Deep Learning Methods for Learning Pauli Errors and Reconstructing Density and Process Matrices

Marcus Edwards
Electrical and Computer Engineering
University of British Columbia
Vancouver, BC, Canada
msedwards@ece.ubc.ca
0000-0001-6591-9585

Gabriel Bottrill
Electrical and Computer Engineering
University of British Columbia
Vancouver, BC, Canada
bottrill@student.ubc.ca

***Abstract*—We implement several alternative approaches to encoding quantum information in matrix representations and leverage these representations to reconstruct the density and process matrices describing the noisy dynamics of a defined circuit using parameters (Pauli error rates) learned by machine learning; this is a useful task for experimentalists looking to characterize the noisy dynamics of their quantum information processors. We find that using a new method of evolving the quantum state by updating a graph Laplacian is in certain cases more efficient than traditional left and right multiplication of density matrices by unitaries and use this to reconstruct the final quantum state corresponding to circuits with learned error rates. We test the hypothesis that our machine learning techniques can perform dimensionality and input reduction to solve for more efficient approaches to learning the Pauli noise in a specific circuit by creating a benchmark dataset with standard methods (i.e. process tomography) and comparing the results from our new neural network implementation against it. We find convergence for the Pauli error rates, and our approach with process matrix reconstruction in the training loop generalizes well, with a mean loss dropping from the original 1.03 to 0.4.**

***Index Terms*—quantum, deep learning, tomography, characterization, graph laplacians, graph learning, quantum network science, simulation, jax**

I. Introduction

The tomography methods that exist for finding the quantum state or process matrix [1] such as Standard Quantum Process Tomography (SQPT) and Direct Characterization of Quantum Dynamics (DCQD) scale exponentially in the system size (the number of qubits). These methods are required if we are to learn about the states that result from quantum computations and the processes that evolved the systems to those states. These complex states cannot be directly observed due to the Heisenberg uncertainty principle. We explore a deep learning approach to learning the structure of the process matrix corresponding to a specific circuit involving many-body interaction and Pauli noise. We test the hypothesis that the model will be able to learn ways to reduce the required number of measurement outcomes — DCQD requires 4^n measurement outcomes and SQPT requires 16^n — and still predict the correct Pauli error rates that correspond to a given circuit.

Quantum Machine Learning (QML) has been the focus of many publications in the past ten years. A notable book on the subject is Dr. Wittek's 2014 publication "Quantum machine learning: what quantum computing means to data mining" [2], which covers quantum versions of pattern recognition, classification, regression and boosting algorithms. Since this work, many diverse proposals have been made including Amin's Quantum Boltzmann Machine from 2016 [3], Bai's Quantum Kernels of 2017 [4], Verdon's Graph Neural Networks published in 2019 [5], and recently, models that are trained with continuous measurements [6], to sample a few differing approaches to quantum machine learning. In general, QML can be divided (as is done in [7]) into "quantum-inspired" models which can have classical implementations, and ones that are inherently quantum (the use of a quantum implementation affords us an exponential or quadratic speedup, for example). i.e. Grover's search based quantum algorithms would have a quadratic speedup over a classical implementation. We can further divide QML approaches into those that deal with quantum data versus classical data.

Quantum graph computing and graph learning is an area that has seen the publication of its own swath of papers since 2015 [8]. In terms of quantum graph computing, there are Hamiltonian encoding-based solvers, quantum random walk based solvers and quantum search based solvers. Hamiltonian encoding-based solvers most notably include those that treat the 2-local Ising Hamiltonian of D-Wave's famous quantum annealers as graphs of couplings with weights on the nodes that correspond to local magnetic field strengths in the physical annealers [8].

One approach to quantum graph learning, quantum kernel-based graph learning, aims to encode graphs into quantum feature spaces [8]. This has taken different experimental forms. For example, Schuld et. al. proposed an encoding of the graph into Hilbert space using Gaussian Boson Sampling [9]. A Continuous Time Quantum Random Walk (CTQRW) can also be used to construct a density matrix which can in turn be used by a Quantum Jensen-Shannon Graph Kernel to infer the properties of a graph [10]. All these embodiments of quantum kernel-based graph learning enable us to decompose a graph into substructures and compare these to one another. In this work we are however most interested not in the best ways to represent graphs by quantum states, but how to represent states by graphs. Therefore, these kinds of uses of quantum computers of various kinds to represent graphs are only tangentially related to our topic.

Most immediately relevant to this work is the approach taken in [11] to define the density matrix ρ_L of a graph Laplacian $\mathbf{L}$. Here β is the inverse temperature, a real number so named in reference to its origin in thermodynamics. ρ would give the Gibbs state of the system in equilibrium at the finite temperature specified by β.

$$\rho_L = \frac{e^{-\beta \mathbf{L}}}{Tr[e^{-\beta \mathbf{L}}]}$$

This matrix is defined thus so that its eigenvalues sum to unity. It is of the same form, for example, as the initial thermal equilibrium state of nuclear spins in a nuclear magnetic resonance quantum computing setup [12, p.328] where $\beta = \frac{1}{KT}$ and K is Boltzmann's constant, T is the physical temperature of the system.

Domenico and Biamonte also use density matrices to describe complex networks in a similar fashion in [13] and [14]. In [13] we learn of some benefits to relating a graph representation with a density matrix. In particular, the authors highlight information-theoretic tools for complex network characterization. The graph can be said to have a von Neumann entropy, for example.

$$S(\rho) = -Tr[\rho log_2 \rho] = -\sum_{i=1}^{N} \lambda_i log_2 \lambda_i$$

[14] agrees that this is a promising direction for the development of an information theory of complex networks. This is a part of a promising new area called "quantum network science".

In our work we propose to bring the utility of classical machine learning to bear on the problem of modeling quantum channels ans states, which is once again a bit different from the perspective taken in the pre-existing literature.

Not directly related to machine or graph learning approaches, previous work has been done in the characterization of Pauli noise [15], which is the type of noise we would like to learn. The authors investigate whether approaches including tomography and cycle benchmarking can enable a researcher to learn about the Pauli noise in a quantum Pauli channel by doing a number of experiments. We will aim to instead use machine learning to characterize Pauli noise with a minimized number of input parameters.

II. Methods

Our methods attempt to address whether we can design a neural network to learn an error model simplified to one and two qubit Pauli errors on a specific circuit for the structure provided in 1. This alternative method should simplify the process tomography procedure for a specific circuit and can act as a rapid benchmarking process.

We began by generating training and benchmarking datasets using a provided process tomography software module within Qiskit Experiments. We generated 100 samples for each pair of qubit size and circuit depth, saving the resulting Choi matrix, the pauli error rates and the measurement outcomes with each sample. An example Choi matrix is shown below 2, with its real and imaginary components shown separately.

The model that was implemented involved several components. The first allowed us to learn Pauli errors via supervised learning from expectation values acquired by process tomography. This component begins with a pooling layer which allows us to summarize the information contained in the expectations (the inputs) resulting from the measurements involved in process tomography in a learnable way. This allows us to learn dimension reductions that allow our Pauli error prediction to remain effective. Following this is a fully connected NN with several hidden layers, and finally a normalized square activation which results in three sets of probabilities which we interpret as our Pauli error rates. We don't consider convolution as we don't expect translation or rotation in our data. This component of the model was trainable in isolation from the second component of the model, which in some iterations was graph-based and in others based on typical approaches to the simulation of non-unitary quantum dynamics using density matrices and Kraus channels.

The second component of our model takes the output probabilities from the first component described above and uses them to construct a representation of a channel. We create a Kraus representation and a process matrix known as the Choi matrix. This was implemented so that flags could be provided to specify matrix operations be used on a representation of the quantum information that was realized by either the left and right multiplication of unitaries on several unitarily evolving subsystems (density matrices), the left multiplication of unitaries by unitaries to create a quantum channel independent of state (a Kraus channel), or by the evolution of Laplacians encoding the density matrix of the quantum subsystems according to the equation $\rho_L = \frac{e^{-\beta \mathbf{L}}}{Tr[e^{-\beta \mathbf{L}}]}$ (graph Laplacians).

The overall model had the depicted structure (Figure 7), keeping in mind that training could be performed either with the first component in isolation or with the quantum simulation in-the-loop. When training the first part of our model in isolation our training data consisted of the input: expectation values, and the expected output: sets of Pauli error probabilities. When training the entire model including the second component, training data consisted of the same inputs: expectations, but the expected outputs: Choi matrices, with a loss function based on a Frobenius norm comparison of the expected and output Choi matrix. In either case, we assess the quality of our learned error model by comparison of the resulting channel characterization with the actual Choi matrix learned from process tomography. The difference lies in whether the matrix multiplication required to construct a Choi matrix from the error model is in the loop or

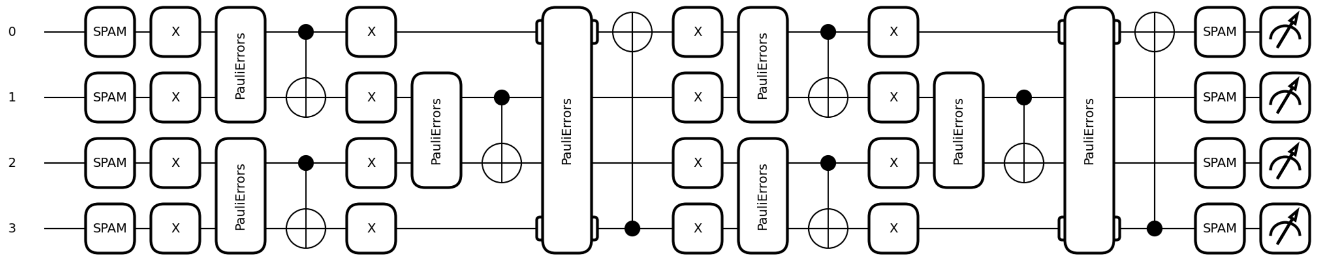


Fig. 1: Quantum Circuit. This is an example of the circuit that our model will be attempting to characterize.

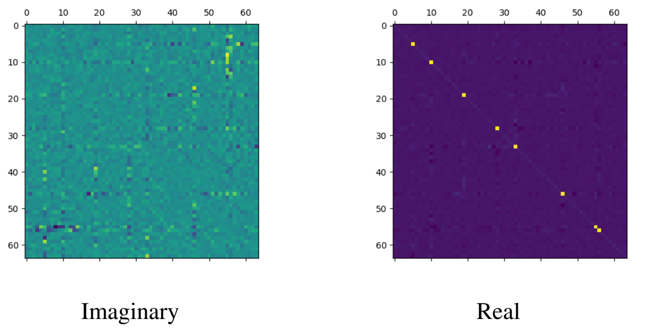


Fig. 2: Choi matrix. These are colour-map representations of a Choi matrix of a 3 qubit circuit.

not. We found that the method with this matrix multiplication in-the-loop had drawbacks including time and space inefficiencies that were dependent on the system size, in comparison to the first method. For example, a 2-qubit 2-layer circuit was represented internally by a Kraus channel with 65536 operators A_i. Converting this to a Choi matrix was costly. We compared the approaches using graph Laplacians, density matrices and unitaries and found the approaches using graph Laplacians and density matrices more suited to quantum state tomography than process tomography for reasons we will expand on in subsection II-B. A summary for this is that they represent the density matrix rather than the quantum channel.

Our pooling layer already implements a dimensionality reduction for us, but we also make use of this to inform a reduction of our inputs using LASSO as a final step. The model essentially learns which inputs are relevant to the problem, and removes the ones that are not. These removals reduce the input size of the model allowing for users to apply this model on a sub-set of the expectation values process tomography would usually require.

A. Optimizations

A part of the goal of the study as originally proposed was to investigate whether any advantages could be gained by the representation of quantum information by graph Laplacians in terms of efficiency.

Instead of unitary matrix multiplication, the equivalent transformation of a Laplacian to represent a unitary evolution of the represented quantum system involves the addition of a matrix logarithm when the Laplacian and the operation commute (this is the case about 23% of the time in our circuit), according to the equation $\rho_L = \frac{e^{-\beta \mathbf{L}}}{Tr[e^{-\beta \mathbf{L}}]}$ from which it follows that

$$U\rho_L U^\dagger = Ue^{-\beta L}U^\dagger = e^{log_m(U)-\beta L+log_m(U^\dagger)} \text{ up to normalization.}$$

In the case that $||U - I||_2 < 1$, the matrix logarithm can be expanded as a power series which may be truncated. This truncation may impact the efficiency of such a computation favourably, while it may also impact the fidelity of the representation of the quantum information. We therefore support this approach to updating the Laplacian as an option in our implementation, when the conditions are met. Caching is also used since our circuit involves repeated layers of repeated gates so that after a matrix logarithm is computed for a matrix once, this computed value may be used again for similar gates without requiring a re-computation.

The other $\approx 77\%$ of the time, we can work how the unitary applies to the Laplacian by performing a truncated expansion of the matrix exponential.

$$e^{-\beta L} = -\beta \sum_{k=0}^{t} \frac{1}{k!} L^k$$

When we evaluate operations on our Laplacian in the cases where the operators do not commute with the current state, there is another trick which we may use. Specifically, it is true that $e^{BAB^{-1}} = Be^A B^{-1}$ for square matrices A and B with invertible B. A unitary matrix is a matrix whose inverse is its Hermitian adjoint. Therefore, for all of our unitary operators, this holds, and we may simply multiply ULU^{-1} which has the same complexity as multiplying $U\rho_L U^\dagger$ and does not require us to evaluate the Taylor expansion of $e^{-\beta L}$.

When converting between density matrix and Laplacian representations, a matrix exponential is also involved, which is also a power series. We similarly support the truncation of this power series to a variable depth in our implementation. An additional quantum approach can be used to optimize this conversion, in theory. Namely, an Approximate Quantum Fourier Transform (AQFT) may be used to calculate the terms L^k with exponential improvement in calls to L in certain cases [16]. This coupled with an efficient approach to AQFT synthesis [17] may reduce the cost of calculating a matrix exponential significantly. In particular, a T-count of $O(nlog(n))$ is involved in this AQFT synthesis. Even better, the same quantum circuit in reverse can also be used to optimize the calculation of the matrix logarithm in the case where we can update the Laplacian by an addition of a matrix logarithm. While this will always incur a greater cost than simply applying U to ρ, it can be used to optimize a pre-compute step that brings us into the Laplacian picture, after which some operations can be run with increased efficiency over the naive approach to executing $U\rho U^\dagger$, as we will see.

B. Drawbacks and Limitations

The graph Laplacian is well suited to representing a Kraus channel in the form

$$E(\rho) = \sum_i A_i \rho A_i^\dagger$$

where A_i are unitaries, since it encodes a density matrix and may be updated in a way that is consistent with unitary evolution of the represented system. However, since it represents the system state at a time t, ρ_t, it is difficult to extract the channel operators themselves from this representation. Indeed, process tomography itself is an approach to take quantum states and reconstruct a process matrix from them. These operators A_i are most useful to us when we are addressing process tomography since in process tomography we are interested in the quantum process and not the state. Therefore, we implemented a more relevant alternative approach which simply constructs a set of unitaries from Pauli error probabilities that, when applied to a density matrix, would evolve the state according to our circuit.

A future direction worth looking at would be the applicability of a graph Laplacian-based approach to addressing state tomography, but that was outside the scope of this work.

Another drawback of this graph Laplacian representation is that in order to represent the channel $E(\rho)$ we need i graphs. In order to overcome the space complexity required, we implemented a flag to specify that after each operation A_i, the sum $\sum_i A_i \rho A_i^\dagger$ may be evaluated, combining the graphs into one. However, this only trades space complexity for the time complexity required to compute this sum.

III. Results

A. Pauli Error Rate Learning

We first benchmarked the performance of standard process tomography on our circuit. This involved running a script similar two what was used to generate our model training and benchmark data. The following execution times are representative of the times taken by the process tomography procedure for circuits of each qubit number and depth. One can see from this data I that the performance is most sensitive to the number of qubits rather than the circuit depth. However, the sensitivity to the number of layers increases with the number of qubits.

Qubits	Layers	Samples	Performance (milliseconds)
2	2	144	2115.195
2	3	144	2003.156
2	4	144	2280.095
3	2	1728	30218.309
3	3	1728	31700.879
3	4	1728	31397.207
4	2	20736	537522.753
4	3	20736	584622.010
4	4	20736	634056.321

TABLE I: Tomography Performance

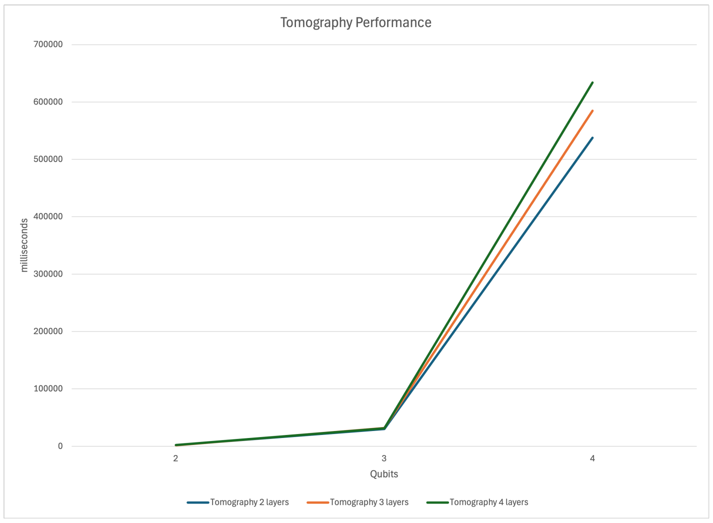
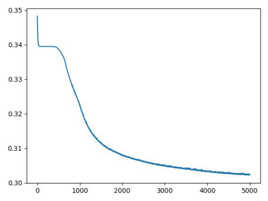


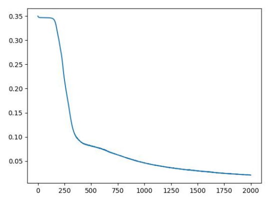
Fig. 3: Tomography Scaling

We then attempted to learn the Pauli error rates from a set of expectations using supervised learning of the first component of the model in isolation, with an L2 loss function and the expectations as our input and the Pauli error rates as our output. The model quickly exhibited learning based on its loss history. Example loss histories are provided in Figure 4 for a 2 qubit and a 4 qubit circuits. Both experiments took approximately 45 minutes to train.

We had post training losses for example around 0.38 (with 200 training examples). The model was able to reduce the input dimension from $4^{qubits} \times 3^{qubits}$ per training example by five passes which each dropped one input from the 3^{qubits} rows in each training data sample, for a total of 81×5 rows dropped in the case of a 4 qubit circuit. Unfortunately, the model suffered from over fitting and did not generalize well when tested against our test / benchmark dataset. We attempted to rectify this by adding dropout layers to the model. This was somewhat effective for reducing the overfitting, however made training much less effective. However, we believe that training on more data is necessary to truly overcome this



2 qubit, 2 layer circuit, 2 removal iterations, 2600 samples



4 qubit 4 layer circuit, 5 removal iterations, 100 samples

Fig. 4: Loss history of probability output per epoch of training without dropout layers.

and unfortunately, as we have shown, data generation is time consuming with process tomography.

We were also able to train using the expected Choi matrix as the output for our model. We used the Frobenius norm of the expected and output Choi matrices as our loss function. The Choi matrix is represented as

$$\Lambda = \sum_{i,j=0}^{2^n} |i\rangle\langle j| \otimes E(|i\rangle\langle j|).$$

This is calculated based on the set of probabilities produced by the first part of our model, Figure 7. The resulting training results of training a dataset of 2 qubit 2 layer circuits with 600 data-points and 1 removal iterations is shown in Figure 5

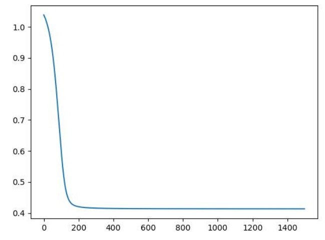


Fig. 5: Loss history of output choi matrices per epoch.

This approach generalized much better with the mean loss of the 100 training dropping from the original 1.03 to 0.4. When solely training the probabilities the loss would remain approximately the same for the training set before and after training.

B. Density Matrix and Process Matrix Reconstruction

The second component of the model reconstructs quantum state or process matrices using various methods. When comparing the graph Laplacian versus the density matrix and the Kraus channel representations at first, the graph Laplacian did not perform as well as the density matrix approach. This is mainly due to the overhead in computing the update to the Laplacian that would correspond to a unitary evolution of a density matrix. For example, for the number of qubits and number of layers below III, the different methods took the following times to compute the final density matrix / graph Laplacian corresponding to a circuit execution with provided Pauli error rates. It is most reasonable to compare the graph Laplacian method and the density matrix method since they both compute state.

Method	Qubits	Layers	Performance (milliseconds)
graph Laplacian	2	2	658.330
Density matrix	2	2	199.239
graph Laplacian	2	3	649.971
Density matrix	2	3	187.284
graph Laplacian	2	4	728.343
Density matrix	2	4	218.800
graph Laplacian	3	2	954.567
Density matrix	3	2	347.223
graph Laplacian	3	3	1149.943
Density matrix	3	3	455.754
graph Laplacian	3	4	1237.421
Density matrix	3	4	517.130

TABLE II: Performances of Kraus operator, density matrix and graph Laplacian constructions

Note that a version of the graph Laplacian implementation was used to generate the non-ab-initio data which does not allow the use of cached matrix logarithms to be used again for *similar* matrices, but only *exact* ones since the array comparisons required to find similar matrices had a significant overhead, and for example resulted in a time of 58558.636 ms for 2 qubits and 2 layers. Instead, an approach using an exact and unique hash of the arrays was used for comparison in the final implementation, speeding it up by over an order of magnitude.

The complexity of matrix multiplication for square $n \times n$ matrices is $O(n^3)$ where addition is only $O(n^2)$ and in many CPU architectures, each addition is more efficient in terms of clock cycles than multiplication i.e. 3 versus 5 cycles on a Sandy Bridge CPU. This is the basis for an argument that graph Laplacian representations, with updates involving matrix sums, can be more efficient at certain scales. However, this advantage is overshadowed again in the data shown so far by the need to compute Laplacian updates from unitaries. It was conceivable that an approach that pre-computes these or uses them ab initio would unlock an advantage. So, in order to investigate the potential advantage of an "ab initio Laplacian update" approach we pre-computed the matrices that would be used in the Laplacian updates and executed the evolution again without the overhead of computing these mid-execution, as peers with the unitaries usually used. We compared a re-play of the circuit execution using unitaries with a re-play using these pre-computed matrix logarithms and the results showed strongly that the Laplacian update was more efficient almost across the board.

This suggests that linear algebra based simulations and representations based from the start on the theory of quantum network science can yield greater efficiency in applications including but not limited to reconstructing quantum process matrices and states. There is no reason why we should have to compute these updates from "first-order" unitaries instead of vice versa. We checked that the Laplacian update method

Method	Qubits	Layers	Performance (milliseconds)
graph Laplacian	2	2	2.018
Density matrix	2	2	3.958
graph Laplacian	2	3	3.421
Density matrix	2	3	2.907
graph Laplacian	2	4	2.054
Density matrix	2	4	7.291
graph Laplacian	3	2	2.892
Density matrix	3	2	3.015
graph Laplacian	3	3	3.750
Density matrix	3	3	4.464
graph Laplacian	3	4	4.220
Density matrix	3	4	6.626
graph Laplacian	4	2	2.488
Density matrix	4	2	4.923
graph Laplacian	4	3	6.973
Density matrix	4	3	9.254
graph Laplacian	4	4	7.999
Density matrix	4	4	9.017

TABLE III: Comparing the density matrix and graph Laplacian constructions with matrix logarithms pre-computed

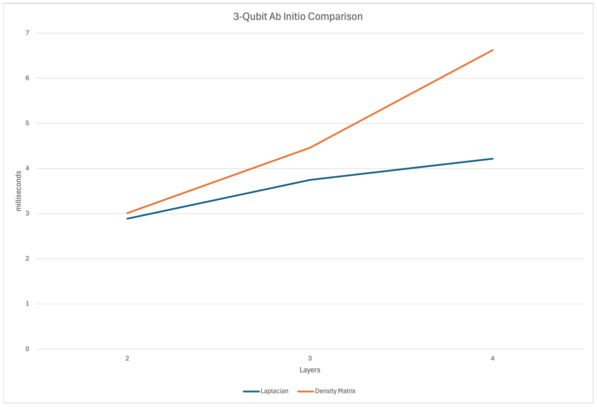


Fig. 6: 3-Qubit Ab-Initio Approach Comparison

is more efficient than the density matrix method up to n=4 truncation length on matrix sums and exponentials, with 2 and 4 qubit circuits.

IV. Discussion

When directly comparing the actual Laplacian updates (the sums not counting the computation of the matrix logarithms) with their equivalent density matrix multiplications in isolation, we found the following. For 2 qubits and 2 layers, we saw that 70% of the time (for 32/46 gates), a Laplacian update was more efficient by an average of 0.001765788 ms. In the case of 2 qubits and 3 layers, the Laplacian update was more efficient 60% of the time (27/46 gates) by an average of 0.003567448 ms, etc. showing that the reason for the Laplacian approach's increased efficiency is indeed the improved speed of the individual operations. During a circuit execution, these gates are each applied very many times. Whether this kind of Laplacian update can be performed is of course something that needs to be checked. We need to check if the matrices commute. It is also notable that evaluating the sum (as was done in the collection of the data presented in previous sections) is much more efficient than not. In the later case, the model blows up in size and we have to do many more matrix multiplications in each step, for example yielding times of 859.937 and 833.930 ms for 2x2 density matrix and Laplacian evolutions respectively as well as 14003.262 and 13732.297 ms for 2x3 evolutions with the same approaches.

On the topic of the training of the neural network, we faced several challenges. The first challenge we faced was the long time it took to generate datasets due to the exponential scaling of process tomography. This meant we were limited to a smaller dataset than we would have liked and this led to overfitting in our model. We tried to combat this by introducing a dropout layer which was somewhat helpful. However, in the end we resorted to generating more data which was of course only so feasible given time constraints. An alternative direction to explore in the future would be to change the way we generate training data i.e. by replacing the full process tomography done by Qiskit (this is needed when you cannot directly observe a quantum state) with a method that takes advantage of the fact that a simulated quantum state can be inspected. Process tomography often includes a state tomography step which could be bypassed.

During the course of this work we investigated whether the updates we applied to our Laplacian were realizable via spectral filtering of the form $L_1 = V h_\theta(\Lambda) V^T$ where the filtering function h_θ operates on the entries of the diagonal Λ, with $L_0 = V \Lambda V^T$. This would resemble graph convolution except that we would be learning the structure of the graph rather than updating the node features by combining the nearest neighbours. Indeed, in our work, we learned parameters that governed how we updated the graph structure, not node features. In our work therefore we do not employ graph convolution. However, we concluded that a spectral filtering approach would lose the advantage we demonstrated by re-introducing matrix multiplication of h_θ with Λ. Of course, this could be mitigated somewhat with the use of Chebyshev polynomials and graph wavelets, but our approach to Laplacian updating can also benefit from similar approximations (series truncation). Instead, an interesting future direction would be to create a framework for the type of learning we used, where the learned parameters are in a matrix h_{update} summed with $-\beta L$ in the form $L_1 = 2h_{update} - \beta L_0$. It would be interesting to further explore what graph properties are preserved by this kind of update.

V. Conclusion

We conclude that the deep learning techniques we employed such as pooling and LASSO based dimensionality reduction allowed us to learn Pauli errors for a specific circuit involving Pauli noise and many-body interaction using less parameters than is required by standard characterization techniques. This was done with a loss decreasing from an original 1.03 to 0.4 and dropping as many as 81×5 rows. In addition, we found evidence that an alternative graph-based representation of quantum information based on the graph Laplacian can be used to more quickly reconstruct quantum state in-the-loop during training, when a pre-computation step precedes this.

VI. Proofs

Proof that for commuting U and L we have $U\rho_L U^\dagger = Ue^{-\beta L}U^\dagger = e^{log_m(U)-\beta L+log_m(U^\dagger)}$ up to normalization. with L absorbing $-\beta$ and $U = U^\dagger$ which is true in our case.

$$\begin{aligned}
e^U e^L e^U &= (\sum_m \frac{U^m}{m!})(\sum_n \frac{L^n}{n!})(\sum_l \frac{U^l}{l!}) \\
&= (\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{U^m L^n}{m!n!})(\sum_l \frac{U^l}{l!}) \\
&= (\sum_{p=0}^{\infty}\sum_{m=0}^{p} \frac{U^m L^{p-m}}{m!(p-m)!})(\sum_l \frac{U^l}{l!}) \\
&= (\sum_{p=0}^{\infty}\frac{1}{p!}\sum_{m=0}^{p} \frac{p!}{m!(p-m)!}U^m L^{p-m})(\sum_l \frac{U^l}{l!}) \\
&= (\sum_{p=0}^{\infty} \frac{(U+L)^p}{p!})(\sum_l \frac{U^l}{l!}) \\
&\text{Note: } [L+U,U] = LU + U^2 - UL - U^2 = [L,U] = 0 \\
&= \sum_{p=0}^{\infty} \frac{(U+L)^p U^l}{p!l!} \\
&= \sum_{q=0}^{\infty}\sum_{p=0}^{q} \frac{(U+L)^p U^{q-p}}{p!(q-p)!} \\
&= \sum_{q=0}^{\infty}\frac{1}{q!}\sum_{p=0}^{q} \frac{q!}{p!(q-p)!}(U+L)^p U^{q-p} \\
&= \sum_{q=0}^{\infty} \frac{(U+L+U)^q}{q!} \\
&= e^{2U+L}
\end{aligned}$$

Another proof that is leveraged in our code when evaluating updates in the non-commuting cases is the following. We prove that $e^{ULU^{-1}} = Ue^L U^{-1}$ for square matrices L, U and invertible U. This is valid for our case because a unitary matrix is a matrix whose inverse is its Hermitian adjoint, so $U^{-1} = U^\dagger$.

$$\begin{aligned}
e^{ULU^{-1}} &= I + ULU^{-1} + \frac{1}{2}(ULU^{-1})^2 + \frac{1}{6}(ULU^{-1})^3 + \dots \\
&= UU^{-1} + ULU^{-1} + \frac{1}{2}UL^2U^{-1} + \frac{1}{6}UL^3U^{-1} + \dots \\
&= U(I + L + \frac{1}{2}L^2 + \frac{1}{2}L^3 + \dots)U^{-1} \\
&= Ue^L U^{-1}
\end{aligned}$$

VII. Acknowledgments

We would like to acknowledge Dr. Renjie Liao whose teaching inspired and informed this work, and who encouraged us to publish these results.

References

[1] M. Mohseni, A. T. Rezakhani, and D. A. Lidar, "Quantum-process tomography: Resource analysis of different strategies," *Phys. Rev. A*, vol. 77, p. 032322, Mar 2008. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.77.032322

[2] P. Wittek, *Quantum machine learning: what quantum computing means todata mining*, 1st ed. Amsterdam: Elsevier, AP, 2014.

[3] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, "Quantum Boltzmann Machine," Jan. 2016, arXiv:1601.02036. [Online]. Available: http://arxiv.org/abs/1601.02036

[4] L. Bai, L. Rossi, L. Cui, Z. Zhang, P. Ren, X. Bai, and E. Hancock, "Quantum kernels for unattributed graphs using discrete-time quantum walks," *Pattern Recognition Letters*, vol. 87, pp. 96–103, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786551630232X

[5] G. Verdon, T. McCourt, E. Luzhnica, V. Singh, S. Leichenauer, and J. Hidary, "Quantum Graph Neural Networks," Sep. 2019, arXiv:1909.12264. [Online]. Available: http://arxiv.org/abs/1909.12264

[6] K. Tucker, A. K. Rege, C. Smith, C. Monteleoni, and T. Albash, "Hamiltonian Learning using Machine Learning Models Trained with Continuous Measurements," Apr. 2024, arXiv:2404.05526. [Online]. Available: http://arxiv.org/abs/2404.05526

[7] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, Sep. 2017, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nature23474

[8] Y. Tang, J. Yan, and H. Edwin, "From Quantum Graph Computing to Quantum Graph Learning: A Survey," Feb. 2022, arXiv:2202.09506. [Online]. Available: http://arxiv.org/abs/2202.09506

[9] M. Schuld, "Measuring the similarity of graphs with a Gaussian boson sampler," *Physical Review A*, vol. 101, no. 3, 2020.

[10] L. Bai, L. Rossi, A. Torsello, and E. R. Hancock, "A quantum Jensen–Shannon graph kernel for unattributed graphs," *Pattern Recognition*, vol. 48, no. 2, pp. 344–355, Feb. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314001265

[11] M. Steinberg, M. Bandic, S. Szkudlarek, C. G. Almudever, A. Sarkar, and S. Feld, "Resource Bounds for Quantum Circuit Mapping via Quantum Circuit Complexity," Feb. 2024, arXiv:2402.00478 version: 1. [Online]. Available: http://arxiv.org/abs/2402.00478

[12] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information 10th Anniversary Edition.* Cambridge;New York;: Cambridge University Press, 2010.

[13] M. De Domenico, "Spectral Entropies as Information-Theoretic Tools for Complex Network Comparison," *Physical Review X*, vol. 6, no. 4, 2016.

[14] J. Biamonte, M. Faccin, and M. De Domenico, "Complex networks from classical to quantum," *Communications Physics*, vol. 2, no. 1, pp. 1–10, May 2019, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s42005-019-0152-6

[15] S. Chen, Y. Liu, M. Otten, A. Seif, B. Fefferman, and L. Jiang, "The learnability of Pauli noise," *Nature Communications*, vol. 14, no. 1, p. 52, Jan. 2023, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41467-022-35759-4

[16] L. Sheridan, D. Maslov, and M. Mosca, "Approximating Fractional Time Quantum Evolution," *Journal of Physics A: Mathematical and Theoretical*, vol. 42, no. 18, p. 185302, May 2009, arXiv:0810.3843 [quant-ph]. [Online]. Available: http://arxiv.org/abs/0810.3843

[17] Y. Nam, Y. Su, and D. Maslov, "Approximate Quantum Fourier Transform with $O(n \log(n))$ T gates," *npj Quantum Information*, vol. 6, no. 1, p. 26, Mar. 2020, arXiv:1803.04933 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1803.04933

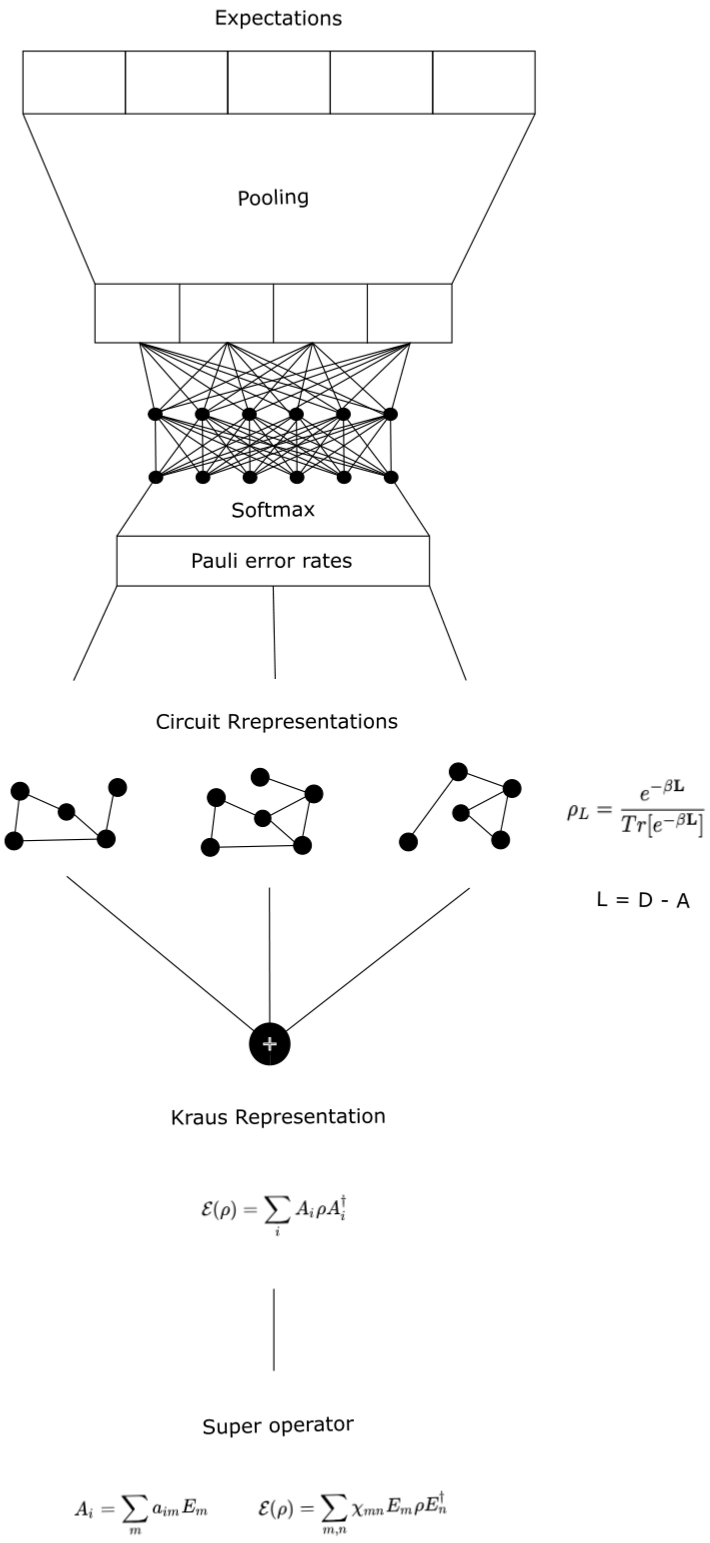

Expectations
Pooling
Softmax
Pauli error rates
Circuit Rrepresentations
$\rho_L = \frac{e^{-\beta \mathbf{L}}}{Tr[e^{-\beta \mathbf{L}}]}$
L = D - A
Kraus Representation
$\mathcal{E}(\rho) = \sum_i A_i \rho A_i^\dagger$
Super operator
$A_i = \sum_m a_{im} E_m$
$\mathcal{E}(\rho) = \sum_{m,n} \chi_{mn} E_m \rho E_n^\dagger$

Fig. 7: Model