Graph Neural Network Surrogate for PauliPropagation.jl

Andrew Sutcliffe, Matteo Casiraghi, and Hugo Izadi
École Polytechnique Fédérale de Lausanne, 1015 Lausanne

Abstract: *We implement a GNN as a surrogate model to improve truncation strategies for Pauli Propagation (PP). Our model predicts with high accuracy future expectation-values of propagated Pauli strings under heisenberg evolution. This method shows a strong promise for improving expectation-value estimation using PP and possibly increase the framework's utility for near-term quantum simulation.*

Keywords: Graph Neural Network, Sparse Pauli Dynamics, Pauli Propagation, Truncation Surrogate

1 Introduction

Simulating classically the dynamics of interacting many-body quantum systems is a computationally intensive task. Yet, it is central in the verification of NISQ computations and in establishing quantum-inspired advantage before large-scale FTQC [1, 2]. Among classical approaches, Pauli Propagation [3] has emerged as one of the most efficient methods by expressing the evolved observable as a sparse expansion over Pauli strings,

$$UOU^\dagger = \sum_i c_i P_i, \tag{1}$$

where the coefficients c_i are updated under successive gate evolution. To remain tractable, practical implementations rely on aggressive coefficient- and Pauli weight-based truncation to curb the exponential scaling of the number of Paulis throughout this evolution. While effective, such truncation strategies can introduce systematic bias.

In this work, we seek a better truncation strategy that would predict future evolution of a given Pauli, enabling more informed truncation decisions. We restrict our work to Trotterized evolution described by unitaries $U \equiv V(H; T, L_1)^{L_1}$ with $T = 1$ and $L_1 = 32$ and generated by transverse-field Ising Hamiltonians,

$$H = \sum_{i=1}^{n} X_i + 2 \sum_{(i,j)\in E} Z_i Z_j, \tag{2}$$

for varying system sizes n and graphs E shown in Fig. 1. To do so, we use the open-source library `PauliPropagation.jl` [4] which provides a robust implementation of SPD and allows to collect the data for this project. We implement a surrogate model $M_{\{c_p,L_2\}}(P)$ aiming to predict the label

$$t(P) = \log\big(\, \big|\mathrm{Tr}\big[(V^\dagger)^{L_2}\, P\, V^{L_2}\, |0^n\rangle\langle 0^n|\big]\big|\, \big)\,, \tag{3}$$

where the logarithmic scaling prioritizes predicting magnitude rather than exact values.

This is motivated by the practical usage of our surrogate within a new `PauliPropagation` truncation pipeline. Indeed, the model is supposed to be evaluated at the end of each Trotter layer $\ell \in \{1, \cdots, L_1 - L_2\}$ and decides on the truncation of any term (c, P) of the `PauliSum` such that $c\, M(P) < \delta$ where δ is a new truncation threshold. In this way, truncation decisions are informed not only by the current coefficient c, but also by a learned estimate of the Pauli string's future relevance, enabling a more balanced trade-off between computational efficiency and physical accuracy.

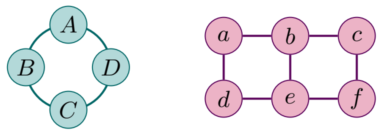


Figure 1: Examples of interaction graphs E.

A key observation underlying our approach is that the interaction graph E induces non-trivial symmetries in Pauli dynamics. Specifically, many Pauli strings have symmetrically equal Heisenberg evolution (1) and yield the same overlap with $|0^n\rangle\langle 0^n|$. These symmetries are captured by graph automorphisms i.e. qubit permutations $\pi \in \mathcal{A}(E)$ that preserve the edge set E. They partition the set of Pauli strings into orbits, each characterized by a representative. For instance, the cyclic permutation $\pi_1 = (ABCD)$ is an automorphism of the ring graph, while $\pi_2 = (ad)(be)(cf)$ is an automorphism of the rectangular lattice shown in Fig. 1. In general, the full automorphism group $\mathcal{A}(E)$ can be identified using the `VF2` algorithm [5].

This symmetry motivates the choice of a message-passing Graph Neural Network (GNN) as our surrogate, which is invariant under any $\pi \in \mathcal{A}(E)$ [6], and therefore ensures identical predictions for Pauli from the same orbit.

The remainder of this report is organized as follows. We first describe the dataset generation pipeline, then present the GNN architecture and finally report numerical results together with a discussion of limitations, sources of bias, and future steps.

2 Data Harvesting

The dataset generation pipeline proceeds in two stages. In the first stage, we construct a set of Pauli strings that we denote by $\mathcal{S}$. This set corresponds to the Pauli strings on which the surrogate model is trained. It is designed to capture the Pauli that are most likely to be encountered during the Trotter evolution of common Paulis. In the second stage, we generate training data for each $P \in \mathcal{S}$ according to (1).

Step 1: Construction of $\mathcal{S}$.
Let $\mathcal{P} = \{Z_{1,2}, Z_{1,3}, \ldots, Z_{n-1,n}\}$. In order to avoid redundant propagation, we have reduced $\mathcal{P}$ to a set of canonical representatives $\tilde{\mathcal{P}}$. Each Pauli string $P \in \mathcal{P}$ is then propagated through L_1 Trotter layers using a coefficient truncation threshold c_1. During this propagation, all Pauli strings encountered are recorded, including those discarded by truncation. The union of all encountered Pauli strings defines the set $\mathcal{S}$. In practice, L_1 and c_1 are tuned to have a dataset close to $100K$.

Step 2: Dataset Generation.
Each Pauli string $P \in \mathcal{S}$ is propagated using a small $c_2 \approx 0$ through L_2 additional Trotter layers. From the evolved operator, we extract the overlap with 0 as the training target. Additionally, we extract the overlap with with all single-Z and double-Z observables, but these remain unused for now. Paulis with labels below 1e-40 prior to logging are discarded to avoid infinities in (3).

3 Model Architecture

The surrogate model is a message-passing Graph Neural Network (GNN), shown in Fig. 2, and is fully specified by the number of blocks B. Each node corresponds to a qubit and is initialized using a Pauli one-hot encoding, $\mathbb{I} \mapsto [000]^\top, Z \mapsto [100]^\top, X \mapsto [010]^\top, Y \mapsto [001]^\top$. Subsequently, a linear lifting operation $\mathbf{x}_i^{(0)} = W_{\text{lift}}(\mathbf{x}_i)$ is applied to increase the internal feature dimension; for simplicity, this step is omitted figure.

Each block ℓ consists of a message-passing (MP) operation followed by a local multilayer perceptron (MLP). The MLP preserves the feature dimension and acts on each node as,

$$\tilde{\mathbf{x}}_i^{(\ell)} = \text{MLP}_\ell\Big(\mathbf{x}_i^{(\ell)}\Big) \tag{4}$$

$$= W_2^{(\ell)}\,\text{GELU}\Big(W_1^{(\ell)}\mathbf{LN}(\mathbf{x})_i^{(\ell)} + \mathbf{b}_1^{(\ell)}\Big) + \mathbf{b}_2^{(\ell)} \tag{5}$$

while MP aggregates neighbors information according to the interaction graph E,

$$\mathbf{x}_i^{(\ell+1)} = \text{MP}_\ell^{[E]}\Big(\tilde{\mathbf{x}}_i^{(\ell)}\Big) \tag{6}$$

$$= \frac{1}{|\mathcal{N}(i) \cup \{i\}|} \sum_{j \in \mathcal{N}(i)\cup\{i\}} msg(\tilde{\mathbf{x}}_j^{(\ell)}) \tag{7}$$

where $\mathcal{N}(i)$ denotes the neighbors of node i and self-loops are included. While $msg(\tilde{\mathbf{x}}_j^{(\ell)})$ information is computed as:

$$\begin{aligned} msg(\tilde{\mathbf{x}}_j^{(\ell)}) &= \text{MLP}_\ell\Big(\tilde{\mathbf{x}}_j^{(\ell)}\Big) \\ &= W_2^{(\ell)}\,\text{ReLU}\Big(W_1^{(\ell)}\mathbf{LN}(\tilde{\mathbf{x}})_j^{(\ell)} + \mathbf{b}_1^{(\ell)}\Big) + \mathbf{b}_2^{(\ell)} \end{aligned} \tag{8}$$

After B blocks, node features are aggregated by component-wise mean pooling, $\mathbf{y} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i^{(B)}$, and mapped to a scalar through a linear layer:

$$M = \mathbf{w_2}^\top \text{GELU}(\mathbf{w_1}^\top \mathbf{y} + b_1) + b_2 \tag{9}$$

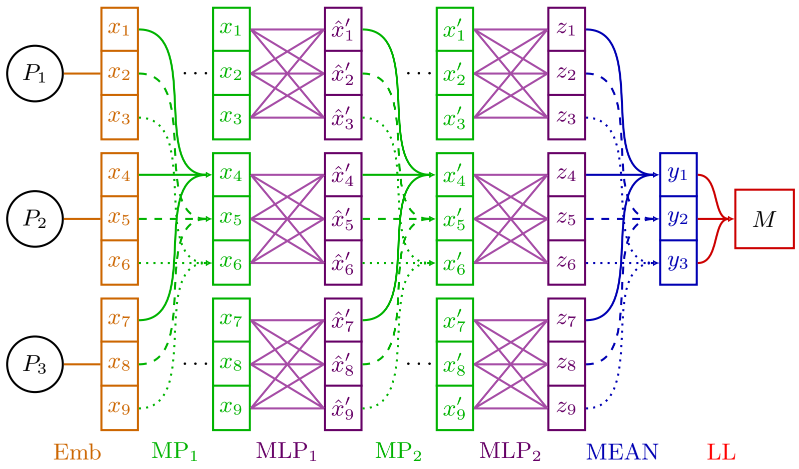
We benchmarked the proposed GNN against simpler linear and quadratic baseline models defined as $M_{\text{lin}}(P) = \mathbf{w}^\top \cdot \mathbf{x} + b$, $M_{\text{quad}}(P) = \mathbf{w}^\top \mathbf{x} + \mathbf{x}^\top Q \mathbf{x} + b$. Additional analyses using a more expressive baseline model are reported in one of the following section.

4 Training Methodology

Our current models implement the standard MSE loss:

$$\mathcal{L}(\theta) = \sum_{P \in \mathcal{S}} \big(M_\theta(P) - t(P)\big)^2, \tag{10}$$

Our flagship model, with 20 qubits on a cyclic ring (fig 1a), has parameters: $L_1 = 32, c_1 = 0.00123418$, producing $|S| = 100K$, $L_2 = 2, c_2 = 1e-15$. Hyperparameter tuning was carried out with a hold-out validation set, however to iterate through all hyperparameter sets, a 5K training set was used. This was followed by final training on the full 90K training set with the best parameters. Final performance is reported on the last 5K held-out test set. Hyperparameters tuned are the hidden dim, expansion ratio, and dropout. The number of message-passing blocks B was fixed to one per



P_1
P_2
P_3
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9
$\hat{x}'_1$ $\hat{x}'_2$ $\hat{x}'_3$ $\hat{x}'_4$ $\hat{x}'_5$ $\hat{x}'_6$ $\hat{x}'_7$ $\hat{x}'_8$ $\hat{x}'_9$
x'_1 x'_2 x'_3 x'_4 x'_5 x'_6 x'_7 x'_8 x'_9
z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8 z_9
y_1 y_2 y_3
M
Emb
MP$_1$
MLP$_1$
MP$_2$
MLP$_2$
MEAN
LL

Figure 2: GNN architecture. The model consists of B blocks, each with a message-passing layer (green) followed by a per-node MLP (purple); the figure shows B=2. The network uses Pauli one-hot encoding with linear lifting (Emb), followed by mean pooling (blue) and a linear output layer (red).

trotter layer, reflecting the math, specifically, the second term in (1).

We train the model using the Adam optimizer with a learning rate of 10^{-3}, and weight decay 10^{-4}, which are standard and solid choices for neural network optimization. Moreover, Adam's adaptive update is a reasonable choice for GNNs, since gradients may vary frequently across layers and nodes. Finally, to promote stable convergence, we took advantage of a scheduler, enabling larger updates in early epochs.

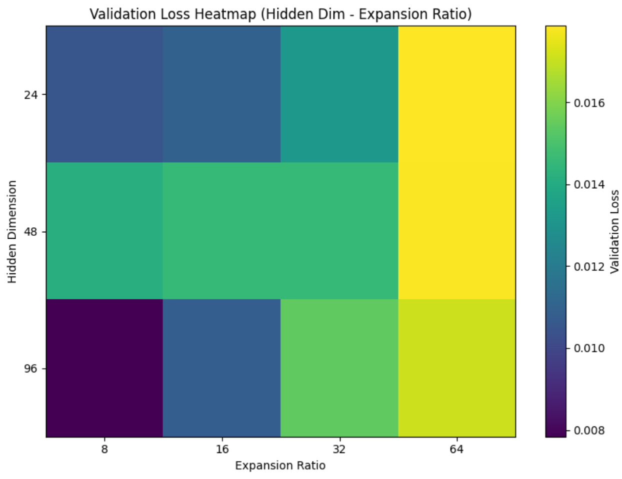


Figure 3: Example hyper-parameter heatmap.

We find the following optimal hyperparameters between 36 possible combinations:

- `hidden_dim` $= 96$,
- `expansion_ratio` $= 8$,
- `dropout` $= 0$,

where `hidden_dim` denotes the internal feature dimension of the per-node MLP blocks. The parameter `expansion_ratio` controls the dimensional expansion applied during the linear lifting step, with the output dimension given by `out_dim` $= B \times$ `expansion_ratio`. This choice reflects the assumption that the representation dimensionality should scale with the number of layers, as deeper models must aggregate and encode progressively richer information. Finally, while dropout is included in the per-node MLP blocks for regularization, it is fixed to zero for the selected hyperparameter configuration.

We employ early stopping based on the validation loss and retain the best-performing checkpoint. As we can see in 4, although the validation loss shows small fluctuations, it continues to follow the overall downward trend of the training loss and does not exhibit divergence. This behavior suggests that the model neither overfits nor underfits, as both losses converge to low values.

5 Results and Discussion

$$\text{Test Loss} = 2.93 \times 10^{-4},\ R^2 = 0.999983$$

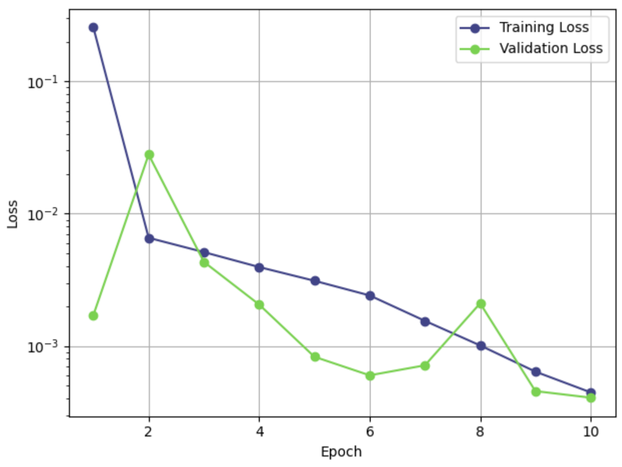


Figure 4: Training and validation loss in log-scale

These values demonstrate our model is able to capture the trotter dynamics very precisely, and can effectively predict the future evolution of unseen paulis. The truncation of `Pauli` term (c, P) defined by $c\,M(P) < \epsilon$ will therefore accurately ascertain which paulis to keep at the present layer more effectively than only using the coefficient c.

5.1 Ablation Studies

To validate our decisions throughout this project, we conduct several simplifications to our model. Firstly, we sought to justify the use of a GNN instead of simpler models. Therefore, we compare to both a standard neural network (NN), mimicking the structure and size of our tuned flagship model. To ensure a fair comparison, we tune the hyperparameters of the neural network, and achieve the following:

$$\text{Test Loss} = 3.02 \times 10^{-3},\ R^2 = 0.999829$$

This shows that incorporating GNN structure, more precisely message passing step, yields a performance improvement of approximately one order of magnitude.

We then conduct a consecutive ablation study targeting the initial linear lifting step and the LayerNorm applied at the beginning of per-node MLP. Removing the linear lifting from the pipeline introduces a strong bias in the model outputs, which collapse toward the mean of the training labels. This behavior indicates that an insufficient embedding dimension prevents the model from learning. In contrast, when LayerNorm is omitted, the model continues to fit the data reasonably well, but the predictions exhibit larger fluctuations, highlighting the stabilizing role of normalization.

Finally, we check to see the importance of including in the dataset all Paulis, even those truncated during propagation in step 1. We create a new dataset which retains only final Paulis, and to maintain the same training set size, L_1 is increased from 32 to 34. This model reaches a loss of 3.02×10^{-3}, however this may result from the fact that this is a harder training set given the extra 2 layers. See section 6 for more meaningful comparisons.

6 Further Steps

The true utility of our model lies in its performance in the overall propagation in (1). Using package `pyquest`, we are able to efficiently simulate the exact state vector evolution, which enables an accuracy calculation of our method. An analysis of runtime Vs accuracy of a propagation demonstrates that, for a given desired accuracy, normal SPD evolution is faster, due to our model query time. The choice of whether to include Paulis which were truncated makes an imperceptible difference.

However, there are many potentially highly fruitful extensions and modifications we foresee to improve our model's accuracy and query time.

6.1 Implemented Improvements

The following changes have already been implemented, but are yet to be rigorously trained and tested.

Updated dataset pipeline. As illustrated in Fig. 5, we introduce an updated data generation pipeline based on symmetry-orbit representatives motivated by the permutation invariance of both the training target (overlap with 0) and the GNN.

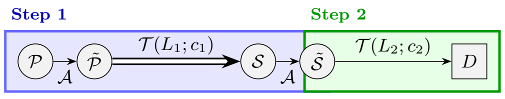


Figure 5: Dataset generation pipeline.

More precisely, in Step 2, we sample 100K random representatives from the set $\tilde{\mathcal{S}}$ instead of propagating all strings in $\mathcal{S}$. By using multithreading additionally, we managed to reduce data generation under 5 hours for small systems while obtaining a dataset exceeding virtually the 100K Pauli strings, as each representative belongs to an orbit larger than one.

Updated loss function. Leveraging this new dataset, we carefully maintain orbit size balance by using the following loss,

$$\mathcal{L}(\theta) = \frac{1}{\sum_{P\in\mathcal{B}} w(P)} \sum_{P\in\mathcal{B}} w(P) \left(M_\theta(P) - t(P;\epsilon)\right)^2, \tag{11}$$

where $\mathcal{B}$ denotes a mini-batch, $w(P) = |\mathrm{Orb}(P)|$ is the size of the orbit associated with the representative P. Also note that the regression target is updated by introducing a parameter ϵ to explicitly deal with the $\log(0)$ instability,

$$t(P;\epsilon) = \log\left(\left|\mathrm{Tr}\left[(V^\dagger)^{L_2}\, P\, V^{L_2}\, |0^n\rangle\langle 0^n|\right]\right| + \epsilon\right). \tag{12}$$

This prevents the discarding of Pauli strings with zero overlap with the idea that improving performance on those is important for the practical utility of the surrogate. For this, we consider two choices for ϵ: a fixed value $\epsilon_J = 10^{-15}$, comparable to the machine precision of Julia's `Float64`, and a larger value chosen between ϵ_J and the smallest non-zero absolute overlap observed in the dataset.

6.2 Non Implemented Improvements

Firstly, model inference time has proven to be a bottleneck of model utility. To reduce this time, we should consider balancing performance with size during hyperparameter tuning. Currently, larger models were favoured. For example, in figure 3, 24 layers, whose loss is only just above 96, could have been chosen instead of 96. However, our flagship model contains only 7937 parameters, thus speed-up is likely to be minimal. Model recompilation and overhead should therefore be investigated.

Finally, training and inference must be migrated to a GPU, as current access is restricted to CPUs.

6.3 Further extensions

The PauliPropagation package allows for easy extension to different toplogies and hamiltonians. Datasets and models were created for both 2D graphs and a more complex hamiltonian than (1), the TTFI. Due to our lack of computing power, datasets were small and inexact, thus model quality was also poor.

7 Conclusion

In this work, we successfully implemented a Graph Neural Network (GNN) surrogate that predicts the overlap with 0 of Pauli strings after an $L_2 = 2$ Trotter evolution with an R^2 of 0.999983. Our results demonstrate that incorporating message-passing steps allows the model to leverage interaction graph symmetries, yielding a performance improvement of approximately one order of magnitude over standard neural network. While model inference time currently presents a bottleneck for real-time truncation, the high predictive accuracy indicates that this approach could significantly enhance the efficiency of Pauli Propagation for near-term quantum simulations once optimized for GPU deployment.

8 Ethical Risks

The EU AI act identifies classes of AI, and enforces regulation rules depending on the class. Despite its laudable intentions, the AI act has been criticised for delivering far less than it promises [7, 8]. Furthermore, the quantum field is developing especially quickly, and attempting to translate AI ethics directly to quantum computing is unlikely to capture as intended [9]. Unique concerns arise in the quantum field, such as the use of quantum algorithms for harm, and inherent opaqueness. However our current application sidesteps all concerns, as we describe below:

The primary stakeholders for our method are researchers who may use this method to accelerate Pauli propagation in quantum state simulations. The main ethical risk for this group is misleading simulation results caused by the GNN. This poses no risk, as our model will only be deployed in further research if it beats other methods, in which case, the risk is not present. Metrics used to evaluate this risk are described above, including MSE, and accuracy comparison to exact state vector evolution.

Indirect stakeholders are researchers who may rely on conclusions derived from simulations accelerated using this method. The ethical concern here is the propagation of incorrect scientific claims. This risk was ruled out by explicitly stating the current ineffectiveness of our model, by validating all results against exact evolution for small systems. No claims are made beyond regimes where exact validation is possible, and the method is not used to generate standalone scientific conclusions.

Typical ML ethical risks related to bias, fairness, or representational harm do not apply. The model operates exclusively on synthetic, mathematically defined Pauli operators, with training data generated synthetically rather than sampled from human populations. As a result, no demographic bias metrics are relevant, and no groups can be systematically advantaged or disadvantaged by the model's behavior.

Finally, the model introduces additional computational overhead due to GNN inference. This was assessed by measuring runtime and compute cost relative to standard Pauli propagation packages. Since the ML-based approach is currently less efficient, it is not used in production, therefore there is no unnecessary energy consumption. Future adoption would require demonstrable net computational savings, which are highly likely given the tiny model size.

Overall, given its restricted scope, lack of human data, quantitative validation, and non-deployment, the project poses minimal ethical risk.

References

[1] A. J. Daley et al. Practical quantum advantage in quantum simulation. *Nature*, 607:667, 2022.

[2] Reza Haghshenas et al. Digital quantum magnetism at the frontier of classical simulations. *arXiv preprint arXiv:2503.20870*, 2025.

[3] T. Begušić et al. Fast and converged classical simulations of evidence for the utility of quantum computing. *Sci. Adv.*, 10:eadk4321, 2024.

[4] Manuel S. Rudolph et al. Pauli propagation: A computational framework for simulating quantum systems. *arXiv preprint arXiv:2505.21606*, 2025.

[5] L. P. Cordella et al. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26:1367–1372, 2004.

[6] Michael M. Bronstein et al. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[7] European Parliament and Council of the European Union. Regulation (eu) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence and amending regulations (ec) no 300/2008, (eu) no 167/2013, (eu) no 168/2013, (eu) 2018/858, (eu) 2018/1139 and (eu) 2019/2144 and directives 2014/90/eu, (eu) 2016/797 and (eu) 2020/1828 (artificial intelligence act), 2024. Published: July 12, 2024. Entered into force: August 1, 2024.

[8] Nathalie A. Smuha and Karen Yeung. The european union's ai act: beyond motherhood and apple pie? In Nathalie A. Smuha, editor, *The Cambridge Handbook of the Law, Ethics, Policy and Artificial Intelligence*, pages 228–258. Cambridge University Press, 2025.

[9] R. Coates, D. Douglas, and M. Per. Ai and quantum computing ethics- same but different? towards a new sub-field of computing ethics. *Quantum Science and Technology*, 10(3):035030, 2025.