IOP Publishing Mach. Learn.: Sci. Technol. 6 (2025) 035034 https://doi.org/10.1088/2632-2153/adf811

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
12 April 2025

REVISED
6 July 2025

ACCEPTED FOR PUBLICATION
5 August 2025

PUBLISHED
19 August 2025

PAPER

Predicting the von Neumann entanglement entropy using a graph neural network

Anas Saleh

Department of Physics and Astronomy, University of Iowa, Iowa City, IA, United States of America

E-mail: anas-saleh@uiowa.edu

Keywords: graph neural networks, quantum information, machine learning

Abstract

Calculating the von Neumann entanglement entropy from experimental data is challenging due to its dependence on the complete wavefunction, forcing reliance on approximations such as classical mutual information (MI). We propose a machine learning approach using a graph neural network to predict the von Neumann entropy directly from experimentally accessible bitstrings. We test this approach on a Rydberg ladder system and achieve a mean absolute error of 3.6×10^{-3} when evaluating within the training range on a dataset with entropy values ranging from 0 to 1.9. The model achieves a mean absolute percentage error of 1.44% and outperforms MI-based bounds. When tested beyond the training range, the model maintains reasonable accuracy. Furthermore, we demonstrate that fine-tuning the model with small datasets significantly improves performance on data outside the original training range.

1. Introduction

Quantum entanglement is a cornerstone of quantum mechanics and a widely studied subject [1–4], and the von Neumann entanglement entropy serves as a key measure of entanglement with broad applications across physics, it is defined as

$$S_A^{\mathrm{vN}} = -\mathrm{Tr}\left(\rho_A \ln\left(\rho_A\right)\right), \tag{1}$$

where $\rho_A = \mathrm{Tr}_B \rho_{AB}$ is the reduced density matrix of subsystem A.

However, experimental measurement of S_A^{vN} faces a fundamental challenge: it requires complete knowledge of the quantum wavefunction, which is not directly accessible through measurements. Traditional approaches rely on full quantum state tomography, which scales exponentially with system size and becomes prohibitively expensive for many-body systems. This scaling bottleneck severely limits our ability to probe entanglement in the large-scale quantum systems where many-body phenomena emerge.

Recent work has begun to address this challenge by developing bounds and approximations for entanglement entropy. Notably, a study of Rydberg ladder systems demonstrated that classical mutual information (MI) derived from measurement bitstrings can provide a lower bound for S_A^{vN} [1]. While this approach represents significant progress, MI-based bounds are often loose and may not capture the full entanglement structure.

In this work, we propose a fundamentally different approach: using graph neural networks (GNNs) to predict S_A^{vN} directly from the same experimental bitstring data used for MI calculations. Our method bypasses the need for full wavefunction reconstruction by leveraging the correlation structure embedded in measurement outcomes.

GNNs provide a natural framework for this task because their graph structure naturally mirrors quantum systems: nodes represent individual particles while edges encode the correlations and interactions that give rise to entanglement. Unlike tomography, which requires exponentially many measurements, our approach can predict entanglement entropy from polynomial-scale experimental data, making it practically feasible for larger quantum systems.

The paper is structured as follows: section 2 reviews machine learning in quantum physics and GNNs, section 3 details our methodology (including data generation and model architecture), and section 4 presents our results and demonstrates how fine-tuning the model improves performance after initial training.

2. Background

2.1. Machine learning for quantum systems

Machine learning (ML) attracted interest from all fields of study and showed its ability to tackle many complex problems [5–8]. Quantum physics has emerged as one of the heavily affected fields and it was shown that ML can offering novel solutions to problems once deemed intractable due to the inherent complexity of quantum systems. Quantum mechanics, governed by principles such as superposition, entanglement, and exponential state-space scaling, poses unique challenges for both theoretical modeling and experimental data analysis. Traditional computational methods, such as exact diagonalization or quantum Monte Carlo, often struggle with the 'curse of dimensionality' in many-body systems or the noise susceptibility of quantum hardware. Machine learning, with its capacity to approximate high-dimensional functions, extract patterns from sparse data, and optimize complex systems, has become an indispensable tool in advancing quantum research [9].

In quantum many-body physics, neural networks revolutionized the representation of wavefunctions. For instance, neural quantum states leverage deep learning architectures to approximate ground and excited states of correlated quantum systems, bypassing the limitations of conventional variational methods. These approaches have enabled precise simulations of spin lattices, fermionic systems, and quantum phase transitions [10, 11].

Quantum computing has also benefited from ML. In [12], it was shown that neural-network-based reinforcement learning can discover quantum error correction (QEC) strategies for protecting logical qubits against noise in few-qubit systems.

Meanwhile, generative models like quantum autoencoders assist in QEC, leading to extended lifetimes of logical qubits [13]. These advances are critical for bridging the gap between theoretical quantum advantage and practical implementation.

2.2. GNNs in physics

GNNs have emerged as a powerful framework for modeling systems with inherent relational or topological structure, such as atomic lattices, particle interaction networks, and various other physical systems. Unlike traditional neural architectures, GNNs operate directly on graph-structured data, leveraging message-passing mechanisms to propagate and aggregate information between nodes and edges [14–16]. This capability aligns naturally with physics problems where locality, symmetry, and connectivity play defining roles, such as in material science, quantum chemistry, and particle physics [17].

Examples of GNN applications in physics include predicting material properties in condensed matter physics, as they are especially suited for such tasks [18], and top tagging as discussed in [19]. In quantum physics applications, GNNs have been successfully applied to particle track reconstruction in high-energy physics and quantum state analysis. They have been implemented in [20] to predict the Rényi entropy, demonstrating their versatility across different domains of physical sciences.

The field has also seen significant developments in quantum machine learning frameworks that exploit graph structures. Recent works have established quantum GNNs as powerful tools for quantum machine learning of graph-structured data, with applications ranging from materials science to high-energy physics [21, 22]. These approaches leverage the natural graph representation of quantum systems to improve learning performance through specialized quantum neural network architectures.

3. Methodology

3.1. Dataset

In this work, we studied a Rydberg system described by the following Hamiltonian:

$$H = \frac{\Omega}{2}\sum_i \left(e^{i\phi}\left|g\right\rangle_i\left\langle e\right|_i + e^{-i\phi}\left|e\right\rangle_i\left\langle g\right|_i\right) - \Delta\left(t\right)\sum_i \hat{n}_i + \sum_{i<j}\frac{C_6}{\left|\vec{r}_i - \vec{r}_j\right|^6}\hat{n}_i\hat{n}_j. \tag{2}$$

Where $C_6 = 5.42 \times 10^{-24}\,\frac{m^6}{s}\,\text{rad}$, Ω is the Rabi drive amplitude, and Δ is the detuning.

The atoms were arranged in a two-leg ladder configuration, with the y-separation being twice the x-separation. This system is based on the Aquila quantum computer developed by QuEra.

To generate our dataset, we randomly sampled specific ranges of the Hamiltonian parameters. But instead of sampling over Δ, Ω, and a (where a is the lattice x-spacing), we normalized our Hamiltonian by dividing by Ω. Then, we sampled the dimensionless ratios $\frac{\Delta}{\Omega}$ and $\frac{R_b}{a}$, where R_b is defined as $R_b = \left(\frac{C_6}{\Omega}\right)^{1/6}$. Additionally, we sampled the number of rungs (ladder legs) in the system. The sampling ranges were as follows:

$$\begin{aligned} \frac{\Delta}{\Omega} &\in [0,6]\,, \\ \frac{R_b}{a} &\in [0.1,5]\,, \\ N_{\text{rungs}} &\in [1,6]\,. \end{aligned} \tag{3}$$

The number of samples taken for $\frac{\Delta}{\Omega}$ and $\frac{R_b}{a}$ were evenly distributed. However, for the number of rungs, we increased the number of samples for larger systems to account for the growing complexity with system size. Specifically, the number of samples for rungs 1 through 6 were 30 000, 60 000, 100 000, 200 000, 350 000, 500 000, respectively. These sampled regions span three distinct ordered phases along with a disordered phase covering a region of great theoretical interest [23].

For each sample, we diagonalized the Hamiltonian, computed the ground state, randomly selected a subsystem, and calculated the von Neumann entanglement entropy.

3.2. Preprocessing

Each data point was transformed into a fully connected graph, where nodes represent the Rydberg atoms, and edges represent the interactions between these atoms. We then defined a set of experimentally accessible features for the nodes, edges, and the graph as a whole.

3.2.1. Node features

For each node, we assigned the following features:

- **Coordinates**: The atoms were placed on a grid with x-spacing $= 1$ and y-spacing $= 2$.
- **Rydberg state probability**: The probability of the atom being in the Rydberg state, calculated using the ground state probabilities.
- **Subsystem mask**: A binary value indicating whether the node belongs to subsystem 'A' or 'B'.

3.2.2. Edge features

For each edge, we defined the following features:

- **Normalized distance**: The distance between the connected nodes divided by the square root of the total number of atoms.
- **Angle with horizontal axis**: The angle that the edge makes with the horizontal axis.
- **Two-point correlation**: Defined as $C_{ij} = \langle r_i\, r_j\rangle - \langle r_i\rangle\langle r_j\rangle$.

The first two features were chosen to encode the geometric structure of the system, while the third feature captures the correlations between Rydberg atoms.

3.2.3. Global features

Finally, we defined two global features for the graph:

- n_A: The fraction of the system in subsystem 'A'.
- n_B: The fraction of the system in subsystem 'B'.

The features are summarized in table 1.

We found that this set of features effectively captures the relationships between the atoms and the entanglement entropy, while avoiding unnecessary complexity that could overwhelm the model.

3.3. Model architecture

Our model was constructed using PyTorch Geometric library and consists of the following components:

Table 1. Summary of graph neural network features.

Type	Feature
Node features	Coordinates Rydberg state probability Subsystem mask
Edge features	Normalized distance Angle with horizontal Two-point correlation
Global features	n_A n_B

3.3.1. Node & edge encoding

We implemented node and edge encoders as 2-layer multilayer perceptrons (MLPs) with BatchNorm and SiLU (Sigmoid linear unit) activation. Each MLP transformed the input features (four-dimensional for nodes, three-dimensional for edges) to a hidden representation of dimension 512.

3.3.2. Edge attention layers

We introduced an MLP designed to give the model learnable weights for the importance of separate edges. The edge attention mechanism consisted of a 3-layer MLP applied at each message-passing layer, producing attention weights in the range [0, 1] through a sigmoid activation.

3.3.3. Edge Representation MLP

We took the output of the edge encoding MLP as input and generated another edge representation with a MLP consisting of 2 layers, BatchNorm, SiLU activation, and dropout.

3.3.4. Message passing layers

We implemented a multi-layer architecture for edge-node co-processing, alternating between two distinct message-passing mechanisms across 6 layers. For even-indexed layers, we employed GINEConv [14] operations with enhanced edge features. Each GINEConv utilized a 3-layer MLP that processed node representations, consisting of two linear transformations with BatchNorm, SiLU activation, and dropout ($p = 0.4$) between layers. For odd-indexed layers, we leveraged TransformerConv [15] operations with edge-aware attention mechanisms. Each TransformerConv implemented multi-head attention with 8 attention heads, where each head processed features of dimension 64. The TransformerConv incorporated beta-transformations and concatenated the outputs from different attention heads. Both convolution types incorporated edge features multiplied by their weights of dimension 512 to guide the message-passing process, enabling rich interactions between node and edge representations throughout the network.

Residual connections were added after each message-passing layer to stabilize training and improve gradient flow. Specifically, the output of each convolution operation h_{new} was added to the previous node features h.

After every even-indexed message-passing layer, we applied an intermediate processing step to further refine the node features. This step consisted of a 2-layer MLP with BatchNorm, SiLU activation, and dropout. The MLP transformed the node features while preserving their dimensionality, allowing for additional feature extraction and regularization.

At the end of each message-passing layer, we updated the edge representation and weights using separate MLPs and used the updated representations as input for the next message-passing layer.

3.3.5. Graph-level readout

For obtaining a graph-level readout, we implemented a multi-head readout mechanism using two Set2Set modules with 4 processing steps each.

To manage the increased dimensionality from the multi-head readout (2048), we employed a dimension reduction projection. This projection consisted of a linear transformation followed by BatchNorm, SiLU activation, and dropout regularization, reducing the representation to 1024.

3.3.6. Global features MLP

We added an MLP consisting of three layers, BatchNorm, SiLU activation, and dropout to process the global features.

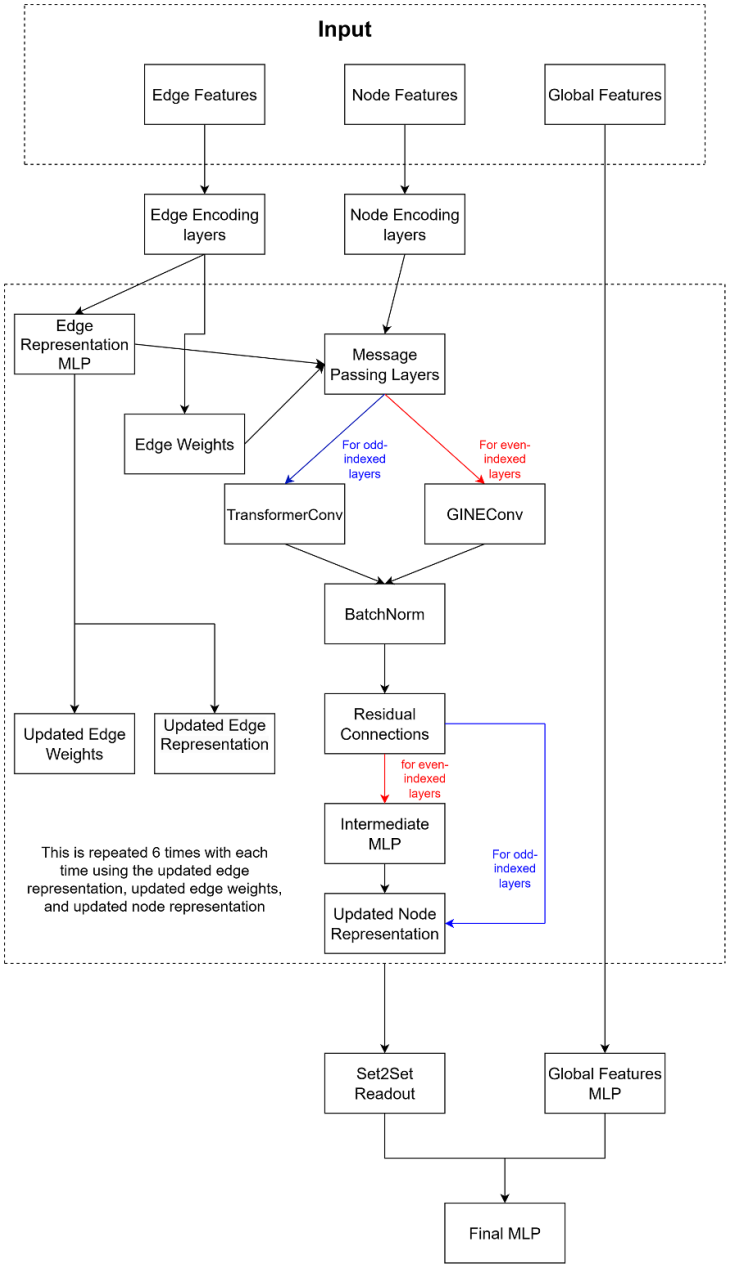

Figure 1. A flow chart of the GNN model, red paths are only taken by even-indexed layers, and blue paths are only taken by odd-indexed layers.

3.3.7. Final MLP

The final MLP combined the graph-level readout and the global features MLP output, producing a single scalar value representing the von Neumann entropy. The final MLP consisted of four layers with BatchNorm, SiLU activation, and dropout, followed by a Softplus activation to ensure non-negative output.

A flow chart of the model is shown in figures 1 and 2.

3.4. Training

We trained the model for 500 epochs using Kaiming initialization [24]. We used the AdamW optimizer [25] with a learning rate of 1.5×10^{-4} and weight decay of 10^{-4}. To stabilize the training, we applied gradient clipping with a maximum gradient norm of 1.0. We also used the CosineAnnealingWarmRestarts scheduler [26] with the following parameters: $T_0 = 50$, $T_{\text{mult}} = 2$, and $\eta_{\text{min}} = 10^{-6}$. The exact values for the optimizer parameters, scheduler parameters, dropout rate, hidden layer dimensionality, and the number of message-passing layers were determined using optimization code that minimized our validation loss over 100 epochs. We specifically used a Tree-structured Parzen Estimator [27] over those parameters and obtained the values used in the final training code.

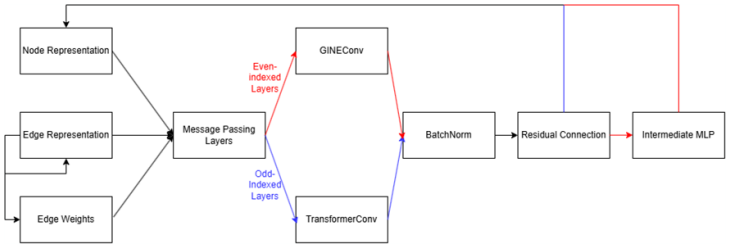

Figure 2. A flow chart of the message passing blocks, red paths are only taken by even-indexed layers, and blue paths are only taken by odd-indexed layers.

Table 2. Mean absolute error for each system size.

System size	MAE$\times 10^{-3}$	System size	MAE$\times 10^{-3}$
2	7.386	8	3.035
4	4.439	10	3.013
6	2.801	12	3.868

Table 3. Mean absolute percentage error for each system size.

System size	MAPE	System size	MAPE
2	4.33%	8	1.47%
4	1.62%	10	1.67%
6	3.89%	12	1.27%

4. Results

We used three metrics to evaluate our model. The first was our loss function defined as:

$$\text{loss} = \frac{1}{N}\sum \log\left(\cosh\left(S_{\text{pred}} - S\right)\right), \tag{4}$$

the second was the mean absolute error defined as:

$$\text{MAE} = \frac{1}{N}\sum \left|S_{\text{pred}} - S\right|, \tag{5}$$

and the third was the mean absolute percentage error defined as:

$$\text{MAPE} = \frac{100}{N}\sum \left|\frac{S_{\text{pred}} - S}{S}\right|. \tag{6}$$

However, we only evaluated the MAPE for points that had an actual entropy value higher than 0.01 of the maximum value in the dataset. This threshold approach prevented artificially inflated error metrics in regions of near-zero entropy, where even small absolute errors could result in disproportionately large percentage errors.

After training the model for 500 epochs, the model achieved a validation loss of 1.5×10^{-5} and a validation MAE of 3.6×10^{-3}. We evaluated the entire dataset after training and recorded the results in tables 2 and 3, and plotted them in figure 3.

4.1. Uncertainty quantification

We evaluated the predictive uncertainty of our model using Monte Carlo dropout [28]. We ran our prediction 50 times per point and took the 2.5th and 97.5th percentiles to construct a 95% confidence interval (CI). However, the original model was overconfident, only giving 83.6% coverage, so we scaled the spread of our predictions by a temperature parameter T while maintaining the mean of our predictions. We obtained an optimal scaling temperature $T = 1.11$ for our 95% CI, which gives us an average interval width of 0.053. We plotted the coverage vs temperature in figure 4, and the model predictions with their calibrated intervals are shown in figure 5.

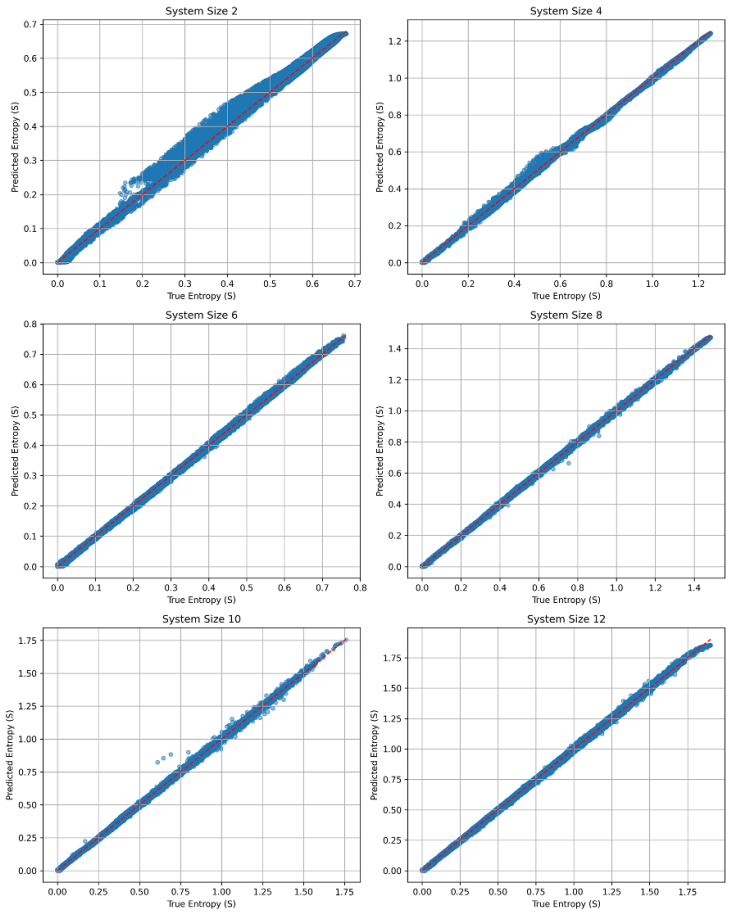

Figure 3. Predicted (y-axis) vs. actual (x-axis) von Neumann entanglement entropy. The red dashed line represents perfect prediction.

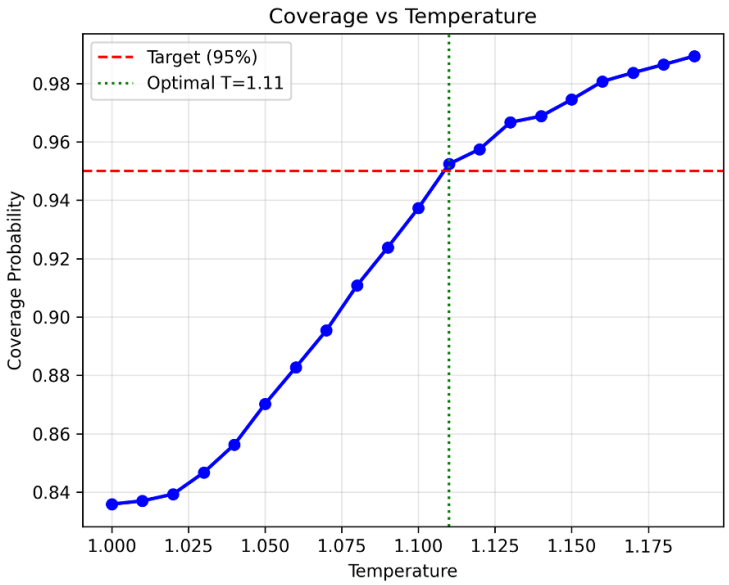

Figure 4. Coverage of the model predictions(y-axis) vs temperature parameter(x-axis).

4.2. Comparative analysis

We conducted a paired-t test to compare the accuracy of our model compared to the classical MI estimation on 50 000 points. Both methods showed statistically significant systematic bias (GNN: mean bias = 0.0007,

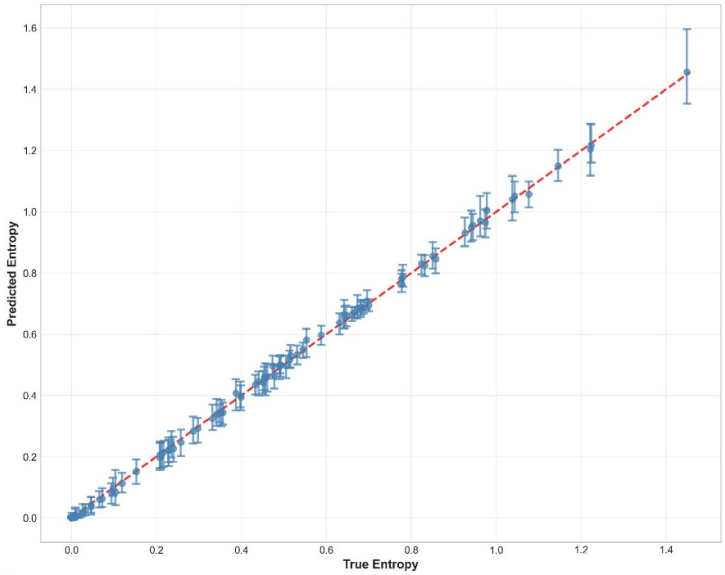

Figure 5. Predicted vs. actual von Neumann entanglement entropy with calibrated 95% confidence intervals. A representative subset of 100 points is shown for clarity. The red dashed line indicates perfect prediction.

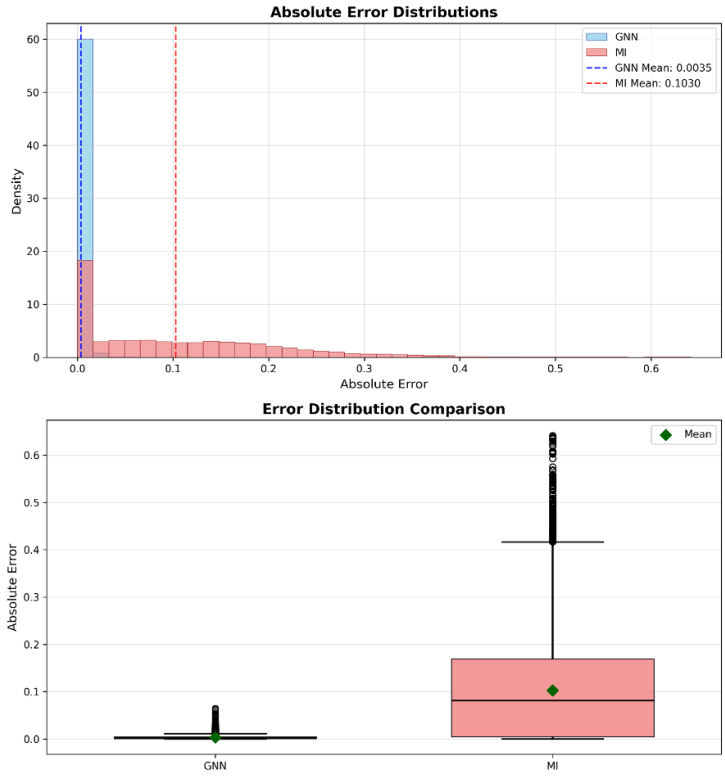

Figure 6. (Top) Probability density distributions of absolute errors for GNN-based (blue) and MI-based (red) Von Neumann entropy estimation. (Bottom) Box plots comparing error distributions of both methods. Both plots were made on a 50 000 points sample.

$t(49\,999) = 32.51$, $p < 0.001$; MI: mean bias $= 0.069$, $t(49\,999) = 122.51$, $p < 0.001$), though the GNN bias was substantially smaller.

Comparing the errors of both methods we see that the GNN significantly outperformed the MI approach ($t(49\,999) = 223.73$, $p < 0.001$, Cohen's d $= 1.40$). The mean absolute error was reduced from 0.103 (MI) to 0.0035 (GNN), representing a 96.6% improvement. We Plotted the error distributions of both methods in figure 6.

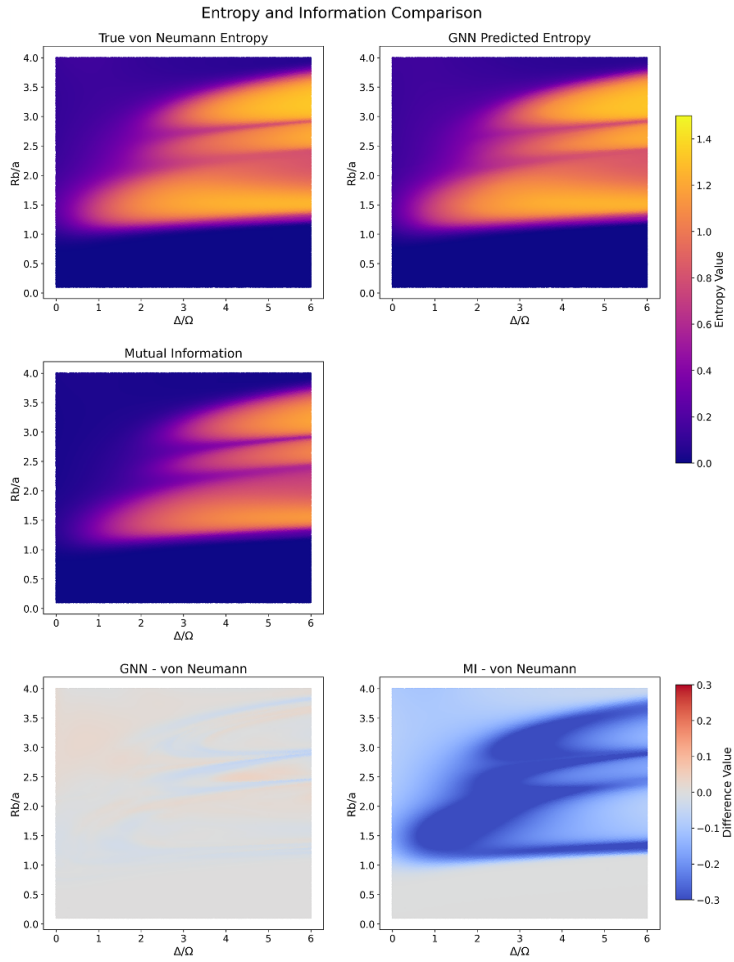

Figure 7. Comparison of von Neumann entropy estimates across parameter space. Heat maps show (top row) true von Neumann entropy and GNN-predicted entropy, (middle) mutual information, and (bottom row) prediction errors (GNN—von Neumann and MI—von Neumann) as functions of $\frac{\Delta}{\Omega}$ and $\frac{R_b}{a}$.

We also plotted in figure 7 the von Neumann entropy, MI, and our model predictions in the phase space of $\frac{R_b}{a}$ and $\frac{\Delta}{\Omega}$ for a symmetrical partition of a 6-rung system, and we can clearly see that our model prediction is much better than the classical MI estimation.

4.3. Model robustness

We also tested the model for robustness against small fluctuations in the input, which is very important for any model that depends on experimental data. First we tested the model with limited input data, we accomplished this by instead of using the full probabilities from our matrix diagonalization data, we sampled 10 000 points from the probability distribution, this effectively simulates an experiment with 10 000 readings. We obtained a MAE of 0.007 738 and a MAPE of 4.12% on the symmetric partition dataset, compared to the MAE and MAPE obtained before truncating our wavefunction, which was 0.008 720 and 3.84% respectively. Our results can be seen in figure 8.

Another experimental error simulation we did was a bit-flip readout error where each bit has a 1% chance of flipping, we did this by randomly applying the bit flip on a sampled 10 000 states from the probability distribution. We achieved a MAE of 0.016 812 and a MAPE of 5.33%, the results are plotted in figure 9.

We also tested the model robustness for wrong classification of the atoms subsystem, we did this by flipping the subsystem of the atoms at the boundary between the two subsystems with a probability of 1%, the results are shown in figure 10, and all of the experimental errors analysis are summarized in table 4.

Looking also at Cohen's d of the two prediction datasets we can see that it increases when we include experimental errors, signaling that our model might have learned quantum correlations that the classical ML does not have causing its prediction degradation to be lower, we listed the Cohen's d values in table 5.

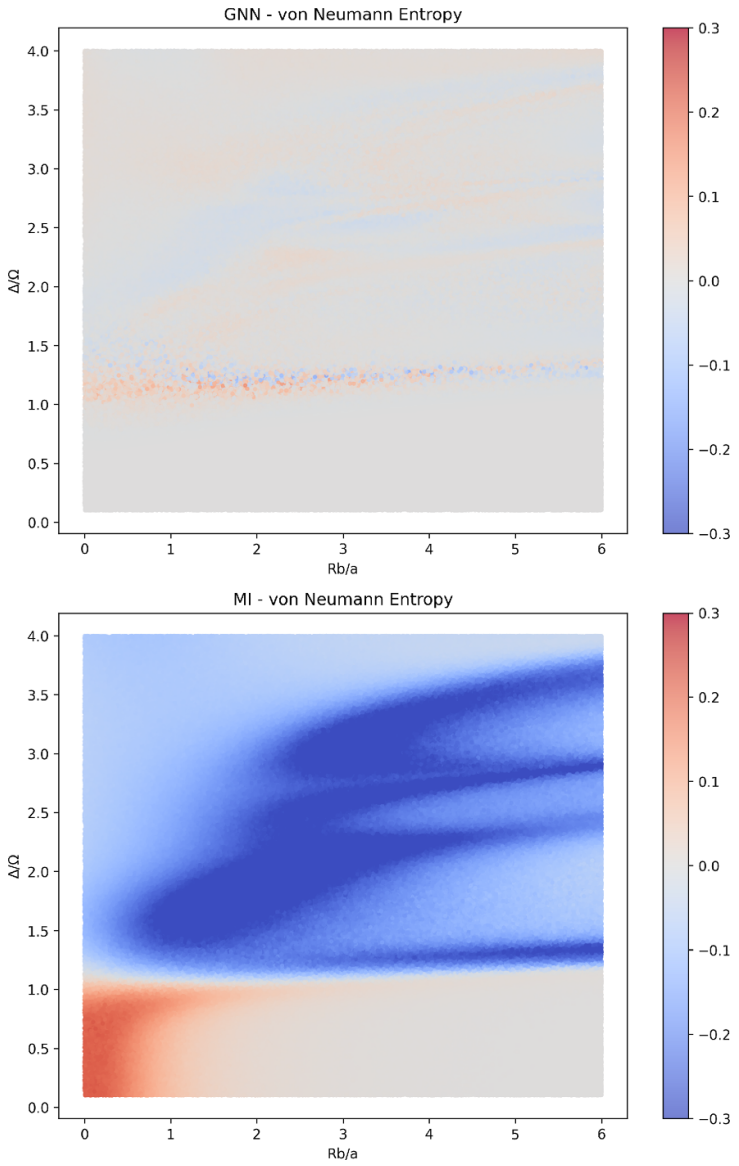

Figure 8. Heat maps of the (Top) GNN—von Neumann entropy, and the (Bottom) MI—von Neumann entropy with limited input data.

4.4. Fine-tuning the model

Another important feature the model should possess is generalization, as generating data for more than 6 rungs in the same volumes we used to train the model becomes computationally expensive rapidly. Therefore, we tested our model on datasets of 7–10 rungs at specific values for $\frac{\Delta}{\Omega}$. The results are shown in tables 6 and 7.

While our model outperformed MI for sizes larger than those it was trained on, the MAE and MAPE were not within acceptable error margins. This degradation occurs because the model was trained exclusively on systems with 1–6 rungs, limiting its ability to capture the scaling behavior of larger quantum systems. To address this limitation, we explored fine-tuning as a practical solution for extending the model's applicability to larger systems of interest.

We fine-tuned the model weights for 7 and 8 rungs using a significantly smaller dataset of 10 000 samples, compared to the much larger original training dataset. We trained the model for 200 epochs using a 5×10^{-5} learning rate and 1.5×10^{-4} weight decay. We achieved a validation loss of 1.84×10^{-4} and MAE of 1.0495×10^{-2}, which represented a significant improvement over the original model. We plotted our results for symmetrical partitions in figure 11, and the MAE and MAPE are listed in tables 6 and 7. We also computed the relative improvements the fine tuning provide and listed them in in table 6.

Fine-tuning achieved substantial improvements across all system sizes, with an overall average improvement of 49.1%. Notably, the model showed excellent transferability, with 9–10 rung systems (outside the fine-tuning range) achieving lower, but significant improvements (13.9%–54.6%) compared to the 7–8

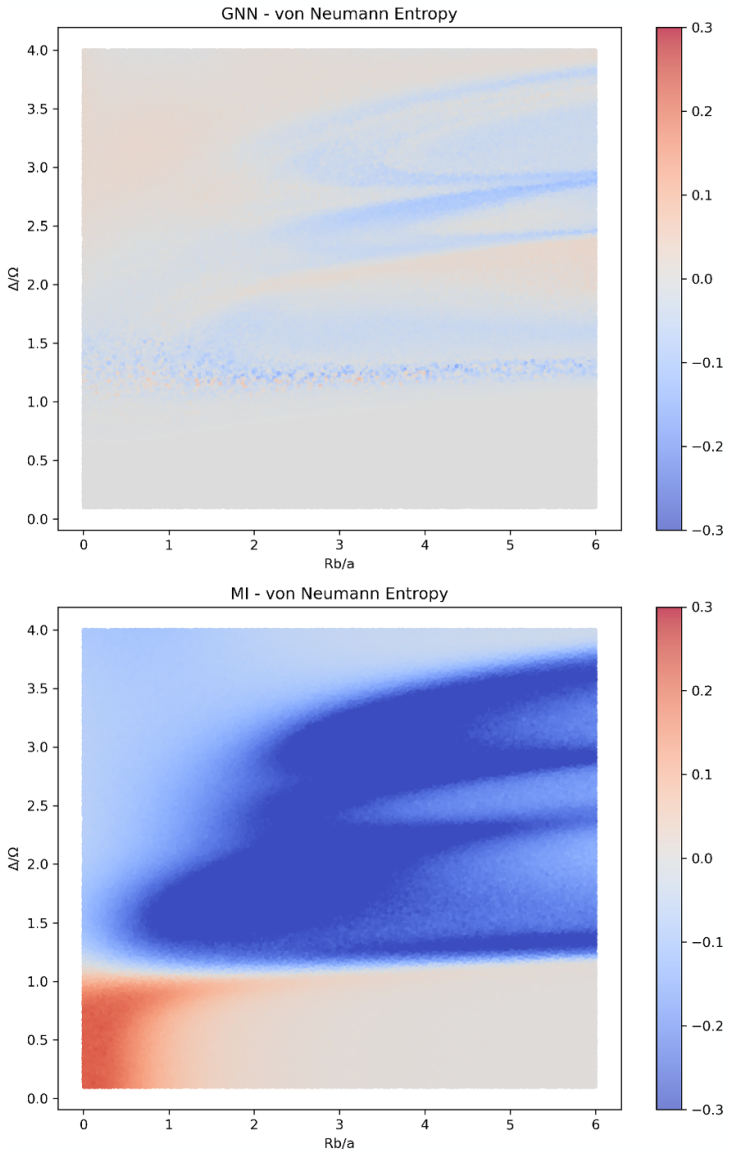

Figure 9. Heat maps of the (Top) GNN—von Neumann entropy, and the (Bottom) MI—von Neumann entropy with limited input data and readout bit-flip errors.

rung systems (44.3%–66.1%) used for fine-tuning. This demonstrates that the learned representations capture scalable entanglement features that generalize beyond the immediate training scope.

We investigated data augmentation for fine-tuning by generating multiple bipartition configurations per parameter set, effectively multiplying our training samples by a factor of n. However, this approach degraded performance, likely because the model overfit to the limited range of $\frac{R_b}{a}$ and $\frac{\Delta}{\Omega}$ values in the small dataset.

4.5. Targeted fine-tuning

Up to this point, all of our training and fine-tuning was on random partitions and sampled over the entire phase space, while this is great to maintain the generality of the model we also tested how would fine-tuning for a specific partition and part of the phase space would preform, so we took the model trained on 1–6 rungs and fine-tuned it separately on the 4 datasets in figure 11, taking only 100 points for the training, all the data points are partitioned symmetrically and have a $\frac{\Delta}{\Omega} = 2.5$, the results are shown in figure 12 and table 8. These results show that fine-tuning for a specific set of parameters could be very effective without requiring much data.

5. Limitations

This study has several limitations that warrant further investigation. First, while our model demonstrated reasonable generalization performance, the significant increase in MAE for larger systems and the substantial

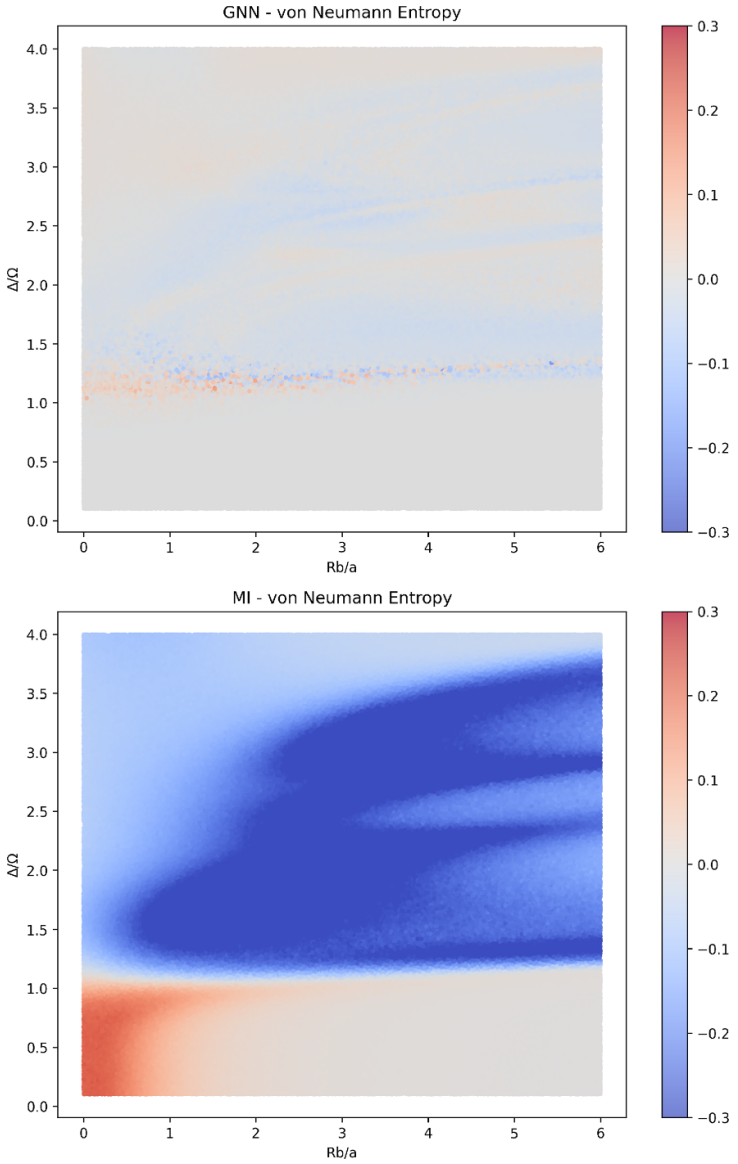

Figure 10. Heat maps of the (Top) GNN—von Neumann entropy, and the (Bottom) MI—von Neumann entropy with limited input data and boundary miss-classification errors.

Table 4. Comparing the MAE and MAPE for different experimental errors.

Kind of error	MAE$\times 10^{-3}$	MAPE
No experimental errors	8.72	3.84%
Limited sample	7.738	4.12%
Limited sample and readout errors	16.812	5.33%
Limited sample and classification errors	13.682	4.74%

Table 5. Cohen's d between our GNN predictions and the MI for different experimental errors.

Kind of error	Cohen's d
No experimental errors	1.803
Limited sample	1.935
Limited sample and bit-flip errors	1.870
Limited sample and classification errors	1.911

improvement from fine-tuning suggests that the model learned entropy representations but failed to fully capture system size scaling laws. Future studies training on datasets with broader size distributions may better teach the model proper scaling behavior.

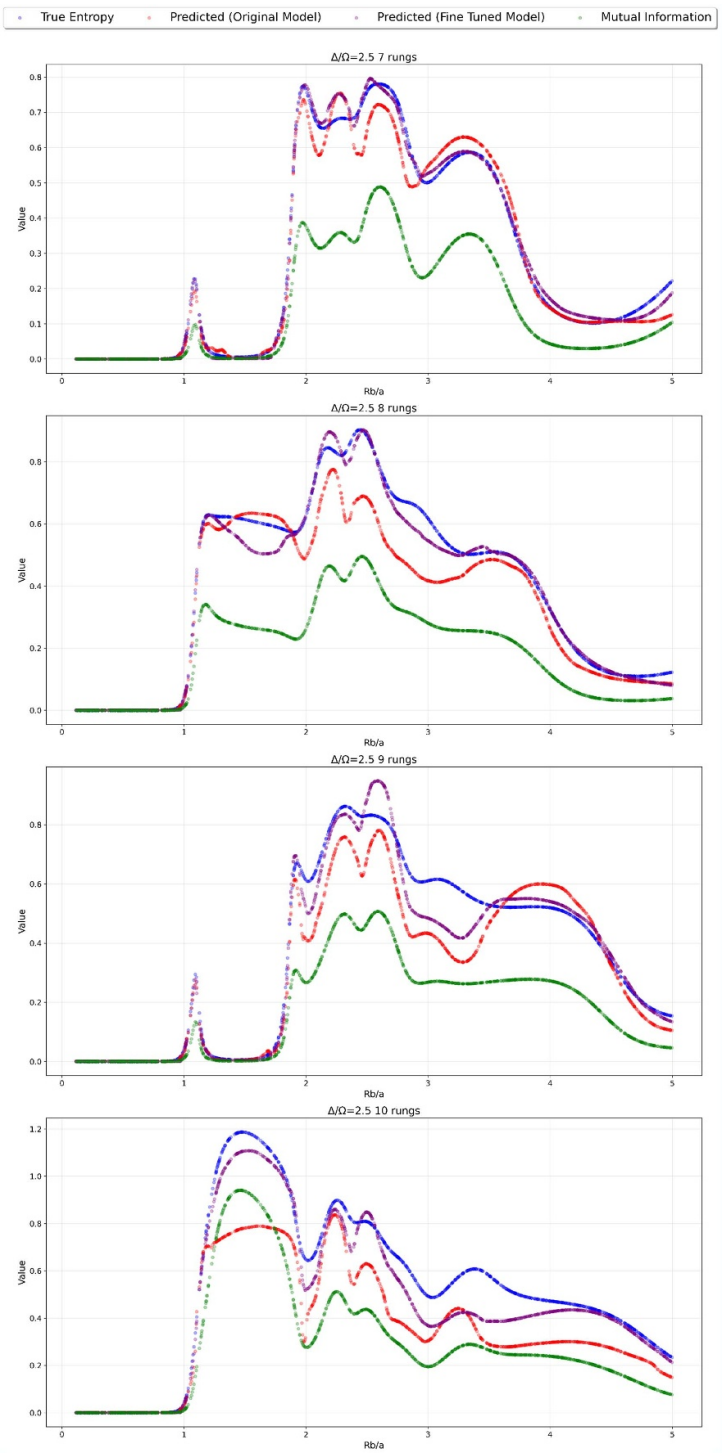

Figure 11. Von Neumann entropy (blue), the original model (red), the fine-tuned model (purple), and the MI (green) as a function of $\frac{R_b}{a}$.

Second, the model's physical interpretation requires deeper analysis. Comparing results with DMRG calculations and examining the hidden layer representations could provide valuable insights into the underlying physics.

Third, while fine-tuning significantly improved performance on larger system sizes (7–10 rungs), this approach assumes consistent entanglement scaling behavior across system sizes. If the underlying physics changes significantly with scale, fine-tuning with small datasets may fail to generalize, potentially requiring more comprehensive retraining with larger datasets.

Table 6. Performance comparison before and after fine-tuning with improvement metrics.

Number of rungs	$\frac{\Delta}{\Omega}$	MAE$\times 10^{-2}$ (before fine-tuning)	MAE$\times 10^{-2}$ (after fine-tuning)	Improvement (%)
7	2.5	3.145	1.388	55.9%
7	3.5	3.819	1.516	60.3%
8	2.5	6.555	2.223	66.1%
8	3.5	5.528	3.081	44.3%
9	2.5	6.936	3.834	44.7%
9	3.5	4.2231	3.637	13.9%
10	2.5	14.977	6.801	54.6%
10	3.5	12.747	5.967	53.2%

Table 7. Comparing the MAPE for the original and fine-tuned model.

Number of rungs	$\frac{\Delta}{\Omega}$	MAPE (before fine-tuning)	MAPE (after fine-tuning)
7	2.5	15.22%	12.92%
7	3.5	13.92%	8.61%
8	2.5	15.83%	6.99%
8	3.5	11.85%	6.82%
9	2.5	22.21%	12.73%
9	3.5	10.66%	8.44%
10	2.5	30.26%	14.35%
10	3.5	19.28%	8.71%

6. Discussion and conclusion

In this paper, we proposed an architecture for a GNN model that processes experimentally accessible data and predicts the von Neumann entanglement entropy. We tested this model on a Rydberg system similar to the one used in the Aquila quantum computer developed by QuEra.

Our model demonstrated excellent performance within its training range, achieving a mean absolute error of 3.6×10^{-3} and a mean average percentage error of 1.44%. When tested outside its training range, the model still outperformed the classical ML approximation, though with noticeably reduced accuracy. This limitation could be addressed by fine-tuning the model for specific system sizes of interest using small datasets. We demonstrated this approach by starting with our base model and fine-tuning it on 7- and 8-rung systems. We observed significant improvements in our predictions for 7–10 rung systems.

Furthermore, we showed that when focusing on a small region of the phase space and a specific partitioning scheme, we could achieve a mean absolute error comparable to our original training performance ($\sim 10^{-3}$) using very small fine-tuning datasets (100 points in our examples). This approach could be extended to any number of rungs by generating small datasets using appropriate approximation methods.

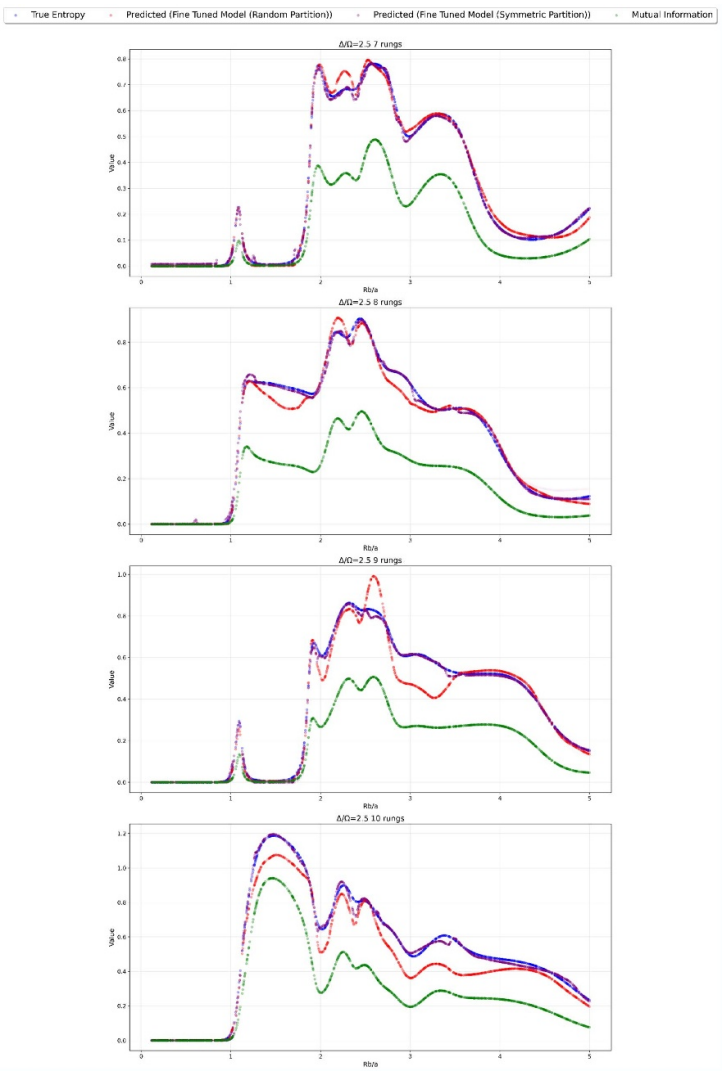

Figure 12. Von Neumann entropy (blue), the model fine-tuned on 10 000 points of 7 and 8 rungs randomly partitioned(red), the model fine-tuned on 100 point of a symmetric partition of the specific rung with $\frac{\Delta}{\Omega} = 2.5$(purple), and the MI (green) as a function of $\frac{R_b}{a}$.

Table 8. Performance comparison of 2 different fine-tuning approaches discussed in sections 4.4 and 4.5.

Number of rungs	MAE$\times 10^{-2}$ (fine-tuning)	MAE$\times 10^{-2}$ (Targeted Fine-tuning)	Improvement (%)
7	1.388	0.769	44.6%
8	2.223	0.879	60.5%
9	3.834	0.775	79.8%
10	6.801	1.282	81.1%

Data availability statement

The codes used to generate the dataset, preprocess it, train the model, and fine-tune it, along with the weights of the original and fine-tuned models, are openly available at the following URL/DOI: https://github.com/AsalehPhys/Predicting-von-Neumann-Entropy.

Acknowledgments

We would like to thank Professor Yannick Meurice for the valuable discussions that shed light on the specific system we studied and its properties. We would also like to thank Leen Saleh for her guidance in exploring the possible machine learning models that could work for our study.

This work was supported in part by the Department of Energy under AwardNumber DE-SC0010113.

Appendix. Architecture design choices

We evaluated several alternative GNN architectures and plotted the models MAE as a function of training epoch. First we Tested a pure GINEConv model and a pure TransformerConv model in opposition to our hybrid approach, the results can be seen in figure 13.

We also evaluated different readout mechanisms and plotted our results in figure 14.

Looking at the edge features, we tested the inclusion of higher order moments to see if that improve the information propagation through our model, we tested the forth cross-moment $E[(x_i - p_i)^2(x_j - p_j)^2]$ where the x is a binary value representing if the site is excited and p is the excitation probability. We did not find any improvements, the results are shown in figure 15.

Lastly, we evaluated how well does the model preform if we had a distance cutoff for graph edges instead of creating fully connected graphs. Our results can be shown in figure 16, while we can clearly see improvements for choosing a cutoff for edge creation, this makes the model generalization worse especially

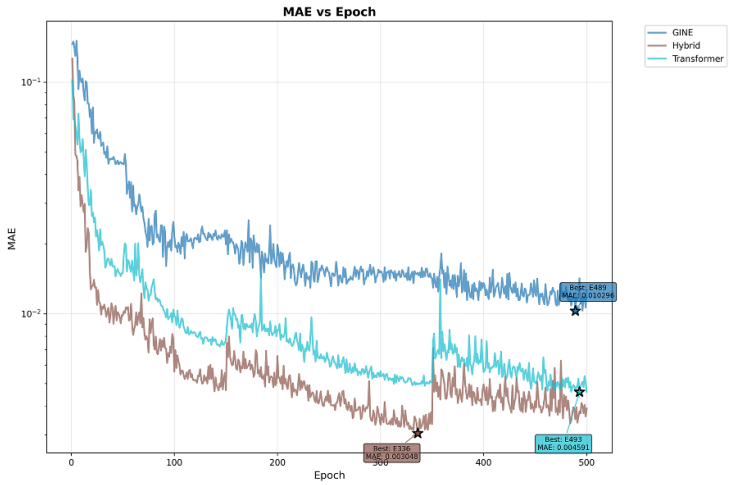

Figure 13. Validation set MAE as a function of training epochs for a pure GINEConv model, a pure TransformerConv model and our hybrid approach.

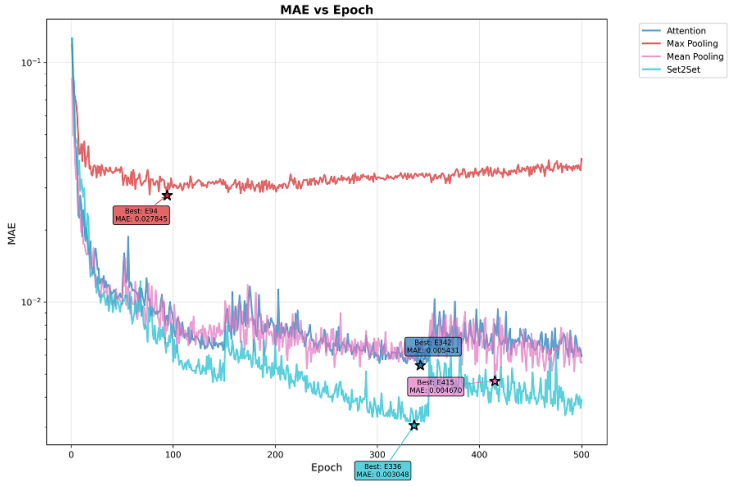

Figure 14. Validation set MAE as a function of training epochs for a max pooling, mean pooling, set2set, and attention readout.

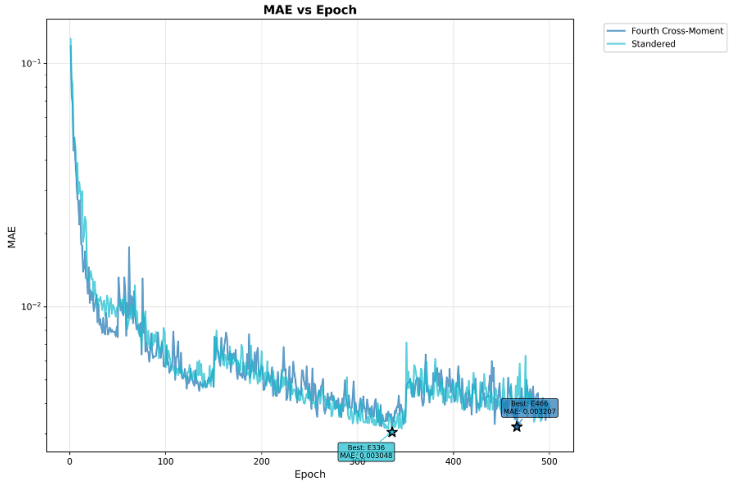

Figure 15. Validation set MAE as a function of training epochs for 2 model with the forth cross-moment added as an edge feature to one of them.

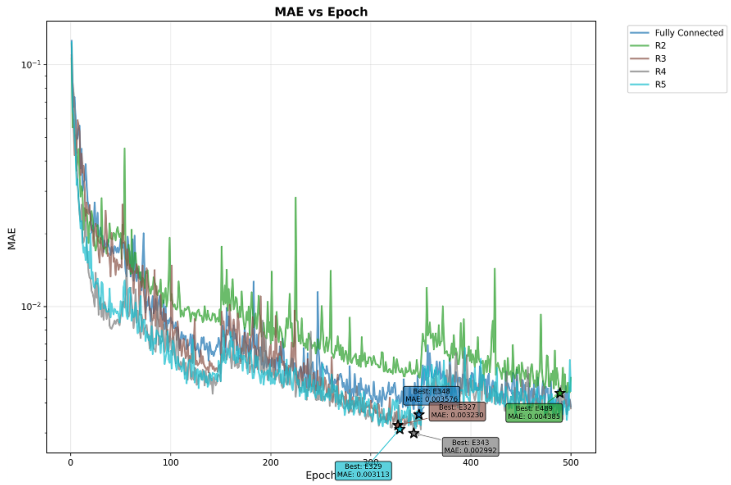

Figure 16. Validation set MAE as a function of training epochs for different values of edge creation cutoff distance from 2 units to fully connected.

when trying to predict far larger systems than what it was trained on. This is likely due to the inherit non-locality of the entanglement entropy. We show in figure 17 the generalization of 2 models, one that was trained on fully connected graphs and the other was trained on graphs with an edge creation cutoff of 4 units.

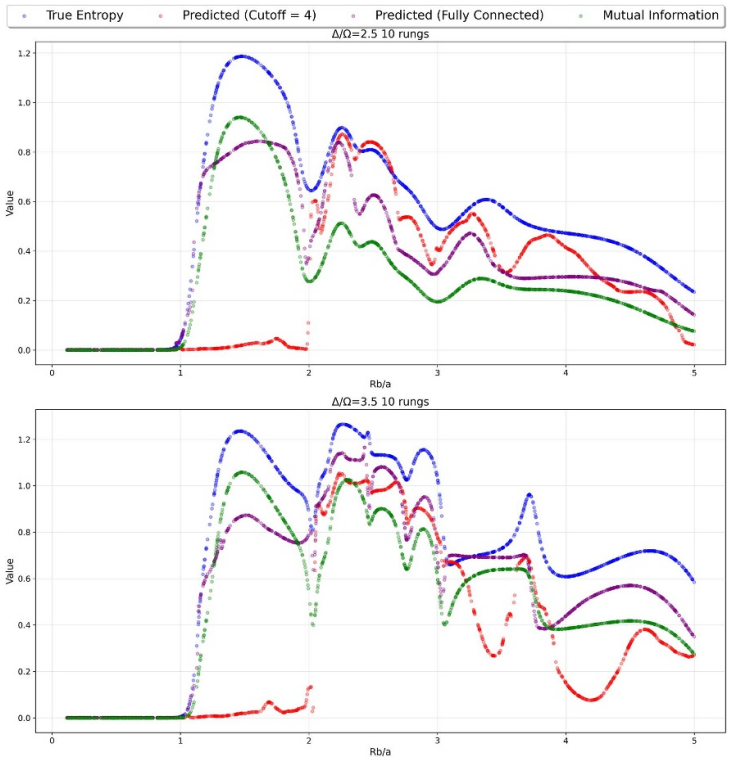

Figure 17. Comparision of the genralization of 2 models, one that was trained on partially connected graphs(red) and one that was trained on fully connected graphs(purple).

ORCID iD

Anas Saleh 0009-0006-8620-1867

References

[1] Meurice Y 2025 Lower bounds on entanglement entropy without twin copy *Phys. Rev. Research* **7** L022023
[2] Sun W, Jin Y and Gang L 2024 Genuine multipartite entanglement from a thermodynamic perspective *Phys. Rev. A* **109** 042422
[3] Huang J *et al* 2025 Integrated optical entangled quantum vortex emitters *Nat. Photon.* **19** 1–8
[4] Barr A J, Fabbrichesi M, Floreanini R, Gabrielli E and Marzola L 2024 Quantum entanglement and Bell inequality violation at colliders *Prog. Part. Nucl. Phys.* **139** 104134
[5] Shi G, Deng S, Wang B, Feng C, Zhuang Y and Wang X 2023 One for all: a unified generative framework for image emotion classification *IEEE Trans. Circuits Syst. Video Technol.* **34** 7057–68
[6] Shi H, Hayat M and Cai J 2024 Unified open-vocabulary dense visual prediction *IEEE Trans. Multimedia* **26** 8704–16
[7] Zhou Z, Ziwei Li, Zhou W, Chi N, Zhang J and Dai Q 2025 Resource-saving and high-robustness image sensing based on binary optical computing *Laser Photon. Rev.* **19** 2400936
[8] Zhang Z, Yuelei X, Song J, Zhou Q, Rasol J and Linhua M 2023 Planet craters detection based on unsupervised domain adaptation *IEEE Trans. Aerosp. Electron. Syst.* **59** 7140–52
[9] Carleo G and Troyer M 2017 Solving the quantum many-body problem with artificial neural networks *Science* **355** 602–6
[10] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L and Zdeborová L 2019 Machine learning and the physical sciences *Rev. Mod. Phys.* **91** 045002
[11] Pfau D, Spencer J S, Matthews A G D G and Foulkes W M C 2020 *Ab initio* solution of the many-electron Schrödinger equation with deep neural networks *Phys. Rev. Res.* **2** 033429
[12] Fösel T, Tighineanu P, Weiss T and Marquardt F 2018 Reinforcement learning with neural networks for quantum feedback *Phys. Rev. X* **8** 031084
[13] Locher D F, Cardarelli L and Müller M 2023 Quantum error correction with quantum autoencoders *Quantum* **7** 942
[14] Weihua H, Liu B, Gomes J, Zitnik M, Liang P, Pande V and Leskovec J 2019 Strategies for pre-training graph neural networks (arXiv:1905.12265)
[15] Shi Y, Huang Z, Feng S, Zhong H, Wang W and Sun Y 2020 Masked label prediction: unified message passing model for semi-supervised classification (arXiv:2009.03509)
[16] Huang C, Wang Y, Jiang Y, Ming Li, Huang X, Wang S, Pan S and Zhou C 2024 Flow2GNN: flexible two-way flow message passing for enhancing gnns beyond homophily *IEEE Trans. Cybern.* **54** 6607–18
[17] Battaglia P W *et al* 2018 Relational inductive biases, deep learning, and graph networks (arXiv:1806.01261)
[18] Xie T and Grossman J C 2018 Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties *Phys. Rev. Lett.* **120** 145301
[19] Huilin Q and Gouskos L 2020 Jet tagging via particle clouds *Phys. Rev. D* **101** 056019

[20] Rieger M, Reh M and Gärttner M 2024 Sample-efficient estimation of entanglement entropy through supervised learning *Phys. Rev. A* **109** 012403
[21] Ryu J-Y, Elala E and Rhee J-K K 2023 Quantum graph neural network models for materials search *Materials* **16** 4300
[22] Tüysüz C, Rieger C, Novotny K, Demirköz B, Dobos D, Potamianos K, Vallecorsa S, Vlimant J-R and Forster R 2021 Hybrid quantum classical graph neural networks for particle track reconstruction *Quantum Mach. Intell.* **3** 1–20
[23] Keesling A *et al* 2019 Quantum Kibble–Zurek mechanism and critical dynamics on a programmable Rydberg simulator *Nature* **568** 207–11
[24] Kaiming H, Zhang X, Ren S and Sun J 2015 Delving deep into rectifiers: surpassing human-level performance on imagenet classification *Proc. of the IEEE Int. Conf. on Computer Vision* pp 1026–34
[25] Loshchilov I and Hutter F 2017 Decoupled weight decay regularization (arXiv:1711.05101)
[26] Loshchilov I and Hutter F 2016 SGDR: stochastic gradient descent with warm restarts (arXiv:1608.03983)
[27] Bergstra J, Bardenet R, Bengio Y and Kégl B 2011 Algorithms for hyper-parameter optimization *Advances in Neural Information Processing Systems* p 24
[28] Gal Y and Ghahramani Z 2016 Dropout as a bayesian approximation: representing model uncertainty in deep learning *Int. Conf. on Machine Learning* (PMLR) pp 1050–9