

Research Proposal

Uncertainty-Aware Neural Decoding for Surface-Code Quantum Error Correction

Andrew Gao, Lucas Trachtenberg

Supervisor: Mr. Mark Hannum

Submitted in partial fulfillment of the research requirements for an Advanced TJHSST Diploma

Contents

1	Introduction	3
1.1	Intellectual Merit	3
1.2	Broader Impact	3
2	Background and Prior Work	4
3	Research Goals	6
	Goal 1: Build a Reproducible Surface-Code Dataset and Decoder Baselines	6
	Goal 2: Isolate the Value of Soft Error Probabilities	6
	Goal 3: Test Robustness Beyond the Training Noise Model	7
4	Proposed Methods and Research Design	7
5	Expected Results and Data Analysis	8
6	Feasibility and Resources	9
7	Safety and Risk Considerations	9
8	Potential Pitfalls and Alternative Strategies	9
9	Research Timeline	10
10	Group Responsibilities	11
	Annotated Bibliography	12

1 Introduction

Quantum computers must detect and correct physical errors without directly measuring the encoded quantum state. Surface codes address this problem by repeatedly measuring local stabilizers on a two-dimensional qubit lattice. Changes in those measurements, called detection events, reveal where an error may have occurred, but they do not identify a unique physical-error pattern. A decoder must instead choose any correction that returns the state to the correct logical equivalence class [1]. Minimum-weight perfect matching (MWPM) is a strong standard decoder, while recent neural decoders show that learned models can exploit spatial, temporal, and device-specific error correlations [2, 3].

This project will test one narrow question: **does preserving a neural model’s uncertainty about physical errors improve surface-code decoding?** A spatiotemporal neural network will map repeated syndrome measurements to per-qubit probabilities for the X and Z components of Pauli errors. A second, syndrome-consistent correction stage will receive those probabilities rather than a binary error map. We will compare this soft-information pipeline with MWPM, a direct neural decoder, and an otherwise matched model whose probabilities are thresholded before correction. The primary outcome will be logical error rate, not exact physical-error classification accuracy.

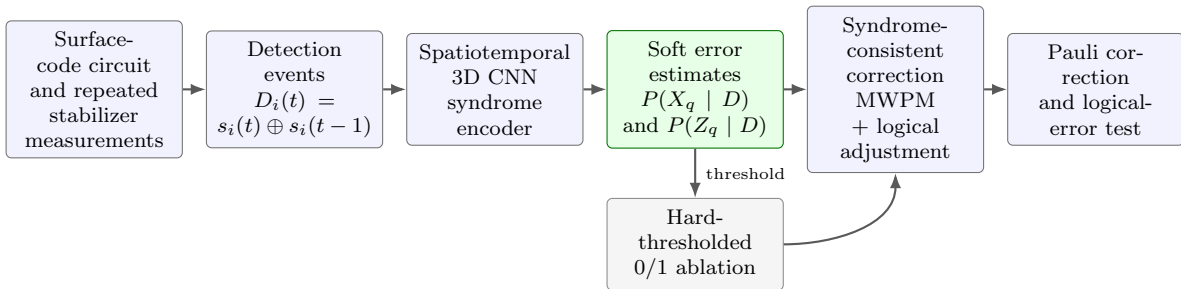


Figure 1: Proposed project pipeline. The same inferred error information is evaluated as continuous probabilities and as thresholded bits; final performance is determined by logical failures.

1.1 Intellectual Merit

The project will isolate the value of uncertainty through controlled ablation rather than evaluating only whether a neural decoder can correct errors. It will distinguish improvements caused by retaining probability information from improvements caused by model size, a learned syndrome representation, or a strong conventional decoding stage. The study will also compare physical-error inference metrics with the logically meaningful outcome, testing whether better per-qubit predictions actually produce fewer logical failures.

1.2 Broader Impact

Reliable decoding is necessary for useful fault-tolerant quantum computation, including future quantum-machine-learning workloads. A reproducible benchmark showing when soft neural information helps, has no effect, or hurts would help researchers avoid judging decoders by misleading physical-error accuracy alone. The data pipeline, ablation protocol, and logical-error analysis could also be reused by future QLab projects studying other codes, noise models, or quantum devices.

2 Background and Prior Work

A surface code encodes logical information nonlocally in a two-dimensional lattice of data qubits and measurement qubits. Local X -type and Z -type stabilizers are measured without revealing the encoded state. An X -type stabilizer detects nearby Z errors, while a Z -type stabilizer detects nearby X errors. If stabilizer i has binary measurement $s_i(t)$ in round t , the corresponding detection event is

$$D_i(t) = s_i(t) \oplus s_i(t-1). \quad (1)$$

Using changes between rounds is important because a data-qubit error and a faulty stabilizer measurement leave different patterns in space and time. For a code of distance d observed over repeated rounds, the decoder input can therefore be represented as a tensor with time, row, column, and stabilizer-type dimensions.

A syndrome does not uniquely specify the physical error. Different Pauli chains can have the same boundary and therefore produce the same detection events. If the physical error is E and the decoder applies correction C , success does not require $C = E$; it requires CE to be a stabilizer rather than a nontrivial logical operator. Logical error rate is consequently the central metric. MWPM operationalizes this idea by pairing detection events along low-weight paths and is an efficient, widely used baseline [1, 4].

Neural decoders attempt to learn structure that a fixed independent-noise model may miss. Torlai and Melko demonstrated a neural decoder for topological codes, and later work has used recurrent, convolutional, and attention-based architectures to model repeated syndrome rounds [2, 5]. AlphaQubit showed that a learned decoder can use experimental syndrome histories and outperform several established baselines on Google surface-code data [3]. Neural models are particularly attractive when errors are correlated, as in crosstalk or device-dependent noise, although the model must generalize across error rates and hardware drift [6].

Syndrome input encoding. Rather than flattening the raw measurement record, we will place each detection event at its stabilizer’s lattice coordinate and its measurement time. Separate maps will store events from X -type and Z -type stabilizers; these name the measured stabilizers, not the error components they detect. Two additional binary masks will indicate where each type has a valid detector. A zero at a valid detector means “no event,” whereas a zero with mask value zero means a nonexistent or unavailable detector. This distinction preserves boundary geometry and avoids interpreting missing entries as reliable measurements. Initial and final events will follow the circuit’s detector definitions, including known preparation parities and comparisons with final data-qubit readout, rather than assuming every missing reference measurement is zero [7].

The resulting batch will have the channels-first layout

$$\mathcal{D} \in \{0, 1\}^{B \times 4 \times T \times H \times W}, \quad \text{channels} = (D^X, D^Z, M^X, M^Z), \quad (2)$$

where B is the number of independent circuit trials, T is the number of retained detector-time slices, and H and W span the stabilizer grid, including its boundaries. Thus T need not equal the number of syndrome rounds when terminal events add another slice. The superscripts identify stabilizer type, and M denotes a binary validity mask. The network will receive these binary values as floating-point inputs, with consistent coordinates and a fixed window length within each comparison. Physical-error labels and logical outcomes are training targets only; neither will be included in the input.

How the CNN processes the input. Our initial encoder will use three 3D convolutional layers with 32, 64, and 64 output feature channels. Each layer will apply a learned $3 \times 3 \times 3$ kernel over time, row, and column, followed by a rectified linear unit (ReLU). Filters mix the incoming

channels and reuse the same learned weights at every location. This gives the model a way to recognize nearby event pairs within a round and patterns recurring at one location across rounds. Stacking layers expands the region of the syndrome history available to each feature. Stride one and one-cell zero padding will preserve the $T \times H \times W$ dimensions; the planar code will not use wraparound padding [8]. We will avoid early spatial pooling so that small event patterns and boundary locations remain distinguishable.

From encoded features to error probabilities. After the convolutional layers, averaging over time will produce a $B \times 64 \times H \times W$ feature map. Flattening the spatial map and applying a dense layer with 256 ReLU units will yield a shared syndrome embedding $\mathbf{h}$. Keeping spatial positions distinct until this dense layer allows the readout to combine evidence from across the lattice. Two separate dense sigmoid heads will then output

$$\mathbf{p}^X = \sigma(A_X \mathbf{h} + \mathbf{b}_X), \quad \mathbf{p}^Z = \sigma(A_Z \mathbf{h} + \mathbf{b}_Z), \quad (3)$$

with one entry per data qubit in a fixed coordinate ordering. For the rotated code, each head therefore has d^2 outputs per trial. The sigmoid maps each score to $[0, 1]$; both outputs can be high for the same qubit, representing the X and Z components of a Y error. The correction stage will receive both probability vectors and $\mathbf{h}$. These layer widths and the temporal averaging are proposed starting choices, not established optimal settings; model selection will use validation data, and separate readouts will initially be trained for each code distance.

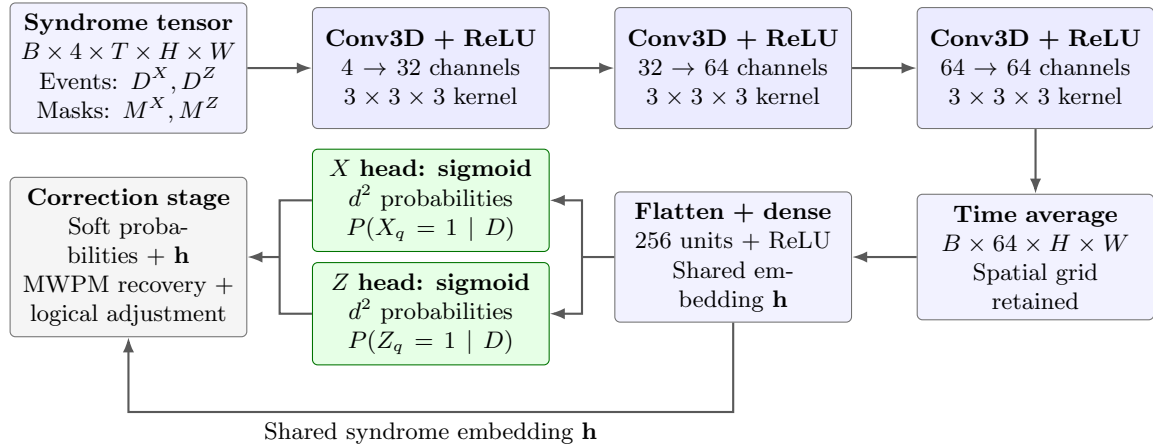


Figure 2: Proposed CNN encoder and readout. Convolutions preserve the time and spatial dimensions while learning feature channels; the shared embedding feeds both per-qubit probability heads and the correction stage. The correction stage also uses the original syndrome for its MWPM reference recovery.

Retaining these marginal probabilities matters because thresholding maps, for example, 0.51 and 0.99 to the same binary value. Work on soft-information decoding has similarly shown that confidence information can improve quantum error correction when incorporated into decoder weights [9]. The probabilities are not treated as exact recovered errors; they are uncertain features used by a correction stage whose success is judged logically.

Project Need / Research Gap. Prior work establishes that neural decoders and soft information can both be useful, but a focused comparison is still needed for the architecture proposed here: infer interpretable per-qubit X/Z probabilities, preserve them through correction, and directly ablate them by replacing the probabilities with thresholded bits. Holding the syndrome data, model

capacity, training budget, and final correction rule constant will show whether uncertainty itself improves logical error rate. Comparisons with MWPM and a direct syndrome-to-logical-class neural decoder will determine whether the intermediate error model adds useful information or only complexity.

3 Research Goals

Research goals describe what will be determined, not what the authors hope the neural model will do. All decoder comparisons will therefore report improvements, no meaningful differences, and degradations with equal emphasis.

Goal 1— Build a Reproducible Surface-Code Dataset and Decoder Baselines

Question / Objective: Establish syndrome datasets and conventional baselines that make later neural comparisons valid and reproducible.

Approach: Simulate rotated surface-code memory experiments at distances $d = 3, 5$, and 7 across a pre-specified range of physical error rates and repeated measurement rounds. Store raw stabilizer results, detection events, available physical X/Z labels, and logical observables. Decode the same held-out samples with MWPM and a direct syndrome-to-logical-class neural baseline.

Measurement / Output: Versioned training, validation, and test partitions; documented noise models; logical error curves for MWPM; and reference training behavior for the direct neural decoder.

Success Criterion: The goal is complete when every sample can be regenerated from recorded seeds and configurations, train/test leakage checks pass, and each retained distance–noise condition has enough independent test trials to estimate logical error with a reported confidence interval.

Goal 2— Isolate the Value of Soft Error Probabilities

Question / Objective: Determine whether preserving per-qubit uncertainty improves final surface-code correction compared with thresholding the same inferred errors.

Approach: Train a shared spatiotemporal syndrome encoder with separate X - and Z -error probability heads. Feed its continuous outputs and syndrome features to a correction stage, then run an otherwise identical hard-input ablation in which the probabilities are thresholded. Match data splits, encoder weights or capacity, optimizer settings, stopping rules, and random seeds.

Measurement / Output: Per-qubit discrimination and calibration metrics, logical error rate, logical error per round, decoder latency, convergence curves, and paired soft-versus-hard results for each code distance and physical error rate.

Success Criterion: The goal is complete when at least five independent training runs per retained architecture have been evaluated on identical held-out syndromes and the direction, magnitude, and uncertainty of the logical-error difference are reported. A null result or worse soft-input performance satisfies the goal.

Goal 3— Test Robustness Beyond the Training Noise Model

Question / Objective: Determine whether the proposed decoder remains useful when noise strength, correlations, or data source differ from those used in training.

Approach: Evaluate the frozen decoders at neighboring physical error rates and on at least one correlated-noise condition. If compatible data are available, also test on a public experimental surface-code dataset or on small-distance circuits run on an accessible quantum processor. Compare zero-shot performance with limited recalibration or fine-tuning.

Measurement / Output: Generalization curves, calibration shift, logical error rate before and after limited adaptation, and a record of failure modes caused by simulation-to-hardware or independent-to-correlated-noise shift.

Success Criterion: Produce a reproducible out-of-distribution comparison against MWPM and the direct neural baseline. This goal does not require the proposed decoder to outperform its controls; identifying where it ceases to generalize is a successful result.

4 Proposed Methods and Research Design

The study will first generate rotated surface-code memory data with Stim, a high-speed stabilizer-circuit simulator [10]. Initial experiments will use code distances $d = 3, 5$, and 7 , approximately d syndrome rounds, and a circuit-level depolarizing noise model over several physical error probabilities below and near the useful decoding regime. Independent random seeds will define training, validation, and test partitions. Each sample will include stabilizer measurements, detection events, logical observables, and physical X/Z error labels where the simulation pipeline can expose them. A public experimental dataset from Google Quantum AI will provide an optional hardware-data test [11, 12].

The conventional baseline will be MWPM implemented with PyMatching [4]. A second baseline will use the same syndrome encoder as the proposed model but predict the logical correction class directly. As detailed in Fig. 2, the proposed encoder will receive a $B \times 4 \times T \times H \times W$ tensor containing two detection-event channels and two detector-validity masks. Three $3 \times 3 \times 3$ convolutional layers with 32, 64, and 64 feature channels and ReLU activations will preserve the time and spatial dimensions. Temporal averaging, spatial flattening, and a 256-unit dense layer will produce the shared syndrome embedding. Separate sigmoid heads will estimate $P(X_q = 1 \mid D)$ and $P(Z_q = 1 \mid D)$ for each of the d^2 data qubits. These are marginal probabilities, not claims that the exact physical error has been reconstructed. A separate model will initially be trained for each distance, with the same encoder configuration used in the soft-input, hard-input, and direct-neural comparisons.

The correction stage will receive the syndrome embedding and the two probability vectors. Its primary implementation will predict the logical adjustment class I , X_L , Z_L , or Y_L to be combined with a syndrome-consistent reference recovery from MWPM. Because multiplying a recovery by a logical operator does not change its syndrome, this design prevents the neural network from proposing a correction inconsistent with the observed detection events. A secondary implementation may use learned probabilities to reweight matching-graph edges. The central ablation will replace every probability entering the correction stage with a thresholded bit while holding the encoder, correction-stage capacity, and training data fixed.

The error-inference heads will use weighted binary cross-entropy, and the logical correction head will use four-class cross-entropy. Loss weights will be selected using validation data only.

At least five initializations will be trained for each retained architecture. Model size, batch size, learning-rate schedule, early-stopping rule, and maximum optimization budget will be recorded; hyperparameter selection will be performed without examining the final test set.

The primary metric will be logical error rate $P_L = N_{\text{logical failures}}/N_{\text{trials}}$. Secondary metrics will include logical error per round, Brier score and calibration error for the probability heads, area under the precision–recall curve, decoder latency, throughput, parameter count, and training time. Correlated-noise tests will be included because learned decoders may be most useful when fixed independent-noise assumptions fail [6].

Methods may change. If per-qubit labels cannot be extracted reliably for a selected circuit-level dataset, the probability heads will first be trained on labeled phenomenological-noise data and then evaluated as frozen features or fine-tuned through the logical-class objective. If hardware lacks repeated mid-circuit measurement and reset, hardware validation will be replaced by the public experimental dataset. Every change and its effect on comparability will be documented.

5 Expected Results and Data Analysis

MWPM is expected to be difficult to beat under the same simple independent-noise model used to assign its graph weights. The proposed neural pipeline may be more useful under correlated noise or hardware data, where a learned spatiotemporal representation can capture structure absent from a fixed decoder. Recent neural-decoder results support this possibility but do not guarantee it for smaller data and models [3, 5].

For the soft-versus-hard comparison, no direction of effect is assumed. One possible outcome is that continuous probabilities lower logical error because the correction stage can distinguish weak evidence from high-confidence errors, consistent with the value of soft information in other decoding settings [9]. A second outcome is no meaningful difference because the syndrome embedding already contains the same information. A third is worse soft-input performance caused by poor calibration, overfitting, or reliance on confidence values that shift under a new noise model.

Logical error will be plotted against physical error probability separately for each code distance. Binomial confidence intervals will be reported for every test point. Because all decoders will evaluate the same held-out syndromes, paired bootstrap intervals and paired success/failure tests will be used for decoder differences. Training runs will be summarized using the mean, median, standard deviation, and full range where useful; conclusions will emphasize effect sizes relative to seed-to-seed and sampling variation rather than the single best run.

Physical-error inference will be analyzed separately from decoding success. Precision–recall curves will handle the large class imbalance between error and no-error qubits, while Brier score and reliability diagrams will measure probability calibration. A model may classify physical errors well but still choose the wrong logical class, or it may infer a physically different yet logically equivalent correction. Those cases will not be collapsed into one accuracy score.

The most informative result may be a tradeoff. For example, the neural decoder could lower logical error while increasing latency, improve at one code distance but fail to generalize to another, or benefit correlated noise while matching MWPM on independent noise. A negative or null result remains valid if the comparison is reproducible. Failure of every decoder or suspiciously strong results caused by duplicated circuits would instead indicate a data or evaluation problem that must be repaired before drawing conclusions.

6 Feasibility and Resources

The project requires a reliable syndrome dataset, classical computing for simulation and preprocessing, and GPU access for practical neural-network training. It does *not* strictly require a physical quantum computer: Stim can generate large labeled stabilizer-circuit datasets efficiently on a classical computer [10]. Quantum-computer access remains useful for a small external validation set, provided the device supports the repeated mid-circuit measurement and reset needed by a surface-code memory experiment. Otherwise, the public Google Quantum AI surface-code data can serve that role [11].

The main constraints are dataset quality and rare logical failures. Training, validation, and test circuits must be independently generated, noise settings must be recorded, and the test set must contain enough failures for meaningful confidence intervals. At very low error rates, this may require many simulated trials or targeted importance sampling. Code distances will begin at $d = 3$ and expand to $d = 5$ and 7 only after storage and training-time measurements confirm feasibility. Standard Python machine-learning tools, Stim, and PyMatching are sufficient; no custom laboratory apparatus is required. Configurations, random seeds, model checkpoints, and logs will be version-controlled.

7 Safety and Risk Considerations

This project does not involve lasers, high voltage, vacuum systems, chemicals, or other significant laboratory hazards. The primary risks are computational and operational rather than physical. Long GPU jobs will be executed only on approved systems with normal thermal management and resource limits; students will not modify workstation cooling, power hardware, or system-level GPU settings.

Quantum processors will be accessed remotely through approved institutional or cloud interfaces, so the students will not handle cryogenic, microwave, or high-voltage hardware. Software packages and external code will be obtained from reputable sources and used in approved project environments. The simulated and public hardware datasets contain device measurements rather than personal information. Training outputs, configuration files, and source code will be stored in the approved research workspace with version control and regular backups.

8 Potential Pitfalls and Alternative Strategies

Challenge 1

At low physical error rates, most samples contain no data-qubit error or no logical failure. A model could achieve high apparent accuracy by predicting the majority class, while a test set with too few failures would give an unstable logical-error estimate.

Solution: Report precision–recall and calibration metrics rather than raw physical-error accuracy, use class-weighted or focal losses only when selected on validation data, and generate test trials until a pre-specified minimum number of logical failures or confidence-interval precision is reached.

Challenge 2

Training the error-inference network against one sampled physical pattern may penalize a different but logically equivalent explanation of the same syndrome.

Solution: Treat per-qubit labels as auxiliary marginal targets, not as the final definition of success. Select and evaluate models primarily by held-out logical error rate, and retain the syndrome-consistent MWPM recovery so that the final correction is not required to reproduce the sampled physical error exactly.

Challenge 3

The neural model may appear better because of train/test leakage, unequal tuning effort, a different stopping rule, or one unusually favorable initialization.

Solution: Generate partitions from disjoint random seeds and verify that circuits are not duplicated. Freeze the test set until model selection is complete, match optimization budgets across neural ablations, repeat retained configurations across at least five seeds, and report distributions rather than only the best run.

Challenge 4

A decoder trained on simulated independent noise may fail on correlated noise or hardware data, and available quantum processors may not support a useful repeated surface-code circuit.

Solution: Measure generalization explicitly rather than hiding the domain shift. Add controlled correlated faults to simulation, recalibrate or fine-tune only on a small documented adaptation split, and use the public experimental dataset if direct hardware collection is infeasible. The primary simulation study can still be completed without quantum-hardware access.

9 Research Timeline

Milestone 1: Late October: Dataset and Decoder Baselines Established

Task — What will you do? Implement the surface-code simulation and preprocessing pipeline; verify detection-event tensors and logical labels; run MWPM and direct-neural baselines.

Objective — What measurable result should this work produce? Produce reproducible train, validation, and test partitions for the first distance-noise matrix, plus baseline logical-error curves with confidence intervals.

Goal — What larger research capability or milestone does this establish? Establish trustworthy data and conventional controls before evaluating the uncertainty-aware architecture.

Milestone 2: Mid-December: Error-Inference Model Complete

Task — What will you do? Implement the spatiotemporal syndrome encoder and X/Z probability heads; compare candidate model sizes and calibration methods using validation data.

Objective — What measurable result should this work produce? Complete at least five retained training runs and produce per-qubit precision–recall, Brier-score, calibration, runtime, and convergence results.

Goal — What larger research capability or milestone does this establish? Establish a stable, interpretable error-probability model for use by the correction stage.

Milestone 3: Mid-March: Soft-Information Ablation and Robustness Study Complete

Task — What will you do? Integrate the correction stage; run matched soft-versus-hard, direct-neural, and MWPM comparisons; test neighboring error rates and correlated noise.

Objective — What measurable result should this work produce? Produce paired logical-error comparisons for each retained code distance and noise condition, including uncertainty intervals, latency, and generalization results.

Goal — What larger research capability or milestone does this establish? Determine whether preserving uncertainty causes a reproducible improvement and identify the conditions under which any benefit disappears.

Milestone 4: Early May: Final Analysis and Research Product

Task — What will you do? Complete any feasible public-data or quantum-hardware validation; regenerate figures from archived runs; document method changes; and write the final paper and presentation.

Objective — What measurable result should this work produce? Produce a reproducible results package containing configurations, seeds, dataset manifests, model checkpoints, benchmark tables, final figures, and a conclusion addressing every research goal.

Goal — What larger research capability or milestone does this establish? Deliver a defensible answer to whether an uncertainty-aware intermediate error model improves surface-code decoding, including the possibility that it changes nothing useful or makes performance worse.

10 Group Responsibilities

Andrew Gao

Primary responsibility for the machine-learning portion of the project: locating and documenting datasets, preprocessing syndrome data, implementing and comparing neural architectures, selecting hyperparameters without test-set leakage, calibrating the error-probability heads, and carrying out statistical and error analyses. Andrew will maintain the shared training and evaluation pipeline so that all neural ablations use identical data partitions and stopping rules.

Lucas Trachtenberg

Primary responsibility for the quantum-error-correction portion of the project: defining the surface-code geometry and stabilizers, constructing repeated syndrome-extraction circuits and noise models, validating detection events and logical observables, implementing the MWPM reference recovery, and coordinating any public-hardware-data or quantum-processor experiments. Lucas will maintain records of circuit, code-distance, and decoder-graph configurations.

Both students will jointly define the final comparison matrix, verify that neural outputs correspond to valid surface-code corrections, review failed or anomalous runs, interpret logical-error results, and write the final paper. Each student is expected to understand the syndrome data, machine-learning pipeline, correction rule, benchmark metrics, and conclusions rather than treating the project as two independent software tasks.

Annotated Bibliography

- [1] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, “Surface codes: towards practical large-scale quantum computation”, *Physical Review A* **86**, 032324 (2012).
What did this work do? Reviewed the surface-code architecture, stabilizer measurement, decoding, logical operations, and resource requirements for fault-tolerant quantum computation.
Why is it relevant? It supplies the quantum-error-correction foundation for the proposal and explains why a decoder must return the state to the correct logical equivalence class.
How will we use it? We will use its surface-code and matching-decoder framework to define code distance, repeated syndrome extraction, logical failure, and the MWPM baseline.

- [2] G. Torlai and R. G. Melko, “Neural decoder for topological codes”, *Physical Review Letters* **119**, 030501 (2017).
What did this work do? Demonstrated that a neural-network model can decode topological quantum codes by learning the relationship between syndromes and logical-error classes.
Why is it relevant? It is an early direct precedent for replacing or augmenting a hand-specified decoder with a learned syndrome representation.
How will we use it? We will include a direct syndrome-to-logical-class neural baseline and compare it with the proposed explicit error-probability intermediate representation.

- [3] J. Bausch, A. W. Senior, F. J. H. Heras, T. Edlich, A. Davies, M. Newman, A. Milsted, et al., “Learning high-accuracy error decoding for quantum processors”, *Nature* **635**, 834–840 (2024).
What did this work do? Introduced AlphaQubit, a neural decoder trained on simulated and experimental surface-code data, and reported improved logical-error performance over several established decoders.
Why is it relevant? It demonstrates that learned models can exploit long syndrome histories and device-specific error correlations, while also setting a strong modern comparison point.
How will we use it? We will adopt its emphasis on logical error, repeated syndrome rounds, held-out hardware data, and careful comparison with standard decoders at a smaller student-project scale.

- [4] O. Higgott, “Pymatching: a python package for decoding quantum codes with minimum-weight perfect matching”, *ACM Transactions on Quantum Computing* **3**, 16:1–16:16 (2022).
What did this work do? Presented an open-source implementation of minimum-weight perfect matching for decoding graphlike quantum error-correcting codes.
Why is it relevant? MWPM is the proposal’s main conventional baseline and supplies a syndrome-consistent reference recovery for the neural correction stage.
How will we use it? We will decode every held-out syndrome with PyMatching and combine its reference recovery with any logical adjustment predicted by the proposed model.

- [5] B. M. Varbanov, M. Serra-Peralta, D. Byfield, C. Blank, F. Battistel, T. E. O’Brien, B. M. Terhal, N. Muthusubramanian, X. Fu, M. Moreira, H. Babu, F. Mueck, M. Beekman, N. Haider, J. Vlothuizen, L. DiCarlo, K. Bertels, and L. Lao, “Neural-network decoder for near-term surface-code experiments”, *Physical Review Research* **7**, 013029 (2025).

What did this work do? Developed a recurrent neural-network decoder for realistic near-term surface-code experiments and studied scaling, data efficiency, and performance under experimentally motivated conditions.

Why is it relevant? It is a close precedent for using spatial and temporal syndrome information in a decoder designed for small experimental surface codes.

How will we use it? We will use its benchmarking considerations when selecting code distances, repeated rounds, training-set sizes, and hardware-like validation tests.

- [6] J. Zhou, X. Gao, S. Liu, Z. Han, and R. Li, “Neural network decoder for the surface code under crosstalk”, arXiv preprint arXiv:2503.04642, 10.48550/arXiv.2503.04642 (2025).

What did this work do? Studied neural surface-code decoding under crosstalk and proposed a modified attention-based architecture for correlated-error conditions.

Why is it relevant? Crosstalk is an example of correlated noise for which a learned decoder may capture structure absent from an independently weighted matching graph.

How will we use it? We will include at least one controlled correlated-noise test and analyze whether any soft-information advantage survives the change from independent noise.

- [7] Stim Developers, *Gates supported by stim: detector and coordinate annotations*, <https://github.com/quantumlib/Stim/blob/main/doc/gates.md> (visited on 10/01/2026).

What did this work do? Documents how detector annotations identify deviations from expected measurement parities and how coordinate annotations locate qubits and detectors.

Why is it relevant? The CNN input must distinguish actual detection events from invalid or absent detector entries and preserve the circuit’s space–time geometry.

How will we use it? We will use circuit-defined detector parities and coordinates to build the event channels and validity masks, including the preparation and final-readout boundaries.

- [8] PyTorch Contributors, *Conv3d: pytorch documentation*, <https://docs.pytorch.org/docs/stable/generated/torch.nn.Conv3d.html> (visited on 10/01/2026).

What did this work do? Defines the channels-first input layout, learned 3D convolutional operation, and the effects of kernel size, stride, and padding on output dimensions.

Why is it relevant? It provides an implementation reference for processing the syndrome tensor over time, row, and column without prematurely flattening its geometry.

How will we use it? We will use stride-one convolutions with one-cell padding and explicit channel counts to implement and verify the proposed encoder’s tensor shapes.

- [9] C. A. Pattison, M. E. Beverland, M. P. da Silva, and N. Delfosse, “Improved quantum error correction using soft information”, arXiv preprint arXiv:2107.13589, 10.48550/arXiv.2107.13589 (2021).

What did this work do? Developed decoding methods that incorporate analog or confidence information rather than reducing every measurement to a hard binary decision.

Why is it relevant? The proposal’s central hypothesis is that retaining uncertainty can improve correction compared with thresholding the same inferred errors.

How will we use it? We will pass calibrated probabilities to the correction stage and compare them directly with an otherwise matched hard-thresholded ablation.

- [10] C. Gidney, “Stim: a fast stabilizer circuit simulator”, *Quantum* **5**, 497 (2021).

What did this work do? Introduced Stim, a stabilizer-circuit simulator designed to sample large quantum-error-correction circuits and detector outcomes efficiently.

Why is it relevant? Large, reproducible labeled datasets are essential for training and evaluating the proposed neural decoder across code distances and error rates.

How will we use it? We will use Stim to generate repeated surface-code memory experiments, detection events, and logical-observable labels from fixed circuit and noise configurations.

- [11] Google Quantum AI, *Data for “suppressing quantum errors by scaling a surface code logical qubit”*, (2022) (visited on 09/29/2026).

What did this work do? Released the circuit files, detection-event samples, observable outcomes, and related data underlying Google’s distance-3 and distance-5 surface-code experiments.

Why is it relevant? The dataset offers an external experimental benchmark without requiring the project to generate a large hardware dataset independently.

How will we use it? We will use a documented compatible subset for out-of-distribution evaluation if the data representation can be aligned with the proposed decoder.

- [12] R. Acharya et al., “Suppressing quantum errors by scaling a surface code logical qubit”, *Nature* **614**, 676–681 (2023).

What did this work do? Reported repeated surface-code experiments on Google quantum processors and demonstrated logical-error suppression as code distance increased under selected operating conditions.

Why is it relevant? It provides a real-hardware reference for the circuit structure, detector data, and logical-error measurements used in modern surface-code experiments.

How will we use it? We will use its experimental setting to guide hardware-like simulation and, if feasible, evaluate trained decoders on the associated public data.