Lightweight Machine Learning Decoding for Quantum Error Correction

Rihaan Sodhani

7 July 2026

Abstract

Practical quantum computation is limited by physical errors arising from environmental noise. Decoders interpret noisy information produced by quantum computers to predict logical errors. Many high-performance machine learning (ML) decoders rely on large, computationally intensive architectures, increasing inference latency and complicating real-time deployment. This study developed a lightweight feedforward neural network for decoding rotated surface codes on consumer-grade hardware. The model was trained at a single physical error rate under a depolarising noise model, then evaluated across physical error rates and under biased Pauli noise. It outperformed minimum-weight perfect matching (MWPM) at distance d=3 under both noise models and achieved comparable performance at d=5 under depolarising noise, though it performed slightly worse under biased noise. Additional classification metrics and receiver operating characteristic and precision-recall analyses demonstrated effective discrimination despite strong class imbalance. Inference time increased with distance and physical error rate for MWPM, with throughput showing opposite trends. By contrast, both quantities remained approximately constant for the ML decoder, indicating favourable scaling characteristics. However, MWPM was more memory-efficient. Ensembling and postselection were investigated to increase the accuracy of the ML decoder at d=5. An ensemble of three models or postselection with a reject ratio of 0.01 was sufficient for the ML decoder to exceed MWPM accuracy, although both approaches introduce deployment tradeoffs. Overall, this study demonstrates the potential of lightweight neural networks for low-distance, low-latency QEC decoding. It also shows the feasibility of conducting lightweight ML-based QEC research using accessible computational resources.

Keywords

Quantum error correction, rotated surface code, machine learning decoder, feedforward neural network, latency, ensembling, postselection

1. Introduction

Quantum computing has the potential to outperform classical computing in applications such as cryptography, materials modelling, and drug discovery. However, quantum states are inherently fragile and are easily disrupted by interactions with their environment. These interactions lead to decoherence and other noise processes, introducing errors that accumulate during computation. As a result, quantum error correction (QEC) is essential for building scalable and reliable quantum computers.

One of the leading strategies is the surface code. In a surface code of distance d, redundancy is introduced by encoding a logical qubit into a $d \times d$ grid of physical (data) qubits. In quantum systems, errors can occur in several forms corresponding to Pauli X, Y, and Z operations. A bit-flip (X) error changes the state of a qubit from $|0\rangle$ to $|1\rangle$ or vice versa when measured in the Pauli Z-basis. A phase-flip (Z) error manifests as a change from $|+\rangle$ to $|-\rangle$ or

vice versa when measured in the Pauli X-basis. A Y error involves a simultaneous bit-flip and phase-flip; any general error that decoheres a qubit can be expressed as a combination of these X, Y, and Z errors (1).

Errors are detected by periodically measuring X and Z stabilizer checks using $d^2 - 1$ stabilizers. The task of a decoder is to use the history of stabilizer measurements, the error syndrome, to apply a correction and obtain the correct logical outcome. Figure 1 shows the working of the rotated surface code. An important concept in QEC is the error threshold, a critical physical error rate below which increasing the size of the code exponentially decreases the error rate (1).

A practical quantum error-correcting decoder must satisfy two key requirements: it must accurately identify logical errors while processing syndrome information quickly enough to keep pace with the quantum processor (2). To enable large-scale fault-tolerant quantum algorithms, such as Shor's algorithm, logical error rates must ultimately be reduced to below approximately 10^{-10} per logical operation (3). Minimum-weight perfect matching (MWPM) is the most widely used decoder for the surface code because of its high decoding performance. However, as the code distance increases, the matching graphs become larger and more complex, increasing computational cost and decoding latency (4). This motivates the search for alternative decoding approaches that can scale more efficiently.

Machine learning (ML) techniques have recently been explored as an alternative approach to QEC. Decoding can be formulated as a supervised binary classification task, in which the input is a syndrome pattern and the output is whether a logical error has occurred.

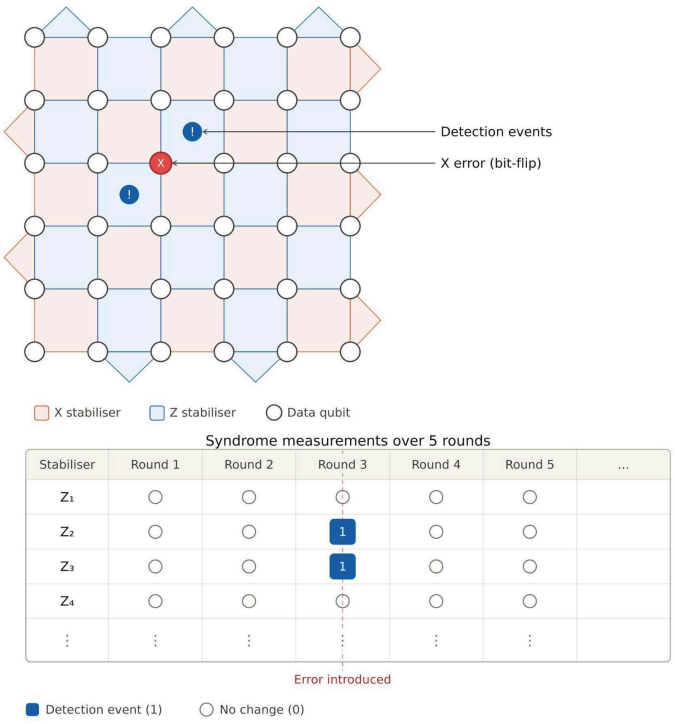


Figure 1. Detection events generated by a bit-flip error in the rotated surface code. Top: A single X (bit-flip) error occurring on a data qubit in the rotated surface code lattice produces

adjacent detection events on neighbouring Z-stabilisers. White circles represent data qubits, while blue and red plaquettes denote Z- and X-stabilisers respectively. Bottom: Syndrome measurements over multiple rounds showing how detection events are identified through changes in stabiliser outcomes between consecutive measurement cycles. A value of 1 indicates a detection event (parity change), while 0 indicates no change.

2. Background

Recent research has increasingly explored machine-learning techniques for QEC. One study compared several neural network (NN) decoder designs for the surface code, specifically feedforward and recurrent architectures, and found that recurrent NN architecture could achieve performance comparable to MWPM. Furthermore, the authors emphasised that decoder design must balance decoding performance with execution time, identifying training complexity and rapidly growing dataset requirements as major barriers to scaling neural-network decoders to larger code distances (5). Another work has demonstrated that NN decoding frameworks can be applied across different topological codes (triangular colour code and toric code) and multiple noise models, including correlated noise (6). This highlights the flexibility and generalisability of NN architectures compared to traditional decoding algorithms.

A recent paper developed a recurrent transformer-based NN decoder, evaluated on real-world data from Google's Sycamore quantum processor for distances d=3 and d=5. It achieved lower logical error rates compared to MWPM. This demonstrates that a neural decoder can outperform traditional decoders even on experimental quantum hardware. In further work, the approach was evaluated on simulated surface codes up to d=11, where it continued to outperform MWPM. However, the authors concluded that further optimisation was required before the decoder could satisfy the latency requirements of real-time quantum error correction (2).

Many high-performance ML decoders rely on large, computationally intensive architectures, increasing inference latency and complicating real-time deployment. This motivates the investigation of lightweight architectures. Hence, this research project aims to use simulation-based approaches to design and evaluate an ML decoder for the rotated surface code at low distances that can be run on consumer-grade hardware.

3. Dataset

The STIM (Stabilizer Circuit Simulator) library was used to generate data. A rotated surface code was defined with distance d and fixed physical error rate p. Two error models were used: a depolarizing noise model where the probability of each of the X, Y, and Z Pauli errors are equal to $p/3$ and a biased Pauli noise model in which Z errors occurred more frequently than X or Y errors. This reflects the type of bias common in many quantum computing architectures, such as superconducting qubits, quantum dots, and trapped ions (7). A moderate bias was introduced by setting the probability of a Z error to ten times the probability of an X or Y errors, i.e., the probability of a Z error was set to $10p/12$ while the probabilities of X and Y errors were each set to $p/12$. The surface code was implemented in

the X-basis, i.e., logical Z errors were detected. Under the symmetric noise model, the probabilities of X- and Z-type physical errors and thus logical errors are equal, so the two bases are equivalent and are expected to produce similar decoding performance. Under the biased noise model used in this study, Z errors occur more frequently than X or Y errors; consequently, the chosen basis captures the dominant error channel.

Four types of stochastic errors can occur: after-Clifford depolarization models errors occurring immediately after quantum gate operations; reset flip probability models imperfect qubit initialisation, where a qubit may be prepared in the wrong state; measurement flip probability represents readout errors that occur immediately before measurement; and before-round data depolarization models errors affecting idle data qubits between consecutive rounds of stabilizer measurements (8). All four error probabilities were set equal to the physical error rate p. This choice provides simplicity while serving as a conservative benchmark for decoder performance. Distributing noise isotropically across the error channels prevents an ML decoder from overfitting to a hardware-specific noise profile. The number of rounds (measurements per shot) was set to d. This is necessary as errors can also occur during measurement, so the decoder checks whether a stabiliser measurement changes between consecutive rounds. Fewer than d rounds would allow a chain of measurement errors alone to produce an undetectable logical error, reducing the effective code distance.

Logical failures are rare events at low physical error rates, resulting in an imbalanced dataset in which shots without logical errors greatly outnumber those containing logical failures. Consequently, enormous numbers of simulated shots were required simply to observe sufficient logical failures for training. To avoid storing these large datasets, training data were generated dynamically during training. Furthermore, using STIM, independent datasets were produced for training and testing, eliminating the need to store pre-generated datasets.

Each data sample corresponds to a single QEC shot or experiment. The input is in the form of a 2D tensor of dimensions (a, b), where a is the batch size used for training and b is the number of detector measurements in the circuit. 10^6 shots were sampled at a time, divided into batches of size 256. The large number of shots was chosen as logical errors are rare at low physical error rates. These rare logical errors allow the model to learn and reduce variance in the estimated logical error rate, producing tighter error bars. The input tensor represents the syndrome reading. Initially, STIM functions output an array that maps an odd parity (change) to 1 and an even parity (no change) to 0. Thus, the arrays are converted to tensors that can be inputted into the neural network decoder. Moreover, each value is multiplied by 2 and 1 is subtracted to map an odd parity (change) to 1 and an even parity (no change) to -1 to ensure that the model interprets the outcomes as distinct categories rather than values on a numerical scale.

The input syndrome data are labelled using a separate 1D tensor of length equal to the batch size. STIM functions output an array, which is then converted to a tensor. It maps each shot to 1 (logical error) or 0 (no logical error). Since the data are generated through simulation, the

true logical outcome of every shot is known exactly, providing ground-truth labels for supervised learning.

4. Methodology

4.1 *Threshold Estimation*

As a benchmark, an MWPM decoder was implemented using the PyMatching package (9). The MWPM threshold was found by plotting logical error rate against physical error rate for MWPM at different distances and estimating the point of intersection of these lines (10). ML models were trained at a high physical error rate below the threshold: a very low error rate would require a large number of shots to get sufficient failures to train the model, while training a model above threshold would make it inaccurate for the noise experienced by future quantum computers that operate below the threshold.

4.2 *Decoder Development, Training, and Evaluation*

A fully connected feedforward neural network (FNN) decoder was developed using Pytorch (11). The model takes syndrome vectors generated from surface-code detection events as input and outputs the probability of a logical error. Nonlinear activation is applied after each hidden layer using the SiLU (Sigmoid Linear Unit) function, while the final layer uses a sigmoid activation to produce a probability output (12). The classification threshold was set to 0.5: outputs greater than 0.5 were classified as logical errors, while outputs less than or equal to 0.5 were classified as logically correct states. The model was trained using binary cross-entropy loss, with parameters optimised using the Adam optimiser with a learning rate of 10^{-3} (13). The architecture of the model is shown below.

```
decoder = nn.Sequential(
  nn.Linear(num_input, 256),
  nn.SiLU(),
  nn.Linear(256, 1024),
  nn.SiLU(),
  nn.Linear(1024, 512),
  nn.SiLU(),
  nn.Linear(512, 256),
  nn.SiLU(),
  nn.Linear(256, 1),
  nn.Sigmoid()
)
loss_fn = nn.BCELoss()
optimizer = torch.optim.Adam(decoder.parameters(), lr=1e-3)
```

The ML decoder was trained at a single physical error rate below threshold under the depolarising noise model, providing a simple training regime that does not depend on a hardware-specific noise profile. During training, sufficient shots were used for the logical error rate to stabilise without significant increases with more training data (i.e. to achieve convergence). To test its ability to extrapolate beyond its training regime and generalise

across noise distributions, the model was evaluated at different physical error rates near or below the threshold under both the depolarising and biased Pauli noise models. The accuracy of a decoder is defined as $1 - P_L$, where P_L is the logical error rate. Both quantities were calculated for ML and MWPM decoders. However, since accuracy and logical error rate are complementary, only logical error rate was plotted against the physical error rate for both decoders.

However, the logical error rate does not distinguish between the different types of classification errors. Each decoding outcome was therefore classified as a true positive (TP), true negative (TN), false positive (FP), or false negative (FN), and confusion matrices were constructed for both decoders. From these, recall, false negative rate (FNR), specificity, and false positive rate (FPR) were calculated. Since recall is the complement of FNR and specificity is the complement of FPR, it was sufficient to plot FNR and FPR against physical error rate for both the ML and MWPM decoders.

Decoding near or below the threshold constitutes an imbalanced classification problem because logical failures are rare. Consequently, overall accuracy can be misleading, as a decoder that predominantly predicts "no logical error" may achieve high accuracy while performing poorly on the minority class. Thus, precision and recall were calculated to assess the decoder's performance in the logical error class, with the F1-score used to balance these two measures. Balanced accuracy was also calculated to give equal weight to the logical-error and no-logical-error classes (14). Finally, the Matthews correlation coefficient (MCC) was calculated as it incorporates all four entries of the confusion matrix—TP, TN, FP, and FN—and is therefore generally the best measure of classifier performance in the presence of strong class imbalance (15). The MCC was plotted against the physical error rate for both the ML and MWPM decoders.

4.3 *Receiver Operating Characteristics and Precision-Recall Analysis*

The accuracy of the ML decoder might be influenced by a poorly calibrated classification threshold. To assess classifier performance independently of the selected classification threshold, receiver operating characteristic (ROC) and precision-recall (PR) curves were plotted. Varying the classification threshold applied to the decoder's output from 0.0 to 1.0, the ROC curve plots the corresponding sensitivity against the FPR (16), while the PR curve plots precision against the recall. PR curves are particularly useful for highly imbalanced datasets, as they are generally more informative than ROC curves in such settings (17). The areas under both curves were subsequently calculated.

4.4 *Latency, Throughput, and Memory*

The model’s latency, defined as its inference time per shot; throughput, defined as the number of shots evaluated per second; and memory consumption were also evaluated. It was necessary to isolate the true execution latency from systemic hardware overhead, as initial experiments on the ML decoder exhibited significant fluctuations. Hence, single-threaded execution was enforced, and gradient tracking was disabled. Furthermore, the first timing trial for each data point was discarded prior to statistical calculation to eliminate initial memory

allocation and graph compilation overhead. Inference time and throughput were plotted against both code distance and physical error rate for the ML and MWPM decoders to compare their scalability and ability to process noisy syndromes. Since inference time for a fixed architecture is independent of the amount of training, the models were given minimal training before testing, given the impracticality of training models at high distances.

Model size and memory consumption also influence the feasibility of deploying a decoder within a real-time quantum computing system. Therefore, working RAM consumption was measured for both decoders. The number of trainable parameters and memory required to store the ML model weights were also recorded at d=3 and d=5. The experiments were conducted on two CPUs: an Apple M1 and an Intel Core i5-1135G7.

4.5 *Ensembling*

Ensembling involves training a set of classifiers and classifying new data points by taking a vote of their predictions (18). It can result in higher accuracies than the individual models. Ensembling was used to try to improve the model's accuracy at d=5. 15 models were trained with different initial weights assigned randomly, and each of them was made to classify the same data. The geometric mean of their sigmoid probability outputs was taken as it penalises strong disagreement between models more heavily; a result on which the models disagree is less reliable. To determine the number of decoders required to outperform MWPM, accuracy was plotted against the number of models in the ensemble.

4.6 *Postselection*

Postselection is a procedure for discarding the runs of a quantum computation that yield a specific measurement outcome (19). This is an alternative method to increase the accuracy of the ML decoder at d=5. The sigmoid activation function outputs a probability that can be used as an indicator of the confidence of the model: a value close to 0 or 1 indicates that the decoder's prediction is reliable, while a value closer to 0.5 indicates that the decoder is uncertain. Therefore, an output x was mapped to a confidence rating $|2x-1|$, so that 1 indicates certainty and 0 indicates a random choice. A specific fraction of the data in which the model was most uncertain, the reject ratio, was discarded. The extent of accuracy increase and the optimum reject ratio were investigated by plotting the accuracy of an ML decoder with postselection against the reject ratio. To evaluate the reliability of the decoder's confidence estimates, calibration analysis was conducted by plotting a calibration curve, with the actual flip rate plotted against the predicted flip probability. Additionally, the expected calibration error, maximum calibration error, and Brier score were calculated (20).

5. Results and Discussion

5.1 *Threshold Estimation*

Unless otherwise stated, all simulations used the depolarising noise model. The threshold physical error rate (p) for MWPM was estimated to be approximately $p \approx 0.007$, as shown in Figure 2. Previous studies have estimated the MWPM threshold for the surface code to lie between approximately 0.005 and 0.014, depending on the assumed noise model and

decoding framework (21). The threshold estimated in this study is therefore consistent with values reported in literature.

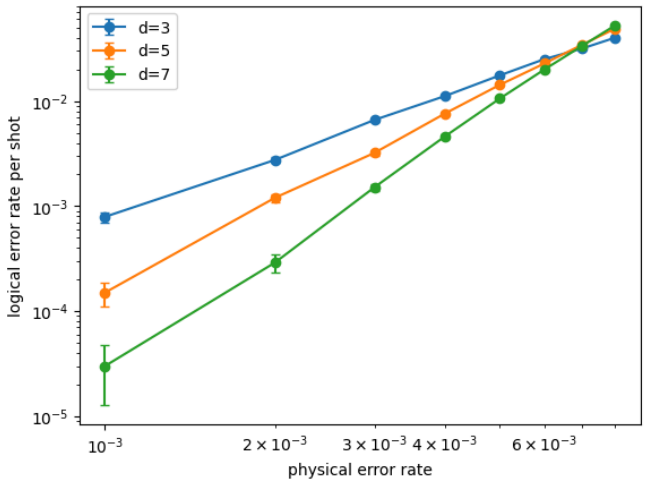


Figure 2. Logical error rate plotted against physical error rates from p=0.001 to p=0.008 for a rotated surface code at different distances using MWPM decoding. The logical error rate per shot is plotted against the physical error rate for code distances d=3 (blue), d=5 (orange), and d=7 (green) on logarithmic axes. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.

5.2 *Decoder development and training*

The ML decoder was trained at p=0.006 under the depolarising noise model. Though the precise threshold of the ML decoder may differ from the MWPM threshold, p=0.006 provides a sufficient margin below the MWPM threshold of p≈0.007. The model generalised to physical error rates outside its training regime and to biased Pauli noise, though it is unclear whether this behaviour is observed for all neural-network decoders. The logical error rates of the ML and MWPM decoders at d=3 and d=5 under both noise models are compared in Figure 3. Values below p=0.001 were not investigated because the corresponding logical error rates were extremely low, requiring impractically large numbers of trials to observe sufficient logical failures for reliable statistical analysis. At d=3, the ML decoder achieved lower logical error rates than MWPM across all sampled physical error rates under both noise models. At d=5, the two decoders exhibited comparable logical error rates under depolarising noise, while the ML decoder performed slightly worse under biased Pauli noise. The architecture and training procedure investigated in this study did not achieve performance comparable to MWPM at d=7.

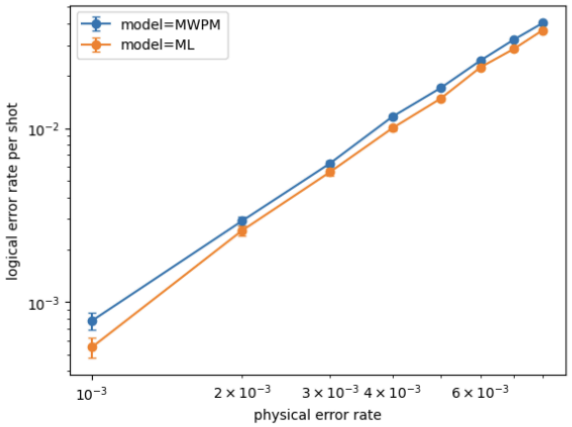
The logical error rate P_L is related to the physical error rate p in the equation $P_L \approx p^{\lfloor \frac{d+1}{2} \rfloor}$ (22). Hence, when P_L is plotted against p on logarithmic axes, the gradient m is expected to be approximately $\lfloor \frac{d+1}{2} \rfloor$. The gradients were calculated from the depolarising-noise results using NumPy (23) as a validation of the observed scaling behaviour. For d=3 and d=5, the theoretical gradients are 2.0 and 3.0, respectively. Experimentally, the fitted gradients were m=2.01±0.03 for d=3 and m=2.83±0.02 for d=5. Both values are in close agreement with the

theoretical predictions, indicating that the ML decoder's behaviour is consistent with the expected scaling behaviour of the surface code in the below-threshold regime.

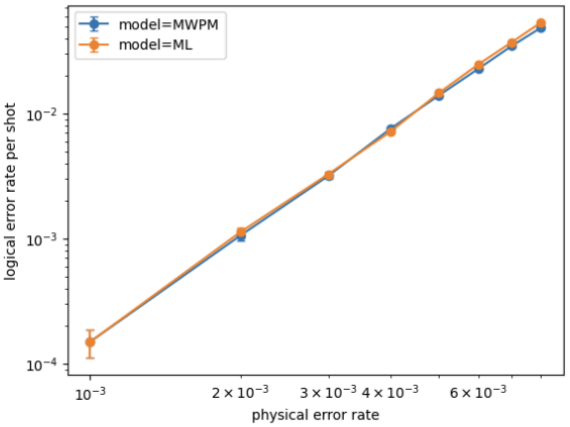
The confusion matrices for both decoders at p=0.006 under the depolarising noise model are shown in Figure 4. At d=3, shots with a ground-truth logical error were substantially outnumbered by those without a logical error, resulting in a strong class imbalance. This imbalance was moderate at d=5, where logical failures occurred more frequently. Though increasing code distance suppresses logical errors exponentially in the below-threshold regime (15), p=0.006 is close to the estimated threshold, resulting in a relatively higher logical error rate at d=5, potentially compounded by the greater number of possible error paths permitted by larger surface codes.

The classification metrics for both decoders at p=0.006 are presented in Table 1. At d=3, the ML decoder had lower recall (0.879±0.003) and higher specificity (0.9919±0.0003) than MWPM (0.891±0.003 and 0.9884±0.0004, respectively), indicating that the ML decoder was more conservative in predicting logical errors. It also achieved higher precision (0.938±0.002) and F1-score (0.9072±0.0009) than MWPM (0.913±0.003 and 0.9022±0.0009, respectively), indicating that its positive predictions were more reliable. MWPM achieved higher balanced accuracy (0.9398±0.0008 versus 0.9353±0.0008), whereas the ML decoder achieved a higher MCC (0.8954±0.0007 versus 0.8890±0.0007). Thus, the two decoders exhibited different strengths at d=3: MWPM performed slightly better when the two classes were weighted equally, while the ML decoder achieved a higher overall correlation with ground truth. At d=5, however, MWPM outperformed the ML decoder across all evaluated classification metrics, despite their comparable logical error rates. Since hyperparameter tuning was impractical given the available computational resources, it was necessary to use other methodologies, including ensembling and postselection, to improve the accuracy of the ML model while preserving its architecture.

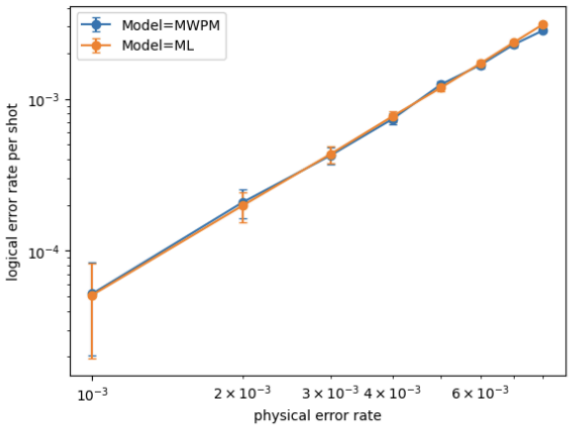
The MWPM and ML decoders are compared at d=3 and d=5 using FNR, FPR, and MCC in Figures 5–7, respectively. At both distances, the FNR of the two decoders was comparable for $p \leq 0.004$, while MWPM achieved lower FNR at higher physical error rates. At d=3, the ML decoder consistently achieved lower FPR than MWPM, whereas the two decoders had comparable FPR at d=5. The ML decoder also achieved higher MCC than MWPM across all sampled physical error rates at d=3. At d=5, the two decoders had comparable MCC for $p \leq 0.004$, while MWPM achieved higher MCC at larger physical error rates.



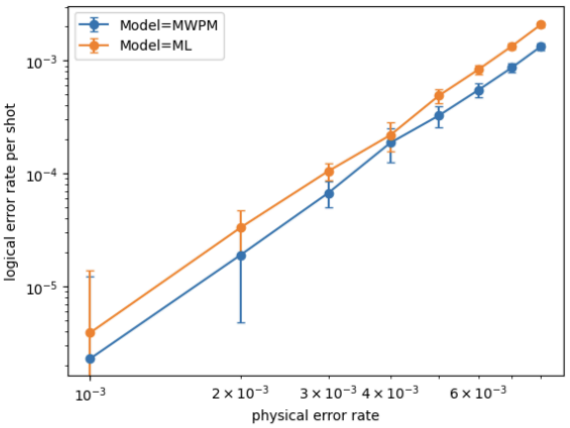
(a) MWPM vs ML decoder performance (d=3, depolarising noise)



(b) MWPM vs ML decoder performance (d=5, biased Pauli noise)

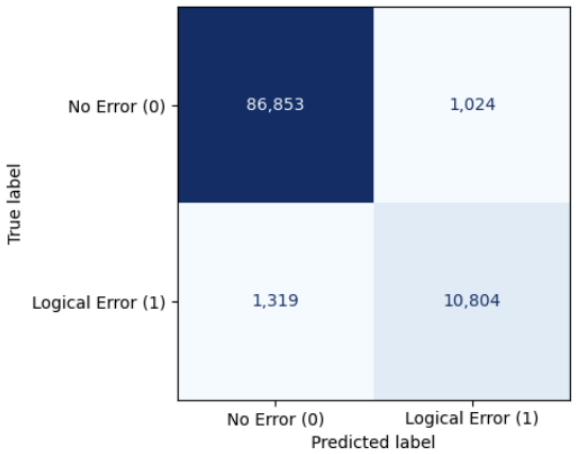


(c) MWPM vs ML decoder performance (d=3, biased Pauli noise)

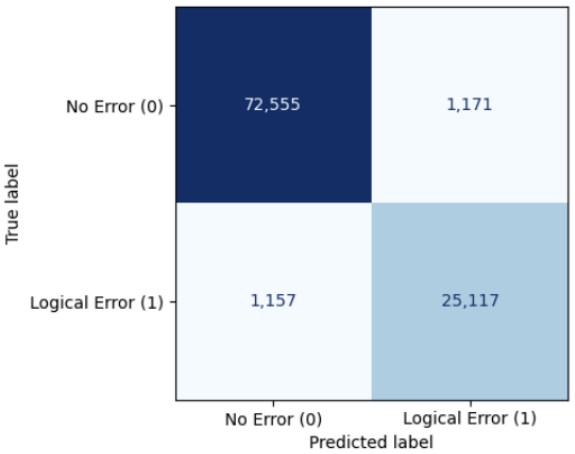


(d) MWPM vs ML decoder performance (d=5, biased Pauli noise)

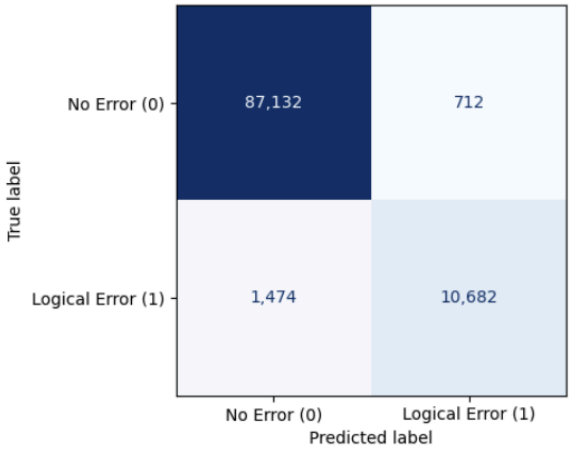
Figure 3. Comparison of ML and MWPM decoder performance based on logical error rate. Logical error rate was plotted against physical error rates from p=0.001 to p=0.008 for the MWPM (blue) and ML (orange) decoders on logarithmic axes under symmetric noise at distances (a) d=3 and (b) d=5 and under biased Pauli noise at (c) d=3 and (d) d=5. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.



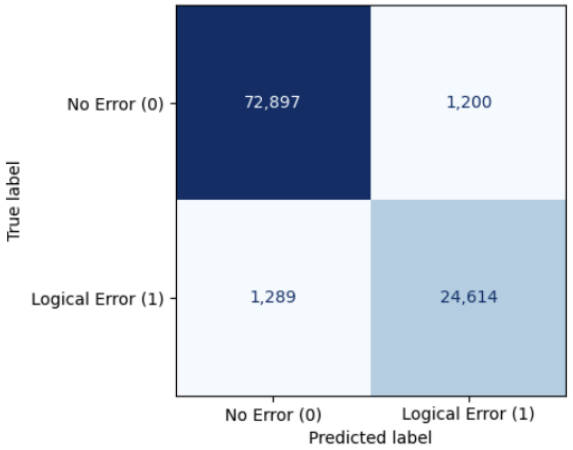
(a) Confusion matrix for the MWPM decoder (d=3)



(b) Confusion matrix for the MWPM decoder (d=5)



(c) Confusion matrix for the ML decoder (d=3)



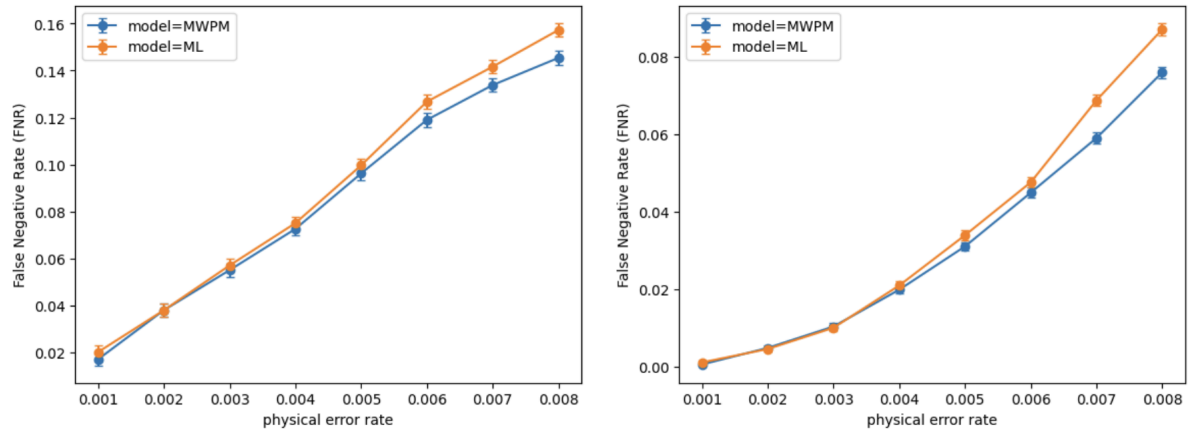
(d) Confusion matrix for the ML decoder (d=5)

Figure 4. Confusion matrices for the ML and MWPM decoders. Each matrix shows the number of true positives (bottom right), true negatives (top left), false positives (top right), and false negatives (bottom left) obtained from the evaluated shots at p=0.006 for MWPM at distances (a) d=3 and (b) d=5 and for the ML model at (c) d=3 and (d) d=5.

Table 1. Classification performance metrics of the ML and MWPM decoders at p=0.006 for distances d=3 and d=5.

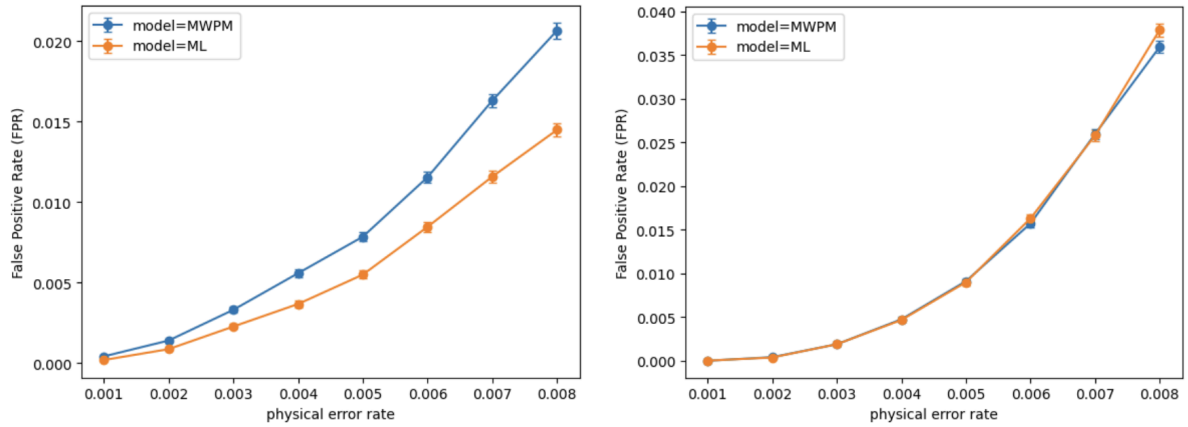
Metric	Distance d=3		Distance d=5	
	MWPM decoder	ML decoder	MWPM decoder	ML decoder
Recall	0.891 ± 0.003	0.879 ± 0.003	0.956 ± 0.001	0.950 ± 0.001
FNR	0.108 ± 0.003	0.121 ± 0.003	0.044 ± 0.001	0.050 ± 0.001
Specificity	0.9884 ± 0.0004	0.9919 ± 0.0003	0.9841 ± 0.0005	0.9838 ± 0.0005
FPR	0.0117 ± 0.0004	0.0081 ± 0.0003	0.0159 ± 0.0005	0.0162 ± 0.0005
Precision	0.913 ± 0.003	0.938 ± 0.002	0.955 ± 0.001	0.954 ± 0.001

Metric	Distance d=3		Distance d=5	
	MWPM decoder	ML decoder	MWPM decoder	ML decoder
F1-score	0.9022 ± 0.0009	0.9072 ± 0.0009	0.9557 ± 0.0007	0.9519 ± 0.0007
Balanced accuracy	0.9398 ± 0.0008	0.9353 ± 0.0008	0.9700 ± 0.0005	0.9670 ± 0.0006
MCC	0.8890 ± 0.0007	0.8954 ± 0.0007	0.9399 ± 0.0005	0.9351 ± 0.0006
Overall accuracy	0.9766 ± 0.0005	0.9781 ± 0.0005	0.9767 ± 0.0005	0.9751 ± 0.0005



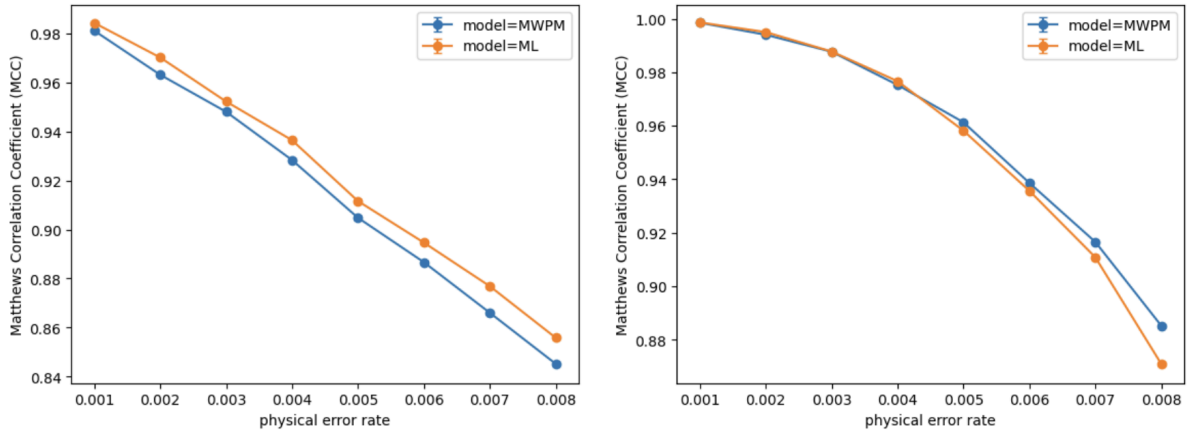
(a) MWPM vs ML decoder performance (d=3) (b) MWPM vs ML decoder performance (d=5)

Figure 5. Comparison of ML and MWPM decoder performance based on false negative rate (FNR). FNR was plotted against physical error rates from p=0.001 to p=0.008 for the MWPM (blue) and ML (orange) decoders on logarithmic axes at distances (a) d=3 and (b) d=5. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.



(a) MWPM vs ML decoder performance (d=3) (b) MWPM vs ML decoder performance (d=5)

Figure 6. Comparison of ML and MWPM decoder performance based on false positive rate (FPR). FPR was plotted against physical error rates from p=0.001 to p=0.008 for the MWPM (blue) and ML (orange) decoders on logarithmic axes at distances (a) d=3 and (b) d=5. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.

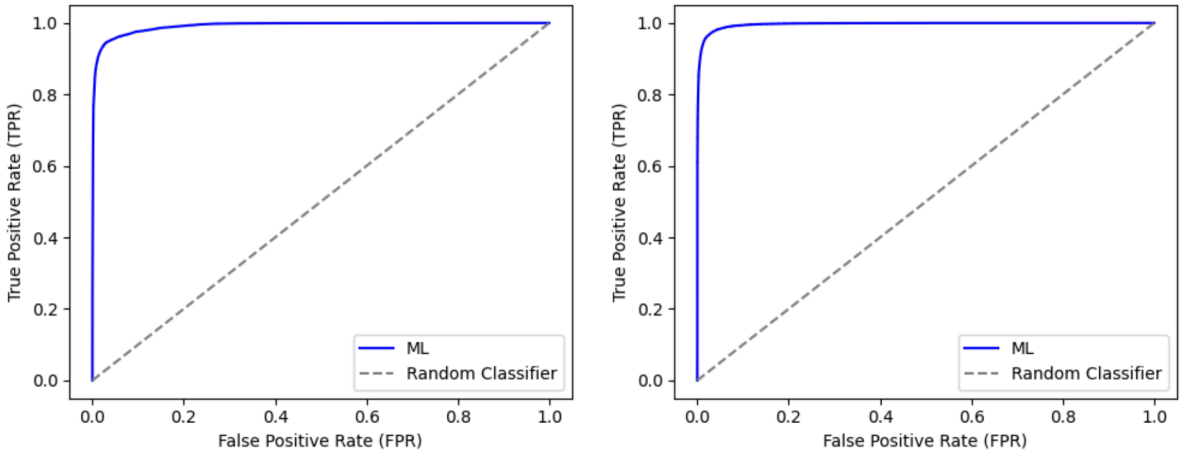


(a) MWPM vs ML decoder performance (d=3) (b) MWPM vs ML decoder performance (d=5)

Figure 7. Comparison of ML and MWPM decoder performance based on the Matthews correlation coefficient. The MCC was plotted against physical error rates from p=0.001 to p=0.008 for the MWPM (blue) and ML (orange) decoders on logarithmic axes at distances (a) d=3 and (b) d=5. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.

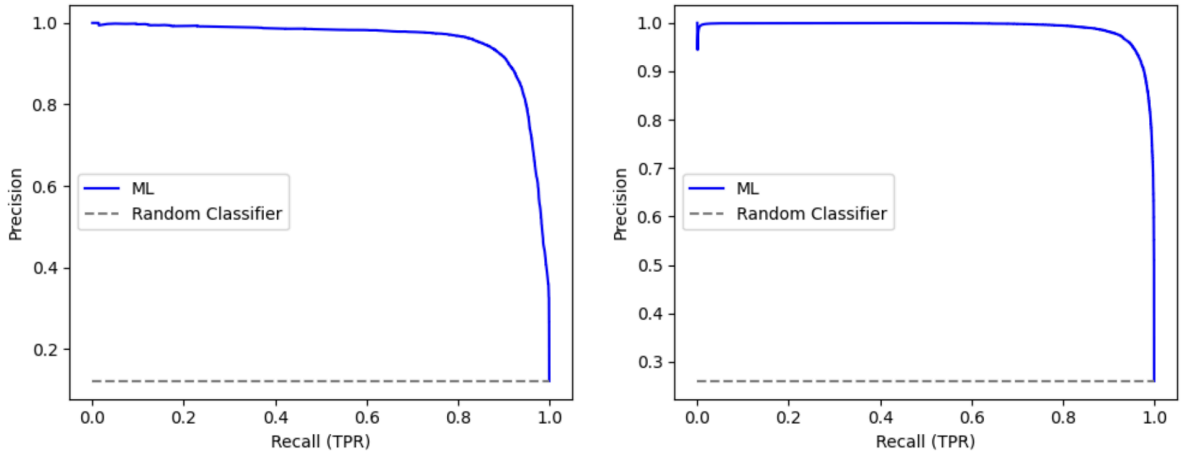
5.3 *Receiver Operating Characteristics and Precision-Recall Analysis*

The ROC and PR curves for the ML decoder at both distances are shown in Figures 8 and 9, respectively. The areas under the ROC curves (AUROC) were 0.9916±0.0006 and 0.9963±0.0003 at d=3 and d=5, respectively. Both values were close to 1, indicating that the model could effectively distinguish between the two classes across the range of classification thresholds (16). The areas under the PR curves (AUPRC) were 0.957±0.001 and 0.9905±0.0004 at d=3 and d=5, respectively. These values also indicate strong performance in identifying the minority class while maintaining high precision. At both distances, the AUROC was higher than the AUPRC, consistent with the tendency of PR analysis to provide a more conservative assessment of classifier performance in imbalanced datasets (17).



(a) ROC curve for the ML decoder (d=3) (b) ROC curve for the ML decoder (d=5)

Figure 8. Receiver operating characteristic (ROC) curves for the ML decoder at distances d=3 and d=5. The classification threshold was varied from 0.0 to 1.0 and the corresponding TPR was plotted against the FPR for the ML decoder as a blue line. The line y=x, representing a random classifier, is shown as a grey dashed line.



(a) PR curve for the ML decoder (d=3) (b) PR curve for the ML decoder (d=5)

Figure 9. Precision-recall (PR) curves for the ML decoder at distances d=3 and d=5. The classification threshold was varied from 0.0 to 1.0 and the corresponding precision was plotted against the recall for the ML decoder as a blue line. The dashed grey line represents a random classifier.

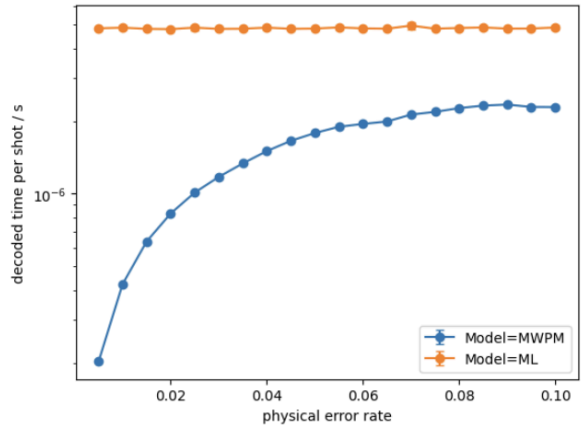
5.4 *Latency, Throughput, and Memory*

The inference time and throughput of the MWPM and ML decoders at d=3 and d=5 across physical error rates are shown in Figures 10 and 11, respectively, for the Apple M1 and Intel Core i5-1135G7 processors. Although the absolute values of the inference time and throughput varied across the two processors, the trends observed remained identical across all experiments. For MWPM, inference time increased with physical error rate, while throughput decreased. In contrast, both quantities remained approximately constant for the ML decoder on both processors. This behaviour is expected because inference for the feedforward neural network consists of a fixed sequence of matrix operations, such that its computational

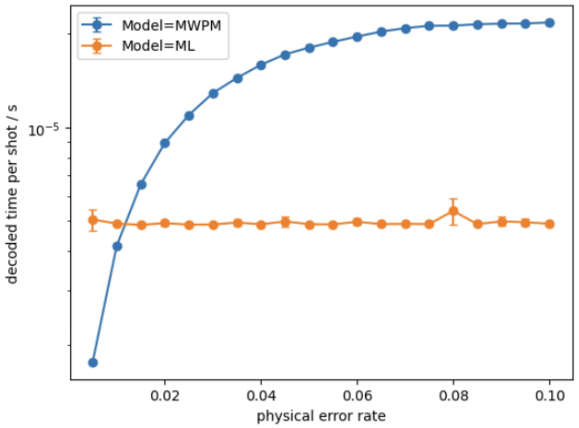
complexity depends primarily on the network architecture rather than the syndrome. Throughput and inference time show the opposite trends as they are inversely proportional. The approximately constant latency and throughput indicate that the ML decoder provides more predictable computational performance as the physical error rate increases.

The inference time and throughput of the two decoders across code distances at p=0.006 are shown in Figures 12 and 13, respectively, for both processors. For MWPM, inference time increased and throughput decreased with increasing code distance on both processors, whereas both remained approximately constant for the ML decoder. For MWPM, the increase in inference time is consistent with the increasing complexity of the syndrome-matching graphs at larger code distances. For the ML decoder, inference consists of a fixed sequence of matrix operations determined primarily by the network architecture, resulting in comparatively little variation with distance. These results suggest that lightweight neural-network decoders may have favourable latency scaling as code distance increases. However, the architecture did not achieve satisfactory decoding performance at d≥7. Consequently, a different architecture may therefore be required at higher distances, and its inference latency may differ substantially from that measured here.

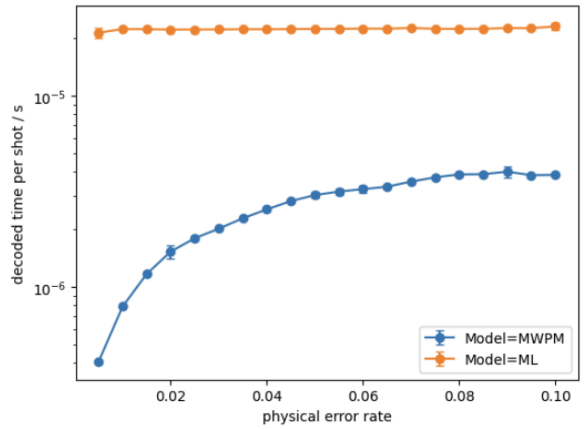
Analysis of model size showed that the ML decoder contained 925,953 trainable parameters at d=3 and 950,529 at d=5, corresponding to 3.5322 MB and 3.6260 MB of weight storage, respectively. MWPM had no trained model weights and therefore did not incur an equivalent model-storage requirement. Additionally, for 100,000 shots, working RAM consumption was 9.54 MB for ML compared with 2.38 MB for MWPM at d=3, and 46.16 MB compared with 11.54 MB at d=5, respectively. Overall, MWPM is significantly more memory-efficient than the ML model.



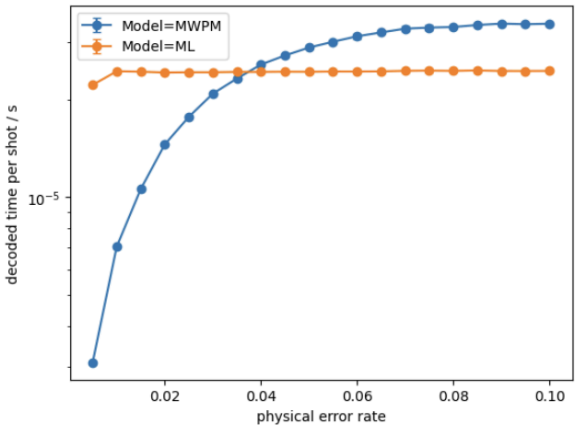
(a) MWPM vs ML decoder inference time (d=3, Apple M1)



(b) MWPM vs ML decoder inference time (d=5, Apple M1)

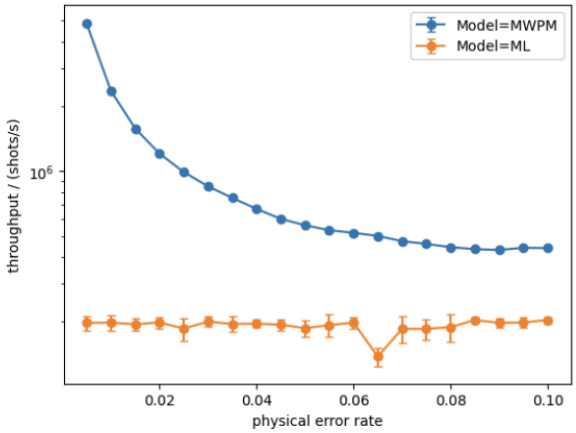


(c) MWPM vs ML decoder inference time (d=3, Intel Core i5-1135G7)

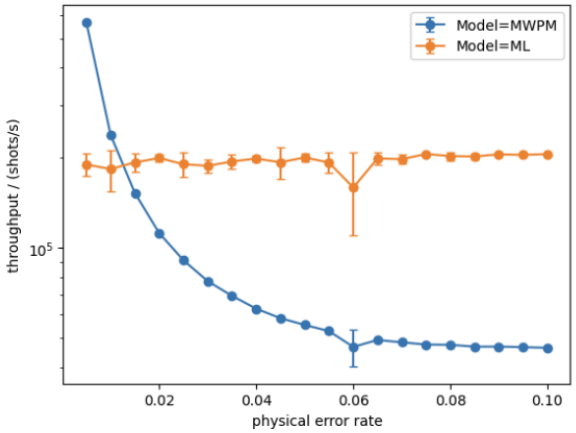


(d) MWPM vs ML decoder inference time (d=5, Intel Core i5-1135G7)

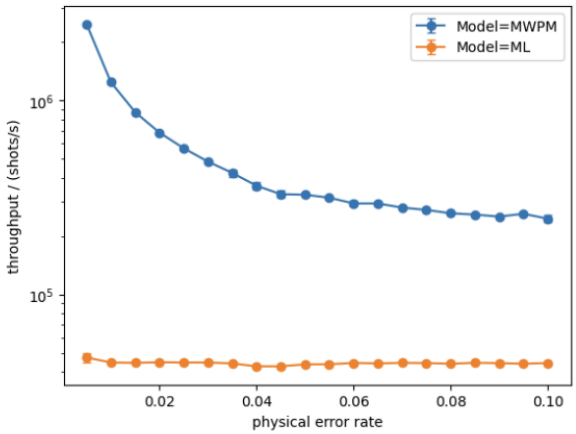
Figure 10. Inference time of the ML and MWPM decoders across physical error rates on two CPU platforms. Inference time per shot was plotted against physical error rate from p=0.01 to p=0.10 for the MWPM (blue) and ML (orange) decoders at (a) d=3 and (b) d=5 on an Apple M1 processor, and (c) d=3 and (d) d=5 on an Intel Core i5-1135G7 processor. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.



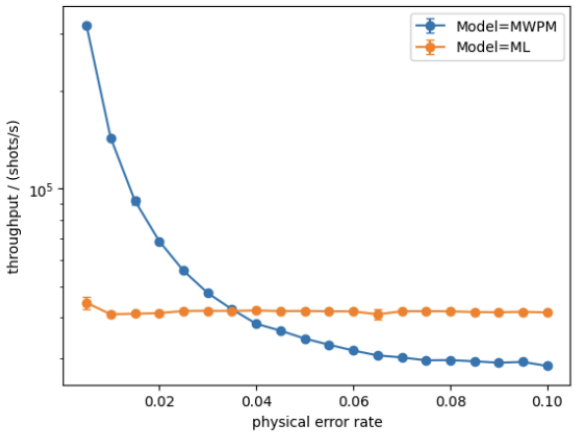
(a) MWPM vs ML decoder throughput (d=3, Apple M1)



(b) MWPM vs ML decoder throughput (d=5, Apple M1)

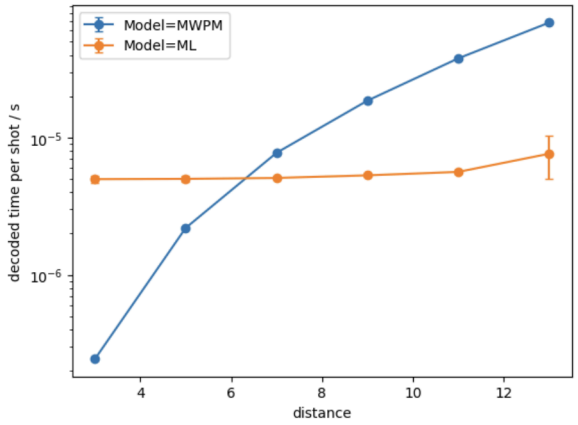


(c) MWPM vs ML decoder throughput (d=3, Intel Core i5-1135G7)

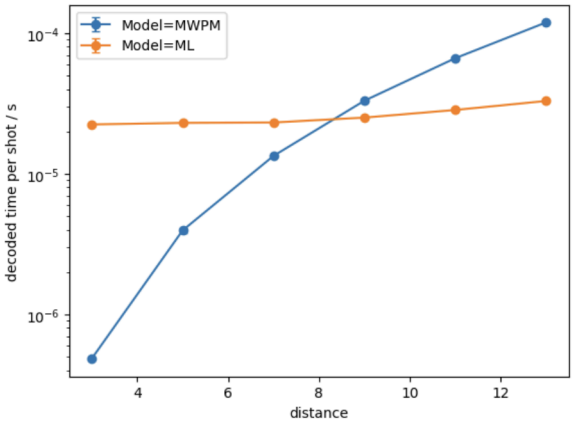


(d) MWPM vs ML decoder throughput (d=5, Intel Core i5-1135G7)

Figure 11. Throughput of the ML and MWPM decoders across physical error rates on two CPU platforms. Throughput, measured in shots per second, was plotted against physical error rate from p=0.01 to p=0.10 for the MWPM (blue) and ML (orange) decoders at (a) d=3 and (b) d=5 on an Apple M1 processor, and (c) d=3 and (d) d=5 on an Intel Core i5-1135G7 processor. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.

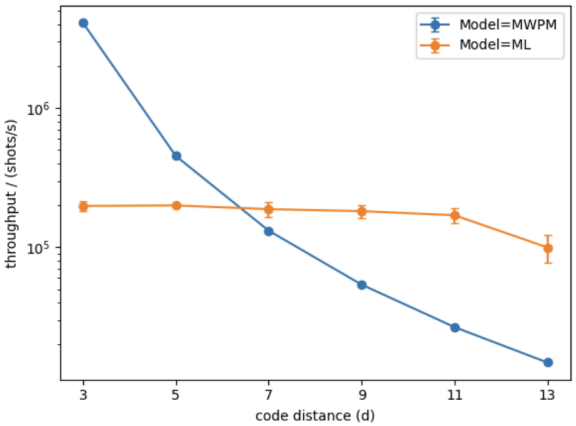


(a) MWPM vs ML decoder inference time (Apple M1)

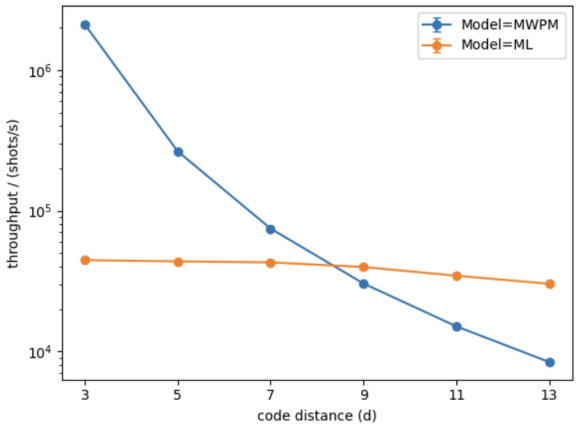


(b) MWPM vs ML decoder inference time (Intel Core i5-1135G7)

Figure 12. Inference time of the ML and MWPM decoders across code distances on two CPU platforms. Inference time per shot was plotted against code distance at odd values from d=3 to d=13 for the MWPM (blue) and ML (orange) decoders at p=0.006 on (a) an Apple M1 processor and (b) an Intel Core i5-1135G7 processor. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.



(a) MWPM vs ML decoder throughput (Apple M1)



(b) MWPM vs ML decoder throughput (Intel Core i5-1135G7)

Figure 13. Throughput of the ML and MWPM decoders across code distances on two CPU platforms. Throughput, measured in shots per second, was plotted against code distance at odd values from d=3 to d=13 for the MWPM (blue) and ML (orange) decoders at p=0.006 on (a) an Apple M1 processor and (b) an Intel Core i5-1135G7 processor. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.

5.5 *Ensembling*

The results of plotting accuracy against the number of models in the ensemble at d=5 and p=0.006 are shown in Figure 14. The accuracy of MWPM at this distance and physical error rate was estimated to be 0.9767±0.0005. The ensemble exceeded the accuracy of MWPM when the number of models was three or greater. Accuracy increased with the number of models before reaching a plateau. This saturation occurred at around ten models, at which point the accuracy was 0.9789±0.0002.

Ensembling increases computational costs, as both the memory footprint required to store model weights and the working RAM consumption scale linearly with the number of models. Furthermore, the latency of the ensemble can remain comparable to that of a single model only if sufficient computational resources are available to execute the models in parallel, increasing hardware requirements. Hence, a smaller ensemble may provide a more favourable trade-off between accuracy and computational cost while still exceeding the accuracy of MWPM.

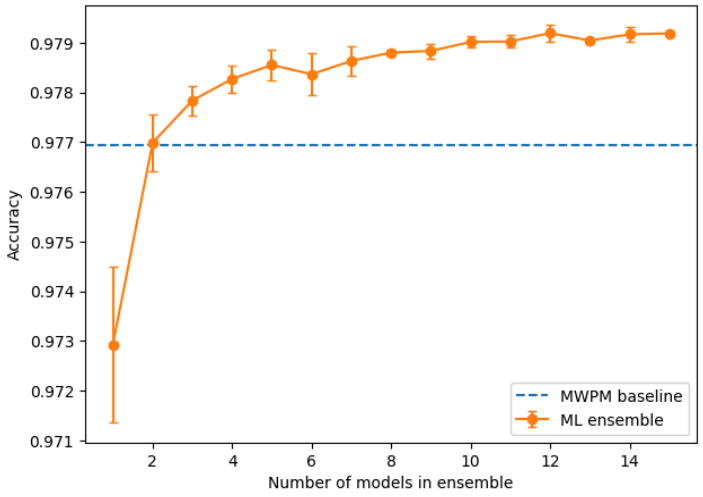


Figure 14. Accuracy of the ML decoder ensemble as a function of ensemble size. The number of models in the ensemble was varied from n=1 to n=15, and corresponding accuracy was plotted as an orange curve. The MWPM baseline accuracy is shown as a blue dashed line. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.

5.6 *Postselection*

The results of plotting accuracy against the reject ratio (r) for an ML decoder with postselection at d=5 and p=0.006 are shown in Figure 15. Although accuracy increases as more uncertain predictions are rejected, the magnitude of this improvement decreases as more data are rejected. A reject ratio of r=0.01 yields an accuracy of 0.9793±0.0005, exceeding the MWPM baseline of 0.9767±0.0005.

While rejecting more low-confidence samples generally improves the reliability of the retained predictions, it necessitates discarding a fraction of decoding outcomes. In real-time quantum computing, rejecting shots may require the computation to be aborted (24), reducing throughput and requiring additional data to obtain a usable outcome. Consequently, postselection may offset the computational advantages of the decoder for long quantum circuits. Thus, the reject ratio should be minimised while retaining the desired improvement in decoding accuracy.

The calibration curve for the ML decoder at d=5 and p=0.006 is shown in Figure 16. It lies close to the line y=x, and the expected calibration error, maximum calibration error, and Brier score are 0.00229, 0.03110, and 0.01851, respectively. Together, these results indicate that the

predicted flip probabilities closely correspond to the observed flip frequencies, supporting the use of the decoder's confidence score as the basis for postselection.

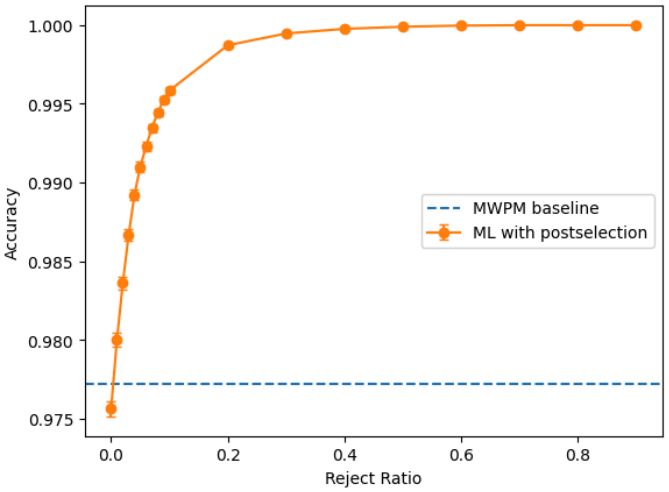


Figure 15. Accuracy of the ML decoder with postselection as a function of reject ratio. The reject ratio was varied at intervals of 0.01 from r=0.01 to r=0.1 and at intervals of 0.1 from r=0.1 to r=0.9, and the corresponding accuracy was plotted as an orange curve. The MWPM baseline accuracy is shown as a blue dashed line. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.

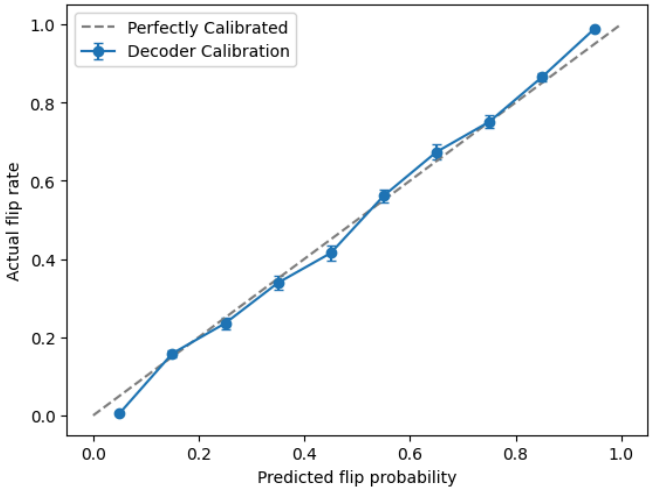


Figure 16. Calibration curve for the ML decoder at d=5 and p=0.006. Predicted flip probabilities are plotted against the corresponding observed flip rates as a blue curve. The line y=x, representing a perfectly calibrated curve, is shown as a grey dashed line. Error bars represent statistical uncertainty; where not visible, they are smaller than the marker size.

6. Conclusion

This study aimed to investigate whether a lightweight ML decoder could provide scalable and low-latency QEC that outperforms the MWPM benchmark at low distances while minimising computational costs and remaining deployable on consumer-grade hardware. A feedforward neural network trained under depolarising noise successfully generalised to physical error rates outside its training regime and to biased Pauli noise. The ML decoder outperformed MWPM at d=3 under both noise models; at d=5, it achieved comparable performance under

depolarising noise but performed slightly worse than MWPM under biased noise. Additional classification metrics and ROC and PR analyses further demonstrated that the model could effectively distinguish logical errors despite the strong class imbalance. Latency analysis revealed that inference time increased and throughput decreased with both distance and physical error rate for MWPM, whereas both quantities remained approximately constant for the ML decoder, indicating more predictable scaling. However, MWPM was more memory-efficient. Ensembling with an ensemble size of 3 and postselection with a reject ratio of 0.01 improved ML accuracy beyond MWPM, though both methods introduce significant tradeoffs for real-time fault-tolerant quantum computing.

Overall, this work highlights the potential of lightweight machine-learning decoders to achieve competitive decoding performance while maintaining favourable latency characteristics. Although a single feedforward neural network did not consistently outperform MWPM at d=5, ensembling and postselection enhanced its accuracy beyond the MWPM baseline. These results demonstrate that relatively simple architectures can learn the relationship between syndrome patterns and logical failures while maintaining predictable inference latency. Additionally, this study provides an initial demonstration that meaningful ML-based QEC research can be conducted using accessible computational resources.

Certain limitations remain. Firstly, the model was evaluated only at the low distances d=3 and d=5, and attempts to extend the architecture to d=7 were unsuccessful. Investigating the possible causes of this deterioration in performance was impractical given the available computational resources. Future work could therefore investigate factors such as network depth, hidden-layer width, and parameter count, as well as alternative lightweight architectures such as residual multilayer perceptrons or convolutional neural networks, to enable decoding at higher distances. Secondly, this work relied on simulated data produced by STIM; real quantum hardware exhibits additional effects such as leakage and crosstalk. Thus, testing lightweight models on hardware-generated syndrome data would provide a more realistic assessment of decoder performance under experimental noise. Finally, all experiments were performed on two consumer-grade CPUs; the reported inference times may differ on GPUs or dedicated accelerators. Future work evaluating lightweight decoders across a wider range of hardware platforms would provide a more comprehensive assessment of their computational performance and deployment requirements.

Acknowledgements

I would like to thank my IB Physics HL teacher, Ms. Meena Rajesh, for her guidance and feedback throughout the process of this project. Her feedback and discussions provided support in the development, review, and presentation of the research. I acknowledge the use of Claude (Anthropic) during the preparation and revision of this research report. Claude was used once in the drafting stage to generate Figure 1 based on my specifications. In the final editing stage, it was used regularly to review grammar, phrasing, sentence structure, and clarity. Following the use of any AI tool, I reviewed, edited, and confirmed the scientific accuracy of all the generated text and figures. All experimental design, data analysis, interpretations, and conclusions are my own. All experiments were conducted on personal

devices using Apple M1 and Intel Core i5-1135G7 processors. This work received no external funding.

References

1. Dennis, E., Kitaev, A., Landahl, A., & Preskill, J. (2002). *Topological quantum memory*. *Journal of Mathematical Physics, 43*(9), 4452–4505. https://doi.org/10.1063/1.1499754

2. Bausch, J., Senior, A. W., Heras, F. J. H., Edlich, T., Davies, A., Newman, M., et al. (2024). *Learning high-accuracy error decoding for quantum processors*. *Nature, 635*(8040), 834–840. https://doi.org/10.1038/s41586-024-08148-8

3. Gidney, C., & Ekerå, M. (2021). *How to factor 2048-bit RSA integers in 8 hours using 20 million noisy qubits*. *Quantum, 5*, 433. https://doi.org/10.22331/q-2021-04-15-433

4. Fowler, A. G., Mariantoni, M., Martinis, J. M., & Cleland, A. N. (2012). *Surface codes: Towards practical large-scale quantum computation*. *Physical Review A, 86*(3), 032324. https://doi.org/10.1103/PhysRevA.86.032324

5. Varsamopoulos, S., Bertels, K., & Almudever, C. G. (2020). *Comparing neural network based decoders for the surface code*. *IEEE Transactions on Computers, 69*(2), 300–311. https://doi.org/10.1109/TC.2019.2948612

6. Maskara, N., Kubica, A., & Jochym-O'Connor, T. (2019). *Advantages of versatile neural-network decoding for topological codes*. *Physical Review A, 99*(5), 052351. https://doi.org/10.1103/PhysRevA.99.052351

7. Tuckett, D. K., Bartlett, S. D., & Flammia, S. T. (2018). *Ultrahigh error threshold for surface codes with biased noise*. *Physical Review Letters, 120*(5), 050505. https://doi.org/10.1103/PhysRevLett.120.050505

8. Gidney, C. (2021). *Stim: A fast stabilizer circuit simulator*. *Quantum, 5*, 497. https://doi.org/10.22331/q-2021-07-06-497

9. Higgott, O. (2022). *PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching*. *ACM Transactions on Quantum Computing, 3*(3), Article 16. https://doi.org/10.1145/3505637

10. Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. *Computing in Science & Engineering, 9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

11. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). *PyTorch: An imperative style, high-performance deep learning library*. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural*

Information Processing Systems (Vol. 32). Curran Associates, Inc. https://doi.org/10.48550/arXiv.1912.01703

12. Elfwing, S., Uchibe, E., & Doya, K. (2017). *Sigmoid-weighted linear units for neural network function approximation in reinforcement learning*. *arXiv*. https://doi.org/10.48550/arXiv.1702.03118

13. Kingma, D. P., & Ba, M. (2015). *Adam: A method for stochastic optimization*. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*. https://doi.org/10.48550/arXiv.1412.6980

14. Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). *The balanced accuracy and its posterior distribution*. In *2010 20th International Conference on Pattern Recognition* (pp. 3121–3124). IEEE. https://doi.org/10.1109/ICPR.2010.764

15. Chicco, D., & Jurman, G. (2020). *The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation*. *BMC Genomics, 21*(1), 6. https://doi.org/10.1186/s12864-019-6413-7

16. Fawcett, T. (2006). *An introduction to ROC analysis*. *Pattern Recognition Letters, 27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

17. Davis, J., & Goadrich, M. (2006). *The relationship between precision-recall and ROC curves*. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 233–240). ACM. https://doi.org/10.1145/1143844.1143874

18. Dietterich, T. G. (2000). *Ensemble methods in machine learning*. In J. Kittler & F. Roli (Eds.), *Multiple classifier systems* (pp. 1–15). Springer. https://doi.org/10.1007/3-540-45014-9_1

19. Aaronson, S. (2005). *Quantum computing, postselection, and probabilistic polynomial-time*. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 461*(2063), 3473–3482. https://doi.org/10.1098/rspa.2005.1546

20. Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). *On calibration of modern neural networks*. In *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 1321–1330). *Proceedings of Machine Learning Research*. https://doi.org/10.48550/arXiv.1706.04599

21. Stephens, A. G. (2014). *Fault-tolerant thresholds for quantum error correction with the surface code*. *Physical Review A, 89*(2), 022321. https://doi.org/10.1103/PhysRevA.89.022321

22. Google Quantum AI and Collaborators. (2025). *Quantum error correction below the surface code threshold*. *Nature, 638*(8052), 920–926. https://doi.org/10.1038/s41586-024-08449-y

23. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., et al. (2020). *Array programming with NumPy*. *Nature, 585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

24. Ryan-Anderson, C., Bohnet, J. G., Lee, K., Gresh, D., Hankin, A., Gaebler, J. P., et al. (2021). *Realization of real-time fault-tolerant quantum error correction*. *Physical Review X, 11*(4), 041058. https://doi.org/10.1103/PhysRevX.11.041058