between qubits; these parasitic effects can manifest when a control signal targeted at one qubit inadvertently alters the state of a neighboring qubit.

This can occur through residual electromagnetic coupling, shared control lines, or physical proximity.

Crosstalk is typically modeled by a weak, unwanted Hamiltonian interaction such as:

$$H_{\text{crosstalk}} = \varepsilon X_i X_j,$$

where ε quantifies the interaction strength between qubits i and j, which are not intended to be coupled. The effect may induce spurious entanglement, phase errors, or unwanted rotations, and it becomes particularly problematic in densely connected qubit arrays.

1.1.5.4 Measurement Error

Measurement errors occur when the classical result obtained from a quantum measurement does not reflect the system's true quantum state. These errors can be introduced by detector inefficiencies, thermal noise during readout, timing errors, or miscalibrated signal thresholds during analog-to-digital conversion.

A confusion matrix can model measurement outcomes, which captures the conditional probabilities of reporting one classical outcome when the system is in a different state. For a single-qubit measurement, this can be written as:

$$M = \begin{bmatrix} 1-p & q \\ p & 1-q \end{bmatrix}.$$

where p is the probability of misidentifying $|0\rangle$ as $|1\rangle$, and q is the reverse.

1.1.5.5 Impact of Errors on Quantum Computation

Quantum error manifestations are hardware-dependent and context-sensitive. This is particularly evident in resource-intensive algorithms, such as quantum Hamiltonian simulation, where long circuit depths are needed to approximate the evolution operator $U(t) = e^{-i\hat{H}t}$. Over time, accumulated errors from gate imperfections, decoherence, crosstalk, and measurement reduce fidelity and may render results indistinguishable from statistical noise.

Thus, practical quantum computation demands robust theoretical design and tight integration with hardware-aware compilation and error mitigation. Strategies such as noise-aware transpilation, pulse-level control, and variational error suppression are critical to pushing beyond the current limitations of noisy intermediate-scale quantum (NISQ) devices. Ultimately, the development of scalable, fault-tolerant quantum computing depends on our ability to mitigate and correct quantum errors at every layer of the quantum stack.

1.1.6 Hybrid Quantum-Classical Neural Networks

In the NISQ era, quantum devices are constrained by limited qubit counts and high error rates, which currently preclude the implementation of fully fault-tolerant quantum algorithms. Nevertheless, hybrid quantum-classical algorithms have emerged as a promising approach to extracting computational advantage from these imperfect devices. These algorithms iteratively optimize an objective function by distributing the computational workload between a quantum processor and a classical optimizer. In particular, hybrid algorithms form the foundation of many quantum machine learning (QML) models, where quantum circuits encode and process data. At the same time, classical routines guide the optimization of circuit parameters based on measured outputs.

A typical hybrid QML model involves a parameterized quantum circuit (PQC), called a variational quantum circuit (VQC), composed of tunable quantum gates. These gates are configured according to a set of parameters that define the circuit's behavior for a given input. After executing the quantum circuit, measurements are performed to estimate expectation values or probabilistic outcomes related to the problem. The classical computer then evaluates a cost or loss function based on the difference between predicted and target values. It employs an optimization algorithm (e.g., gradient descent or one of its quantum-aware variants) to update the circuit parameters. This feedback loop continues until convergence or a stopping criterion is met. A visualization of this is in Fig. 1.4

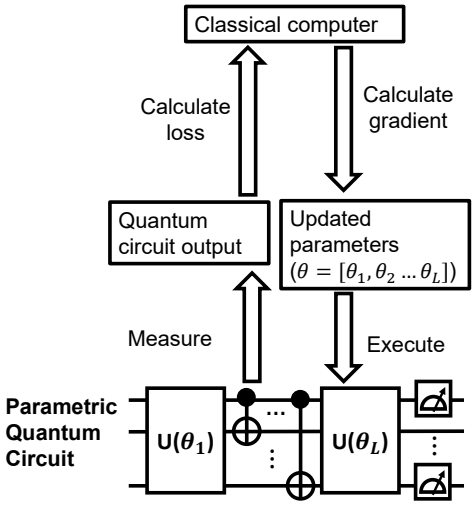


Figure 1.4: The standard model for hybrid-quantum classical machine learning. The parameter θ executes the quantum algorithm; a classical machine then calculates the loss/gradient and updates θ accordingly.

Hybrid quantum-classical approaches are especially valuable in the NISQ regime because

they reduce the quantum resource requirements by offloading a significant portion of the computational burden to classical hardware. Prominent examples of such algorithms include the Variational Quantum Eigensolver (VQE), Quantum Approximate Optimization Algorithm (QAOA), and quantum neural networks (QNNs), which are being explored for applications in chemistry, optimization, and machine learning. Despite their potential, these models remain sensitive to noise and hardware limitations, and ongoing research is focused on improving their robustness, scalability, and training efficiency.

1.2 Overview of Machine Learning

1.2.1 Neural Networks (NN)

Neural networks (NNs) are a class of machine learning models inspired by the structure and function of the human brain. They consist of individual computational units called neurons or perceptrons, which are organized into layers, typically including an input layer, one or more hidden layers, and an output layer. Each neuron is connected to neurons in the subsequent layer through weighted connections, where each weight is a learnable parameter that determines the strength and direction of the signal passed between neurons. A simple neural network example is in Fig. 1.5.

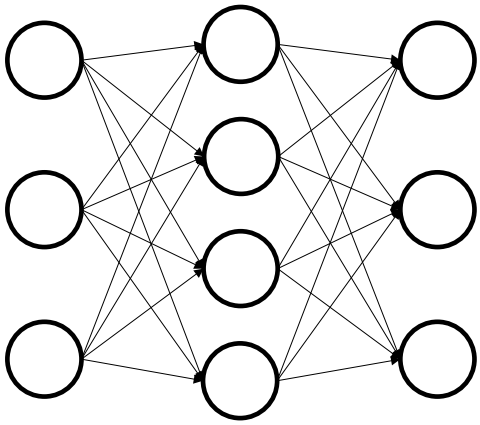


Figure 1.5: A simple neural network consisting of an input layer, hidden layer, and output layer

During the feedforward phase, input data is passed through the network layer by layer, with each neuron applying a nonlinear activation function to the weighted sum of its inputs. The output produced by the network is then compared to the expected result, and the discrepancy is quantified using a loss function. Common loss functions include mean squared error (MSE) for regression tasks and cross-entropy for classification tasks.

The network undergoes a process called backpropagation to minimize this loss and improve prediction accuracy. This involves computing the gradient of the loss function with respect to each weight in the network using the chain rule of calculus. The weights are then updated, typically using an optimization algorithm such as stochastic gradient descent (SGD) or one of its variants (e.g., Adam or RMSprop), to reduce the error in future predictions. Through iterative training on large datasets, neural networks can learn complex patterns and representations in data, making them highly effective for tasks such as image recognition, natural language processing, and time-series forecasting.

1.2.2 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) are a class of NN architectures designed explicitly for modeling sequential data. They build upon the foundational principles of NNs but introduce a critical enhancement: the ability to maintain and utilize temporal dependencies through internal memory. Unlike feedforward models, where inputs and outputs are treated as independent, RNNs incorporate feedback loops that allow information from previous time steps to influence the current output. This is achieved by passing the hidden state of a neuron from one time step to the next, effectively enabling the network to "remember" prior inputs. This is visualized in Fig. 1.6, where H_{t-1} contains the prior historical information and H_t contains the current historical information.

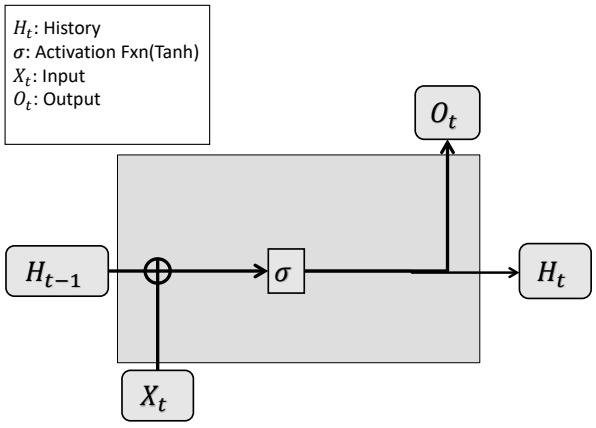


Figure 1.6: A vanilla recurrent neural network. X_t is the current input, H_{t-1} is the previous historical state. These are combined and passed through the activation function, σ, which creates the output, O_t, and a new hidden state, H_t.

This temporal dependency makes RNNs particularly well-suited for tasks in which the meaning or prediction of the current input is influenced by its context within a sequence.

Applications include time-series forecasting (e.g., stock market trends or weather prediction), physical system modeling, and various domains of natural language processing, such as machine translation, speech recognition, and sentiment analysis.

RNNs naturally support variable-length input sequences, making them flexible for real-world data where input sizes may not be fixed. They can be implemented as unidirectional models, which process sequences in a forward direction, or as bidirectional models, which process input in both forward and backward directions, allowing the network to consider both past and future context simultaneously.

Several variants of RNNs have been developed to address challenges such as vanishing and exploding gradients during training. The most widely used among these are the Elman Network (a basic RNN architecture), the Long Short-Term Memory (LSTM) network, and the Gated Recurrent Unit (GRU). Both LSTM and GRU architectures introduce gating mechanisms that regulate the flow of information through the network, enabling them to capture long-range dependencies and control how much past information is retained or forgotten at each time step, enhancing the model's ability to learn from sequences with complex temporal dependencies.

1.2.3 Graph Neural Networks

Graph Neural Networks (GNNs) are a powerful class of neural networks designed to operate on data naturally represented as a graph. Graphs are flexible data structures consisting of nodes (or vertices) and edges that encode pairwise relationships. They are well-suited to various domains, including electronic circuits, neural architectures, molecular structures, transportation systems, and social networks. Unlike traditional neural networks, which assume fixed-size input and Euclidean structure (e.g., grids in images), GNNs can directly leverage the non-Euclidean topology of graph-structured data.

The core computational mechanism behind most GNNs is message passing, a process through which nodes iteratively exchange information with their neighbors to update their internal representations (embeddings). In each message-passing step, a node aggregates information from its adjacent nodes and combines it with its current state, enabling it to build a richer, context-aware representation of its local neighborhood.

One of the most common implementations of this concept is found in Graph Convolutional Networks (GCNs), which perform simultaneous message passing across all nodes in the graph. During a single 1-hop message passing step, each node updates its representation based on the features of its immediate neighbors. An example of this process is in Fig. 1.7. This process can be extended to multiple hops—for example, 2-hop message passing allows a node to incorporate information from its neighbors' neighbors, thereby capturing a broader

structural context. Repeating this process over several layers enables the model to learn from increasingly distant parts of the graph. However, doing so must be balanced carefully to avoid over-smoothing, where node representations become indistinguishable.

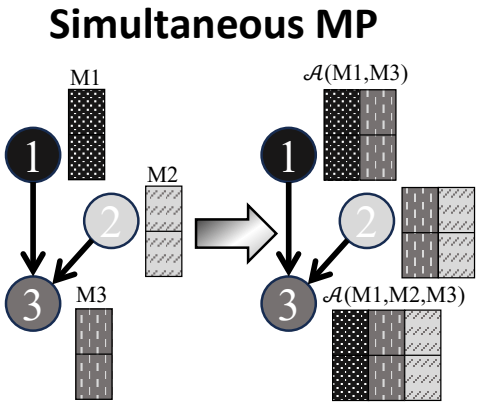


Figure 1.7: A simultaneous message passing scheme. The left side shows the initial state, where each node has its unique representation. After 1-hop message passing occurs, each node state is updated to account for the directly connected nodes.

GNNs typically comprise a stack of message-passing layers, often followed by pooling and readout operations to generate graph-level representations. These models are trained end-to-end using task-specific loss functions, allowing them to learn effective representations for various tasks such as node classification, link prediction, and graph classification.

1.3 Motivation

The core challenge in quantum computing lies in the efficient design and optimization of quantum circuits. Traditional circuit optimization methods are conservative and reliant on expert knowledge, employing local, greedy techniques that reduce gate counts only when exact functional equivalence can be guaranteed. While effective, this approach limits the potential for discovering more efficient circuit configurations that may initially appear non-equivalent but ultimately preserve functionality. In contrast, generative models provide a more exploratory framework for circuit transformation, enabling the identification of novel, functionally equivalent circuits with reduced gate counts and depths, as well as key metrics for the feasibility and performance of quantum computations on noisy intermediate-scale quantum (NISQ) devices.

Recent progress in graph-based generative models, particularly those using Variational Autoencoders (VAEs) trained on directed acyclic graph (DAG) representations, has demonstrated

significant advantages over non-graph-based counterparts, especially in domains such as neural architecture search. These models can capture the structural intricacies of the data more effectively and produce output architectures that closely align with the design principles of the input space. Compared to reinforcement learning (RL) techniques, VAEs also exhibit a higher degree of exploratory capability, enabling the generation of a broader range of potential solutions. Motivated by these advancements, we adopt a similar approach to the generation and optimization of quantum circuits, aiming to automate the discovery of low-depth, low-gate-count designs with equivalent functionality.

Manual quantum circuit construction remains a significant bottleneck, particularly for applications such as QML, where designing appropriate ansatz (heuristic quantum circuits) is crucial to performance. Given the fragile nature of NISQ hardware and the high barrier to entry for quantum algorithm design, automation becomes essential for advancing QML. Current approaches for quantum circuit generation include RL, large language models (LLMs), and VAEs. While RL methods often struggle with exploratory diversity and LLMs, despite their generative prowess, lack an inherent understanding of circuit structure, VAEs can capture circuit details through graph representations. More recently, diffusion models have emerged as a promising alternative, combining the structural advantages of graph-based representations with improved training stability and output quality. These models offer a powerful new avenue for generating novel and efficient quantum circuits.

Despite the limitations in the NISQ era, characterized by limited qubit counts, high noise levels, and limited quantum circuits, hybrid quantum-classical approaches such as QML offer a promising path toward quantum supremacy. Classical machine learning has demonstrated capabilities that surpass those of humans in various domains, such as image recognition. Domains like these are likely to see no improvement when attempting to leverage QML because QML suffers from something known as barren plateaus. When a problem is too simple for a QNN, it cannot perform proper parameter tuning due to the curse of dimensionality. However, there are many complex domains where significant improvement is still possible. One of which is the field of material discovery. In particular, the processes of retrosynthesis and molecular property identification. Where textual inputs such as SMILES and SELFIES encodings provide structured representations of chemical compounds. While convolutional networks and linear layers can process text-based inputs, it is usually better to use an RNN-based design or a transformer. Models such as the Quantum Long Short-Term Memory (QLSTM), Quantum Kernel-based LSTM (QK-LSTM), and Quantum Transformers have demonstrated potential in handling sequential textual data. However, the runtime cost of running a quantum transformer is almost three orders of magnitude higher for even just a 4-qubit simulation.

The rapid acceleration of molecular discovery, especially in materials science, has created a pressing need for more efficient and accurate computational models to aid in the design and validation of new compounds. Among the most critical challenges is the early identification of side effects and undesirable properties, a costly and risk-laden process in the traditional pipeline. Although classical machine learning models have made progress in this area, they often require large-scale computational resources and are susceptible to overfitting or poor generalization across diverse chemical spaces. These limitations constrain their utility and accessibility, particularly for exploratory research.

In light of these challenges and opportunities, this dissertation explores the intersection of quantum circuit optimization and quantum machine learning to address key limitations in current methodologies. By leveraging advanced generative models, particularly graph-based VAEs and emerging diffusion models, this work aims to automate and enhance quantum circuit design, improving efficiency and scalability for near-term quantum devices. Simultaneously, it investigates the application of quantum learning architectures to complex, high-impact domains such as materials discovery, where classical approaches struggle to generalize or scale effectively. The overarching goal is to contribute foundational methods that reduce the barriers to practical quantum algorithm design and accelerate scientific discovery in chemistry and beyond.

1.4 Thesis Contribution

This thesis contributes to the growing fields of drug discovery and quantum computing by exploring and developing hybrid quantum-classical algorithms to address key challenges in molecular property prediction and quantum circuit design. The contributions of this thesis span theoretical development, algorithmic innovation, and empirical validation and are summarized as follows:

1. *AltGraph: Generative Graph Models for Quantum Circuit Optimization*: A generative modeling framework that optimizes quantum circuits for NISQ hardware constraints. Utilizing DAG-based Variational Autoencoders (D-VAE) and the DeepGMG framework, AltGraph generates functionally equivalent circuits with significant reductions in gate count (37.55%) and depth (37.75%), while maintaining functional fidelity (MSE of 0.0074). This surpasses prior rule-based, and RL approaches in circuit transformation.

2. *Diffusion-Based Quantum Architecture Search for Automatic Circuit Design*: A novel diffusion-based algorithm for Quantum Architecture Search (QAS) within the LayerDAG

framework, enabling the automated synthesis of valid quantum circuits. Unlike existing RL- or LLM-based models, the proposed method consistently generates 100% syntactically valid circuits, significantly reducing the time and expertise needed for quantum algorithm design.

3. *Application of Quantum Recurrent Neural Networks to Chemical Retrosynthesis*: This chapter presents the first application of Quantum Long Short-Term Memory (QLSTM) networks for the problem of chemical retrosynthesis, a fundamental task in material discovery. Unlike classical LSTMs, the QLSTM model demonstrated superior generalization capabilities in substructure identification, achieving higher testing accuracy (80%) despite lower training accuracy. This indicates that QLSTM networks are more robust in avoiding overfitting and better suited for real-world generalization in chemical sub-structure prediction.

4. *Lightweight GRU-Based Model for Molecular Side Effect Prediction*: A novel, heuristic-based method was developed to identify molecular properties, particularly adverse side effects, using a parameter-efficient GRU architecture. This approach reduces model complexity by over 98% relative to large-scale language models while maintaining competitive performance. This contribution makes side effect prediction more accessible for research environments with limited computational resources, democratizing early-stage drug safety screening.

5. *Hybrid Quantum-Classical Models for Molecular Property Prediction*: The Quantum Kernel-based LSTM (QK-LSTM) is introduced to blend the representational power of quantum kernel methods with classical sequence modeling. This hybrid model shows promise for molecular property prediction tasks by offering compression benefits without compromising performance. This is the first study to explore augmented SMILES and SELFIES representations in both classical and hybrid quantum-classical settings, with SELFIES augmentation leading to statistically significant improvements (5.97% in classical, 5.91% in hybrid models).

Collectively, these contributions push the boundaries of what is currently achievable with molecular property prediction and quantum computers, with a strong focus on real-world impact in computational chemistry and quantum algorithms. This thesis lays foundational steps toward more generalizable, and scalable QML systems and opens new avenues for future research in quantum-enhanced scientific discovery.

1.5 Thesis Organization

We organize the rest of the dissertation as follows. Chapter 2 evaluates the potential for using generative graph models to perform quantum circuit transformation, specifically targeting optimized quantum circuits. It compares the results to common approaches and discusses future research, potential directions, and limitations with the approach. In Chapter 3, the thesis examines the role of graph diffusion in designing new quantum circuits, detailing the methodology, design choices, datasets, and future limitations. Chapter 4 offers a detailed comparison between classical Long Short-Term Memory (LSTM) models and hybrid quantum-classical LSTM (QLSTM) models. It highlights the potential of quantum models and how they may be able to replace classical ones in the near future. Chapter 5 presents a comprehensive analysis of the advantages of using task-aware design techniques, specifically targeting molecular property identification. It also discusses constraints, ethical concerns, and clinical insights that can be gained. The study of Quantum-kernel LSTM (QK-LSTM) for molecular property prediction and the first-ever evaluation of augmented SELFIES is explored in Chapter 6. Finally, the Concluding Remarks in Chapter 7 summarize the key findings of the thesis, highlight its contributions to the field, and outline potential directions for future research.

Chapter 2
AltGraph: Redesigning Quantum Circuits Using Generative Graph Models for Efficient Optimization

Quantum circuit transformation aims to optimize circuits for depth, gate count, and compatibility with Noisy Intermediate Scale Quantum (NISQ) devices that suffer from various error sources. Prior methods use combinations of expert-defined rules and Reinforcement Learning (RL). We introduce *AltGraph*, a novel approach employing generative graph models to generate functionally equivalent quantum circuits using—specifically, Direct Acyclic Graph (DAG) Variational Autoencoder (D-VAE) variants (GRU and GCN) and Deep Generative Model for Graphs (DeepGMG). AltGraph perturbs the latent space to generate quantum circuits optimized for hardware coupling maps, reducing gate count by 37.55% and circuit depth by 37.75% post-transpiling, with 0.0074 Mean Squared Error (MSE) in the density matrix—outperforming state-of-the-art methods by 2.56%.

2.1 Introduction

Quantum circuits are the fundamental component of modern-day quantum computing. They are analogous to classical electronic circuits, where various classical (quantum) gate operations are performed on bits (qubits) to obtain the desired output. Typically, each quantum circuit has a set of quantum gate operations followed by a measurement operation on desired qubits at the end. Special quantum circuits, such as Parametric Quantum Circuits (PQC), also have a state preparation component at the start to prepare the initial state of qubits. Each quantum gate is a unitary operation that can be represented as unitary matrix U (for conjugate transpose U^*, $UU^* = UU^{-1} = I$) and works on either a single qubit or multiple qubits. A few well-known

single qubit gates are Hadamard (H), Pauli X/Y/Z gates, rotation gates like RX/RY/RZ and U, and multi-qubit gates are CNOT, Toffoli, controlled Pauli and controlled rotation gates.

Ideal quantum circuits with no noise produce correct output with 100% fidelity. Unfortunately, NISQ computers have gate errors, readout errors, decoherence errors, crosstalk errors, and other noise sources that degrade the overall fidelity and correctness of computation. In such noisy environments, gate count and circuit depth are salient when crafting a circuit. Typically, circuits with low depths and gate counts are ideal for maximum fidelity. Therefore, generating equivalent quantum circuits to obtain lower-gate functionally equivalent quantum circuits is a fundamental research problem in the NISQ era.

There are two methods of performing *quantum circuit transformation* (referred to as *circuit transformation* for brevity) [7] namely, *rule-based* and *search-based*. The former uses gate-cancellation rules to optimize quantum circuits [8], while the latter uses machine learning techniques to find functionally equivalent quantum circuits from the search space of quantum circuits [7, 9–12]. These techniques primarily use Reinforcement Learning (RL) methods e.g., [13] uses an RL agent to determine the correct set of gate cancellations to reach optimal depth on ICM+H circuits. Since ICM+H circuits contain only CNOT and Hadamard gates which are easy to optimize, this technique may not necessarily work on quantum circuits involving other gates. [10] overcomes this issue by using parametric quantum circuits but require circuits to be hardware compliant prior to performing optimization. [11] uses generative methods by generating circuits as Quantum Assembly Language (QASM) instructions, however the search space of the generation algorithm is limited and also does not sample new circuit representations that may fundamentally address the issues like hardware constraint. We address these shortcomings by proposing *AltGraph*: usage of generative graph models that produce an equivalent Directed Acyclic Graph (DAG) of the original quantum circuit DAG from which we reconstruct the corresponding quantum circuit. We perturb the latent space of the models to generate equivalent quantum circuits some of which may be more compatible with the hardware coupling map and/or enable better optimization leading to compact circuits. We use multiple graph generation models, specifically DAG Variational Autoencoder (D-VAE) Gated Recurrent Unit (GRU), D-VAE Graph Convolutional Network (GCN) and Deep Generative Model for Graphs (DeepGMG) and perform comparative analysis.

Novelty: The D-VAE and DeepGMG models were able to process DAGs and generate stylistically similar graphs as the original DAGs. However, there was no way to guarantee a reconstructed DAG could be interpreted as a valid quantum circuit. We extend D-VAE and DeepGMG's graph reconstruction capabilities to generate always valid quantum circuits. Furthermore, we enable the capability to generate novel quantum circuits equivalent to the

original quantum circuit by sampling from the latent space of the generative models. Finally, we perform detailed comparative analysis of the effectiveness of the generative models in producing compact representation of the original circuits.*To the best of our knowledge, this is the first attempt to use generative graph model-based circuit transformation to produce optimized quantum circuits.*

The rest of the chapter's organization is as follows: Section 2.2 illustrates the background information required to understand the chapter and related work. Section 2.3 presents the approach to convert the models to valid quantum circuit graph generators. Section 2.4 contains visualizations of each model's abilities to process quantum circuits. Section 2.5 analyzes the results, and Section 2.6 summarizes the work.

2.2 Background and Related Works

2.2.1 Fundamentals of Quantum Computing

Please refer to Section 1.1.1 for a description of qubits, Section 1.1.2 for an introduction to quantum gates, and Section 1.1.3 for a description of quantum circuits.

2.2.1.1 Transpilation

Before running a quantum circuit, more complex gates, like Tofolli, multi-controlled NOT gate, etc. are broken into sets of gates that the quantum device natively understands. Converting gates to hardware native gates is known as *decomposition*. NISQ devices have restricted connectivity i.e. not every qubit is connected to every other qubit. Physical qubit connections are represented in a graph called a coupling map. Because not all qubits are connected, compilation requires mapping logical qubits to physical qubits, termed as *qubit mapping*. Decomposition combined with qubit mapping is formally known as transpilation. If a gate operation is required between qubits that are not connected, swap operations will be required to bring them to physically connected qubits. Each swap operation introduces 3 CNOT gates which increases the circuit depth and gate count.

2.2.2 Quantum Circuit Transformation

There are two types of circuit transformation: rule-based approach and search-based approach. A rule-based approach involves utilizing gate-cancellation rules on particular subcircuits to reduce them into smaller amounts of gates with lower depths. For example, two back-to-

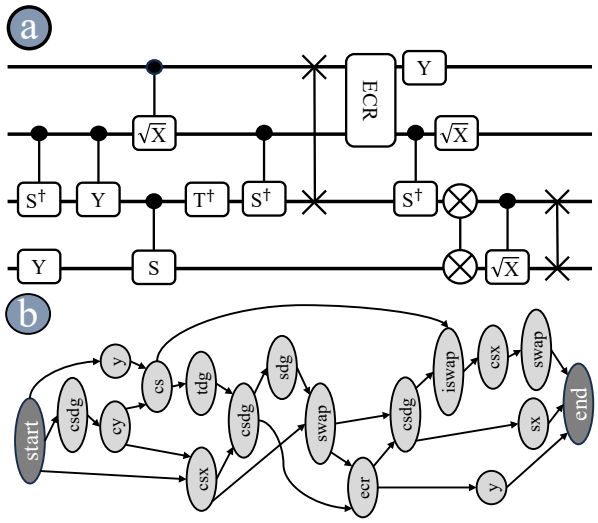


Figure 2.1: (a) Sample 4 qubit 16 gate quantum circuit with (b) the DAG rendering of the circuit.

back Pauli X gates computationally equate to identity operation (I) as $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Similar behavior exists for other single-qubit gates and multi-qubit gates. This kind of circuit transformation is typically incorporated in quantum compilers like Qiskit [8]. Search-based approaches offer flexibility in terms of optimization. Rather than relying on gate-cancellation rules, this approach searches for a functionally equivalent circuit. Typically, Machine Learning (ML) techniques such as Reinforcement Learning (RL), generative models, or other methods aid the approach. In RL-based search, the current quantum circuit is the state, and the original version of the circuit is the environment. The circuit transformation performed in each step is the action, and based on some input, the agent performs the most rewarding action. Some RL-based works include ① [13], that uses D-VAE to encode a quantum circuit DAG and send that encoding to an RL agent which then decides what action to perform among the possible set of updates, ② [7], that uses Hierarchical Advantage Learning (HAL) combined with Proximal Policy Optimization (PPO) [9] to perform reinforcement learning, and ③ [10] that also uses PPO along with Advantage Actor Critique (AAC) scheme-based agent. In contrast, generative approaches such as [11] generate quantum circuits as a Quantum Assembly Language (QASM) sequence of instructions. Other methods include ① [12] uses Monte Carlo tree search framework to generate hardware compliant quantum circuits, and ② [14] performs ZX-calculus on graphs to perform circuit transformation.

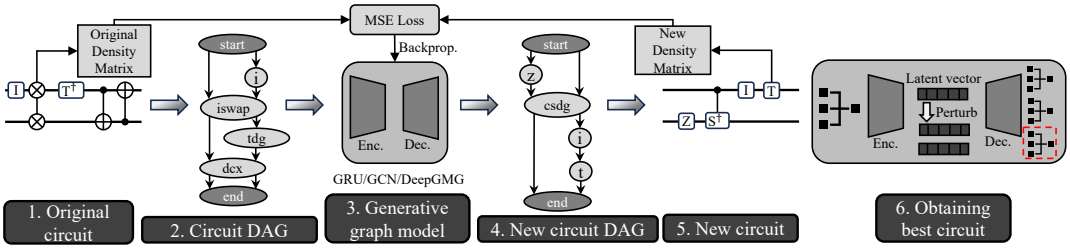


Figure 2.2: Overview of AltGraph process for generating alternate representation of quantum circuits.

2.2.3 Message Passing

Generative graph models and graph neural networks, in general, require nodes to share information to build either a global or a local understanding of themselves and the nodes around them in hopes of generating a (semi-)unique representation. While there are multiple techniques, only two approaches exist in D-VAE and DeepGMG: simultaneous message passing and asynchronous message passing.

2.2.3.1 Simultaneous

The more common approach in graph neural networks, specifically Graph Convolution Networks (GCN), is simultaneous message passing. In D-VAE, simultaneous message passing starts by taking the initial type of each node and sharing it with all its immediate neighbors (downstream neighbors if uni-directional). These messages are then combined to create a new localized representation of each node based on its type and its neighbors. The localization extends to additional neighbors by iteratively repeating this process.

2.2.3.2 D-VAE Asynchronous

D-VAE proposes an “asynchronous" message-passing scheme. This name may seem misleading since D-VAE does require some synchronization. Analysis of the first few gates of Fig. 2.1b is done for clarity. At the top is the initial or the “start" node. Once the representation of this node is created the children begin to process, however not all three children can process immediately. D-VAE requires processing all predecessor nodes before updating a node’s state. So, only the “csdg" and the “y" gate may asynchronously process. The “csx" will not be active until the completion of the “cy" node following “csdg". While the synchronization is time-consuming, it allows each node a “complete" understanding of all predecessor nodes.

2.2.4 Generative Graph Models

Generative graph models create new equivalent models based on input graphs and various constraints. D-VAE encodes a DAG to the latent space, then the encoding is used to obtain reconstructed DAGs. For detailed analysis, different variations of the D-VAE generative graph model [15] are used. Namely, D-VAE GRU, D-VAE GCN, and D-VAE DeepGMG [16]. All three versions use topological ordering to create their DAGs, differing mainly in the encoding method.

2.2.4.1 D-VAE GRU

The standard D-VAE approach uses the Gated Recurrent Unit (GRU), which encodes the current node's state using the node's predecessors' states and the current node type. D-VAE uses asynchronous message passing [17] and topological sorting to ensure all information is available to the node before encoding. D-VAE defaults to bi-directional encoding, where upon completion of forward encoding D-VAE reverses the graph and the node states are updated by repeating the same process.

2.2.4.2 D-VAE GCN

The second D-VAE approach uses GCN. D-VAE allows the designer to decide how many rounds of simultaneous message passing [18] they desire to create the encodings of the nodes. To initialize the encoding D-VAE uses the neighbor (or predecessors if uni-directional) node types to initialize each node's state. In additional rounds of encoding neighbor (or predecessor) node states are used to update the state. Standard GCN models are primarily for encoding, so D-VAE applies the GRU method for GCN decoding.

2.2.4.3 D-VAE DeepGMG

The third D-VAE approach uses DeepGMG [16]. The original DeepGMG work focuses on undirected graph generation. It also does not use topological ordering and uses a random order instead. To make a more valid comparison, D-VAE updates the design of DeepGMG to use topological ordering, allowing for multiple iterations of message passing and the processing of directed graphs. DeepGMG relies on a similar simultaneous message-passing scheme as GCN with an important update. DeepGMG, instead of only relying on surrounding node information, also accounts for an edge feature vector to better understand node connections.

2.3 AltGraph: Proposed Approach

2.3.1 Approach

To leverage the generative abilities of D-VAE for the quantum domain, we propose AltGraph, a robust approach to generate valid equivalent quantum circuits. In quantum graph representation, nodes represent quantum gates in a system, and edges represent the connection between the gates. We discuss the changes required in original D-VAE to realize AltGraph in the following paragraphs.

The overall process of generating quantum circuits using AltGraph is shown in Fig. 2.2. First, the original quantum circuits (step ①) are converted into DAGs (step ②). These DAGs are sent to the generative graph model (step ③) for reconstructing new DAGs (step ④), from which the new quantum circuit is reconstructed (step ⑤). For training, we use MSE Loss between the density matrices of original and new quantum circuits to ensure similar functionality. Finally, to obtain the best quantum circuit, we perturb the latent representation of the circuit with some noise (step ⑥) and obtain multiple candidate circuits. From these, one can select the circuit with matching density matrix and least depth and gate count.

2.3.2 Loss

D-VAE used teacher forcing to get faster convergence of models. For loss evaluation, D-VAE used encoder KLD and the model's ability to predict the correct node. D-VAE also evaluated the loss based on the model's ability to predict edges using the "true" node. While these features are viable when dealing with graphs and ensuring similarly designed graphs, the goal is to generate equivalent quantum circuits. Structurally similar graphs are not a guarantee of equivalent quantum circuits. It is necessary to compare density matrices to evaluate quantum circuit equivalence. The loss process is updated to generate the nodes and edges rather than forcing edge predictions based on the "true" node. This forces complete graph reconstruction for each input, allowing density matrix comparison. The density matrices are taken from the target hardware post-transpilation using Qiskit's default optimization level 1. This ensures the circuits are compared based on their real values rather than the ideal state. AltGraph's loss calculation uses the KLD of the encoder, the mean squared error (MSE) of the original circuit's density matrix, and the reconstructed circuit's density matrix.

2.3.3 Graph Generation

D-VAE requires that each graph has a single input node and a single output node. In cases with multiple input/output nodes, D-VAE requires a virtual node to represent the final input/output. To match D-VAE requirements, input qubit wires map to the start node, and all qubit measurements map to the end node. An example mapping is in Fig. 2.1.

Quantum circuits have numerous requirements to ensure validity. Ideally, a neural network could learn the basic requirements of quantum circuit structures. However, the loss calculations use the difference between the density matrices of the original and reconstructed circuit, so even the initial circuits need to be valid constructed designs.

2.3.3.1 Number of Qubits

The first requirement for density matrix comparison is that both density matrices are the same size. Density matrices are size $2^N * 2^N$ where N is the number of qubits. To ensure reconstructed circuit size equivalence, the reconstructed DAG start node out-degree must match the out-degree of the original DAG's start node. The reconstructed DAG end node must have the same in-degree as the original DAG's end node.

2.3.3.2 Gates & Gate Connections

D-VAE uses a topological ordering for both encoding and reconstruction. During reconstruction, it allows for the selection of edges between the current node and any previous node. D-VAE also allows the model to learn the appropriate amount of in and out edges. However, quantum gates have a specified in and out edge limitation. To ensure the use of all out-edges, the model maintains a queue of available nodes for the current node to connect. The node is removed from the queue when the out-edges are complete. When the model predicts the "end" node it connects all remaining nodes in the queue. The model predicts the most likely connection from the available list of previous nodes for the in edge. If it is a multiple-wire gate, the prediction process repeats until all in-degrees are filled.

2.3.3.3 Circuit Types

To reduce the search space complexity of quantum circuits training/testing uses graphs made of non-parametric circuits that consist of one and two-qubit gates. Specifically, the single qubit gates *x*, *y*, *z*, *h*, *s*, *t*, *id*, *sxdg*, *sdg*, *sx*, *tdg* and *cx*, *cy*, *cz*, *swap*, *dcx*, *iswap*, *csdg*, *ecr*, *ch*, *cs*, *csx* two-qubit gates are used.

Table 2.1: Average gate reduction (%) and loss per model per gate count.

	2 Qubit		4 Qubit		6 Qubit	
Model Type #Gate	**Loss**	**Reduction**	**Loss**	**Reduction**	**Loss**	**Reduction**
GRU 16	0.09	19.59%	0.01	24.63%	0.01	30.18%
GRU 24	0.09	33.94%	0.01	43.86%	0.01	43.96%
GRU 32	0.10	36.27%	0.01	51.37%	0.01	54.16%
GCN 16	0.22	19.83%	0.12	26.17%	0.11	32.29%
GCN 24	0.31	33.02%	0.22	41.09%	0.19	41.99%
GCN 32	0.54	35.34%	0.44	51.74%	0.43	53.1%
DeepGMG 16	0.14	19.57%	0.03	23.67%	0.06	30.34%
DeepGMG 24	0.18	31.99%	0.10	31.71%	0.08	42.6%
DeepGMG 32	0.28	40.77%	0.11	40.44%	0.12	51.50%

2.4 AltGraph Results

2.4.1 Setup and Evaluation Framework

In generating equivalent quantum circuits, the maximum gate predictions are capped at the original circuit gate count. This can yield higher loss values but penalizes models for larger circuits, aligning with the aim of more hardware-friendly circuits. AltGraph's performance is assessed with qubit counts of 2, 4, and 6 and gate counts of 16, 24, and 32 per qubit. A total of 300 random circuits for each size are produced using gates from Section 2.3.3.3. The data follows a 90-10 training/testing split. The same 300 samples are used for all models, ensuring consistency. Each model is trained thrice per circuit size, and reported values represent the average outcomes.

2.4.2 Results and Analysis

Table 2.1 presents the mean training loss and gate reduction percentages for three models: AltGraph GRU, AltGraph GCN, and AltGraph DeepGMG across various circuit sizes. These percentage reductions are derived from comparing the number of gates between the transpiled original and the transpiled reconstructed circuits.

The GRU model's loss tends towards zero faster than both the GCN and DeepGMG models, irrespective of the qubit and gate counts. Moreover, DeepGMG's loss approaches zero faster than GCN. Interestingly, despite GCN's generally higher loss, it surpasses both GRU and DeepGMG when circuits have 16 gates, irrespective of the qubit count. Conversely, for circuits with 24 gates, GRU is superior, irrespective of qubit count. For 32-gate circuits, performance varies by qubit count: DeepGMG excels for 2-qubit circuits, GCN for 4-qubit circuits, and GRU for 6-qubit circuits.

Delving deeper into each model's capability to yield a circuit with fewer gates post-transpilation provides further insights into their efficiencies. This analysis involves three

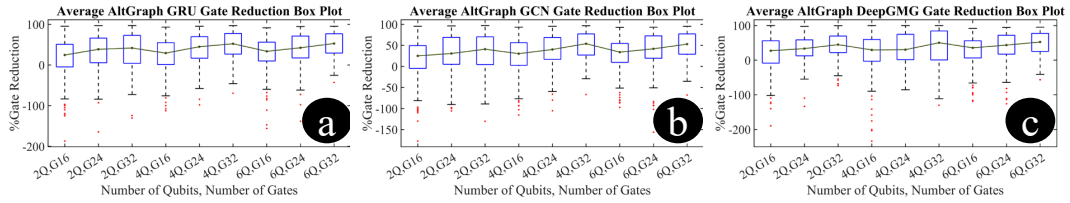


Figure 2.3: Averaged gate reduction per circuit size for (a) AltGraph GRU, (b) AltGraph GCN,(c) AltGraph DeepGMG.

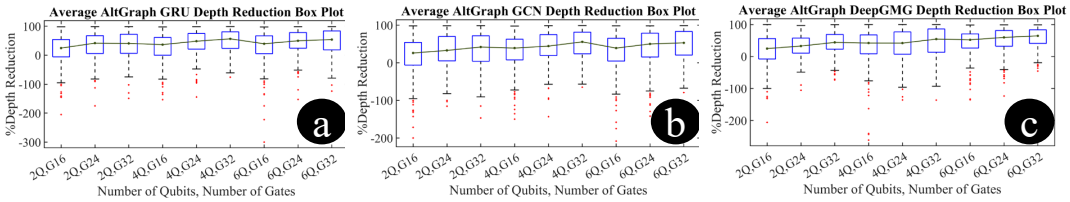


Figure 2.4: Averaged depth reduction per circuit size for (a) AltGraph GRU, (b) AltGraph GCN, (c) AltGraph DeepGMG.

different encodings for each testing gate and three decodings for each encoding sample, resulting in nine reconstructions per circuit. Upon generating these samples, both the reconstructed and original circuits are transpiled using Qiskit's *FakeWashingtonV2* to emulate performance on cutting-edge hardware. Fig. 2.3a showcases the mean gate reduction for the AltGraph GRU model per transpiled circuit size.

2.4.2.1 Gate Reduction Trends Across Models

From Fig. 2.3a, a consistent positive gate reduction is observed across different qubits and gate configurations. Median gate reductions range from 24% (2 qubits, 16 gates) to 53% (6 qubits, 32 gates). As the gate count increases within each qubit category, there is a proportional increase in gate reduction percentages. Specifically, for 6 qubit circuits with 16, 24, and 32 gates, the median reductions stand at 33.6%, 42.4%, and 52.9%, respectively. Comparable patterns are seen for 2 and 4-qubit circuits. There are few reconstructions that result in an increased gate count, like a 4 qubit, 16 gate circuit showing a surge of 73%. However, such cases are exceptions, as Fig. 2.3a indicates that most alternative configurations yield a reduction in gate count.

Trends seen in GCN and DeepGMG match those of GRU, as demonstrated in Figs. 2.3b and 2.3c. GCN's median gate reduction spans from 25% (2 qubits, 16 gates) to 54% (4 qubits, 32 gates), and DeepGMG's range is from 27% (2 qubits, 16 gates) to 52.4% (6 qubits, 32 gates). For both models, as the number of gates grows, gate reduction percentages correspondingly increase. Comparatively: ① For 2-qubit circuits, DeepGMG achieves the peak in median

Table 2.2: Average gate reduction (%) per model per gate count

Model	16 Gate	24 Gate	32 Gate
AltGraph GRU	24.80%	40.59%	47.27%
AltGraph GCN	26.10%	38.70%	46.73%
AltGraph DeepGMG	24.53%	35.44%	44.24%

gate reduction, followed by GRU and then GCN, ② in 4-qubit circuits, GCN leads in median gate reduction, with GRU second and DeepGMG last, and ③ for 6-qubit circuits, all models display similar median reductions, though DeepGMG has an advantage.

2.4.2.2 Depth Reduction Trends Across Models

From Fig. 2.4a, there is a clear and consistent depth reduction across different configurations of qubits and gates. The median depth reduction spans from 24% for circuits with 2 qubits and 16 gates to 56% for those with 4 qubits and 32 gates. Intriguingly, circuits with an identical qubit count display an enhanced depth percentage reduction with the addition of more gates. For 6-qubit circuits, the median depth reductions are 39.6% for 16 gates, 49.9% for 24 gates, and 53.9% for 32 gates. This pattern is consistent for 2 and 4-qubit circuits as well. While there are rare quantum circuits that show an increased depth post-reconstruction, such as a 4-qubit, 16-gate circuit with a 100% depth increase, these cases are rare, predominantly outliers. As illustrated in Fig. 2.4a, the dominant trend is a depth reduction in the alternative representations.

GCN and DeepGMG trends reflect those of GRU, as observed in Figs. 2.4b and 2.4c. GCN's median depth reduction ranges from 26% (2 qubits, 16 gates) to 55% (4 qubits, 32 gates), while for DeepGMG, it is between 24% (2 qubits, 16 gates) and 58% (6 qubits, 32 gates). As gate counts rise for every qubit configuration, depth reduction percentages also climb for both GCN and DeepGMG. Comparative analysis of the models reveals: ① For 2-qubit circuits, GRU tops in median depth reduction, succeeded by DeepGMG and then GCN. ② With 4-qubit circuits, GRU retains the lead in median depth reduction, followed by GCN and subsequently DeepGMG. ③ In 6-qubit circuits, DeepGMG stands out with the highest median depth reduction, with GCN in the second spot and GRU trailing.

2.4.2.3 Comparative Analysis of Model Performances

① From Table 2.2, AltGraph GRU stands out in its gate count reduction capabilities when compared to both GCN and DeepGMG for circuits with 24 and 32 gates. However, for 16-gate circuits, the GCN model surpasses, showing a 5.11% performance difference over the GRU model. The cumulative gate reduction performance of each model is detailed in Table

Table 2.3: Average gate reduction per model

Model	Average Gates Reduced	Average Reduction%
AltGraph GRU	35.67	37.55
AltGraph GCN	33.78	37.17
AltGraph DeepGMG	32.87	34.73
Qiskit	26.15	18.65
t$\|ket\rangle$	73.68	33.5
Quartz [11]	–	30.1
Quarl [7]	–	36.6
Monte Carlo Tree [12]	–	30.0

Table 2.4: Average depth reduction and test circuit MSE per model

Model	Average Depth Reduction%	Average MSE
AltGraph GRU	37.75	.0074
AltGraph GCN	35.54	.27
AltGraph DeepGMG	40.08	.13

2.3. ② Upon inspecting Table 2.3, it is evident that the AltGraph GRU, on average, trims more gates than either the GCN or DeepGMG models. Yet, GCN remains a strong contender, lagging behind GRU by a mere 1.02%. To understand how AltGraph performs compared to state-of-the-art systems we include the results from optimizing the same circuits using two rules based methods, t$|ket\rangle$ and Qiskit. We also include the reported average improvements of 3 state-of-the-art systems ran on IBM Qiskit non-parametric circuits. [9, 11, 12].

The collective depth reduction performance for each model is compiled in Table 2.4. ③ A closer look at Table 2.4 reveals that the AltGraph DeepGMG consistently outperforms in depth reduction when set against the GCN and GRU models. Nevertheless, the GRU model is not far behind, showing only a slight 5.98% performance gap to the DeepGMG. For a comprehensive view of the average MSE for test circuits, refer to the last column of Table 2.4. ④ Finally, analyzing Table 2.4, it becomes clear that AltGraph GRU excels in achieving a smaller density matrix difference than both the GCN and DeepGMG models.

2.5 Discussion

2.5.1 Mixed Optimization Techniques

Drawing from Table 2.2, it is evident that graph generative models present a promising avenue for gate reductions across different circuit dimensions. However, there is scope for refining the models further. The circuits used are randomly generated and did not undergo optimization using conventional rule-based methodologies. If models were trained on optimized circuits, it is plausible that there would be a more significant overall reduction between original and reconstructed circuits, as the model would target only substitutions not already addressed.

Implementing rule-based optimization post-circuit reconstruction could yield added reductions.

2.5.2 MSE Constraints

Despite the models exhibiting robust learning capabilities, using MSELoss as an evaluative metric has its drawbacks. While MSELoss serves well in gauging matrix similarity, there are pitfalls in its application. Being a regressive optimizer, MSELoss does not always ensure the discovery of the best solution. In the quantum realm, this can translate to functional discrepancies. For instance, if a model, during training, predicts a Pauli-X gate ($\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$) for an original Pauli-Z gate ($\left[\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right]$) , a high loss results. But if the model later predicts an S gate ($\left[\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right]$) , the resulting lower loss might misleadingly suggest an "optimized" circuit, which is not functionally accurate. A deeper probe into the relationship between a low MSELoss and a model's functional accuracy is warranted.

2.5.3 Density Matrix Constraints

Another challenge with MSELoss arises with sparse density matrices. In cases with notably high qubit counts and minimal gate counts, MSELoss might fall short. For illustration, consider a 16 qubit circuit with ($2^{16} * 2^{16} = 4,294,967,296$) values that possess only a single qubit gate. The MSELoss for a circuit with any such gate would be nearly zero. Yet, this limitation can be overlooked as the primary objective is to optimize large, intricate circuits, rendering sparse configurations, like the one mentioned, irrelevant to the study's context. Scalability to larger qubit counts initially seems problematic, as density matrices suffer from Nth power scaling. However, scaling can be addressed by performing block replacements of sub-circuits. For example, if there is a 100-qubit quantum circuit, the density matrix will have 2^{100} elements, which will be too large to process. However, one can break it down into blocks of five qubit sub-circuits and generate alternate representations for each sub-circuit one by one.

2.6 Conclusions

In our study, generative graph models have proven to be a promising method for gate reduction in quantum circuits. The AltGraph GRU model showcased an average gate count reduction of 37.55% which is a 2.56% improvement over the reported state of the art. This is coupled with a depth reduction of 37.75%, and achieved an impressively low average MSE of 0.0074. The AltGraph GCN model delivered a comparable gate count reduction of 37.17%, a depth reduction of 35.54%, but with a slightly higher average MSE of 0.27. Finally, the AltGraph

DeepGMG model, while averaging a gate reduction of 34.73% and achieving the highest depth reduction of 40.08%, reported an average MSE of 0.13. These findings underscore the potential and versatility of generative graph models in the realm of quantum circuit optimization.

Chapter 3
Q-Fusion: Diffusing Quantum Circuits

Quantum computing holds great potential for solving socially relevant and computationally complex problems. Furthermore, quantum machine learning (QML) promises to rapidly improve our current machine learning capabilities. However, current noisy intermediate-scale quantum (NISQ) devices are constrained by limitations in the number of qubits and gate counts, which hinder their full capabilities. Furthermore, the design of quantum algorithms remains a laborious task, requiring significant domain expertise and time. Quantum Architecture Search (QAS) aims to streamline this process by automatically generating novel quantum circuits, reducing the need for manual intervention. In this chapter, we propose a diffusion-based algorithm leveraging the LayerDAG framework [19] to generate new quantum circuits. This method contrasts with other approaches that utilize large language models (LLMs), reinforcement learning (RL), variational autoencoders (VAE), and similar techniques. Our results demonstrate that the proposed model consistently generates 100% valid quantum circuit outputs.

3.1 Introduction

Quantum computing is an emerging field with the potential to significantly advance computational capabilities. At the heart of quantum computing lies quantum circuits, which are analogous to their classical counterparts but operate on qubits to perform complex computations. Quantum gates are the fundamental building blocks of these circuits, with common single-qubit gates including Hadamard (H), Pauli X/Y/Z gates, and rotation gates such as RX, RY, RZ, and U. Multi-qubit gates, including CNOT, Toffoli, controlled Pauli, and controlled rotation gates, enable more intricate operations. Each gate is unitary, which means that it can be represented

by a unitary matrix U (where U^*, $UU^* = UU^{-1} = I$), acting on one or more qubits. These operations are crucial for the development of quantum algorithms and the realization of the potential of quantum computing.

Quantum circuits typically consist of a sequence of gate operations applied to various qubits, followed by a measurement of the relevant qubits at the conclusion of the computation. In addition to these functional gate operations, some quantum circuits include a state preparation step prior to the main operations, where the initial state of the qubits is instantiated. This state preparation is crucial for setting up the qubits in the desired configuration before the circuit performs the intended computation.

In the current era of noisy intermediate-scale quantum (NISQ) [20] devices, quantum computers face several constraints that affect the fidelity and accuracy of quantum circuit executions. These challenges include qubit crosstalk, limited gate types, gate errors, short decoherence times, and restricted qubit connectivity. In these noisy environments, factors such as gate count and circuit depth become critical limitations when designing and evaluating quantum circuits, hindering researchers' ability to develop practical and scalable quantum algorithms. These constraints present a significant barrier to fully realizing the potential of quantum computing.

Even if noise issues were negligible, the high cost of entry and the steep learning curve associated with quantum computing still limit the number of algorithms researchers can develop to harness its unique properties [21, 22]. Quantum architecture search (QAS) solutions aim to address these challenges by reducing the reliance on human expertise in quantum algorithm development [23]. These approaches employ a variety of techniques, including reinforcement learning (RL) [24, 25], large language models (LLMs) [26, 27], variational autoencoders (VAEs) [28], and evolutionary algorithms [21, 29]. They can be applied to generate both parametric [30, 31] and non-parametric [32, 33] quantum circuits. Parametric Quantum Circuits (PQC), which use quantum gates requiring parametric input, are particularly prominent in Variational Quantum Algorithms (VQAs). VQAs enable the implementation of quantum machine learning models, potentially overcoming the limitations of classical machine learning approaches. In this work, we leverage a diffusion model to generate quantum circuits, aiming to further expand the possibilities of AI designed quantum algorithms for quantum computing.

The structure of the chapter is as follows: Section 3.2 provides the background and related work. Section 3.3 outlines the approach for converting the models into valid quantum circuit graph generators. Section 3.4 presents visualizations showcasing each model's ability to process quantum circuits. Section 3.5 and 3.6.1 offers an analysis of the results, and Section 3.7 concludes the chapter.

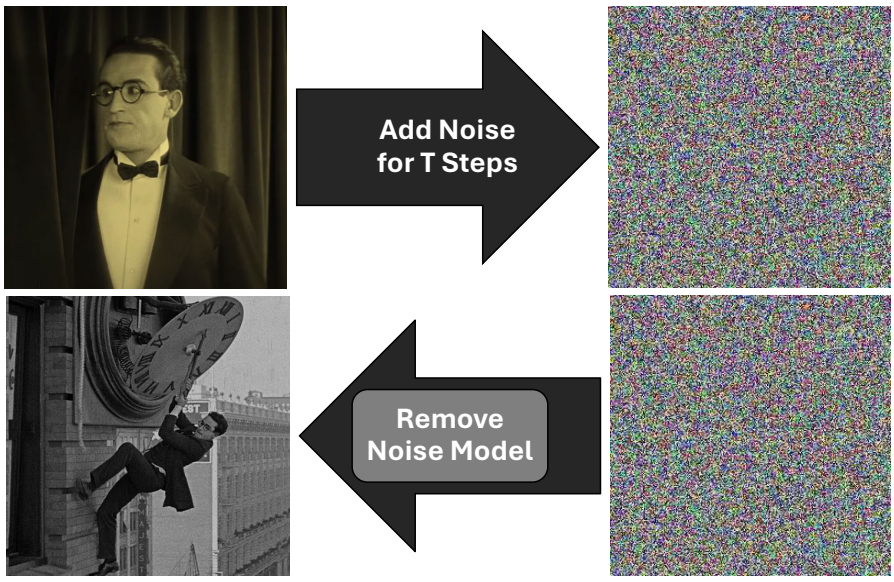


Figure 3.1: Sample of the diffusion process. The top shows how an image structure is destroyed via noise over several steps. The bottom shows how structure is recreated using a model.

3.2 Background and Related Work

3.2.1 Fundamentals of Quantum

For a summary of qubits refer to Section 1.1.1 and for a summary of quantum gates refer to Section 1.1.2.

3.2.2 Diffusion Neural Networks

Diffusion models are widely used in image generation to create or recover data samples that align with the distribution of source data [34]. The diffusion process consists of two phases: a forward diffusion phase and a reverse diffusion phase. In the forward phase, noise is incrementally added over several time steps, progressively obscuring the underlying structure of the data. Once the data is sufficiently corrupted by noise, it is passed through a model during the reverse diffusion phase. Here, the model is trained to learn the noise patterns so it can reverse the process, gradually denoising the data to reveal the original structure. Diffusion models have gained significant popularity in recent years, surpassing Generative Adversarial Networks (GANs) in certain applications. They have also been successfully integrated with multi-modal models to generate images based on textual descriptions [35–37] (Fig. 3.1). Diffusion models offer an advantage over other models such as GANs and reinforcement learning due to their

stability and avoidance of mode collapse during training, which is important when considering the complexity of the quantum domain.

3.2.3 Quantum Architecture Search

Current quantum architecture search (QAS) solutions face several challenges. Large language model (LLM)-based approaches, which leverage vast datasets and are central to neural network research, such as the Qiskit code assistant, have the potential to enable researchers to design quantum circuits more efficiently [27]. However, LLMs lack the ability to fully exploit the physical structures defined by graph-based approaches, which can lead to inaccurate or suboptimal results due to the absence of physical understanding [26]. Reinforcement learning (RL)-based solutions allow for a more thorough exploration of the quantum search space using various techniques [24, 38, 39], but they typically rely on expert-defined sub-circuit replacement rules, making them time-consuming to train and run. Variational autoencoder (VAE)-based solutions, while able to leverage the graph structure of quantum circuits and avoiding the training/runtime inefficiencies of RL, still face scalability issues and struggles with qubit size [28]. Moreover, VAEs are prone to problems such as mode collapse and unstable training. To overcome these limitations, we propose a diffusion model based on the LayerDAG framework [19] as a novel approach for quantum circuit generation.

3.3 Q-Fusion: Approach

3.3.1 Approach

To harness the generative potential of diffusion models in the quantum domain, we introduce Q-Fusion, an innovative approach to generate quantum circuits. Our method builds upon quantum circuits' natural representation using graph notation and draws inspiration from recent graph-based diffusion frameworks, such as LayerDAG [19]. In this graph representation, nodes correspond to quantum gates, whereas edges represent the connections (wires) between qubits. In this section, we detail the functionality of LayerDAG and the necessary modifications to adapt it for the effective creation of Q-Fusion, enabling the generation of valid and efficient quantum circuits.

The process of generating quantum circuits using Q-Fusion is illustrated in Fig. 3.2. Initially, the original quantum circuits (step ①) are transformed into directed acyclic graphs (DAGs), with noise added to "dirty" the graph (step ②). These "dirty" DAGs are then input into the graph diffusion model (step ③), which reconstructs "clean" DAGs (step ④). The noise from

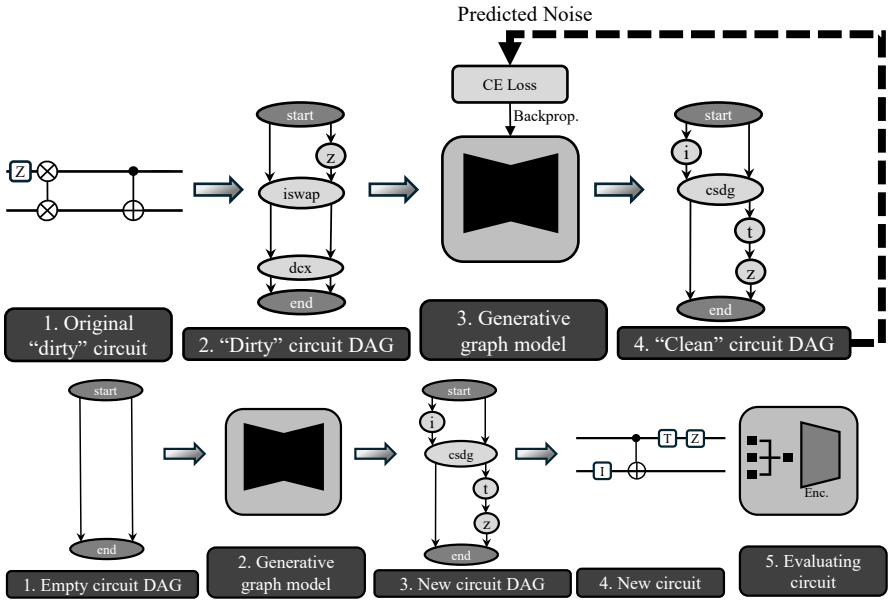


Figure 3.2: Overview of Q-Fusion process for generating quantum circuits. The top image represents the training process of the diffusion model. The bottom image displays the sampling method used to create new circuits from the trained diffusion model.

this process is used to update the model. During training, we employ Cross-Entropy (CE) Loss to compare the real noise applied with the predicted noise, ensuring that the model learns to preserve the desired functionality. In the sampling phase, empty DAGs are generated (step ①) and fed into the diffusion model (step ②). The model then samples to produce new, completed graphs (step ③). Finally, the generated graph is converted into a quantum circuit (step ④), which is validated for correctness as a quantum circuit (step ⑤).

3.3.2 Training

LayerDAG employs a three-pronged layering approach, visualized in Fig. 3.3, with each layer trained using independent parameters. The first layer predicts the node count, the second layer predicts the nodes themselves, and the third layer predicts the edges between the nodes. Each layer utilizes a variation of teacher forcing, where the expected label is fed into the graph encoding portion of the model during training. In the context of quantum computing, several labeling strategies are possible. After experimenting with various options, we chose to use the summation of the density matrix as the label. This choice stems from the sparse nature of most density matrices, where we sacrifice some expressivity to achieve a significant reduction in memory usage (from $O(2^n * 2^n)$ to $0(1)$).

For each of the three layers, the directed acyclic graph (DAG) is processed sequentially. A

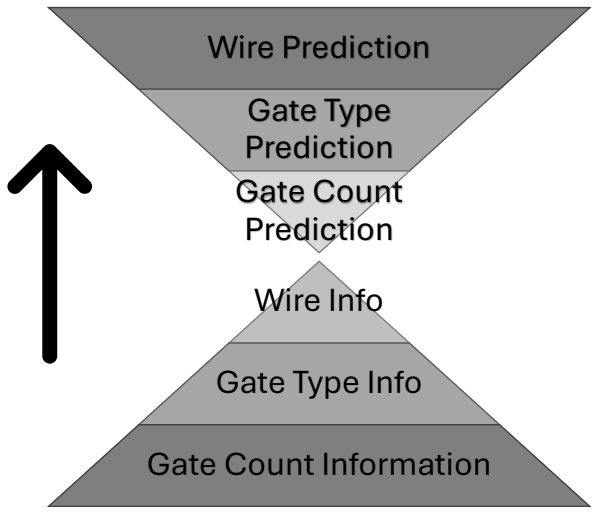


Figure 3.3: Underlying structure of 3 layered diffusion model.

noisy partial DAG is fed into the model, which then predicts the information for the next layer (i.e., the node count, the nodes themselves, and the edges for that layer). The parameters of each model layer are updated using Cross-Entropy (CE) Loss, comparing the predicted noise with the actual noise applied to the model, as is common in modern diffusion models [40]. This approach ensures that each layer progressively refines the structure of the DAG toward a valid quantum circuit.

3.3.3 Sampling

Once the model, with all its separate layers, has been trained, it is sampled for performance evaluation. During the sampling process, empty graphs with noise are fed into the model, which predicts the features of the next layer—specifically, the number of nodes, node types, and edges. The predictions from each layer are then used to update the graph, and this process is iterated until the model predicts no new nodes. Upon completion, we attempt to convert the resulting DAGs into quantum circuits. The model's performance is then evaluated based on its ability to generate graphs that align with valid quantum circuits, ensuring that the generated structures are both functional and meaningful in the context of quantum computation.

3.3.4 Model Updates

3.3.4.1 Graph Generation

LayerDAG is designed to process graphs without requiring a standardized structure, which is advantageous in its ability to handle generic directed acyclic graphs (DAGs). However,

quantum circuits have a highly structured flow of information, governed by strict layering and connection rules. For example, altering the order of gate execution or removing a gate can drastically change the functionality of a circuit. Similarly, if a graph were to bifurcate, the circuit could fail, the qubit count might be reduced, or the execution order could become unpredictable. Dropping a qubit from the structure could render the circuit non-functional or fail to preserve its original design. To address these challenges, we introduce virtual start and end nodes at the beginning and end of the training graphs. These nodes help guide the diffusion model by emphasizing the importance of maintaining the strict structure and order of execution inherent in quantum circuits. This approach aims to ensure that the generated circuits remain consistent with the intended functionality and quantum operations.

Ideally the quantum circuits used would come from a large, well maintained dataset where the circuits are labeled for their functionality. Unfortunately, in the NISQ era, this is not available to the public. So to build a data set, random quantum circuits are generated, and then converted to graphs to train the diffusion model.

3.3.4.2 Wire Information

While LayerDAG is capable of processing generic DAGs, it may overlook critical structural details. In the case of quantum circuits, for instance, there is no inherent way to represent which wires are associated with the edges, making it difficult to determine how the circuit functions when multiple qubits are involved. As the number of qubits increases, the number of potential circuit configurations grows factorially, further complicating the task. The absence of wire information also removes key structural elements that help define the quantum search space. To address this, we introduce wire information into each of the three layers of the model. This is accomplished by adding additional parameters that build a hidden representation of the wire connections, similar to the way node types are represented. These wire parameters are then concatenated with the overall hidden representation of the graph before being processed to make predictions. This addition ensures that the model can better capture the intricate connectivity and structure of quantum circuits, leading to more accurate and functional predictions.

3.3.5 Model Information

3.3.5.1 Gates & Gate Connections

The model employs topological ordering for both graph encoding and reconstruction. During the reconstruction phase, it allows for the selection of edges between the current nodes and any previously created nodes. The model is also trained to learn the appropriate number of

incoming and outgoing edges based on the types of gates involved. While quantum gates have fixed input and output edge constraints, we designed the model to explore this space and autonomously learn the structural limitations. This approach enables the model to discover the optimal configurations for gate connections. When the model predicts that no new nodes remain for a given layer, it halts the graph processing, signifying the completion of the quantum circuit structure.

3.3.5.2 Circuit Types

As a proof of concept to evaluate Q-Fusion's ability to produce quantum circuits a large set of non-parametric gates using the $X, Y, Z, H, S, T, ID, SXDG, SDG, SX$, and TDG single qubit gates and the CX, CY, CZ, $SWAP$, DCX, $ISWAP$, $CSDG$, ECR, CH, CS, and CSX were used to generate 6k samples of 2 qubit 8 gate circuits.

To reduce the complexity of the quantum search space and better emulate the constraints of NISQ machines, the gates used in the design of training and testing graphs are drawn from non-parametric circuits consisting of one- and two-qubit gates from the IBM Heron gate set. Specifically, we utilize single-qubit gates such as X, SX, ID, and the two-qubit gate CZ. Additionally, we evaluate the performance of parametric circuits by incorporating the parametric gate RZ, which allows for more flexible gate operations. This combination of non-parametric and parametric gates facilitates a more controlled exploration of the quantum circuit design space, reflecting the practical limitations of current quantum hardware.

3.4 Results

3.4.1 Setup

The Q-Fusion model has been enhanced over LayerDAG [19] with additional structures, including the ability to predict the qubit wires assigned to gates and the connection constraints between gates, to address the specific requirements of quantum circuits. For an initial evaluation, Q-Fusion is tested using a 2-qubit, 8-gate circuit, with 6,000 samples used for model training. The circuit generation utilized the custom non-parametric gate set described in Section 3.3.5.2. During evaluation, the model's wire predictions are excluded, and the virtual start and end nodes are used to guide the graph generation. Wire ordering is randomly assigned at the initial connection from the virtual start node and maintained throughout the graph-to-circuit translation process.

Table 3.1: Non-Parametric Circuits Produced (%) From Sampling Trained Diffusion Model

Statistic	% Valid	% Unique
2-qubit Circuits, Custom	100%	40.21%
2-qubit Circuits, Custom w/ Meaningful Functionality	99.9%	40.21%
2-qubit Circuits, Heron	100%	37.25%
2-qubit Circuits, Heron w/ Meaningful Functionality	48.92%	76.51%
5-qubit Circuits, Heron	100%	9.77%
5-qubit Circuits, Heron w/ Meaningful Functionality	21.89%	32.71%

3.4.2 Generation of Non-Parametric Circuits

An example of a generated 2-qubit quantum circuit is shown in Fig. 3.4. This demonstrates that the diffusion model is capable of effectively navigating the quantum search space, accurately selecting the correct number of wires per gate and distributing gates across multiple wires to construct a functional and unique circuit.

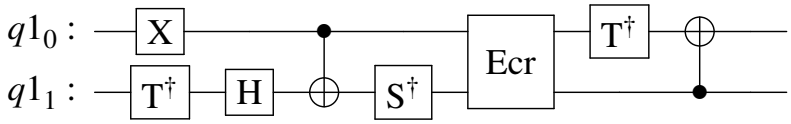


Figure 3.4: Example Quantum Circuit produced by Q-Fusion for the 2 qubit 8 gate trained model

The Q-Fusion model's ability to predict the wire assignment for each gate was reintroduced and evaluated using two smaller circuit sizes: 2 qubits and 5 qubits, each consisting of 32 non-parametric gates. This reduced circuit architecture facilitates the verification of circuit designs without requiring extensive computational resources or runtime for evaluating the density matrices. To train the model, 6,000 random sample circuits were generated for each separate circuit set. The 32-gate circuits were constructed using IBM Heron's non-parametric gate set.

After successfully training Q-Fusion, the model is sampled to generate 4,320 random circuits in order to assess the ability of the diffusion model to reproduce quantum circuits. The results are presented in Table 3.1. The model demonstrated the ability to generate a circuit successfully each time it is sampled. To evaluate the model's effectiveness in producing "meaningful" circuits (i.e., circuits that are not empty), we analyzed the density matrices of each generated circuit. These density matrices vary in size, with 4x4 matrices containing 16 values and 25x25 matrices containing 625 values. A circuit is considered "meaningful" if 10 or more of these values were non-zero, indicating that the circuit exhibited functional quantum behavior.

Table 3.2: PQCs Produced (%) From Sampling Trained Diffusion Model

Statistic	%	Expressibility	% Unique
Valid PQCs Produced From Sample Space	100%	.953	16.94%
PQCs w/ Meaningful Functionality	32.45%	.880	59.12%

3.4.3 Generation of Parametric Circuits

After demonstrating that Q-Fusion can effectively work with quantum circuits, we next evaluated its ability to generate parametric quantum circuits (PQCs) that are fundamental to quantum neural networks (QNNs), the results of which are in Table 3.2. PQCs enable the training and tuning of quantum circuits for machine learning applications. For this evaluation, we generated 6,000 random sample circuits, each using between 1 and 5 qubits and comprising 32 gates. The parametric gates were assigned random values between 0 and 2π. The specific values used for the parameters are not critical, as their precise selection holds limited value without knowledge of the downstream task. These parameters are fine-tuned during the neural network training process, which is where their true utility is realized.

3.5 Discussion

3.5.1 Non-Parametric Circuit Design

When evaluating the diffusion model's ability to generate quantum circuits, we observe promising results. However, a closer examination of Table 3.1 reveals a concerning trend: a decrease in "meaningful" circuits. This decline can be attributed to several factors, with the most significant being the limited gate count and gate types. The circuit is constrained to producing only 32 gates per instance using IBM Heron. As discussed in Section 3.3.5.2, the inclusion of the *ID* gate, which functions as a null operation, exacerbates this issue. Moreover, with such a restricted gate set, there is a high likelihood of repeating gate operations on the same wire in rapid succession. This is problematic, particularly for gates like the X gate, where consecutive applications result in cancellation, effectively behaving like a null operation.

Another contributing factor is the diffusion model's lack of inherent quantum knowledge. While the model understands how to construct a valid layout, select appropriate wiring, and so on, it lacks the built-in understanding of optimization principles like gate cancellation. This absence of quantum awareness further explains the reduction in "meaningful" functionality observed in the results. Addressing this gap in quantum knowledge presents an intriguing avenue for future research, with the potential to improve the model's performance in generating

functional quantum circuits.

Additionally, we evaluate the model's ability to produce unique circuits. For the case of the non-meaningful circuits, the 2-qubit custom gate set achieves the highest percent uniqueness. The 2-qubit Heron set obtains the highest value in terms of achieving meaningful functionality. This is due to the fact that the model has a much larger set of gates available to the system allowing for better performance in the case of the 2-qubit custom gate set. However, when we account for the meaningful nature behind the circuits the 2-qubit Heron set performs the best, this is because the smaller gate set and circuit size allows for the model to build a stronger understanding of the nature of quantum circuit design.

3.5.2 Parametric Circuit Design

Table 3.2 tabulates an additional statistic, expressibility [41], which defines how well a PQC is able to explore the quantum Hilbert space. Values close to 0 are preferred, with no upper limitation on the value observed. The diffusion model is able to produce near 0 values, and we note that the expressibility improves with more meaningful circuits. The performance of LayerDAG is promising, especially considering the expressibility is not considered at training time. Further improvements would be expected if this is taken into account.

Comparing the uniqueness performance between Table 3.1 to Table 3.2, we note that the PQC model's performance sits right between the 2-qubit and 5-qubit non-parametric model. This is expected as the gate set is only increased by a single gate, while consisting of circuits using 1 to 5 qubits.

Continuing to examine Table 3.2, we observe a continued loss in "meaningful" functionality. While the issues identified earlier persist, two additional concerns arise. The first relates to the behavior of the parameters. Since parameter values are randomly generated, they may either cancel out the functionality on the wire or leave the wire with no functionality at all. If the *RZ* gate is assigned a value that results in the inverse of the wire's intended functionality, the operation effectively becomes a null operation. Given that the only current methods for evaluating a PQC rely on training and assessing its performance on a machine learning task (IE tuning the parameter), this loss of "meaningful" functionality should be regarded as a limitation of quick evaluation techniques. Future work should focus on improving methods for evaluating the functionality of PQCs more reliably.

The second issue contributing to the performance drop stems from the use of a functionally smaller dataset. This choice was deliberate, aiming to better simulate the constraints of quantum systems in the current NISQ era. While previous experiments utilized 6k samples for a single circuit size, in this study, the circuit range was expanded to include 1, 2, 3, 4, and 5 qubit

circuits, along with an additional complex parametric gate, all while maintaining the same dataset size. The challenge is that there are not 6k well-understood circuits available, even for these small circuit sizes. Nonetheless, it is important to highlight that meaningful machine learning progress can still be made within this limited data regime, showcasing the potential for developing effective quantum models despite these constraints.

3.6 Limitations

3.6.1 Evaluation

Diffusion models have demonstrated strong performance with large and complex images, offering promising prospects for scaling quantum circuits to incorporate more gates and qubits. However, these models are not free of limitations, and standard challenges associated with diffusion models and graph representations must be considered. A unique constraint in the context of quantum computing is the difficulty in calculating the density matrix associated with the circuit. Density matrices, which represent the functionality of quantum circuits, scale exponentially as $2^n \times 2^n$, where n is the number of qubits. As circuit sizes increase, this exponential growth quickly outpaces the computational power of classical machines, making it infeasible to represent and evaluate large quantum circuits. For Q-Fusion, this means that, beyond a certain scale, it will no longer be possible to evaluate the circuit design using classical methods. One potential solution is to map the circuit to a physical quantum device, though this would limit our ability to assess its "meaningful" functionality using the current definition. As discussed in Section 3.5.2, evaluating PQCs in this manner may be unreliable. Nevertheless, if preserving the evaluation of meaningful functionality is critical for the task at hand, a feasible approach could involve decomposing larger circuits into smaller sub-circuits and evaluating them individually.

3.6.2 Labeling

One limitation of the current approach arises from the use of the label system discussed in Section 3.3.2). If this label were to be replaced with a density matrix representation, the model would struggle to process large circuit designs. Even with the existing label system, the model becomes unfeasible when the computer can no longer process the entire density matrix. This limitation stems from the fact that the label relies on a summation of the density matrix. However, the ultimate goal of the label is to provide a unique representation for functionally distinct circuits. To overcome this constraint and preserve similar functionality, an alternative

approach could involve running the circuit on real hardware and using the resulting output vector to represent the circuit.

3.7 Conclusion

Based on the results from both non-parametric and parametric quantum circuit generation as benchmarks, we believe that diffusion models hold significant potential to enhance QAS solutions. In the future, the proposed approach may facilitate the accelerated exploitation of quantum properties, offering promising advancements for complex problems such as quantum machine learning.

Acknowledgment

The work is supported in parts by NSF (CNS-1722557 and CNS-2129675) and gifts from Intel.

Chapter 4
Quantum Machine Learning for Material Synthesis

Using quantum computing, this chapter addresses a scientifically-pressing and day to day-relevant problem, namely, chemical retrosynthesis which is an important step in drug/material discovery. We show that Quantum Long Short-Term Memory (QLSTM) is a viable tool for retrosynthesis. We achieve 65% training accuracy with QLSTM whereas classical LSTM can achieve 100%. However, in testing we achieve 80% accuracy with the QLSTM while classical LSTM peaks at only 70% accuracy!

4.1 Introduction to Chemical Retrosynthesis

Chemical retrosynthesis attempts to provide reactants that can be combined, using a chemical reaction, to synthesize a desired molecule. This process defines fields such as agriculture, medical treatment, material discovery, and countless others. Fig. 4.1a exemplifies the retrosynthesis process, where the chemical on the left can be formed by the chemical on the right in combination of a chemical reaction. Performing retrosynthesis in the lab using trial-and-error takes years, and possibly cost billions of dollars, to resolve just for a single chemical. This leads to an immense amount of interest in machine learning (ML)-based solutions. Previous work have been able to generate promising results, but suffer from limitations. For example, expert defined rules for retrosynthesis [42] relies on human's incomplete knowledge of retrosynthesis and doesn't scale well as more rules are being defined. To overcome limitations of domain knowledge, models have been created that do not require prior knowledge [43, 44]. These solutions ignore the certainty of domain knowledge, require excessive training time, and still poses scalability issues, making it hard to solve retrosynthesis of large molecules [45]. Another common issue is a dependence on a predefined library of solutions rather than coming up with

unique chemical results [46]. The efforts to resolve these issues run into the difficulty of finding chemically viable solutions, long training times, etc. [47, 48]. Chemical retrosynthesis could benefit from more capability than what modern machines offer, prompting us to search for solutions in new hardware domains.

The promise of exponential growth in computational space has led to the idea of Quantum Neural Networks (QNN) [49] and more recently the Quantum Long Short-Term Memory (QLSTM) [50]. Unfortunately, Quantum Machine Learning (QML) efforts have fallen short of their desired exponential gain in speed [51]. However, they still offer the ability to represent an exponentially growing amount of information with only a linear growth in hardware size.

We evaluate the performance of QLSTM (a quantum-classical hybrid approach) and compare it to the performance of LSTM in its ability to make retrosynthetic predictions using the USPTO-50k dataset [52]. We also introduce two unique approaches to simplify the retrosynthesis process by identifying a specific substring within the reactants that are used to produce the given reaction.

The rest of the chapter is structured as follows: we cover basics on quantum computing, QLSTM, QNN etc. in Section 4.2, discuss the methodology used for chemical synthesis in Section 4.3, present the results of the problem in Section 4.4, and end with closing remarks in Section 4.5.

4.2 Laying the Groundwork: Material Discovery, Machine Learning and Quantum

4.2.1 Material Discovery

Material discovery extensively employs USPTO-50K dataset [52] which consists of 40,000 training, 5,000 validation, and 5,000 testing SMILES formatted chemical examples. SMILES is originally created as a way to use characters to represent chemical chains [53]. The letters

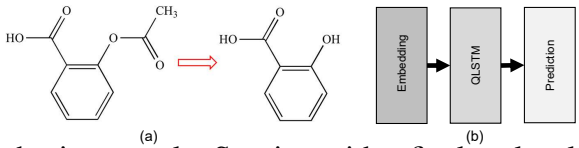


Figure 4.1: (a) Retrosynthesis example. Starting with a final molecule, the goal is the identify its starting molecule; (b) chemical retrosynthesis architecture used for training; embedding step turns information into the proper dimension for the QLSTM; the QLSTM learns and processes the data; the prediction step performs a softmax to convert the dimensional data to a singular value.

represent various elements within the chain where the first letter of an element can be uppercase, denoting that the element is non-aromatic, or lowercase, denoting that the element is aromatic. If an element requires a second letter it will be lowercase, regardless of the casing of the first letter. Numbers are used within the chain to represent the opening and closing of a ring. Finally, parenthesis are used to denote branches from a chain, whereas periods are used to denote the start of a new chemical.

$$<RX\_1> c\ 1\ c\ c\ c\ (\ C\ n\ 2\ c\ c\ c\ 3\ c\ c\ c\ c\ c\ 3\ 2\ )\ c\ c\ 1 \tag{4.1}$$

The input from USPTO-50K consists of two parts, the first part is the reaction type that causes the reaction whereas the second part of the string is the reaction. The reaction type consists of 10 different possible values, ranging from 1-10. The output consists of possible input reactants that can be used in combination with the reaction type to create the final reaction. Exemplifying the SMILES format in Eq. 4.1, the initial six characters, ($<RX\_1>$), represent the reaction type that causes the target molecule given certain reactant(s). Following the reaction type, we have the chemical $c1ccc(Cn2ccc3ccccc32)cc1$, which breaks down into three unique pieces. $c1ccc$ makes the initial chain, while $(Cn2ccc3ccccc32)$ forms a separate chain, which is denoted by the parentheses. Finally, we end with a third smaller chain, $cc1$. Next is the use of C and c, in the uppercase we note there is only a single non-aromatic carbon used, while the rest of the carbon in the chain is aromatic. Finally, we consider the use of numbers. Within the separate chain marked by the parentheses, we note the smallest ring formed, $3ccccc3$, this is the third ring in the set, which is why it is marked by two different 3s. Since the creation of the USPTO-50K, it has frequently been used as an experimental testing ground for chemical retrosynthesis [44, 45, 54]. We note that due to the nature of this difficult problem and unlike familiar benchmarks in other domains, the accuracy of much of this work rarely reaches higher than 50% while predicting the proper reactant for a given input. Although 50% is typically associated with random guessing, in this domain the accuracy relies on the exact match of reactant(s) to the given reaction. Given each reaction can have one or two reactants, and the majority of these reactants are unique to their reaction, it is easy to see 50% accuracy is far higher than random guess. We summarize the results from previous work in Table 4.1.

4.2.2 LSTM

LSTM is an adaptation of the original Recurrent Neural Network (RNN) structure which is designed to keep temporal storage of information. This allows the neural network to maintain previous states of information. However, there is no guarantee as to what information is held

Table 4.1: Small summary of SOTA chemical retrosynthesis results.

Model Type	Resulting Top 1 Accuracy
G2Gs [54]	48.9%
GLN [45]	52.5%
RetroPrime [44]	51.4%
Augmented Transformer [55]	53.5%

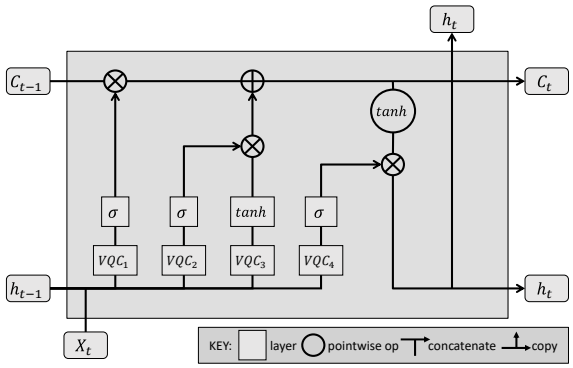


Figure 4.2: QLSTM architecture used for training; (C_{t-1}, C_t) represent the cell state, (X_t) represents the input, (h_{t-1}, h_t) represent the output state. The VQC_1 wire represents the forget gate, deciding if the input should be added to memory. The (VQC_2, VQC_3) wires represent the update gate, updating the cell memory if there is need. The VQC_4 wire represents the output gate, outputting the result of the QLSTM to the rest of the model.

and for how long it will remain, causing saturation issues. To get around these issues the LSTM allows for the neural network to decide when to add/remove pieces of information, helping mitigate context saturation issues.

4.2.3 Quantum Information

As a refresher on quantum, please refer to Section 1.1.1 for a description of qubits and Section 1.1.2 for a description of quantum gates.

4.2.4 QNN

QNN is a promising QML model that has received a lot of attention in recent years. A traditional QNN is made up of a data encoding circuit, a Parametric Quantum Circuit (PQC), and measurement operations. The data encoder transforms classical data into a quantum state. The PQC transforms the quantum state using a chosen ansatz. Measurements determine the output state. The PQC parameters are tuned during the training phase to produce the desired measurement results. We can train QNN models to perform traditional ML tasks such as classification, regression, distribution generation, etc. by selecting appropriate cost functions.

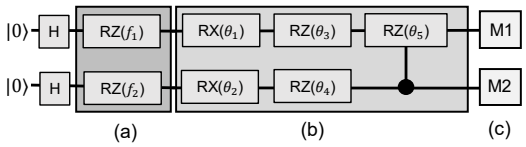


Figure 4.3: QNN architecture used for training; (a) angle encoding to convert classical feature (f_1, f_2) to their corresponding quantum state; (b) parametric quantum circuit used to perform desired transformations; and (c) measurement operation which collapses the qubit state to 0 or 1.

Fig. 4.3 shows the architecture of the 2-qubit QNN we used for training. It consists of the (a) Encoding, (b) PQC and (c) Measurement blocks. Several encoding techniques have been explored, e.g., amplitude encoding, basis encoding, NEQR [56]. We employ angle encoding where we pass classical features (f_1, f_2) as angles of quantum rotations gates (RZ) to transform them to quantum state. Similar to angle encoding, there are a number of PQC ansatz [41] to choose from but almost all of the PQCs consists of two main gate types: single qubit gates which are used to perform design space exploration, and two qubit gates which are used to entangle the qubits. The latter forms a correlation between the qubits based on the input feature values.

In the QNN, we use the 2-qubit Controlled-RZ (CRZ) gate to entangle the qubits and rotation gates along X and Z axes (RX, RZ) for transformation/exploration. The PQC/QNN is analogous to a classical neural network where we adjust the weights (w_i) to reduce the loss value while we adjust the tunable parameters (θ_i) to generate the desired output in QNN. Finally, to measure the qubit state the most widely used measurement technique is Pauli's measurement along any of the X, Y, or Z axes. In our QNN model we used Pauli-Z measurement ($\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$). A measurement in the Pauli Z basis means projecting the state onto one of the states $|0\rangle$ or $|1\rangle$ (the eigenstates of Pauli Z matrix).

4.2.5 QLSTM

There have been many attempts in the quantum computing domain to create trainable networks [49, 57] to solve classification problems. However, selective memory has not been available. QLSTM addresses this challenge and offers the same advantage as classical LSTMs, i.e., the ability to intentionally form a contextual understanding of previous input. This approach is near identical to the classical LSTM. The divergence of the two occurs when the network, instead of taking the information directly from the hidden layer and the input, takes the information and pass it to a Variational Quantum Circuit (VQC) where we can perform a data entanglement of the values. We then perform a measurement on the entangled information and proceed

to process it in the same prediction structure as the classical LSTM. Fig. 4.1b displays the basic overall architecture of the network: embedding, QLSTM and prediction. Embedding is preferred to a bag of words model as it reduces potentially large sparse vectors to smaller dense vectors that require less memory. Fig. 4.2 shows the structure of the QLSTM.

Starting with our representation choices, we use the round edge boxes to represent the external values fed into the QLSTM, each varying in size. For X_t the size is dependent on the embedding layer whereas for h_t and C_t the size is defined by the hidden dimension size. We use the sharp-cornered boxes to represent layers of a network, and circles to represent point wise functions. For the wires, as displayed in the key, we use the wire merging to represent concatenation and wire splitting to represent a copy of the wire. We also use σ to represent the sigmoid activation function, defined by Eq. 4.2, and tanh is the arctan activation function, defined by Eq. 4.3.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{4.2}$$

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{4.3}$$

Working through the QLSTM starting with the bottom left, we have X_t which represents the input to the QLSTM structure. The input is concatenated with the previous hidden layer information, which is represented as h_{t-1}. This combination is fed into four different VQCs; each of them are defined by a modified version of the basic entangler circuit from Fig. 4.4.

The modified basic entangler includes a trainable fully connected layer that squeezes the dimensional space of the information down to the circuit size of the VQC. After each VQC a quantum measurement of the expectation for each wire is fed to the trainable fully connected bloating layer. The bloating layer, increases the size from the quantum circuit back to the required dimensional space of the classical network. This is then processed using classical LSTM approaches. Hence, this is a quantum-classical hybrid approach. The first sigmoid activation is known as the forget gate which is used to decide whether to update the context C_t to include the new input. After the sigmoid, the result is multiplied onto C_{t-1}. The second sigmoid, and the *tanh* activations are known as the input gate which is used to write the new input into the context. The result of the sigmoid and tanh activations are multiplied to either be added to C_{t-1}, or to ensure the input is not added to C_{t-1}. The last sigmoid activation is known as the output gate where the actual prediction is performed. This output is also used to update the hidden layer h_t.

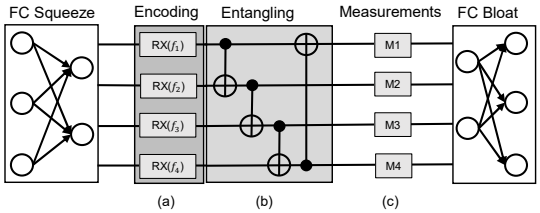


Figure 4.4: Modified basic entangler circuit; fully connected (FC) squeeze layer reduces the input size to be of the same qubit count. (a) Angle encoding converts classical features (f_1, f_4, f_3, f_4) to quantum states, (b) parametric quantum circuit entangles quantum states, (c) qubits measured, and bloated to original higher-dimension space. [6].

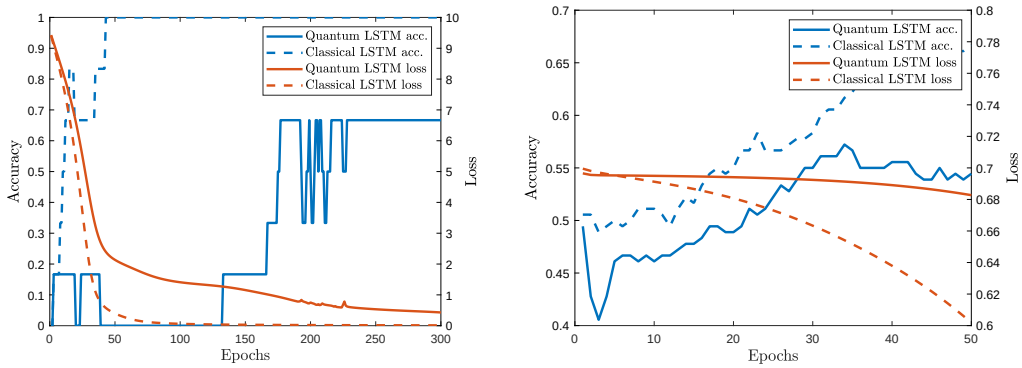
4.3 Fundamentals of Chemical Retrosynthesis

Previous works on retrosynthesis have addressed the problem from many different ways such as, using graph, transformer or some other approach. Prior to transformers, LSTMs were the preferred approach to neural networks that required a memory [58]. However, LSTMs don't work as well as transformers. Therefore, we propose two unique approaches to simplify the problem of retrosynthesis namely, (a) we restrict the reactions by selecting just a single reaction type, $<RX\_1>$, in an attempt to simplify the retrosynthesis process. This subset is reduced from 12,000 to just 9 samples to reduce training time, and emulate the proof of concept proposed by Di Sipio [59](b) We revert back to including all reaction types and change our output from a prediction of the reactants to a prediction of a chemical chain within the reactant. For this we select acetic acid and acetone as the common chemical chains and reduce the input reactions to only options that produce the selected chemical chains. This subset is reduced from 2,100 samples down to 200, which is then split 90:10 between training-validation set so there are 180 training samples and 20 validation samples. We then introduce these approaches to the QLSTM to show the potential of quantum computing in chemical retrosynthesis. In order to implement the encoding and the required layers for both LSTM and QLSTM, as well as the sigmoid activation and arctan activation function for the QLSTM, we use Pytorch [60]. The quantum circuits are trained using pennylane [6].

4.4 Practicality of QML

4.4.1 Experimental Setups

Since we adopt unique approaches to perform chemical retrosynthesis our results cannot be directly compared to other state-of-the-art work. For a fair comparison we create a classical LSTM in the same form as the QLSTM. The QLSTM depends on a 4 qubit VQC structure,



(a) (b)

Figure 4.5: Results of chemical retrosynthesis using a quantum and classical LSTM model; (a) training of 9 chemical sample limited data set of a single reaction type (reaction type 1) where quantum is able to reach reasonable accuracy. (b) training of 180 chemical sample limited data set of two common sub-string chemicals (acetic acid, acetone) where quantum nearly matches classical for the first 35 epochs.

while relatively small, the 4 qubit structure allows for a more manageable run time. Both the QLSTM and the LSTM use a small embedding dimension size of 8, and a small hidden dimension size of 6. The small embedding dimension is used for two reasons: first, it allows for enough memory for the second approach where we are predicting a reactant sub-string, and still uses a smaller vector than a bag of words would use for the proof of concept. The second reason is that when the concatenation of X_t and h_t occurs, it doesn't require a large fully connected layer squeeze/bloat to match the size of the VQC structure. The hidden dimension size is heuristically selected, using values less than the embedding dimension, in expectation to keep the fully connected layer size requirement low. For the concatenation it is performed such that X_t appends to h_t. Table 4.2 contains a summation of parameters. All performance results are reported from execution on an Intel Xeon W-2125 CPU running at 4 GHz, with 16 GB of RAM.

Table 4.2: Chemical retrosynthesis prediction defined parameters for both QLSTM and LSTM models

Parameter	X_t	h_t	C_t	VQC size
Value	8x1 dim	6x1 dim	6x1 dim	4 qubits

4.4.2 Single Reaction Type Retrosynthesis

Before beginning the single reaction type retrosynthesis some preprocessing of the original data is required. For clarity, an example string is provided for each step. To begin we take the initial input strings, as seen below.

$$<RX\_1>F\ \ c\ \ 1\ \ c\ \ c\ \ 2\ \ c\ \ (\ \ N\ \ C\ \ 3\ \ C\ \ C\ \ C\ \ C\ \ C\ \ C\ \ 3\ \ )\ \ n\ \ c\ \ n\ \ c\ \ 2\ \ c\ \ n\ \ 1$$

We note here that the use of multiple lines are meaningless, they are just inserted for readability purposes. For simplicity and legibility, we use superscript numbers to represent a repeating series of a character. This format will be followed for the continuation of the work:

$$<RX\_1>F\ \ c\ \ 1\ \ c^{2}\ \ 2\ \ c\ \ (\ \ N\ \ C\ \ 3\ \ C\ \ ^{6}\ \ 3\ \ )\ \ n\ \ c\ \ n\ \ c\ \ 2\ \ c\ \ n\ \ 1$$

As part of our method, we then compress by removing all of the individual spacing, as this space does not carry any special meaning in the context of SMILES format:

$$<RX\_1>\ F\ c\ 1\ c^{2}\ 2\ c\ (\ N\ C\ 3\ C^{6}\ 3\ )\ n\ c\ n\ c\ 2\ c\ n\ 1$$

We then ensure the reaction type is the first reaction type, matching the $<RX\_1>$. After this we strip off the reaction type as it is no longer helpful:

$$F\ c\ 1\ c^{2}\ 2\ c\ (\ N\ C\ 3\ C^{6}\ 3\ )\ n\ c\ n\ c\ 2\ c\ n\ 1$$

After finishing the input string, we take the output string for processing. Here we match the input string to the output string to find the corresponding output:

$$F\ c\ 1\ c^{2}\ 2\ c\ (\ C\ l\ )\ n\ c\ n\ c\ 2\ c\ n\ 1\ .\ N\ C\ 1\ C^{6}\ 1$$

After we find the matching output string, we simply compress the string by removing the spaces:

$$F\ c\ 1\ c^{2}\ 2\ c\ (\ C\ l\ )\ n\ c\ n\ c\ 2\ c\ n\ 1\ .\ N\ C\ 1\ C^{6}\ 1$$

Once the trimming of the input and output is done, we perform a word encoding for both the input and the output to have a numerical representation of the SMILES strings for use in LSTM. The word encoding requires two unique lists, one for reactions and another for reactants. Each

list consists of unique chemicals, where each chemical is assigned it's numerical value based on it's index within the list it belongs to. After completing the preprocessing we train the LSTM and QLSTM models. The promising results in Fig. 4.5a show that the quantum approach, while unable to match the results of classical approach, is able converge to an accuracy of 65% and a loss of 0.1.

4.4.3 Chemical Chain Prediction

We perform similar preprocessing of the original data as explained with an example string below. We take the initial output strings from the input file:

$$C^2\ (\ C\ )\ (\ C\ )\ O\ C\ (\ =\ \ O\ )\ N\ C^2\ (\ =\ \ O\ )\ C^3\ (\ =\ \ O\ )\ O\ C^4\ (\ =\ \ O\ )\ O\ C\ c\ 1\ c^5\ 1$$

We then remove all of the individual spacing:

$$C^2\ (\ C\ )\ (\ C\ )\ O\ C\ (\ =\ \ O\ )\ N\ C^2\ \ (\ =\ \ O\ )\ C^3\ (\ =\ \ O\ )\ O\ C^4\ (\ =\ \ O\ )\ O\ C\ c\ 1\ c^5\ 1$$

We then ensure the reaction contains the acetic acid chain, $CC(=O)O$ or acetone $CC(=O)C$. After this we dispose of the reactant and simply use the label of the chain the string contains, for example: *acetic*.

After we finish the output string, we take the input string and match it to the output string to find the corresponding input:

$$<RX\_6>C^2\ (\ C\ )\ (\ C\ )\ O\ C\ (\ =\ \ O\ )\ N\ C^2\ (\ =\ \ O\ )\ C^3\ (\ =\ \ O\ )\ O\ C^4\ (\ =\ \ O\ )\ O$$

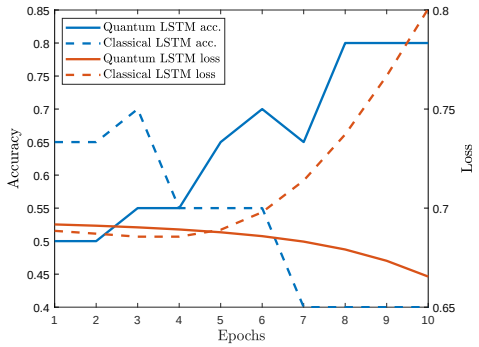


Figure 4.6: Results of 20 chemical sample limited data set testing of two common sub-string chemicals (acetic acid, acetone) where quantum outperforms classical.

After we find the matching input string, we remove the spaces, and the reaction type:

$$C^2\,(\,C\,)\,(\,C\,)\,O\,C\,(=O\,)\,N\,C^2\,(=O\,)\,C^3\,(=O\,)\,O\,C^4\,(=O\,)\,O$$

Once the trimming of the input and output is done, we perform our encoding and we train the LSTM and QLSTM models. The results in Fig. 4.5b show that the classical loss never reaches a point of convergence, where the quantum loss also doesn't reach convergence nor does it reach the same level as the classical. These results hold true for accuracy, where the classical domain reaches 65% and the quantum domain reaches 55%. While there is a small gap in performance, we see that given the task of identifying a common substring within the predicted reactants, quantum is able to nearly match classical performance during training. Validation is ran once every 5 epochs during training and here, there is a flip of performance. The results in Fig. 4.6 show that the classical loss starts to increase after just 25 training epochs, whereas the quantum loss is steadily decreasing for the entirety of the training. As for the accuracy, the classical accuracy reaches a high of 70% and steadily decreases to 40%. For the quantum domain the accuracy starts at 50%, while steadily increasing all the way to 80%, outperforming the classical model by 40% at the end of the model training.

4.5 Conclusion

We have shown that QLSTM is a viable solution to solve chemical retrosynthesis problem, even with just 4 qubits. While QLSTM didn't train as well as its classical counterpart, it is able to reach a reasonable accuracy and loss metrics for the proof of concept. For example, quantum achieves 65% accuracy and classical achieves 100%. It again is able to reach a reasonable accuracy e.g., 55% for quantum and 65% for classical while attempting to predict substrings. However these gaps are misleading since quantum is able to reach an accuracy or 80% whereas classical peaks at an accuracy of 70% during testing of the substring prediction!

Chapter 5
Predicting Side Effect of Drug Molecules Using Recurrent Neural Networks

Identification of molecular properties, like side effects, is one of the most important and time-consuming steps in the process of molecule synthesis. Failure to identify side effects before submission to regulatory groups can cost millions of dollars and months of additional research to the companies. Failure to identify side effects during the regulatory review can also cost lives. The complexity and expense of this task have made it a candidate for a machine learning-based solution. Prior approaches rely on complex model designs and excessive parameter counts for side effect predictions. Reliance on complex models only shifts the difficulty away from chemists rather than alleviating the issue. Implementing large models is also expensive without prior access to high-performance computers. We propose a heuristic approach that allows for the utilization of simple neural networks, specifically the GRU recurrent neural network, with a 98+% reduction of required parameters compared to available large language models while obtaining near identical results as top-performing models.

5.1 Introduction

Molecular property prediction is one of the most fundamental tasks within the field of drug discovery [61, 62]. Applying in silico methods to molecular property prediction offers the potential of releasing safer drugs to the market while reducing test time and cost. Detecting molecular properties before development enables researchers to develop more effective new materials faster and with higher certainty. Detecting known causes of side effects in drugs before release can prevent unnecessary injury and save thousands of lives. Historically, these in

silico approaches relied on complex feature engineering methods to generate their molecule representations for processing [63, 64]. The bias of the descriptors limits these approaches, which means the generated features may not be reusable for different tasks as some valuable identifiers may not be present. The feature vectors also depend on current molecular comprehension; upon discovery, the feature vectors could become redundant. Graph Neural Networks (GNN) remove the dependence on complex and temporal descriptors. GNNs became favorable due to the common practice of drawing molecules using graph representations, which offer a generic form for the input. The generic input format allows machine learning models to build their interpretation of information rather than rely on human capabilities. Through these advances GNNs perform well on multiple cheminformatic tasks, especially molecular property prediction [65, 66]. Despite these improvements GNNs still have limitations. Specifically, GNNs have difficulty understanding shared dependence and have scalability issues. The size of the graphical input increases exponentially with each additional molecule that is represented. With this growth, the cost of communication between graphical nodes also exponentially increases. Compared to other neural network types, GNNs can perform worse at molecular property prediction, despite their built-in generic representation [67]. With the recent success of large language models, newer attempts aim to build transformer-based approaches with promising signs of success [68]. While new large language models offer comparable performance to GNNs, they require up to 120 billion parameters.

Due to the rapid explosion of parameters caused by GNNs, feed-forward neural networks, and transformers, we propose a heuristic approach using a recurrent neural network, specifically the gated recurrent unit (GRU). Our approach can obtain close to state-of-the-art results with 99+% fewer parameters than large graph-based models or large language-based models, such as Galactica [68]. In the following sections, we review the MoleculeNet benchmark [69] and compare the SMILES and SELFIES formats and the basic concepts of a recurrent neural network, and also discuss a few of the related works that are evaluated using the MoleculeNet benchmark (Section 5.2). We then discuss the data pre-processing and model implementation details (Section 5.3), followed by model performances and a comparison to other state-of-the-art options (Section 5.4). Finally, we conclude the chapter by giving a summary (Section 5.5).

5.2 Overview of Molecular Property Prediction and Neural Networks

5.2.1 MoleculeNet Benchmark

MoleculeNet is a benchmark set used to evaluate machine learning techniques [69]. It curates quantum mechanics, physical chemistry, biophysics, and physiology datasets. For each dataset, it establishes the preferred metric for evaluation to enable consistent comparison across models. We describe each dataset selected to evaluate our model.

5.2.1.1 Side Effect Resource (SIDER)

The principal molecular property of human consumption is the side effects associated with the molecule. The Side Effect Resource (SIDER) dataset attempts to create a single source of combined public records for known side effects [70]. The dataset consists of 28 columns; the first column is the SMILES representation of a given molecule, and the 27 subsequent columns are affected system organ classes where side effects are classified by MedDRA [1]. The side effects of each molecule are marked with a one if it is known to have a side effect or a zero otherwise.

5.2.1.2 BACE

BACE is a collection of experimentally reported values from various journals for the binding results for inhibitors of human β-secretase 1 [71].

5.2.1.3 Blood-brain barrier penetration (BBBP)

Here molecules are classified by their ability to permeate through the blood-brain barrier. A drug's ability to permeate through the blood barrier is an important feature for drugs specifically targeting the central nervous system [72].

5.2.1.4 ClinTox

MoleculeNet introduces ClinTox to evaluate drugs previously approved by the FDA and drugs that have failed clinical trials due to toxicity. [69]

[1] https://www.meddra.org/

5.2.1.5 HIV

The HIV dataset is originally from the Drug Therapeutics Program (DTP) [2] consisting of molecules tested to inhibit HIV replication. There are roughly 40k samples within the dataset, where MoleculeNet uses two labels, confirmed inactive and confirmed active.

5.2.1.6 MUV

The Maximum Unbiased Validation (MUV) dataset contains 17 labeling tasks and 90k molecules. The dataset originates from PubChem [73].

5.2.2 ROC-AUC

The receiver operating characteristic curve (ROC curve) measures the true positive rate against the false positive rate at multiple threshold settings for a binary classifier. This measures the ability of a model to distinguish correctly between two classes. ROC-AUC is commonly preferred when evaluating models trained on imbalanced datasets, making it an ideal statistic to evaluate the MoleculeNet datasets.

5.2.3 Simplified Molecular-Input Line Entry System (SMILES)

Simplified molecular-input line-entry system (SMILES) uses characters to build a molecular representation [53]. Letters represent various elements within a molecule, where the first letter of an element can be uppercase, denoting that the element is non-aromatic, or lowercase, denoting that the element is aromatic. Assuming an element requires a second letter, it will be lowercase. Another possible representation of aromaticity is the colon, which is the aromatic bond symbol. Other potential bond symbols are a period (.), a hyphen (-), a forward slash (/), a backslash (\), an equal sign (=), an octothorpe (#), and a dollar sign ($). Periods represent a no bond, hyphens represent a single bond, and the forward slash and backslash represent single bonds adjacent to a double bond. However, the forward slash and backslash are only necessary when rendering stereochemical molecules. The equal sign represents a double bond, the octothorpe represents the triple bond, and the dollar sign represents a quadruple bond. In cases where stereochemical molecules are used, the asperand (@) can be used in a double instance to represent clockwise or in a single occurrence to represent counterclockwise. Numbers are used within a molecule to characterize the opening and closing of a ring structure, or if an element is within brackets, the number can represent the number of atoms associated

[2] https://wiki.nci.nih.gov/display/NCIDTPdata/AIDS+Antiviral+Screen+Data

with an element. Numbers appearing within brackets before an element represent an isotope. A parenthesis (()) denotes branches from the base chain.

5.2.4 Self-Referencing Embedded Strings (SELFIES)

Self-referencing embedded Strings (SELFIES) improve the initial idea of SMILES for usage in machine learning processes by creating a robust molecular string representation [74]. SMILES offered a simple and interpretable characterization of molecules that was able to encode the elements of molecules and their spatial features. The spatial features rely on an overly complex grammar where rings and branches are not locally represented features. This complexity causes issues, especially in generative models, where machines frequently produce syntactically invalid or physically invalid strings. To remove this non-locality, SELFIES uses a single ring or branch symbol, and the length of this spatial feature is directly supplied; ensuring that any SELFIES string has a valid physical representation.

5.2.5 Recurrent Neural Networks (RNN)

Elman networks, more commonly known as vanilla recurrent neural networks (RNN), attempt to introduce the concept of a time-dependent dynamic memory [75]. The idea is to make predictions about inputs based on contextual information. Context-based predictions can be done for four input-output schemes: one-to-one, one-to-many, many-to-one, and many-to-many. One-to-one models are a variation of a classic neural network, one-to-many models are best for image caption generation, many-to-one models are best for sentiment analysis, and many-to-many models are best for translation or video frame captioning. Fig. 5.1 is an example of the basic structure of a vanilla RNN.

In Fig. 5.1, the X_t represents some input, H_{t-1}, H_t represents some hidden state (which is representative of memory), O_t represents some output, and σ represents some activation function. The current input information combines with the previous hidden state, and the resulting combined state is then fed to an activation function to insert some non-linearity. This non-linearity produces the next hidden state, which can be manipulated to create a desired output. The fundamental element is the hidden state. The hidden state theoretically allows for consideration of any historical input and its effects on the current input. For a mathematical description of an RNN, we refer to Equation 5.1 and Equation 5.2.

$$H_t = \sigma(W_{HH}H_{t-1} + W_{XH}X_t) \tag{5.1}$$

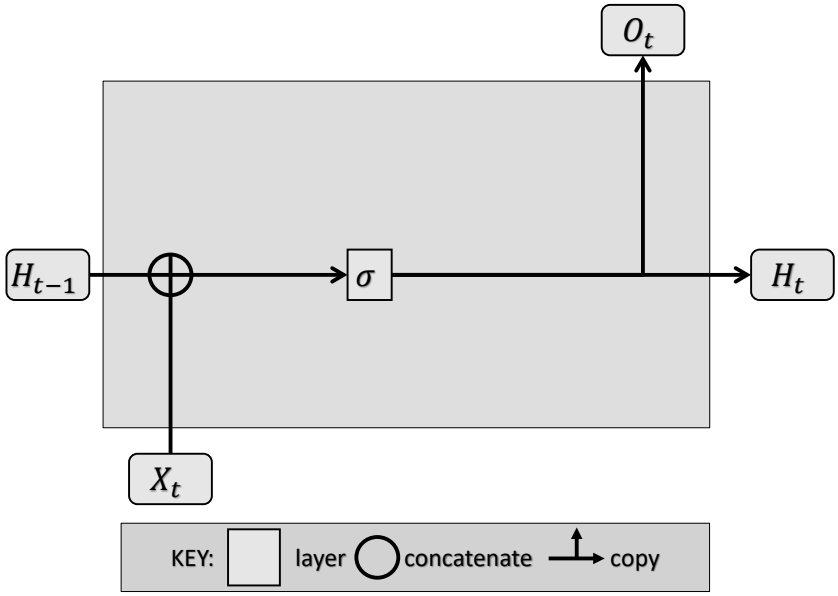


Figure 5.1: Vanilla RNN architecture used for training; (H_{t-1}, H_t) represent the hidden state, (O_t) represents the output state, and (X_t) represents the input information. The σ represents the activation function that operates on the combined input and hidden state.

$$O_t = W_{HO} H_t \tag{5.2}$$

Unfortunately, Vanilla RNNs suffer from memory saturation issues, so they are not always reliable. There have been many methods proposed to overcome this issue, but one of the most popular is the Gated Recurrent Unit (GRU) [76]. The basic structure of a GRU is in Fig. 5.2. We can mathematically describe each of the components using Equation 5.3, Equation 5.4, Equation 5.5, and Equation 5.6. Equation 5.5 represents the candidate hidden state function, representing the potential updated state. Equation 5.6 performs the actual update to the hidden state based on the previous hidden state and the candidate hidden state. Both Equation 5.3 and Equation 5.4 allow the network to tune the importance of the contribution of the previous hidden state to the new hidden state. Because of the r_t and z_t parameters, the GRU can better control its memory state offering a practical improved performance over RNNs.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{5.3}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{5.4}$$

$$\tilde{h}_t = tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{5.5}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{5.6}$$

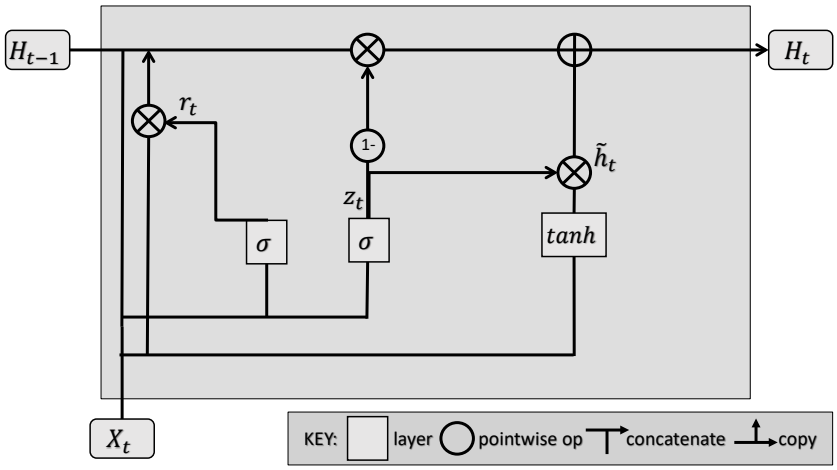


Figure 5.2: GRU architecture used for training; (h_{t-1}, h_t) represent the hidden state, $(\tilde{h}_t)$ represents the candidate hidden state state, and (r_t) and (Z_t) represents the parameters to tune the importance of the previous hidden state versus the updated information. The σ represents the activation function that operates on the combined input and hidden state.

5.2.6 Related Works

5.2.6.1 GROVER

The graph representation from the self-supervised message passing transformer (GROVER) model takes two forms, GROVER$_{base}$ and GROVER$_{large}$ [77]. We only consider GROVER$_{large}$ as it achieves the highest performance of the two. GROVER bases its design on popular large language models such as, BERT and GPT, where a large corpus of information pre-trains a model and fine-tuning is applied for the completion of downstream tasks [78,79]. However, they stray from prior works that attempt training using the SMILES string format [80] and instead use graphs, which they state are more expressive. Previous graph pre-training approaches use the available supervised labels to train their model [65], but GROVER prefers a self-supervised approach to achieve higher performance, so they suggest using contextual property prediction and graph-level motif prediction. Contextual property prediction takes a given element (node) within a molecular graph and predicts the connected elements and the type of bond used for the connection. Graph-level motif prediction takes a given molecule and attempts to predict the recurrent sub-graphs, known as motifs, that may appear within the molecule. To build the model, they designed a new architecture known as the GTransformer, which creates an attention-based understanding of molecular graphs. The pre-training process uses 10 million unlabeled molecules for training and 1 million molecules for validation. The molecules are taken from ZINC15 [81] and Chembl [82]. GROVER is fine-tuned on 11 benchmark datasets from MoleculeNet [69] for final evaluation. While this new architecture and self-supervised training approach offer appealing results the model uses 100M parameters, uses 250 Nvidia

V100 GPUs, and takes four days for pre-training.

5.2.6.2 ChemRL-GEM

Geometry Enhanced Molecular representation learning method (GEM) for Chemical Representation Learning (ChemRL) (ChemRL-GEM) draws inspiration from previous works using a graph-based approach, especially GROVER [65, 77]. ChemRL-GEM uses a large corpus of information to pre-train a model and, like GROVER, finds the ambiguity of SMILES and lack of structural information hard to build a successful model using a string-based approach [83]. ChemRL-GEM blames the low performance of prior graph approaches on neglecting the available molecular 3D information and improper pre-training tasks. ChemRL-GEM pre-training splits tasks into geometry-level and graph-level tasks. The geometry level tasks are again split into two types where bond length prediction, and bond angle prediction are local spatial structure predictions, and atomic distance matrices prediction is a global spatial prediction. The graph-level predictions are the Molecular ACCess System (MACCS) key prediction and the extended-connectivity fingerprint (ECFP) prediction. To build the model they designed an architecture called GeoGNN which trains on the atom-bond graph and the bond-angle graph of molecules to build a 3D structure-based understanding of the molecular graphs. ChemRL-GEM achieves SOTA performance and is an early attempt at a large 3D graph model pre-trained network. The pre-training approach uses 18 million training samples from ZINC15 and 2 million for evaluation [81]. They state that pre-training a small subset of the data would take several hours using 1 Nvidia V100 GPU, and fine-tuning would require 1-2 days on the same GPU. As a rough estimate of the actual training process there was a follow-up work called LiteGEM which removed the 3D input of the model but still uses 74 million parameters and takes roughly ten days of training using 1 Nvidia V100 GPU [84].

5.2.6.3 Galactica

Galactica is inspired directly by previous large language models and their utilization of large datasets to pre-train models for downstream tasks [78, 79]. Differentiating from BERT, they use a decoder-only setup from Vaswani et al. [58]. Unlike GROVER or ChemRL, Galactica focuses on general scientific knowledge and wishes to apply it to the entirety of the scientific domain [68]. The Galactica model takes several forms, but the 120 billion parameter model offers the best performance. Galactica trains over 60 million individual scientific documents and 2 million SMILES strings. Galactica is trained with samples from MoleculeNet, where the molecular properties are converted to text prompts and responses. Galactica acknowledges using SMILES they receive reduced performance gains as their model size increases, but they

state this could be overcome with more samples. Galactica offers a competitive performance to graph-based approaches while offering a simplified architecture design. Unfortunately, the model requires 120 billion parameters and trains using 128 Nvidia A100 80GB nodes. Despite the massive model size, it is not SOTA for a single SMILES metric. Galactica states they need additional samples/fine-tuning to obtain SOTA results.

5.3 Examining Methodology

5.3.1 Data Pre-processing

The available MoleculeNet benchmark [69] uses SMILES for its molecular representation. After reviewing some of the molecule strings, not all are canonical. Including non-canonical SMILES is problematic as SMILES grammar is already complex; the molecules are converted to RDKit's canonical form to reduce complexity. The next issue is caused by RNNs, one of the many advantages of RNN is the allowance of variable length inputs to account for a variable length of history. This is only true theoretically; in practice, RNN memory has limits, which is the focus of many newer works [85]. Despite this limitation, it has been recently shown that RNNs can handle input lengths of around 45-50 before the performance begins to degrade [86, 87]. Using this knowledge, we set a maximum SMILES length of 46 for the molecules. The limitation keeps a minor majority of the molecules while allowing us to ensure the RNN is performing well. After limiting the SMILES molecular length, the SMILES are converted to SELFIES. The intention of converting SMILES to SELFIES is to reduce the grammar complexity and simplify the learning process of the RNN. SELFIES converts each element and structural component, such as rings or branches, into their label. These labels are then encoded into a numerical value based on their dictionary index.

5.3.2 RNN Implementation

Fig. 5.3 offers a visualization of the methodology used to train the RNN. The molecules are first loaded in from a dataset from the MoleculeNet benchmark [69] and converted to SELFIES representation using the method described in Section 5.3.1. The converted SELFIES are then processed through an embedding layer with a dimensional space matching the size of the label dictionary. The dictionary consists of all the unique SELFIES components within the dataset and the embedding dimension equals the dictionary size to maintain as much information as possible. The input, hidden, and output dimensions of the RNN are also equal to the size of the dictionary. Maintaining the dimensional space and not reducing it before

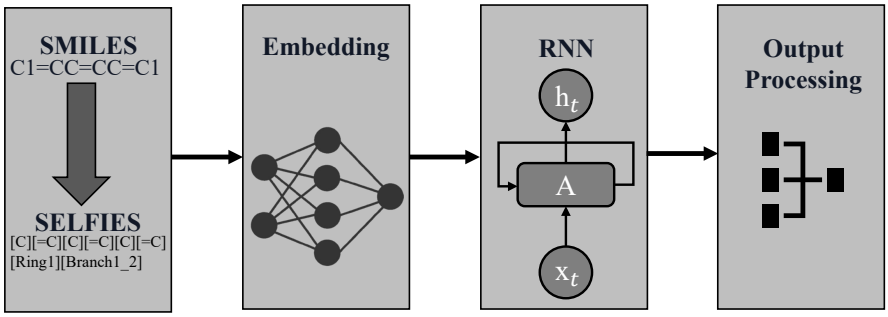


Figure 5.3: Overview of the RNN process.

output generation gives the RNN a chance of learning the molecular context. Fig. 5.1 and Fig. 5.2 are visualizations of the RNN architectures used to process the SELFIES. RNNs historically use the Tanh activation function, but we use the LeakyReLU as it reduces saturation possibilities and typically results in higher performance [88, 89]. In addition to this, we also include a dropout layer on the output of the RNN which helps prevent overfitting and reduce the error rate of RNNs [90]. After processing the SELFIES through the RNN, the final state should have all important prior information encoded into it. This vector then passes through an additional LeakyReLU and dropout layer before being fed to a fully connected layer. The fully connected layer reduces the vector from the dictionary-sized dimension down to the number of classes present in the molecular property. Subsequently, a soft-max operation finds the most likely class.

5.4 Analyzing and Comparing Model Performances

5.4.1 Results

Before training on the selected MoleculeNet datasets referenced in Section 5.2.1, we perform an additional reduction to the dataset by setting the lower bound of 31 molecules to the SMILES string allowing for the search space to remain sufficiently complex while reducing the overall run time. The lower bound reduces the datasets before stratified splitting the data using 80% for training and 20% for testing [91]. The stratified splitting intends to maintain the known sample rate of a given side effect to model real-world testing. However, during training, we want to remove the sampling bias to ensure our model accurately learns the causes of a side effect. The minority samples within the training set are duplicated to have an even sample count between the side effect present and the side effect not present to reduce the sampling bias. After replicating training samples, the SMILES conversion to SELFIES occurs. Typical natural language processing (NLP) methods use a word, sub-word, or character tokenization to convert

strings into numerical values, but we opt for a slightly different method, which we explain by referring to equation 5.7. It shows the SELFIES representation of benzene where each molecule and structural element are between brackets. Using this representation, we decide to tokenize based on each set of brackets that exist within the SELFIES converted dataset. This results in a total of 47 unique values. After tokenizing the SELFIES, the embedding dimension, input dimension of the RNN, and the hidden dimension of the RNN are set to a size of 47 to match the dimensional space of the tokens. To give the RNN model the best opportunity to make accurate classifications, we use a single model to perform a single side effect classification prediction. For SIDER, instead of predicting all 27 potential side effect classifications, we opt to predict 20 side effect classifications due to extreme imbalances present in the side effect data. The vanilla RNN architecture results in a model with 11.5K parameters and the GRU architecture results in a model with 18.8k parameters. Both train in under 2 minutes on an Nvidia GeForce RTX 3090. To compare our performance with other works that use MoleculeNet we evaluate using the suggested metric, the receiver operating characteristic curve (ROC) [61, 92].

While ROC is helpful for comparison, it is commonly misunderstood [93, 94] so we include a small sample of 2 training/testing accuracy and loss curves in Fig. 5.4 as a simple spot check of model performance. Examining Fig. 5.4, we note that training and testing loss is decreasing across all three side effect properties. There are spikes within each of the loss curves, but this is known to have occurred since the inception of RNNs [75]. The training loss for all three side effects saturate faster than the testing. There can be some gap in performance in loss based on the difficulty of new samples, but the gap here is likely accentuated as an unfortunate side effect of the minority sample duplication process. The duplicate samples within the training set help the model learn what molecular components help detect a side effect, but during training, the repeated samples become easier to predict for the model. In the case of accuracy, both training and testing show an upward trending curve where improvement starts to attenuate between the 20th-40th epoch. This attenuation roughly matches the attenuation that occurs with the loss curves. Comparing training and testing accuracy there appears to be a roughly 20+% gap in performance at nearly every epoch, which we again attribute to the duplicate samples within the training set.

$$[C][=C][C][=C][C][=C][Ring1][=Branch1] \tag{5.7}$$

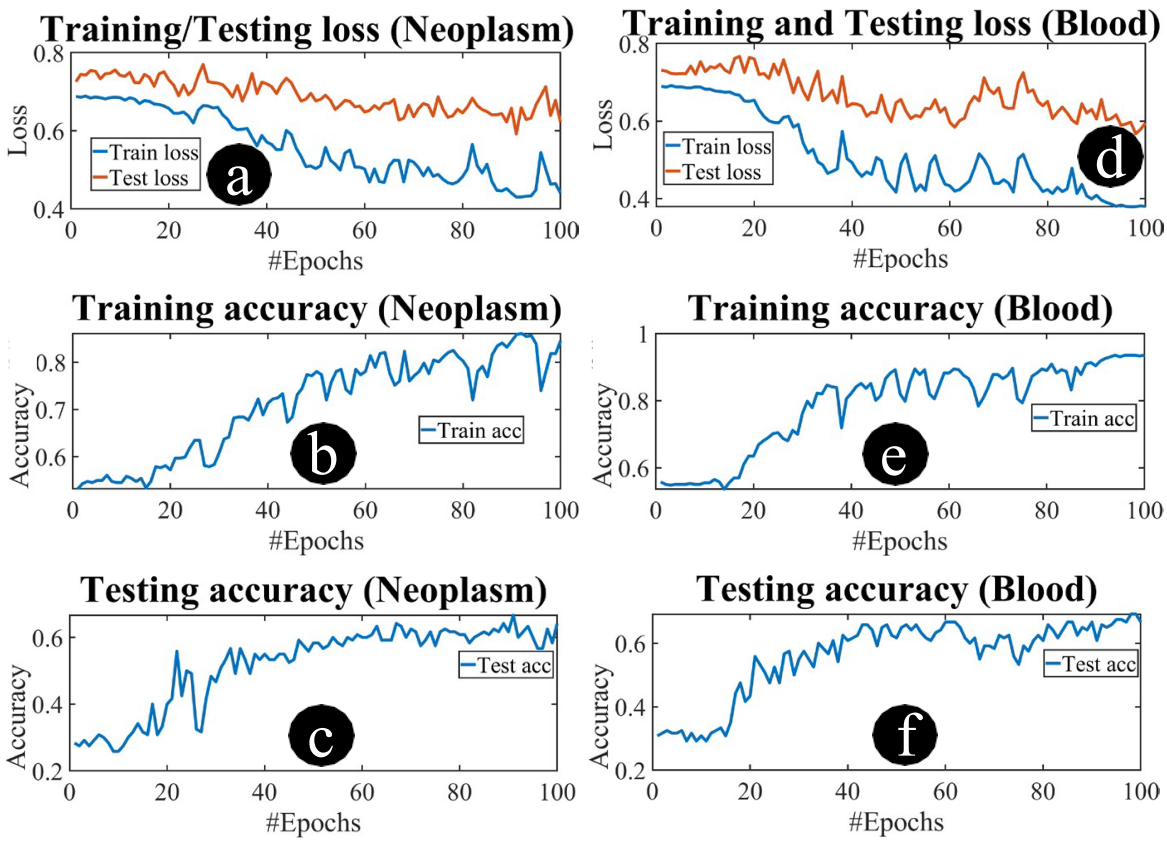


Figure 5.4: Results of three tasks: (a) loss curves, (b) training accuracy, (c) testing accuracy for neoplasms benign, malignant and unspecified (incl cysts and polyps) disorders, (d), (e), (f) for blood and lymphatic system disorders.

5.4.2 Comparisons

To understand the performance of the proposed approach, we compare it across multiple datasets to two top-performing GNN models, ChemRL-GEM [83] and GROVER$_{large}$ [77], and a top-performing NLP model, Galactica [68]. In addition to the top-performing large models, we include the random forest and GCN model from MoleculeNet [69], the DMPNN model [61], and the pre-trained GIN model [65]. For each heuristic model we evaluate we train using 20 different random seeds and evaluate the model by taking the top 3 performing ROC scores per metric. We include standard deviation as a way to account for uncertainty in model performance. Overall results are shown in Table 5.1. Beginning with the SIDER test, the results in Table 5.1 show our approach using the GRU achieves SOTA performance with a 17.8% higher performance over the best model not using the proposed method (RF [69]). While there are no direct statistics available for ChemRL-GEM, we use roughly 99.7% fewer parameters than its follow-up work, LiteGEM [83, 84]. It is worth noting that applying our proposed approach to a CNN network offers a 17.3% higher performance over RF [69]. For the

Table 5.1: Table of ROC performance per molecular property prediction across datasets.

Model	RNN	bi-RNN	GRU	CNN	MLP	ChemRL-GEM [83]	GROVER$_{large}$ [77]	Galactica [68]	RF [69]	GCN [69]	DMPNN [61]	Pre-trained GIN [65]
SIDER	*.557 ± .05*	*.56 ± .071*	**.818 ± .084**	**.814 ± .059**	**.731 ± .063**	.672 ± .004	.658 ± .023	.632	.684 ± .009	.638 ± .012	.676 ± .014	.627 ± .08
BBBP	*.643 ± .133*	*.5 ± 0*	*.937 ± .025*	*.923 ± .068*	*.681 ± .039*	.724 ± .004	.940 ± .019	.661	.714 ± .000	.690 ± .009	.737 ± .001	.687 ± .013
Clintox	*.637 ± .101*	*.6 ± .17*	**.98 ± .012**	**.976 ± .021**	*.860 ± .029*	.901 ± .013	.944 ± .021	.826	.713 ± .056	.807 ± .047	.864 ± .017	.726 ± .015
BACE	*.623 ± .015*	*.733 ± .196*	**.94 ± .006**	**.923 ± .006**	**.943 ± .017**	.856 ± .011	.894 ± .028	.617	.867 ± .008	.783 ± .014	.852 ± .053	.845 ± .007
HIV	*.583 ± .032*	*.61 ± .095*	**.653 ± .031**	**.713 ± .081**	**.683 ± .041**	-	-	.632	-	.763 ± .016	.776 ± .007	.799 ± .007
MUV	*.621 ± .089*	*.627 ± .107*	*.923 ± .034*	*.931 ± .037*	*.867 ± .058*	-	-	-	-	.046 ± .031	.041 ± .007	.813 ± .021
Pre-train data	-	-	-	-	-	18M Mols	10M Mols	60M Docs	-	-	-	2.4M Mols
Rep/Est Train Time	∼ 2m	∼ 2m	∼ 2m	∼ 2m	∼ 2m	∼ 10D	4D	30D	-	-	-	-
GPU Req.	RTX3090	RTX3090	RTX3090	RTX3090	RTX3090	∼ 1 V100	250 V100	512 A100	-	-	-	-

BBBP test, the GRU outperforms ChemRL-GEM and Galactica but performs 0.32% worse than GROVER$_{large}$. While it may be possible that GROVER achieves better performance due to their usage of graph representation, it more likely stems from having 100M parameters, over 5,000x more parameters than our model [77]. For the Clintox test, our performance was again the best of all the models. This test is one of Galactica's best performances, yet we can achieve a 17.05% higher performance with 6Mx less parameters [68]. Comparing the GRU approach to the best performing approach for Clintox and BACE, GROVER$_{large}$ [77], it achieves a 3.74% better performance for Clintox and a 5.01% better performance for BACE.

Further comparing the models we include the pre-train data requirement, the train time of the model and the GPU requirement to achieve the listed runtime in Table 5.1. Additional evaluations of the accuracy, precision, recall, and Wilcoxon statistics can be found in Appendix 1 and Appendix 2.

5.5 Conclusion

While large data models may offer coverage of larger molecular lengths, we have shown that small models (specifically GRUs) are still viable candidates for molecular property prediction. Smaller models are cheaper, more practical, and more accessible solutions as they don't require multiple GPUs and several days of training prior to obtaining a result. There are limitations with RNN based models, but when these limitations are carefully considered and more descriptive languages, such as SELFIES [74], are used RNNs offer SOTA/near SOTA results.

5.5.1 Clinical Insights

Property prediction models allow chemists to perform molecular evaluations prior to physical experimentation. Effective property evaluation can prevent months of wet lab research being spent on molecules that will not be feasible. Reducing failures realized during synthesis has the potential to greatly reduce the drug to market run time, enabling clinical researchers to rapidly treat patients for their medical conditions.

5.5.2 Constraints

Despite the RNN's learning capabilities and ability to process variable length input with no additional parameters required, using such an architecture does have drawbacks. RNN models can scale when dealing with large datasets via batching or even model parallelization, but they do not scale well when considering larger input sequences. Theoretically, RNN models can process large sequences of information with no problem, but in practice, RNNs can suffer from vanishing or exploding gradients causing them to "forget" important information. Even if we could implement the perfect memory model, the RNN still suffers from long run times where each addition to the sequence increases the run time due to the sequential nature of recurrence. One possible method to mitigate the long run time would be chunking, where the sequences are partitioned into smaller processable pieces. Unfortunately, this is unreliable, as sometimes vital state information may be separated from chunks causing inaccurate results.

5.5.3 Ethical Statement

While machine learning models can help identify potential molecular properties, they are not without flaws. Even if machine learning models can accurately identify all molecular properties of the datasets, they are trained with are fully dependent on previous human discoveries. The datasets are subject to flawed understandings of chemistry and even political choices. For example, the NIH only classifies drugs as toxic to the liver after successfully ruling out other potential causes [3]. Therefore, machine learning models should only be used for preliminary evaluation of molecules and not as the only form of molecular evaluation.

[3] https://www.ncbi.nlm.nih.gov/books/NBK548049/

Chapter 6
Evaluating Effects of Augmented SELFIES for Molecular Understanding Using QK-LSTM

Identifying molecular properties, including side effects, is a critical yet time-consuming step in drug development. Failing to detect these side effects before regulatory submission can result in significant financial losses and production delays, and overlooking them during the regulatory review can lead to catastrophic consequences. This challenge presents an opportunity for innovative machine learning approaches, particularly hybrid quantum-classical models like the Quantum Kernel-Based Long Short-Term Memory (QK-LSTM) network. The QK-LSTM integrates quantum kernel functions into the classical LSTM framework, enabling the capture of complex, non-linear patterns in sequential data. By mapping input data into a high-dimensional quantum feature space, the QK-LSTM model reduces the need for large parameter sets, allowing for model compression without sacrificing accuracy in sequence-based tasks. Recent advancements have been made in the classical domain using augmented variations of the Simplified Molecular Line-Entry System (SMILES). However, to the best of our knowledge, no research has explored the impact of augmented SMILES in the quantum domain, nor the role of augmented Self-Referencing Embedded Strings (SELFIES) in either classical or hybrid quantum-classical settings. This study presents the first analysis of these approaches, providing novel insights into their potential for enhancing molecular property prediction and side effect identification. Results reveal that augmenting SELFIES yields in statistically significant improvements from SMILES by a 5.97% improvement for the classical domain and a 5.91% improvement for the hybrid quantum-classical domain.

6.1 Introduction

Molecular property prediction is a cornerstone of drug discovery [61, 62]. In silico methods for predicting molecular properties hold the potential to accelerate the development of safer drugs by reducing testing time and costs. By identifying molecular properties early in the development process, researchers can efficiently create novel materials with greater confidence. Moreover, detecting side effects before drug release can prevent unnecessary harm, ultimately saving lives.

Traditionally, in silico approaches relied on complex feature engineering techniques to generate molecular representations for processing [63, 64]. However, the limitations of these descriptor-based methods often hinder their effectiveness. The generated features are highly task-dependent, meaning they may not be transferable across different problems. Additionally, as molecular understanding evolves, previously used feature vectors may become obsolete, reducing utility.

Graph Neural Networks (GNNs) have emerged as a promising alternative, overcoming many of the limitations of traditional feature engineering. GNNs are particularly well-suited for molecular property prediction due to their ability to model molecules as graphs, a representation that naturally captures the structure of molecules. This graph-based approach removes the reliance on predefined descriptors, allowing the model to learn representations directly from the data. As a result, GNNs have shown strong performance across a range of cheminformatics tasks [65, 66]. Despite advances, GNNs face challenges, including difficulties capturing shared dependencies and scalability issues. As the size of the molecular graph increases, so does the computational complexity, with the communication cost between nodes growing exponentially. Consequently, GNNs can sometimes perform worse than other neural network architectures despite their versatile input representation [67].

Building on the success of large language models in other fields, recent research has explored language-based approaches for molecular property prediction with promising results [68, 95]. Although which model architecture is best suited for chemoinformatics [96] remains an open question, language models offer several advantages that make them attractive for researchers. Notable advancements include the use of large pre-trained datasets [80, 97], multi-modal techniques [98], data augmentation strategies [99, 100], hybrid quantum-classical machine learning [1], and combinations of these approaches [96, 101].

As the focus on language models continues to grow, understanding how to optimize these models is crucial. While prior studies have explored the impact of augmenting SMILES for classical networks, no research has examined the effects of augmenting SELFIES [102], a

language specifically designed to enhance machine understanding of chemical structures to the best of our knowledge. Moreover, no studies have investigated the potential impact of such augmentations in hybrid quantum-classical models. *This study presents the first analysis of augmented SELFIES in classical and hybrid quantum-classical domains to address these gaps*.

In the following sections, we review the necessary background on MoleculeNet benchmark [69], SMILES and SELFIES formats, Long Short-Term Memory, and few of the related quantum networks (Section 6.2). We then discuss the methods used to train the models and their results (Section 6.3), followed by a review of related works (Section 6.4). Finally, we provide concluding remarks (Section 6.5).

6.2 Preliminaries on Molecular Information and Quantum Models

6.2.1 MoleculeNet Benchmark

MoleculeNet is a comprehensive benchmark for evaluating machine learning models in cheminformatics [69]. It curates datasets spanning quantum mechanics, physical chemistry, biophysics, and physiology. MoleculeNet establishes a preferred evaluation metric for each dataset, ensuring consistent and fair comparisons across different models. In this chapter, we describe the datasets chosen to assess the performance of our model.

6.2.2 Side Effect Resource (SIDER)

A critical molecular property of drugs intended for human consumption is their associated side effects, which can significantly impact safety and efficacy. The Side Effect Resource (SIDER) dataset comprehensively collects publicly available data on known drug side effects [70]. This dataset compiles side effect information from various sources, offering a unified resource for understanding the adverse effects of drugs.

The SIDER dataset consists of 28 columns. The first column contains the SMILES representation of each drug molecule, providing a standardized chemical structure notation. The remaining 27 columns represent different organ classes affected by side effects, with side effects classified according to the Medical Dictionary for Regulatory Activities (MedDRA) [1]. Each side effect is encoded with a binary value: "1" indicates that the drug is known to cause a side effect in the corresponding organ class, while "0" indicates the absence of that side

[1] https://www.meddra.org/

effect. This structure enables systematic analysis and model training for predicting adverse drug reactions based on molecular properties.

6.2.3 ROC-AUC

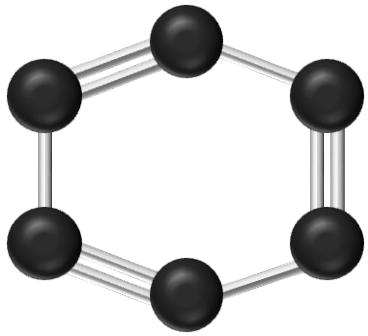
The Receiver Operating Characteristic (ROC) curve is used to evaluate the performance of a binary classifier by calculating the true positive rate (TPR) against the false positive rate (FPR) across various threshold settings. The true positive rate measures the proportion of actual positives correctly identified by the model. In contrast, the false positive rate represents the proportion of negative instances incorrectly classified as positive. The ROC curve provides a comprehensive view of a model's ability to discriminate between the two classes, with the area under the curve (AUC) serving as a summary statistic that quantifies overall performance. The ROC-AUC score, which represents the area under the ROC curve, is particularly valuable when evaluating models trained on imbalanced datasets, as it accounts for the classifier's ability to identify positive instances and avoid misclassifying negative instances correctly. This makes ROC-AUC an ideal metric for assessing models on datasets like those found in MoleculeNet, where the class imbalance is often present [103].

6.2.4 Simplified Molecular-Input Line Entry System (SMILES)

SMILES is a notation used to represent molecular structures using a compact string of characters [53]. In SMILES, atoms are represented by letters, where the first letter of an element is uppercase to indicate it is non-aromatic and lowercase to signify it is aromatic. When an element has two letters, the second letter is always lowercase, regardless of aromaticity. Aromaticity can also be denoted by a colon (:) symbol, representing aromatic bonds.

Bond types in SMILES are depicted using specific symbols. The period (.) represents a bond with no connection (isolated atoms), the hyphen (-) indicates a single bond, and the forward-slash (/) and backslash (\) are used to represent single bonds adjacent to a double bond, primarily in stereochemical molecules. The equal sign (=) signifies a double bond, the octothorpe (#) represents a triple bond, and the dollar sign ($) is used for quadruple bonds.

In stereochemical representations, the "@" symbol is used to denote chirality: a single "@" indicates counterclockwise configuration, while a double "@" denotes clockwise configuration. Additionally, numbers in SMILES denote ring structures by marking the positions where the ring starts and closes. When elements are enclosed in brackets, the number inside indicates the number of atoms involved in the substitution or structure. Isotopes of atoms can be specified by placing a number before the element in square brackets.



Molecule: *Benzene*
SMILES: *c1ccccc1*
SELFIES: *[C][=C][C][=C][C][=C][Ring1][Branch1_2]*

Figure 6.1: Sample benzene molecule that relates to both the SMILES and SELFIES samples supplied

Parentheses (()) indicate branches from the main chain, allowing for more complex molecular structures to be represented. The SMILES notation enables efficient and compact molecular representation, facilitating computational processing and analysis in cheminformatics. An example SMILES representation of benzene, shown in Fig. 6.1.

6.2.5 Self-Referencing Embedded Strings (SELFIES)

SELFIES provide a more robust molecular representation for machine learning applications, building on the foundations of SMILES [74]. While SMILES offers a simple and interpretable way to encode molecular structures, it has limitations, particularly regarding spatial features. SMILES can represent atoms and bonds but rely on a complex grammar to describe features such as rings and branches, which are not locally represented. This complexity can lead to issues, especially in generative models, where machines often produce syntactically or physically invalid strings.

SELFIES addresses these shortcomings by simplifying the representation of spatial features. Instead of relying on complex grammar to handle rings and branches, SELFIES uses a single symbol to represent these structural elements. The length of the feature, such as the size of a ring or the extent of a branch, is explicitly encoded, ensuring that each SELFIES string is guaranteed to correspond to a valid molecular structure. This approach eliminates the risk of generating invalid or impossible molecules, making SELFIES a powerful tool for generative tasks and other machine-learning applications in cheminformatics. An example SELFIES representation of benzene, shown in Fig. 6.1.

6.2.6 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks [85] is a specialized type of Recurrent Neural Network (RNN) [75] designed to address the limitations of standard RNNs, particularly in handling long-term dependencies in sequential data. Unlike traditional RNNs, which struggle to maintain information over long sequences due to issues such as vanishing or exploding gradients, LSTMs have memory cells that can retain information over extended periods, enabling LSTMs to capture and store temporal information across time steps.

RNNs have an inherent ability to maintain state information, but there is no mechanism in basic RNNs to selectively decide which pieces of information are kept and for how long. This can lead to context saturation, where irrelevant information accumulates, making it difficult for the network to focus on more pertinent data. LSTMs address this problem by introducing specialized gating mechanisms, namely, the input, forget, and output gates, which allow the network to control the flow of information. These gates enable the LSTM to decide when to add new information, remove outdated data, and update its internal state, thus mitigating the saturation issues seen in traditional RNNs and improving performance on tasks requiring long-range dependencies.

6.2.7 Quantum Neural Networks (QNN)

QNNs comprises of three core components: a data encoding circuit, a Variational Quantum Circuit (VQC), and measurement operations. Classical data is first transformed into a quantum state using amplitude, basis, or angle encoding schemes. The VQC, defined by a parameterized ansatz, manipulates this quantum state using single- and two-qubit gates, with entanglement enabling the capture of complex feature correlations [41]. Finally, measurement operations extract information via quantum observables. During training, VQC parameters are optimized to minimize a task-specific loss function, enabling QNNs to perform machine learning tasks. Integrating quantum circuits with ML frameworks offers promising capabilities for handling high-dimensional and complex data.

6.2.8 Quantum Long Short-Term Memory (QLSTM)

Despite their promise, prior quantum models such as QNNs struggle to incorporate selective memory mechanisms, key elements of classical machine learning models like LSTM networks. The QLSTM model is proposed to address this, extending the classical LSTM's ability to selectively store and recall contextual information from previous inputs to the quantum realm.

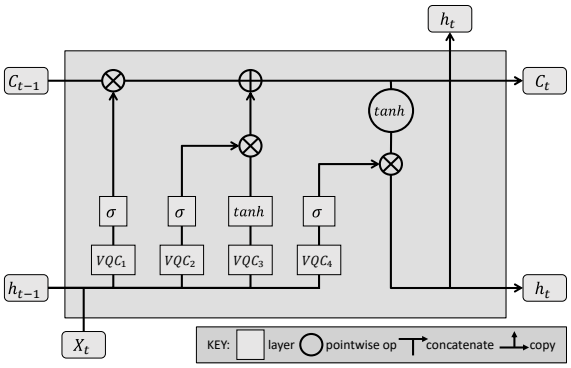


Figure 6.2: QK-LSTM architecture used for training; (C_{t-1}, C_t) represent the cell state, (X_t) represents the input, (h_{t-1}, h_t) represent the output state. The VQC_1 wire represents the forget gate, deciding if the input should be added to memory. The (VQC_2, VQC_3) wires represent the update gate, updating the cell memory if needed. The VQC_4 wire represents the output gate, outputting the result of the QK-LSTM to the rest of the model.

The QLSTM architecture closely mirrors the classical LSTM but rather than directly processing information through traditionally hidden layers, QLSTM passes input data through a VQC. This enables quantum entanglement of the values, which is then measured and processed in the same manner as the classical LSTM's prediction structure [50]. This modification allows QLSTM to exploit quantum phenomena, such as entanglement and superposition, to potentially enhance the model's ability to capture temporal dependencies in sequential data.

6.2.9 Quantum Kernel-Based LSTM (QK-LSTM)

The QK-LSTM [104] enhances the QLSTM framework by integrating quantum kernels, which map data into higher-dimensional feature spaces [105], enabling more expressive representations with fewer trainable quantum parameters. This hybrid approach combines quantum data transformations with classical LSTM mechanisms, improving performance in tasks such as time-series prediction and NLP.

As shown in Fig.6.2, inputs like X_t, h_t, and C_t are processed through a sequence of quantum kernels based on entangler circuits (Fig.6.3). These kernels enrich the input by leveraging quantum feature maps, followed by a fully connected layer that reduces dimensionality for classical processing. The enhanced data is passed through standard LSTM components: a forget gate (sigmoid) determines whether to retain the previous context, an input gate (sigmoid and tanh) decides what new information to store, and an output gate (sigmoid) generates predictions and updates the hidden state. This quantum-classical architecture enables QK-LSTM to effectively model complex temporal dependencies while leveraging quantum-enhanced feature extraction for improved sequence learning performance.

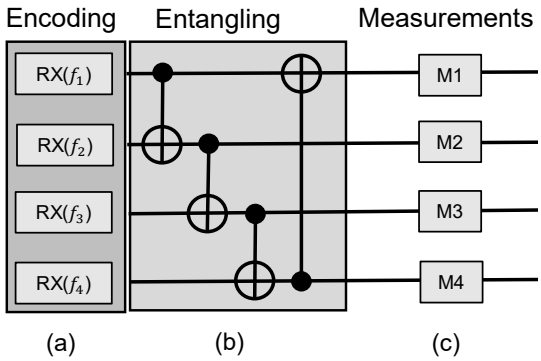


Figure 6.3: QK-LSTM basic entangler circuit; (a) angle encoding converts classical features (f_1, f_4, f_3, f_4) to quantum states, (b) parametric quantum circuit entangles quantum states, (c) qubits measured, and bloated to original higher-dimension space. [6].

6.3 QKLSTM: Methods & Results

6.3.1 Data Pre-processing

The MoleculeNet benchmark dataset [69] represents molecules using the SMILES notation. Upon inspection, we observed that not all SMILES strings in the dataset are canonical. Non-canonical SMILES introduce additional complexity due to the inherent syntactic variability of SMILES representations. To mitigate this, all molecular strings are converted to their canonical forms using RDKit, thereby reducing redundancy and improving consistency in downstream learning.

6.3.1.1 Augmentation

Following canonicalization, SMILES augmentation is optionally applied using augmentation techniques proposed by Bjerrum [100]. This technique generates alternative SMILES strings for the same molecule by rearranging the atom and bond sequences while preserving chemical validity. Augmentation increases the diversity of the training data, aiding in model generalization without altering the underlying molecular structure.

6.3.1.2 SELFIES Conversion

To further alleviate SMILES's syntactic complexity and facilitate more robust sequence learning, the canonical and optionally augmented SMILES strings are converted into SELFIES representations [74]. SELFIES provide a semantically constrained encoding that ensures the generation of valid molecules. Each SELFIES token, representing atoms or structural motifs

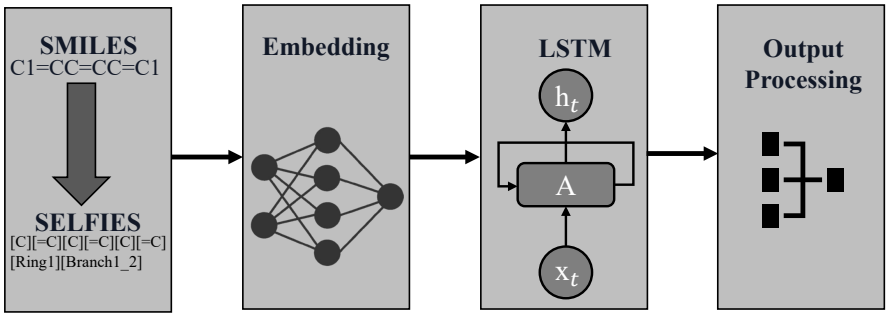


Figure 6.4: Overview of the LSTM process.

such as branches and rings, is mapped to a numerical index based on a predefined vocabulary. This sequence of indices serves as the input to the LSTM-based model.

6.3.2 LSTM Implementation

The pre-processed molecular representations train a recurrent neural network based on the LSTM architecture. Each molecule, represented as a SMILES or SELFIES sequence, is first mapped through an embedding layer, where each token is embedded into a continuous vector space. The embedding dimension corresponds to the size of the molecular vocabulary.

The embedded sequence is then passed into an LSTM network. The LSTM's input and output dimensions match the embedding size, while the hidden state dimensionality is selected empirically to explore different model capacities. The LSTM processes the molecular sequence step-by-step, encoding structural dependencies into its hidden state. The final hidden state is expected to capture the relevant molecular features.

This final representation is fed into a fully connected layer that projects the hidden state to a lower-dimensional space corresponding to the number of output classes—typically, the molecular property to be predicted. This setup lets the model learn end-to-end mappings from molecular structures to target properties. An overview of this process is in Fig. 6.4.

6.3.3 Results

Ideally, we would analyze the performance of QK-LSTM across all four variations: SMILES augmented SMILES, SELFIES, and augmented SELFIES. However, the computational cost of running these variations is prohibitive. Therefore, we focus on the two extreme variations, namely SMILES and augmented SELFIES, while comprehensively analyzing the classical model to allow for meaningful abstraction of the results. To evaluate QK-LSTM performance effectively, we employed Optuna [106] for hyperparameter optimization, randomly selecting four different LSTM and QK-LSTM configurations for training. Each configuration's top three

performing models were selected, and the ROC-AUC scores were averaged to derive the final performance metric.

For data augmentation, we followed the recommendation of Bjerrum et al. [107], generating five augmented samples per original data point, where possible. While increasing the number of augmentations could improve the ROC-AUC, it would significantly increase training time. In practice, we generated 20 augmentation samples and selected the five shortest to minimize training time cost.

Following data augmentation, the conversion from SMILES to SELFIES is performed. SMILES are tokenized at the character level, while SELFIES are tokenized at the bracket level, transforming the strings into numerical representations that are subsequently fed into an embedding layer.

The data split is performed randomly following the methodology suggested by MoleculeNet [69], with 80% of the data allocated for training, 10% for validation, and 10% for testing.

Each model is limited to a maximum of 30 epochs to mitigate the impact of excessive training time. Early stopping is applied if the validation ROC-AUC did not improve over 10 consecutive epochs. We employed a learning rate reduction after five epochs of non-improvement in ROC-AUC to further optimize performance.

Optuna explored the hidden dimension space for the LSTM and QK-LSTM models for hyperparameter optimization. The LSTM's hidden dimension is allowed to range from 32 to 128, with a step size of 16, while the QK-LSTM's hidden dimension varied from 8 to 32, with a step size of 4, resulting in qubit sizes between 3 and 5. The smaller qubit count is chosen to balance reasonable runtime with a realistic comparison of quantum computers' current capabilities against classical implementations. We used a single model for each side effect classification task to ensure accuracy. The models are trained on an Nvidia GeForce RTX 4090 with an Intel i9-13900K. The final results are averaged across all runs and summarized in Table 6.1.

6.3.4 Comparisons

To understand the performance changes from augmentation, we compare the SMILES to augmented SMILES and SELFIES to augmented SELFIES. From Table 6.1, we note a 0.04 increase in the ROC-AUC score (which is outside the standard deviation) when switching between SMILES to augmented SMILES. We get a 0.049 increase in the ROC-AUC score (which is outside the standard deviation as well) for switching between SELFIES and augmented SELFIES. The performance gain for SMILES and SELFIES shows the benefit of increasing the number of available samples to train with, even if they are just variations of the same molecule,

Table 6.1: ROC-AUC Test Score From LSTMs

Setup	ROC-AUC
LSTM SMILES	0.525 ± 0.023
LSTM Augmented SMILES	0.562 ± 0.004
LSTM SELFIES	0.507 ± 0.026
LSTM Augmented SELFIES	0.556 ± 0.013
QK-LSTM SMILES	0.524 ± 0.022
QK-LSTM Augmented SELFIES	0.555 ± 0.009

as this allows the model to build a more robust understanding of the molecular plane.

To quantify the performance changes from using SELFIES, we compare the SMILES to SELFIES for the standard dataset and the augmented data set. From Table 6.1, we note a 0.022 worse performance (that is within the standard deviation) when switching between SMILES to SELFIES. As for switching between augmented SMILES and augmented SELFIES, we get a 0.006 worse performance change (again within the standard deviation). This shows that we continue to see a non-statistically significant performance change as we increase the size of our dataset. This is problematic for the future usage of SELFIES in molecular property prediction as we continue to build larger and more robust molecular datasets.

We compare SMILES and augmented SELFIES for the LSTM and QK-LSTM to quantify the effect of switching between the classical and quantum domains. Analyzing Table 6.1, we note a 0.001 performance loss (that is within the standard deviation) due to switching to the quantum domain for SMILES. As for switching to quantum for augmented SELFIES, we get a 0.001 performance loss (that is within the standard deviation). While quantum cannot outperform the classical domain, we do see the model achieving similar performance gains between SMILES and augmented SELFIES. This is promising for the future of quantum as we continue to advance quantum machine capabilities.

6.4 QKLSTM: Related Works

6.4.1 Language Models for Cheminformatics

6.4.1.1 Classical Models

Prior to Liu et al. [43], most works required a structured rules-based approach to predict properties associated with molecules; Liu et al. removed this requirement and achieved similar

performance by using LSTMs to make predictions. Unfortunately, it did not result in any significant improvement in state-of-the-art results.

With the advent of the transformer [58], Liu et al.'s proof of concept and the availability of more standardized datasets, such as MoleculeNet [69], many works attempt to leverage the abilities of deep neural networks to attempt to achieve state-of-the-art performance in cheminformatics [68, 77, 80, 83, 84, 96, 97, 108, 109]. While deep learning networks generally show improvement, they come with the cost of exponentially larger datasets and parameter counts.

6.4.1.2 Quantum Models

To overcome the exponential increase of classical parameters, some works have explored the application of quantum machine learning models. Works like Wang et al. [110] attempt to use a QLSTM to identify potential adverse drug reactions. However, the model relies only on Twitter data, ignoring potentially helpful information from chemistry datasets. Works like Beaudoin et al. [1] implement hybrid quantum-classical machine learning models using real cheminformatic datasets to show the potential for hybrid quantum models to perform molecular property prediction, which has recently been confirmed by Zhang et al. [111]. Unfortunately, none of these works explore how these models can be further improved.

6.4.2 Representing Molecules

Representing molecular structures is pivotal in machine learning-driven drug discovery and cheminformatics. SMILES has become popular among various representations due to its compatibility with text-based deep learning models. However, SMILES is highly complex and limits neural networks' understanding of the input.

Researchers have explored data SMILES enumeration to overcome this limitation, which generates multiple valid SMILES strings for the same molecule by varying the atom order and traversal path. This approach significantly enhances model robustness and generalization. Bjerrum demonstrated that augmenting datasets with randomized SMILES improves the performance of neural networks on molecular property prediction tasks, reducing the risk of overfitting and improving accuracy on unseen molecules [100, 112]. However, to our knowledge, no works explore how augmentation can improve SELFIES.

6.5 Discussion & Conclusion

While classical models may offer non-statistically significant better performance over near-term hybrid quantum-classical models, we have shown that quantum models are viable candidates for molecular property prediction. Additionally, we have shown that we can leverage classical techniques, such as augmentation, to improve hybrid quantum-classical models further. This will be increasingly important as we continue to scale up our data sizes, as reflected by the results shown by the LSTM.

Interestingly, we see no statistically significant performance gains by converting from SMILES to SELFIES. While SELFIES does guarantee that molecules produced are always real, it does not simplify the learning process for molecules. Guo et al. found that using SELFIES may even result in lower performance for LLMs [113], but we did not find this to be the case. However, further evaluation using additional datasets may be warranted for confirmation. The promising news is that the augmentation of SELFIES offers similar performance gains to that of SMILES.

6.5.1 Clinical Insights

Property prediction models allow chemists to perform molecular evaluations prior to physical experimentation. Practical property evaluation can prevent months of wet lab research from being spent on molecules that will not be viable, or may be harmful. Reducing failures realized during synthesis has the potential to significantly reduce the drug-to-market run time, enabling clinical researchers to treat patients for their medical conditions rapidly.

6.5.2 Constraints

Despite LSTM's capabilities and ability to process variable length input with no additional parameters required, such an architecture has drawbacks. LSTM models can scale when dealing with large datasets via batching or even model parallelization, but they do not scale well when considering larger input sequences. Theoretically, LSTM models can process large sequences of information with no problem, but in practice, LSTMs can suffer from vanishing or exploding gradients just like classic RNNs, causing them to "forget" important information. Even if we could implement the perfect memory model, the LSTM would still suffers from long run times where each addition to the sequence increases the run time due to the sequential nature of recurrence. One possible method to mitigate the long run time would be chunking, where the sequences are partitioned into smaller processable pieces. Unfortunately, this is unreliable, as

sometimes vital state information may be separated from chunks, causing inaccurate results.

6.5.3 Ethical Statement

While machine learning models can help identify potential molecular properties, they have flaws. Even if machine learning models can accurately identify all molecular properties of the datasets they are trained with, they depend entirely on previous human discoveries. The properties associated with molecules are subject to our current understanding of them. For neuvo molecular designs we simply do not have enough understanding of chemistry to be certain of our predictions. The datasets used for property identification are subject to flawed understandings of chemistry and even political choices. For example, the NIH only classifies drugs as toxic to the liver after successfully ruling out other potential causes [2]. Therefore, machine learning models should only be used for the preliminary evaluation of molecules and not as the only form of molecular evaluation.

[2]https://www.ncbi.nlm.nih.gov/books/NBK548049/

Chapter 7
Concluding Remarks

This thesis presents a unified framework for advancing quantum computing and material discovery through efficient, domain-aware classical and hybrid quantum-classical machine learning models. We demonstrate that meaningful progress in quantum circuit optimization and molecular discovery can be achieved without resorting to increasingly large and resource-intensive models. Instead, we advocate for smarter model design, informed architecture selection, and data-centric approaches as viable and often superior alternatives.

This thesis advanced quantum algorithm synthesis by introducing generative, graph-based models for quantum circuit optimization. These models significantly reduced gate counts and circuit depth, outperforming current state-of-the-art methods and highlighting the potential of generative approaches in circuit design.

We also introduced the first graph-based diffusion model for generating novel quantum circuits, an important step toward automating quantum circuit design and development.

In molecular synthesis, we introduced and evaluated hybrid QLSTM models for retrosynthesis. Despite the limitations of noisy intermediate-scale quantum (NISQ) hardware, our results show that QLSTMs can identify molecular substructures with competitive accuracy. While classical models outperform quantum models in absolute metrics, QLSTMs surpass classical models in specific prediction tasks, indicating that quantum circuits can offer non-trivial representational benefits even with minimal qubit resources.

We further explored the role of small, well-designed recurrent neural networks (RNNs) in molecular property prediction, challenging the prevailing trend of scaling up model size for marginal accuracy gains. Our findings suggest that small models, particularly GRUs when paired with expressive molecular representations like SELFIES and enhanced with strategic data augmentation, can achieve near state-of-the-art performance at a fraction of the computational cost.

The thesis establishes that QK-LSTMs, enhanced through classical techniques like data

augmentation, are credible alternatives for chemical property prediction. Though performance differences compared to classical baselines are often not statistically significant, the hybrid approach offers a flexible and scalable path forward, especially as larger datasets and quantum hardware become more accessible.

A critical insight from our comparative studies is that model performance is less dependent on raw size and more on context-aware design. While large language models (LLMs) offer broad generalization, smaller, task-specific architectures can be more effective when paired with problem-relevant datasets and optimization techniques.

This thesis presents scalable, efficient machine-learning strategies for scientific discovery. Leveraging domain knowledge demonstrates how both classical and quantum models can deliver high performance with minimal overhead. As quantum hardware and datasets grow, the approaches developed here provide a strong foundation for addressing key computational challenges in chemistry and quantum computing.

Appendix

1 Accuracy, Precision and Recall Statistics

Data	MLP	CNN	GRU	RNN	BiRNN
SIDER	.453 ± .052	.555 ± .039	.598 ± .049	.314 ± .126	.353 ± .156
BBBP	.346 ± .019	.587 ± .026	.641 ± .008	.129 ± .004	.331 ± .274
ClinTox	.546 ± .031	.797 ± .022	.824 ± .008	.102 ± .033	.241 ± .250
BACE	.695 ± .033	.657 ± .057	.808 ± .029	.404 ± .174	.174 ± .024
HIV	.525 ± .011	.529 ± .013	.513 ± .022	.476 ± .036	.486 ± .014
MUV	.581 ± .110	.683 ± .083	.478 ± .067	.304 ± .191	.311 ± .21

Model accuracy results using proposed method

Data	MLP	CNN	GRU	RNN	BiRNN
SIDER	.734 ± .317	.773 ± .272	.799 ± .214	.526 ± .399	.488 ± .413
BBBP	.845 ± .007	1 ± 0	1 ± 0	.0 ± 0	.6 ± .432
ClinTox	.295 ± .082	.639 ± .104	.667 ± .236	.174 ± .029	.083 ± .029
BACE	.444 ± .079	.528 ± .171	.704 ± .127	.248 ± .12	.207 ± .139
HIV	.536 ± .081	.677 ± .087	.806 ± .056	.739 ± .055	.604 ± .152
MUV	1 ± 0	.495 ± .209	.391 ± .093	.203 ± .140	.229 ± .186

Model precision results using proposed method

Data	MLP	CNN	GRU	RNN	BiRNN
SIDER	$.676 \pm .240$	$.757 \pm .189$	$.739 \pm .173$	$.438 \pm .404$	$.478 \pm .444$
BBBP	$.845 \pm .007$	$.75 \pm .029$	$.724 \pm .007$	0 ± 0	$.364 \pm .452$
ClinTox	1 ± 0	1 ± 0	1 ± 0	1 ± 0	1 ± 0
BACE	1 ± 0	1 ± 0	1 ± 0	$.833 \pm .236$	1 ± 0
HIV	$.736 \pm .086$	$.852 \pm .139$	$.606 \pm .048$	$.516 \pm .152$	$.616 \pm .118$
MUV	$.362 \pm .079$	1 ± 0	1 ± 0	$.899 \pm .199$	$.892 \pm .215$

Model recall results using proposed method

2 Wilcoxon Statistics

To understand the statistical difference between each of the models running with the proposed approach a Wilcoxon Signed-Rank test was performed for each model against the absolute best performance per dataset between the 3 SOTA models (Galactica [68], ChemRL-GEM [83],and GROVER$_{large}$ [77]).

For the GRU the Wilcoxon Signed-Rank test indicated that there is a non-significant large difference between GRU (Mdn = 0.9 ,n = 5) and Top SOTA Results (Mdn = 0.9 ,n = 5), W+ = 1, p = .125, r = -0.7.

For the CNN the Wilcoxon Signed-Rank test indicated that there is a non-significant large difference between CNN (Mdn = 0.9 ,n = 5) and Top SOTA Results (Mdn = 0.9 ,n = 5), W+ = 1, p = .125, r = -0.7.

For the MLP the Wilcoxon Signed-Rank test indicated that there is a non-significant small difference between MLP (Mdn = 0.7 ,n = 5) and Top SOTA Results (Mdn = 0.9 ,n = 5), W+ = 9, p = .813, r = 0.1

For the RNN the Wilcoxon Signed-Rank test indicated that there is a non-significant large difference between RNN (Mdn = 0.6 ,n = 5) and Top SOTA Results (Mdn = 0.9 ,n = 5), W+ = 15, p = .063, r = 0.8.

For the bidirectional RNN the Wilcoxon Signed-Rank test indicated that there is a non-significant large difference between Bidirectional RNN (Mdn = 0.6 ,n = 5) and Top SOTA Results (Mdn = 0.9 ,n = 5), W+ = 15, p = .063, r = 0.8.

In addition to the 3 SOTA models we also performed a Wilcoxon Signed-Rank test comparing the GRU vs MLP and the GRU vs CNN.

Comparing the GRU vs MLP the Wilcoxon Signed-Rank test indicated that there is a non-significant large difference between GRU (Mdn = 0.9 ,n = 6) and MLP (Mdn = 0.8 ,n = 6), W+ = 3, p = .156, r = -0.6.

Comparing the GRU vs CNN the Wilcoxon Signed-Rank test indicated that there is a non-significant very small difference between GRU (Mdn = 0.9 ,n = 6) and CNN (Mdn = 0.9 ,n = 6), Z = -0.2, p = .833, r = -0.09.

Bibliography

[1] BEAUDOIN, C., S. KUNDU, R. O. TOPALOGLU, and S. GHOSH (2022) "Quantum machine learning for material synthesis and hardware security," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–7.

[2] BEAUDOIN, C., K. PHALAK, and S. GHOSH (2024) "Predicting Side Effect of Drug Molecules Using Recurrent Neural Networks," *IEEE Transactions on Emerging Topics in Computational Intelligence*.

[3] BEAUDOIN, C. and S. GHOSH (2025), "Evaluating Effects of Augmented SELFIES for Molecular Understanding Using QK-LSTM," 2504.20789.
URL https://arxiv.org/abs/2504.20789

[4] BEAUDOIN, C., K. PHALAK, and S. GHOSH (2024) "AltGraph: Redesigning Quantum Circuits Using Generative Graph Models for Efficient Optimization," in *Proceedings of the Great Lakes Symposium on VLSI 2024*, pp. 44–49.

[5] BEAUDOIN, C. and S. GHOSH (2025), "Q-Fusion: Diffusing Quantum Circuits," 2504.20794.
URL https://arxiv.org/abs/2504.20794

[6] BERGHOLM, V., J. IZAAC, M. SCHULD, C. GOGOLIN, M. S. ALAM, S. AHMED, J. M. ARRAZOLA, C. BLANK, A. DELGADO, S. JAHANGIRI, ET AL. (2018) "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*.

[7] LI, Z. ET AL. (2023) "Quarl: A Learning-Based Quantum Circuit Optimizer," *arXiv preprint arXiv:2307.10120*.

[8] ANIS, M. S. ET AL. (2021) "Qiskit: An open-source framework for quantum computing," *Qiskit/qiskit*.

[9] SCHULMAN, J. ET AL. (2017) "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*.

[10] FÖSEL, T. ET AL. (2021) "Quantum circuit optimization with deep reinforcement learning," *arXiv preprint arXiv:2103.07585*.

[11] XU, M. ET AL. (2022) "Quartz: superoptimization of quantum circuits," in *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pp. 625–640.

[12] ZHOU, X. ET AL. (2020) "A Monte Carlo tree search framework for quantum circuit transformation," in *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–7.

[13] MOFLIC, I. ET AL. (2023) "Graph Neural Network Autoencoders for Efficient Quantum Circuit Optimisation," *arXiv preprint arXiv:2303.03280*.

[14] DUNCAN, R. ET AL. (2020) "Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus," *Quantum*, **4**, p. 279.

[15] ZHANG, M. ET AL. (2019) "D-vae: A variational autoencoder for directed acyclic graphs," *Advances in Neural Information Processing Systems*, **32**.

[16] LI, Y. ET AL. (2018) "Learning deep generative models of graphs," *arXiv preprint arXiv:1803.03324*.

[17] ——— (2015) "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*.

[18] KIPF, T. N. ET AL. (2016) "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*.

[19] LI, M., V. SHITOLE, E. CHIEN, C. MAN, Z. WANG, S. SRIDHARAN, Y. ZHANG, T. KRISHNA, and P. LI (2024) "LayerDAG: A Layerwise Autoregressive Diffusion Model for Directed Acyclic Graph Generation," *arXiv preprint arXiv:2411.02322*.

[20] PRESKILL, J. (2018) "Quantum computing in the NISQ era and beyond," *Quantum*, **2**, p. 79.

[21] RUBINSTEIN, B. (2001) "Evolving quantum circuits using genetic programming," in *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, vol. 1, IEEE, pp. 144–151.

[22] GEPP, A. and P. STOCKS (2009) "A review of procedures to evolve quantum algorithms," *Genetic programming and evolvable machines*, **10**, pp. 181–228.

[23] MARTYNIUK, D., J. JUNG, and A. PASCHKE (2024) "Quantum Architecture Search: A Survey," *arXiv preprint arXiv:2406.06210*.

[24] KUO, E.-J., Y.-L. L. FANG, and S. Y.-C. CHEN (2021) "Quantum architecture search via deep reinforcement learning," *arXiv preprint arXiv:2104.07715*.

[25] CHEN, S. Y.-C. (2023) "Asynchronous training of quantum reinforcement learning," *Procedia Computer Science*, **222**, pp. 321–330.

[26] LIANG, Z., J. CHENG, R. YANG, H. REN, Z. SONG, D. WU, X. QIAN, T. LI, and Y. SHI (2023) "Unleashing the potential of llms for quantum computing: A study in quantum architecture design," *arXiv preprint arXiv:2307.08191*.

[27] DUPUIS, N., L. BURATTI, S. VISHWAKARMA, A. V. FORRAT, D. KREMER, I. FARO, R. PURI, and J. CRUZ-BENITO (2024) "Qiskit Code Assistant: Training LLMs for generating Quantum Computing Code," *arXiv preprint arXiv:2405.19495*.

[28] HE, Z., J. WEI, C. CHEN, Z. HUANG, H. SITU, and L. LI (2024) "Gradient-based optimization for quantum architecture search," *Neural Networks*, **179**, p. 106508.

[29] SPECTOR, L., H. BARNUM, H. J. BERNSTEIN, and N. SWAMY (1999) "Finding a better-than-classical quantum AND/OR algorithm using genetic programming," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 3, IEEE, pp. 2239–2246.

[30] HE, Z., X. ZHANG, C. CHEN, Z. HUANG, Y. ZHOU, and H. SITU (2023) "A GNN-based predictor for quantum architecture search," *Quantum Information Processing*, **22**(2), p. 128.

[31] DU, Y., T. HUANG, S. YOU, M.-H. HSIEH, and D. TAO (2022) "Quantum circuit architecture search for variational quantum algorithms," *npj Quantum Information*, **8**(1), p. 62.

[32] WILLIAMS, C. P. and A. G. GRAY (1998) "Automated design of quantum circuits," in *NASA International Conference on Quantum Computing and Quantum Communications*, Springer, pp. 113–125.

[33] YABUKI, T. and H. IBA (2000) "Genetic algorithms for quantum circuit design-evolving a simpler teleportation circuit," in *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pp. 421–425.

[34] SOHL-DICKSTEIN, J., E. WEISS, N. MAHESWARANATHAN, and S. GANGULI (2015) "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, PMLR, pp. 2256–2265.

[35] DHARIWAL, P. and A. NICHOL (2021) "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, **34**, pp. 8780–8794.

[36] ROMBACH, R., A. BLATTMANN, and B. OMMER (2022) "Text-guided synthesis of artistic images with retrieval-augmented diffusion models," *arXiv preprint arXiv:2207.13038*.

[37] CROITORU, F.-A., V. HONDRU, R. T. IONESCU, and M. SHAH (2023) "Diffusion models in vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **45**(9), pp. 10850–10869.

[38] MCKIERNAN, K. A., E. DAVIS, M. S. ALAM, and C. RIGETTI (2019) "Automated quantum programming via reinforcement learning for combinatorial optimization," *arXiv preprint arXiv:1908.08054*.

[39] PIRHOOSHYARAN, M. and T. TERLAKY (2021) "Quantum circuit design search," *Quantum Machine Intelligence*, **3**, pp. 1–14.

[40] AUSTIN, J., D. D. JOHNSON, J. HO, D. TARLOW, and R. VAN DEN BERG (2021) "Structured denoising diffusion models in discrete state-spaces," *Advances in Neural Information Processing Systems*, **34**, pp. 17981–17993.

[41] SIM, S., P. D. JOHNSON, and A. ASPURU-GUZIK (2019) "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, **2**(12), p. 1900070.

[42] SOCORRO, I. M., K. TAYLOR, and J. M. GOODMAN (2005) "ROBIA: a reaction prediction program," *Organic letters*, **7**(16), pp. 3541–3544.

[43] LIU, B., B. RAMSUNDAR, P. KAWTHEKAR, J. SHI, J. GOMES, Q. LUU NGUYEN, S. HO, J. SLOANE, P. WENDER, and V. PANDE (2017) "Retrosynthetic reaction prediction using neural sequence-to-sequence models," *ACS central science*, **3**(10), pp. 1103–1113.

[44] WANG, X., Y. LI, ET AL. (2021) "RetroPrime: A Diverse, plausible and Transformer-based method for Single-Step retrosynthesis predictions," *Chemical Engineering Journal*, **420**, p. 129845.

[45] DAI, H., C. LI, C. COLEY, B. DAI, and L. SONG (2019) "Retrosynthesis prediction with conditional graph logic network," *Advances in Neural Information Processing Systems*, **32**.

[46] GÓMEZ-BOMBARELLI, R. ET AL. (2016) "Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach," *Nature materials*, **15**(10), pp. 1120–1127.

[47] ——— (2018) "Automatic chemical design using a data-driven continuous representation of molecules," *ACS central science*, **4**(2), pp. 268–276.

[48] RAPPOPORT, D., C. J. GALVIN, D. Y. ZUBAREV, and A. ASPURU-GUZIK (2014) "Complex chemical reaction networks from heuristics-aided quantum chemistry," *Journal of chemical theory and computation*, **10**(3), pp. 897–907.

[49] KAK, S. C. (1995) "Quantum neural computing," *Advances in imaging and electron physics*, **94**, pp. 259–313.

[50] CHEN, S. Y.-C., S. YOO, and Y.-L. L. FANG (2022) "Quantum long short-term memory," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 8622–8626.

[51] CILIBERTO, C. ET AL. (2018) "Quantum machine learning: a classical perspective," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **474**(2209), p. 20170551.

[52] LOWE, D. M. (2012) *Extraction of chemical structures and reactions from the literature*, Ph.D. thesis, University of Cambridge.

[53] WEININGER, D. (1988) "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules," *Journal of chemical information and computer sciences*, **28**(1), pp. 31–36.

[54] SHI, C., M. XU, H. GUO, M. ZHANG, and J. TANG (2020) "A graph to graphs framework for retrosynthesis prediction," in *International Conference on Machine Learning*, PMLR, pp. 8818–8827.

[55] TETKO, I. V., P. KARPOV, R. VAN DEURSEN, and G. GODIN (2020) "State-of-the-art augmented NLP transformer models for direct and single-step retrosynthesis," *Nature communications*, **11**(1), pp. 1–11.

[56] ZHANG, Y., K. LU, Y. GAO, and M. WANG (2013) "NEQR: a novel enhanced quantum representation of digital images," *Quantum information processing*, **12**(8), pp. 2833–2860.

[57] CONG, I., S. CHOI, and M. D. LUKIN (2019) "Quantum convolutional neural networks," *Nature Physics*, **15**(12), pp. 1273–1278.

[58] VASWANI, A., N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, Ł. KAISER, and I. POLOSUKHIN (2017) "Attention is all you need," *Advances in neural information processing systems*, **30**.

[59] DI SIPIO, R., J.-H. HUANG, S. Y.-C. CHEN, S. MANGINI, and M. WORRING (2022) "The dawn of quantum natural language processing," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 8612–8616.

[60] PASZKE, A. ET AL. (2019) "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Curran Associates, Inc., pp. 8024–8035.
URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`

[61] YANG, K. ET AL. (2019) "Analyzing learned molecular representations for property prediction," *Journal of chemical information and modeling*, **59**(8), pp. 3370–3388.

[62] WIEDER, O. ET AL. (2020) "A compact review of molecular property prediction with graph neural networks," *Drug Discovery Today: Technologies*, **37**, pp. 1–12.

[63] LENGAUER, T. ET AL. (2004) "Novel technologies for virtual screening," *Drug discovery today*, **9**(1), pp. 27–34.

[64] MERKWIRTH, C. ET AL. (2005) "Automatic generation of complementary descriptors with molecular graph networks," *Journal of chemical information and modeling*, **45**(5), pp. 1159–1168.

[65] HU, W. ET AL. (2019) "Strategies for pre-training graph neural networks," *arXiv preprint arXiv:1905.12265*.

[66] WU, Z. ET AL. (2020) "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, **32**(1), pp. 4–24.

[67] MAYR ET AL. (2018) "Large-scale comparison of machine learning methods for drug target prediction on ChEMBL," *Chemical science*, **9**(24), pp. 5441–5451.

[68] TAYLOR, R. ET AL. (2022) "Galactica: A large language model for science," *arXiv preprint arXiv:2211.09085*.

[69] WU ET AL. (2018) "MoleculeNet: a benchmark for molecular machine learning," *Chemical science*, **9**(2), pp. 513–530.

[70] KUHN, M. ET AL. (2016) "The SIDER database of drugs and side effects," *Nucleic acids research*, **44**(D1), pp. D1075–D1079.

[71] SUBRAMANIAN ET AL. (2016) "Computational modeling of β-secretase 1 (BACE-1) inhibitors using ligand based approaches," *Journal of chemical information and modeling*, **56**(10), pp. 1936–1949.

[72] MARTINS ET AL. (2012) "A Bayesian approach to in silico blood-brain barrier penetration modeling," *Journal of chemical information and modeling*, **52**(6), pp. 1686–1697.

[73] ROHRER ET AL. (2009) "Maximum unbiased validation (MUV) data sets for virtual screening based on PubChem bioactivity data," *Journal of chemical information and modeling*, **49**(2), pp. 169–184.

[74] KRENN, M. ET AL. (2020) "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation," *Machine Learning: Science and Technology*, **1**(4), p. 045024.

[75] ELMAN, J. L. (1990) "Finding structure in time," *Cognitive science*, **14**(2), pp. 179–211.

[76] CHO ET AL. (2014) "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*.

[77] RONG ET AL. (2020) "Self-supervised graph transformer on large-scale molecular data," *Advances in Neural Information Processing Systems*, **33**, pp. 12559–12571.

[78] DEVLIN ET AL. (2018) "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*.

[79] RADFORD ET AL. (2018) "Improving language understanding by generative pre-training," .

[80] WANG ET AL. (2019) "SMILES-BERT: large scale unsupervised pre-training for molecular property prediction," in *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*, pp. 429–436.

[81] STERLING ET AL. (2015) "ZINC 15–ligand discovery for everyone," *Journal of chemical information and modeling*, **55**(11), pp. 2324–2337.

[82] GAULTON ET AL. (2012) "ChEMBL: a large-scale bioactivity database for drug discovery," *Nucleic acids research*, **40**(D1), pp. D1100–D1107.

[83] FANG ET AL. (2021) "Chemrl-gem: Geometry enhanced molecular representation learning for property prediction," *arXiv preprint arXiv:2106.06130*.

[84] ZHANG ET AL. (2021) "LiteGEM: Lite Geometry Enhanced Molecular Representation Learning for Quantum Property Prediction," *arXiv preprint arXiv:2106.14494*.

[85] HOCHREITER ET AL. (1997) "Long short-term memory," *Neural computation*, **9**(8), pp. 1735–1780.

[86] YAO, X. ET AL. (2019) "A novel independent rnn approach to classification of seizures against non-seizures," *arXiv preprint arXiv:1903.09326*.

[87] YIN, W. ET AL. (2017) "Comparative study of CNN and RNN for natural language processing," *arXiv preprint arXiv:1702.01923*.

[88] MAAS ET AL. (2013) "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, Atlanta, Georgia, USA, p. 3.

[89] XU ET AL. (2015) "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*.

[90] GAL ET AL. (2016) "A theoretically grounded application of dropout in recurrent neural networks," *Advances in neural information processing systems*, **29**.

[91] THOMPSON, S. (2012) *Sampling*, CourseSmart, Wiley.
URL https://books.google.com/books?id=-sFtXLIdDiIC

[92] ZHOU ET AL. (2023) "Uni-mol: A universal 3d molecular representation learning framework," .

[93] HANLEY ET AL. (1982) "The meaning and use of the area under a receiver operating characteristic (ROC) curve." *Radiology*, **143**(1), pp. 29–36, pMID: 7063747, https://doi.org/10.1148/radiology.143.1.7063747.
URL https://doi.org/10.1148/radiology.143.1.7063747

[94] SAITO ET AL. (2015) "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PloS one*, **10**(3), p. e0118432.

[95] M. BRAN, A., S. COX, O. SCHILTER, C. BALDASSARI, A. D. WHITE, and P. SCHWALLER (2024) "Augmenting large language models with chemistry tools," *Nature Machine Intelligence*, **6**(5), pp. 525–535.

[96] JIN, B., G. LIU, C. HAN, M. JIANG, H. JI, and J. HAN (2024) "Large language models on graphs: A comprehensive survey," *IEEE Transactions on Knowledge and Data Engineering*.

[97] IRWIN, R., S. DIMITRIADIS, J. HE, and E. J. BJERRUM (2022) "Chemformer: a pre-trained transformer for computational chemistry," *Machine Learning: Science and Technology*, **3**(1), p. 015022.

[98] LIU, P., Y. REN, J. TAO, and Z. REN (2024) "Git-mol: A multi-modal large language model for molecular science with graph, image, and text," *Computers in biology and medicine*, **171**, p. 108073.

[99] TETKO, I. V., P. KARPOV, E. BRUNO, T. B. KIMBER, and G. GODIN (2019) "Augmentation is what you need!" in *International Conference on Artificial Neural Networks*, Springer, pp. 831–835.

[100] BJERRUM, E. J. (2017) "SMILES enumeration as data augmentation for neural network modeling of molecules," *arXiv preprint arXiv:1703.07076*.

[101] MAO, Q., Z. LIU, C. LIU, Z. LI, and J. SUN (2024) "Advancing graph representation learning with large language models: A comprehensive survey of techniques," *arXiv preprint arXiv:2402.05952*.

[102] ÖZÇELIK, R. and F. GRISONI "De Novo Drug Design by Chemical Language Modeling," in *An Introduction to Generative Drug Discovery*, CRC Press, pp. 45–66.

[103] FAWCETT, T. (2006) "An introduction to ROC analysis," *Pattern recognition letters*, **27**(8), pp. 861–874.

[104] HSU, Y.-C., T.-Y. LI, and K.-C. CHEN (2024) "Quantum Kernel-Based Long Short-term Memory," *arXiv preprint arXiv:2411.13225*.

[105] SCHULD, M. and N. KILLORAN (2019) "Quantum machine learning in feature Hilbert spaces," *Physical review letters*, **122**(4), p. 040504.

[106] AKIBA, T., S. SANO, T. YANASE, T. OHTA, and M. KOYAMA (2019) "Optuna: A Next-Generation Hyperparameter Optimization Framework," in *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631.

[107] BJERRUM, E. J., C. MARGREITTER, T. BLASCHKE, S. KOLAROVA, and R. L.-R. DE CASTRO (2023) "Faster and more diverse de novo molecular optimization with double-loop reinforcement learning using augmented SMILES," *Journal of Computer-Aided Molecular Design*, **37**(8), pp. 373–394.

[108] ZENG, Z., B. YIN, S. WANG, J. LIU, C. YANG, H. YAO, X. SUN, M. SUN, G. XIE, and Z. LIU (2024) "ChatMol: interactive molecular discovery with natural language," *Bioinformatics*, **40**(9), p. btae534.

[109] BAGAL, V., R. AGGARWAL, P. VINOD, and U. D. PRIYAKUMAR (2021) "MolGPT: molecular generation using a transformer-decoder model," *Journal of chemical information and modeling*, **62**(9), pp. 2064–2076.

[110] WANG, X., X. WANG, and S. ZHANG (2022) "Adverse drug reaction detection from social media based on quantum bi-lstm with attention," *IEEE Access*, **11**, pp. 16194–16202.

[111] ZHANG, L., Y. XU, M. WU, L. WANG, and H. XU (2024) "Quantum Long Short-Term Memory for Drug Discovery," *arXiv preprint arXiv:2407.19852*.

[112] ARÚS-POUS, J., S. V. JOHANSSON, O. PRYKHODKO, E. J. BJERRUM, C. TYRCHAN, J.-L. REYMOND, H. CHEN, and O. ENGKVIST (2019) "Randomized SMILES strings improve the quality of molecular generative models," *Journal of cheminformatics*, **11**, pp. 1–13.

[113] GUO, T., B. NAN, Z. LIANG, Z. GUO, N. CHAWLA, O. WIEST, X. ZHANG, ET AL. (2023) "What can large language models do in chemistry? a comprehensive benchmark on eight tasks," *Advances in Neural Information Processing Systems*, **36**, pp. 59662–59688.

Vita

Collin Beaudoin

Collin Beaudoin pursued his Ph.D. in the Department of Computer Science and Engineering at Pennsylvania State University (PSU) from September 2021 to June 2025, under the supervision of Dr. Swaroop Ghosh. His research focused on the intersection of machine learning and quantum computing. During his time at PSU, he was honored with the Robert M. Owens Memorial Scholarship in recognition of his academic excellence.

Prior to his doctoral studies, Collin earned both his M.Sc. and B.Sc. degrees from Grand Valley State University (GVSU) in 2021. His master's work explored high-performance computing techniques for quantum simulation. At GVSU, he was awarded the Graduate Dean's Citation for Academic Excellence. Alongside his graduate studies, Collin worked as a full-stack and mobile developer at Dominion Systems in Grand Rapids, Michigan, where he developed a range of tools, including a customizable geofencing verification system.

In the summer of 2024, Collin interned with the data engineering team at Coherent Corporation, where he investigated and implemented techniques to improve data services for silicon manufacturing. His contributions had a direct impact on the internal data infrastructure at the company.

During his Ph.D., Collin authored six peer-reviewed research papers, serving as the lead author on five of them. These works laid the foundation for this dissertation and reflect his significant contributions to material discovery and quantum computing. His research has been featured in prominent venues including GLSVLSI, ICCAD, Frontiers in Molecular Medicine, and IEEE Transactions on Emerging Topics in Computational Intelligence (TETCI).

Collin has also contributed to the academic community as a peer reviewer for top-tier conferences and journals, including ICCAD, ISCA, Frontiers, DATE, and DAC. He has delivered invited talks at several institutions and platforms, such as Carnegie Mellon's Software Engineering Institute and Fairfield University.

In addition to his academic work, Collin is dedicated to mentorship and outreach in computer science, particularly in advancing accessibility in machine learning and quantum computing.

At the time of submitting this thesis, Collin is serving as a research intern at Carnegie Mellon's Software Engineering Institute and has accepted a position as an Assistant Professor at Fairfield University in Connecticut.