Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Learning to Correct Quantum Errors

Mayank Goswami

mayankfph@iitr.ac.in

Indian Institute of Technology Roorkee

Shreeyam Bangera

shreekant_sb@ph.iitr.ac.in

Indian Institute of Technology Roorkee

Method Article

Keywords: quantum error correction, surface code, neural decoder, convolutional neural network, minimum-weight perfect matching, Stim, Sycamore

Posted Date: November 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11642492/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Learning to Correct Quantum Errors: A Compact Space-Time Convolutional Decoder that Outperforms Correlated Matching on Simulated and Real Surface-Code Data

Shreeyam Bangera [1] and Mayank Goswami,[1,2]

[1]Department of Physics, IIT Roorkee
[2]MF School of AI and DS, IIT Roorkee
mayank.goswami@ph.iitr.ac.in

***Abstract*—Fault-tolerant quantum computers must continually decode noisy parity-check data, the *syndrome*, to decide whether an encoded logical qubit has been corrupted. The standard decoder, minimum-weight perfect matching (MWPM), assumes independent, graph-like errors and is sub-optimal under realistic circuit-level noise. We present a compact residual 3D convolutional neural network (0.5–0.6 M parameters) that reads the space-time syndrome of a rotated surface code as a volumetric image. Trained on Stim simulations at code distances $d = 3$ and $d = 5$ and evaluated on 3.6×10^6 held-out shots, a three-seed ensemble beats correlated MWPM, the strongest baseline we test, at all 18 noise settings, with 6–46% fewer logical errors (paired McNemar $p < 0.03$ everywhere). On real Google Sycamore data ($d = 3$, five rounds), the network is pre-trained on the device's fitted error model and fine-tuned on half of the shots. It reaches a logical error rate of 13.03%, versus 13.57% for correlated MWPM on the same held-out shots, and wins on all eight cross-validation folds, although it does not reach belief-matching or tensor-network decoders. Its outputs are well calibrated (ROC AUC 0.998): discarding the 10% least-confident shots cuts errors tenfold. An ablation identifies space-time locality as the essential inductive bias. All models train in 49 minutes on two free GPUs, and the data and code are public.**

***Index Terms*—quantum error correction, surface code, neural decoder, convolutional neural network, minimum-weight perfect matching, Stim, Sycamore**

I. Introduction

Qubits are fragile: thermal fluctuations, control imperfections and crosstalk corrupt their state within microseconds. Useful quantum algorithms therefore require *quantum error correction* (QEC), in which one logical qubit is redundantly encoded across many physical qubits [1], [2]. The leading architecture is the surface code. Data qubits sit on a 2D grid, and ancilla qubits repeatedly measure the parities (stabilizers) of their neighbours. Measuring the data qubits directly would destroy the encoded information, so a QEC system only ever observes *detection events*: stabilizer outcomes that change between consecutive rounds. A classical *decoder* must turn this space-time pattern of alarms into one decision, namely whether the logical qubit was flipped.

Decoding is the computational heart of fault tolerance. Every decoder mistake becomes a logical error, so it must be accurate; syndrome data arrive every $\sim 1\,\mu$s, so it must also be fast. The workhorse is minimum-weight perfect matching (MWPM) [1], [3], which pairs detection events along the most probable error chains. MWPM is efficient, but it treats X and Z errors independently and assumes each fault flips at most two detectors. Real *circuit-level* noise violates both assumptions: a single faulty two-qubit gate can produce correlated X/Z (i.e. Y) errors and "hook" errors. Correlated matching [3], [4] partially compensates by re-weighting the graph. Belief matching [5] and tensor-network decoders [6] are more accurate but substantially more expensive.

Neural networks can learn these correlations directly from data [7]–[9]. In 2024, Google DeepMind's AlphaQubit, a recurrent transformer, outperformed state-of-the-art classical decoders on real Sycamore data [10], and Google's Willow processor demonstrated below-threshold error suppression [11]. These systems rely on industrial compute budgets. We ask a practical question: *can a small, interpretable convolutional decoder, trained in under an hour on free hardware, beat the strongest standard matching decoder, both in simulation and on real quantum hardware?*

A. Problem statement and objectives

Problem: given the space-time pattern of detection events from a surface-code memory experiment under realistic circuit-level noise, predict whether the logical qubit was flipped, more accurately than matching-based decoders. *Why it must be solved:* every decoding mistake is a logical error, so the decoder directly sets the error rate of the whole quantum computer, and a decoder that cannot exploit correlated errors wastes part of the benefit of building larger codes. *Objectives:* (i) learn correlated errors from data with a compact neural decoder; (ii) outperform both plain and correlated MWPM, with paired statistical tests; (iii) validate on real quantum-hardware data without data leakage; (iv) keep the full pipeline reproducible on free compute in under an hour.

Contributions.

1) A 0.5–0.6 M-parameter residual 3D CNN that treats the syndrome as a space-time volume. Its three-seed ensemble beats correlated MWPM at all 18 simulated operating points, with paired significance tests.
2) A leakage-free hardware protocol: pre-train on Google's device-fitted error model, fine-tune on half of the real shots, test on the other half. On Sycamore data the CNN beats correlated matching on all eight folds.
3) An inductive-bias ablation: a fully connected network matches the CNN at d=3 but loses error suppression at d=5.
4) A matching-assisted hybrid decoder, whose smaller gain we report as an informative negative result.
5) Analysis of calibration and post-selection, robustness to noise mismatch, and failure modes.
6) Public release of the data (with fixed seeds), code and notebooks.

II. Related Work

Early neural decoders. Torlai and Melko trained Boltzmann machines to decode the toric code [7]. Varsamopoulos *et al.* used feed-forward networks on small surface codes [8], and Chamberland and Ronagh studied deep decoders for small fault-tolerant experiments [9]. Baireuther *et al.* showed that a recurrent (LSTM) decoder can outperform blossom matching by learning correlations between X and Z errors over many rounds [12].

Convolutional decoders. Convolutions exploit the translation invariance of topological codes. Ni scaled a renormalisation-group-inspired network to toric codes of distance 64, but only under code-capacity bit-flip noise [13]. Meinerz *et al.* used a sliding local network as a pre-decoder ahead of a global decoder, reaching tens of thousands of qubits under depolarizing noise [14]. Gicev *et al.* paired a CNN with a "mop-up" decoder for residual errors to obtain a fast, scalable syndrome decoder [15]. These works mostly treat code-capacity or phenomenological noise and compare against plain MWPM or union-find.

Graph and recurrent decoders on circuit-level and hardware data. Lange *et al.* cast decoding as graph classification on the detector graph. Their graph neural network (GNN) outperformed matching under simulated circuit-level noise and matched MWPM on Google's experimental repetition-code data [16]. Varbanov *et al.* trained an LSTM decoder that achieved about 25% lower logical error than MWPM on Google's 2023 surface-code data [17], [18]. AlphaQubit [10], a recurrent transformer trained at industrial scale, surpasses tensor-network decoding on the same device.

Our position. We study the opposite end of the cost spectrum: a single-shot 3D CNN that sees the whole space-time volume at once, with no mop-up stage and under a one-hour free-GPU budget. We benchmark it against *correlated* MWPM rather than plain MWPM, report seed variance and paired significance tests, and use a leakage-free pre-train/fine-tune protocol on real Sycamore surface-code data. This protocol lets us compare directly against Google's own released decoder predictions.

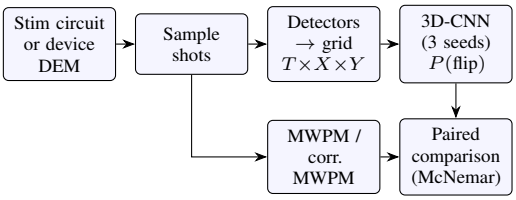


Fig. 1. Workflow. Every decoder sees exactly the same shots, which enables paired significance tests.

III. Theory

A. Rotated surface-code memory experiment

A distance-d rotated surface code uses d^2 data qubits and d^2-1 measurement qubits. In a Z-basis memory experiment, the logical qubit is prepared in $|0\rangle_L$, d rounds of stabilizer measurement are performed, and all data qubits are finally measured. A *detector* is the XOR of two consecutive measurements of the same stabilizer; without errors every detector reads 0. The logical observable is the parity of one row of data qubits, and the decoder predicts whether it flipped. Below a *threshold* physical error rate, increasing d suppresses the logical error rate (LER) exponentially. We quantify this with $\Lambda^{-1} = \mathrm{LER}(d{=}5)/\mathrm{LER}(d{=}3)$, where values below 1 mean the larger code helps.

B. Why decoding is hard

Degeneracy: many physical error patterns produce the same syndrome. Some of these patterns differ by a logical operator, so even a perfect decoder cannot always be right. Decoding is therefore probabilistic classification of the logical class, not identification of "the" error. *Correlations:* one depolarizing fault on a CNOT can trigger detectors in both stabilizer types and in two time steps, which a matching graph cannot represent exactly.

IV. Methodology

A. Simulated data

We use Stim [19] (v1.16) to generate `surface_code:rotated_memory_z` circuits with d rounds. The noise is standard circuit-level depolarizing noise of strength p: two-qubit depolarization after every Clifford gate, single-qubit depolarization on data qubits before every round, and bit flips before every measurement and after every reset. Each shot yields 24 (d=3) or 120 (d=5) detector bits and one label. Training and validation data mix five noise levels, $p \in \{0.2, 0.4, 0.6, 0.8, 1.0\}\%$, so that one model serves all strengths (Table I). The test sets use *different random seeds* and nine noise levels, four of which $(0.1, 0.3, 0.5, 1.2\%)$ are never seen in training. Two out-of-distribution (OOD) sets multiply the measurement and reset error rates by five. All seeds are fixed, and sampling and decoding run in parallel on all CPU cores.

TABLE I
SIMULATED DATASET (ALL SPLITS PUBLIC). SHOTS PER FILE.

Split	p values	$d = 3$	$d = 5$
Train	5 mixed	2.0×10^6	4.0×10^6
Validation	5 mixed	1.0×10^5	1.0×10^5
Test	9 (separate files)	$9 \times 2 \times 10^5$	$9 \times 2 \times 10^5$
OOD (meas. $\times 5$)	0.4%, 0.8%	$2 \times 2 \times 10^5$	$2 \times 2 \times 10^5$

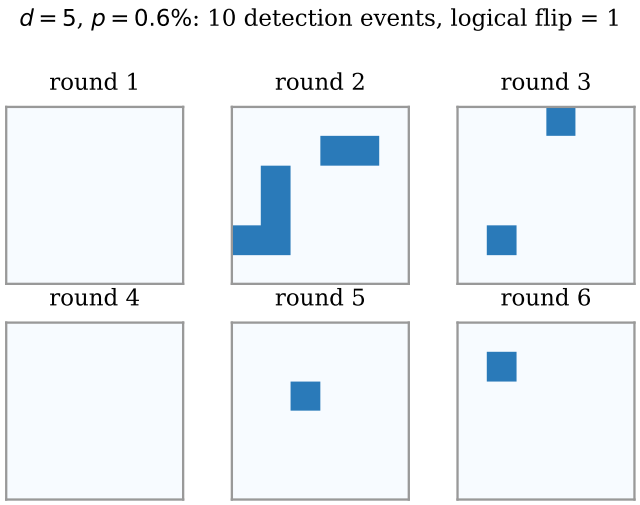


Fig. 2. A d=5 test shot as the CNN sees it: one 6×6 slice per round. Errors appear as local clusters in space and time.

B. Syndrome as a space-time image

Each detector carries coordinates (x, y, t) from its own circuit. We map the sorted unique values onto a dense grid of shape $T \times X \times Y$: 4^3 for d=3, 6^3 for d=5, and $6 \times 5 \times 5$ for the Sycamore circuits. Unused cells are filled with zeros (Fig. 2). Building the grid from each circuit's *own* coordinates is essential. In a preliminary experiment, we fed hardware detectors into the layout of the simulated circuit, and the decoder fell to near-chance accuracy.

C. Decoders

MWPM uses PyMatching v2 (sparse blossom) [3] with the detector error model (DEM) of the *true* simulation circuit, giving the baseline perfect noise knowledge. **Correlated MWPM** is the same decoder run with `enable_correlations=True`, which re-weights edges after a first pass to account for X–Z correlated faults [4]. It is the strongest baseline in this study.

3D-CNN (ours). A 3^3 convolutional stem with C channels (C=48 for d=3, C=32 for d=5) feeds three residual blocks, each containing two 3^3 convolutions with BatchNorm and GELU. A 1^3 convolution then squeezes the features to 8 channels, followed by a dense head (256 hidden units, dropout 0.1) that outputs one logit. The receptive field spans the whole volume. The model (Fig. 3) has 507 k (d=3) and 610 k (d=5) parameters. We train three seeds and average their probabilities to form the *ensemble*.

Hybrid. The same CNN additionally receives the correlated-MWPM decision $s \in \{\pm 1\}$, which is concatenated into the head and added as a residual, $\text{logit} = f(x, s) + \alpha s$ with α initialised to 4. The model therefore starts out *identical* to correlated MWPM and only has to learn when matching fails.

MLP (ablation). This model receives the flattened input and has three hidden layers (512–512–256). It has 4.3×10^5 (d=3) and 5.1×10^5 (d=5) parameters and no notion of which detectors are neighbours.

D. Training

All models use binary cross-entropy loss, AdamW (learning rate 3×10^{-3}, weight decay 10^{-4}), a one-cycle schedule, batches of 4096 and mixed precision. The full training set stays on the GPU. Training jobs are scheduled largest-first on a shared queue served by one worker per GPU. The CNNs train for 8 (d=3) and 6 (d=5) epochs, and the checkpoint with the lowest validation LER is kept. Test data are never used for model selection. Data generation, all baseline decoding, training of 8 simulation models and 16 hardware models, and evaluation take 48.6 minutes on two NVIDIA T4 GPUs (Kaggle free tier).

E. Hardware protocol

We use Google's public Sycamore surface-code data [18], [20]: distance 3, 5 rounds, Z basis, and four patches centred on different device qubits, with 5×10^4 shots each. Google provides two error models fitted to the device, one estimated from the even-indexed shots (for decoding the odd ones) and one from the odd-indexed shots. For fold A we: (i) pre-train the CNN on 4×10^5 synthetic shots sampled from `pij_from_even_for_odd.dem`; (ii) fine-tune it on 90% of the real even shots, using the remaining 10% for validation; and (iii) test on all 2.5×10^4 real odd shots. Fold B mirrors this. MWPM and correlated MWPM use exactly the same fitted model, so no decoder sees the test shots or a model estimated from them. We also report the predictions that Google released for its own decoders on the same shots [20].

F. Statistics

LER is the fraction of test shots whose logical flip is mispredicted. Because all decoders see identical shots, we use the exact two-sided McNemar test on discordant pairs (n_a: only our decoder wrong; n_b: only the baseline wrong). This test is far more sensitive than comparing binomial confidence intervals.

V. RESULTS

A. Simulation: accuracy against matching

Table II and Fig. 4 show the main result. At $d = 3$, the ensemble reduces LER by 8.7–20.0% relative to MWPM and by 6.3–43.7% relative to correlated MWPM. Correlated matching is itself *worse* than plain MWPM at d=3 and low p, because its re-weighting over-corrects on such a small graph. At $d = 5$,

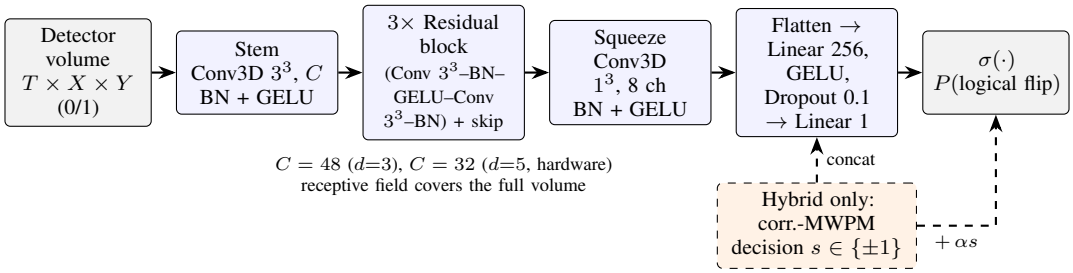


Fig. 3. Decoder architecture. The syndrome of one shot is a binary space-time volume (4^3 for d=3, 6^3 for d=5, 6×5×5 on Sycamore). The hybrid variant (dashed) additionally receives the correlated-MWPM decision and starts out identical to it (α initialised to 4).

TABLE II
SIMULATED TEST LOGICAL ERROR RATE ($\times 10^{-3}$; 2×10^5 SHOTS PER CELL). THE CNN COLUMN IS MEAN ± S.D. OVER THREE SEEDS. BEST PER ROW IN BOLD. LAST COLUMN: RELATIVE CHANGE OF THE ENSEMBLE VS. CORRELATED MWPM (NEGATIVE MEANS CNN IS BETTER).

d	p (%)	MWPM	Corr. MWPM	CNN (3 seeds)	**CNN ens.**	Hybrid	Δ vs Corr.
	0.1	0.840	1.235	0.702±0.006	**0.695**	0.785	−43.7%
	0.2	2.895	3.350	2.338±0.008	**2.315**	2.415	−30.9%
	0.3	6.405	6.845	5.358±0.028	**5.275**	5.395	−22.9%
	0.4	11.050	10.755	9.387±0.053	**9.365**	9.375	−12.9%
3	0.5	16.965	16.445	14.697±0.127	**14.615**	14.895	−11.1%
	0.6	23.770	23.440	21.260±0.177	**21.160**	21.400	−9.7%
	0.8	39.515	38.580	35.812±0.116	**35.580**	36.275	−7.8%
	1.0	59.050	57.565	53.435±0.068	**53.095**	54.065	−7.8%
	1.2	79.995	77.945	73.433±0.183	**73.035**	74.565	−6.3%
	0.1	0.120	0.110	0.068±0.008	**0.060**	0.075	−45.5%
	0.2	1.100	0.860	0.723±0.023	**0.710**	0.790	−17.4%
	0.3	3.370	2.385	2.085±0.036	**1.960**	2.200	−17.8%
	0.4	7.825	5.770	5.330±0.063	**5.060**	5.330	−12.3%
5	0.5	14.240	11.015	10.462±0.136	**9.935**	10.415	−9.8%
	0.6	22.650	18.270	16.892±0.429	**15.795**	17.540	−13.5%
	0.8	49.420	40.810	39.000±0.488	**36.810**	39.130	−9.8%
	1.0	83.080	72.635	70.910±1.076	**67.020**	70.365	−7.7%
	1.2	126.215	114.230	111.993±1.219	**106.415**	111.275	−6.8%

the gains grow: 15.7–50.0% relative to MWPM and 6.8–45.5% relative to correlated MWPM, with 12–18% over correlated matching in the practically relevant range $p = 0.2$–0.4%. Gains shrink at the highest noise levels, where many shots are intrinsically undecodable. Training is reproducible: the seed-to-seed relative standard deviation is below 1% at d=3 and 1–3% at d=5 (11% only at p=0.1%, where there are just 12–15 errors). *Every individual seed* beats correlated MWPM at every operating point, and ensembling the three seeds lowers the LER by a further 0.2–6.5% for $p \geq 0.2\%$.

Table III lists the paired tests. All 36 ensemble comparisons are significant: $p < 0.03$ everywhere, $p < 2 \times 10^{-3}$ against MWPM everywhere, and $p < 10^{-5}$ in 32 of 36. The weaker comparisons all occur at the lowest noise levels, where very few shots are decoded differently (e.g. only 16 and 112 discordant shots against correlated MWPM at d=5, p=0.1% and 0.2%).

B. Matching-assisted hybrid

The hybrid also beats correlated MWPM at all 18 points, significantly so at every one ($p < 0.05$). However, its gains (2.6–8.1% at d=5 for $p \geq 0.2\%$) are consistently *smaller* than those of the pure CNN. Because it starts from the matching decision, its optimisation stays near that solution: at d=5 it disagrees with correlated MWPM on 3–6× fewer shots than the pure CNN does. Giving a network a strong classical answer therefore acts as a conservative prior. It guarantees no degradation, but within a fixed training budget it limits the improvement. We report this as an informative negative result.

C. Inductive bias: why locality matters

In a preliminary run on identical test sets, we trained the MLP and a larger dense-head CNN (1.16/3.03 M parameters, one seed). At d=3 (24 inputs), the MLP comes within 2% of the CNN, since a small code can effectively be memorised. At d=5 (120 inputs), its LER is 1.6–2.5× that of MWPM, and its Λ^{-1} already exceeds 1 at p=0.3% (Fig. 5). With the MLP as decoder, *building a bigger quantum code makes things worse*. The CNN shares weights across space and time and keeps Λ^{-1} below that of MWPM over the whole range, matching correlated MWPM. Replacing the 3 M-parameter dense head with the 8-channel squeeze reduced d=5 parameters by 5×, avoided the over-fitting seen after 8 epochs in the preliminary run, and lowered the single-seed LER at p=0.6% from 1.80% to $1.69 \pm 0.04\%$. This mirrors other physics-ML benchmarks in which locality priors enable efficient learning [21].

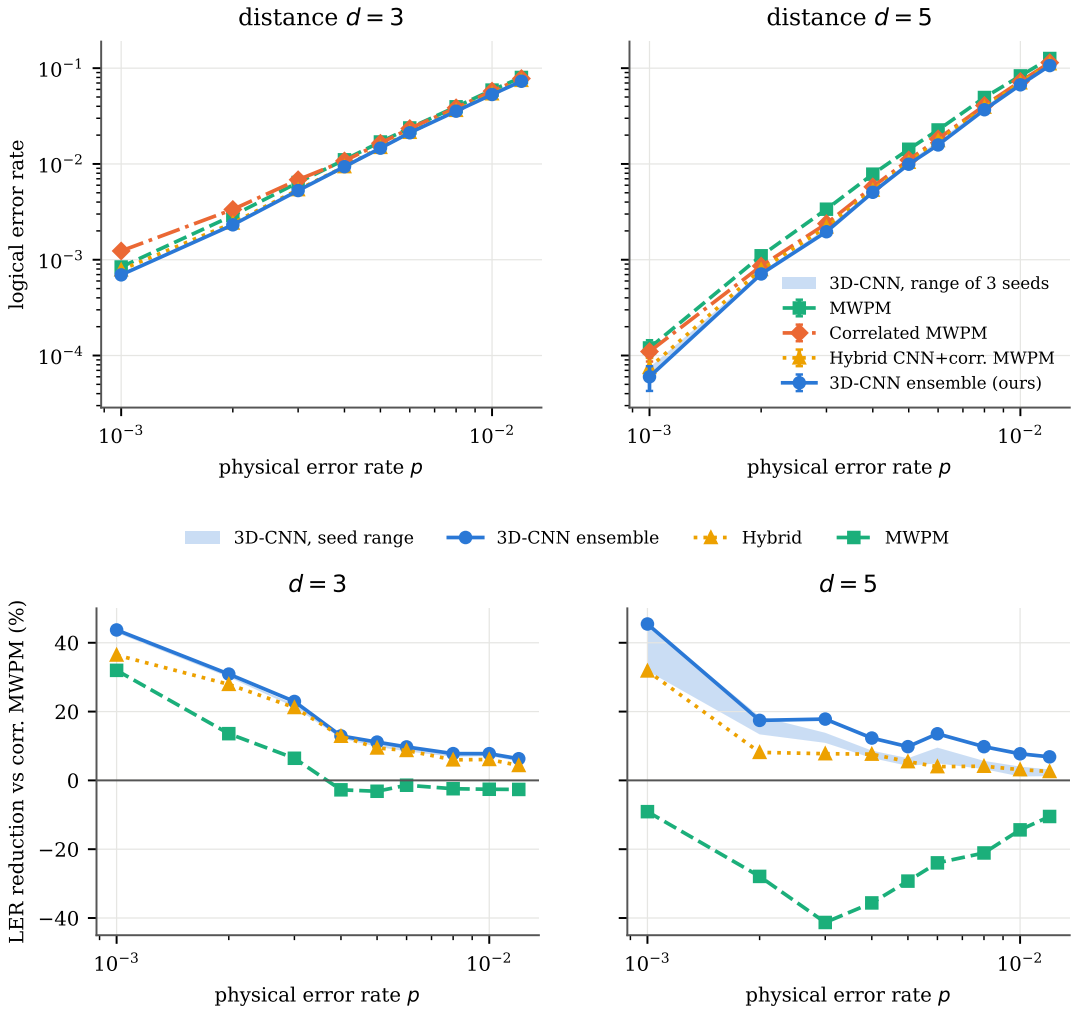


Fig. 4. Top: logical error rate on held-out simulated test sets (2×10^5 shots per point; error bars are binomial standard errors; the shaded band is the range over 3 training seeds). Bottom: LER reduction relative to correlated MWPM (positive means better). The CNN ensemble and every individual seed beat correlated MWPM at all 18 points.

D. Real hardware: Google Sycamore

Table IV and Fig. 6 report the leakage-free hardware evaluation on 2×10^5 real test shots. The fine-tuned CNN averages 13.03% LER, compared with 13.57% for correlated MWPM and 15.06% for MWPM, all using the same device-fitted model. It is lower than correlated MWPM on *all eight* folds (a sign test gives $p = 0.008$), significantly so on six (McNemar $p < 0.04$). It also beats Google's released PyMatching (15.40%) and correlated-matching (13.99%) predictions on the same shots. It does *not* reach belief matching (12.88%) or tensor-network contraction (12.45%). These decoders are more accurate but much slower, and the tensor-network decoder is generally considered too slow for real-time use. Pre-training on the fitted error model alone already gives 13.10%; fine-tuning on 2.25×10^4 real shots adds a modest 0.07 points. This indicates that Google's fitted model captures most of the device's noise structure, and that the CNN's advantage comes from modelling correlations that matching approximates. The hybrid reaches 13.33%.

E. Calibration, confidence and post-selection

Unlike MWPM, the CNN outputs a probability. At d=5, p=0.6%, the ensemble achieves ROC AUC = 0.998, and its predicted probabilities track observed flip frequencies closely (Fig. 7). Confidence is highly informative. Discarding the 5%, 10% and 20% least-confident shots reduces the LER from 1.58% to 0.42%, 0.16% and 0.04%, respectively. This can be used to flag shots for repetition, post-selection or slower re-decoding.

F. Robustness to unseen noise

In the OOD sets, measurement and reset errors are 5× stronger than during training (Fig. 8). Here we compare against matching that *assumes* the training noise model, which is the realistic deployment scenario. The CNN ensemble beats both mis-specified MWPM and mis-specified correlated MWPM in all four settings, for example 2.92% vs. 3.58% and 3.48% at

TABLE III

EXACT PAIRED MCNEMAR TESTS ON IDENTICAL TEST SHOTS. n_a: SHOTS ONLY OUR DECODER GETS WRONG; n_b: SHOTS ONLY THE BASELINE GETS WRONG. TWO-SIDED p-VALUES.

		Ens. vs MWPM			Ens. vs Corr.			Hybrid vs Corr.		
d	p(%)	n_a	n_b	p-val.	n_a	n_b	p-val.	n_a	n_b	p-val.
3	0.1	17	46	3×10^{-4}	20	128	2×10^{-20}	22	112	1×10^{-15}
	0.2	67	183	1×10^{-13}	75	282	2×10^{-29}	82	269	2×10^{-24}
	0.3	154	380	5×10^{-23}	174	488	2×10^{-35}	199	489	5×10^{-29}
	0.4	277	614	4×10^{-30}	308	586	1×10^{-20}	326	602	9×10^{-20}
	0.5	452	922	2×10^{-37}	453	819	7×10^{-25}	516	826	2×10^{-17}
	0.6	672	1194	6×10^{-34}	702	1158	3×10^{-26}	781	1189	3×10^{-20}
	0.8	1217	2004	5×10^{-44}	1225	1825	1×10^{-27}	1356	1817	3×10^{-16}
	1.0	1835	3026	6×10^{-66}	1916	2810	8×10^{-39}	2128	2828	3×10^{-23}
	1.2	2583	3975	1×10^{-66}	2604	3586	8×10^{-36}	2912	3588	5×10^{-17}
5	0.1	1	13	0.002	3	13	0.021	0	7	0.016
	0.2	35	113	9×10^{-11}	41	71	0.006	14	28	0.044
	0.3	105	387	6×10^{-39}	132	217	6×10^{-6}	45	82	0.001
	0.4	264	817	3×10^{-66}	315	457	4×10^{-7}	73	161	9×10^{-9}
	0.5	532	1393	1×10^{-88}	624	840	2×10^{-8}	123	243	3×10^{-10}
	0.6	876	2247	3×10^{-137}	969	1464	9×10^{-24}	227	373	3×10^{-9}
	0.8	2102	4624	1×10^{-212}	2394	3194	9×10^{-27}	449	785	8×10^{-22}
	1.0	4100	7312	4×10^{-201}	4409	5532	2×10^{-29}	748	1202	7×10^{-25}
	1.2	6702	10662	5×10^{-200}	7100	8663	1×10^{-35}	1058	1649	5×10^{-30}

TABLE IV

GOOGLE SYCAMORE HARDWARE (DISTANCE 3, 5 ROUNDS, Z BASIS), LER IN %, 2.5×10^4 HELD-OUT REAL SHOTS PER ROW. MWPM, CORR. (CORRELATED MWPM), CNN AND HYBRID USE THE ERROR MODEL FITTED ON THE *other* HALF OF THE SHOTS. †PREDICTIONS RELEASED BY GOOGLE [20]: PYMATCHING (PM), CORRELATED MATCHING, BELIEF MATCHING (BM) [5] AND TENSOR NETWORK (TN) [6]. CNN_{pre}: PRE-TRAINED ON THE FITTED MODEL ONLY; CNN_{ft}: ADDITIONALLY FINE-TUNED ON REAL SHOTS. LAST COLUMN: MCNEMAR p, CNN_{ft} VS. CORR.

Patch, fold	MWPM	PM†	Corr.	Corr.†	BM†	TN†	CNN_{pre}	**CNN_{ft}**	Hybrid	$p_{\text{CNN vs Corr}}$
(3,5), A	14.72	14.96	13.76	14.02	12.69	12.32	13.03	12.97	13.42	8×10^{-6}
(3,5), B	14.97	15.33	13.90	14.07	12.64	12.65	13.33	12.91	13.90	4×10^{-8}
(5,3), A	15.12	15.52	13.94	14.43	13.37	13.19	13.66	13.58	13.75	0.035
(5,3), B	14.65	15.13	13.47	13.81	13.14	12.65	13.30	13.35	13.27	0.491
(5,7), A	14.63	15.09	12.78	13.37	12.30	11.76	12.74	12.74	12.92	0.876
(5,7), B	15.12	15.62	13.26	14.07	12.63	12.05	12.52	12.52	12.70	3×10^{-5}
(7,5), A	15.72	15.98	13.88	14.26	13.26	12.53	13.19	13.19	13.40	2×10^{-4}
(7,5), B	15.53	15.57	13.57	13.89	13.02	12.43	13.04	13.00	13.29	0.001
Mean	15.06	15.40	13.57	13.99	12.88	12.45	13.10	13.03	13.33	

d=5, p=0.4%. In that setting it even beats an oracle MWPM that knows the true noise (3.04%). In the other three settings the oracle remains best, by 2–7%. Learned decoders thus degrade gracefully under noise mismatch, but they still benefit from training data matched to the device. Our hardware protocol achieves this match through device-fitted models and fine-tuning.

VI. DISCUSSION AND LIMITATIONS

Scale. We studied $d \leq 5$, and on hardware only $d = 3$ with 5 rounds. The parameter count of the dense head grows with the volume. A pooled or recurrent read-out would make the model distance-agnostic, which is needed for $d \geq 7$.

Strongest decoders. On hardware, belief matching and tensor-network decoding remain 1–5% more accurate. AlphaQubit [10] surpasses them using orders of magnitude more training data and compute. Our contribution is to show how much of the neural-decoder advantage over *matching* a compact model captures on a free compute budget.

Noise model. Simulated circuit-level depolarizing noise omits leakage, crosstalk and drift. The hardware results partly address this, since they rely on real device data.

Latency. Superconducting devices need decisions within $\sim 1\,\mu$s per round. In a preliminary measurement, a $5\times$ larger version of our network took 5–18 μs per shot with batched GPU inference. Real-time use would require quantisation and FPGA/ASIC deployment [8].

Hybrid. The hybrid's limited gain suggests that conditioning on a classical decision should be paired with longer training or explicit regularisation towards disagreement. We leave this to future work.

VII. ETHICAL AND SOCIETAL CONSIDERATIONS

Decoders are infrastructure for future quantum computers, which may eventually break today's public-key cryptography. Progress in QEC therefore strengthens the case for migrating to post-quantum cryptography now. Learned decoders are also less auditable than MWPM. Their failures are silent, can correlate

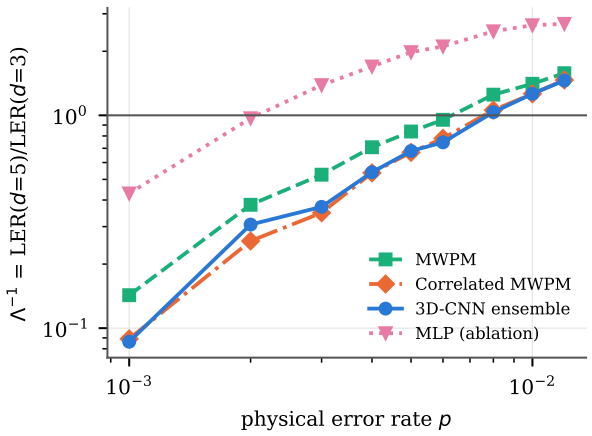


Fig. 5. Error suppression with code size; below the line, d=5 outperforms d=3. The crossing near $p \approx 0.7\%$ is the empirical threshold. The MLP (ablation, preliminary run) loses error suppression already at $p = 0.3\%$.

with untested noise regimes, and are hard to bound formally. We recommend deploying them alongside a classical fallback and calibrated confidence thresholds, which the CNN's probabilistic output enables. Finally, this work uses only open data and free compute, which lowers the barrier for groups without industrial resources to contribute to QEC research.

VIII. Conclusion

A compact residual 3D CNN, trained in under an hour on free GPUs, outperforms correlated minimum-weight perfect matching on surface-code decoding. In simulation it does so at every tested noise level and code distance, with statistical significance and reproducibly across seeds. On real Sycamore hardware, it beats correlated matching on every patch and fold of a leakage-free cross-validation. Space-time locality is the inductive bias that makes this possible. The model's calibrated confidence enables strong post-selection. Future work will scale to $d \geq 7$ with distance-agnostic read-outs, close the gap to belief matching on hardware, and distil the model for real-time hardware inference.

Data and Code Availability

The simulated dataset (training, validation, test and OOD splits, Stim circuits and manifest), the code, result tables, figures and both Kaggle notebooks are available at https://drive.google.com/drive/folders/1vwuUwRRQ8tfFtTnjAoC2KV4-wUqnBrCK. Trained model weights are available from the authors on request. Hardware data are from Google Quantum AI [20] (Zenodo, DOI 10.5281/zenodo.6804040). Or https://github.com/mggm1982/Learning-to-Correct-Quantum-Errors.git

IX. Video

https://youtu.be/Nqtdhu-rgVk

Declarations

Funding: No external funding was received. **Competing interests:** The authors declare no competing interests. **Author contributions:** S. B.: conceptualisation, implementation, experiments, analysis, writing. M. G.: supervision, review. **Ethics:** No human or animal subjects were involved. **Use of AI tools:** An AI assistant (Claude) was used to help write code. All experiments were run, and all results verified, by the authors.

References

[1] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *Journal of Mathematical Physics*, vol. 43, no. 9, pp. 4452–4505, 2002.
[2] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, p. 032324, 2012.
[3] O. Higgott and C. Gidney, "Sparse blossom: correcting a million errors per core second with minimum-weight matching," *Quantum*, vol. 9, p. 1600, 2025.
[4] A. G. Fowler, "Optimal complexity correction of correlated errors in the surface code," *arXiv preprint arXiv:1310.0863*, 2013.
[5] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, "Improved decoding of circuit noise and fragile boundaries of tailored surface codes," *Physical Review X*, vol. 13, p. 031007, 2023.
[6] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code," *Physical Review A*, vol. 90, p. 032326, 2014.
[7] G. Torlai and R. G. Melko, "Neural decoder for topological codes," *Physical Review Letters*, vol. 119, p. 030501, 2017.
[8] S. Varsamopoulos, B. Criger, and K. Bertels, "Decoding small surface codes with feedforward neural networks," *arXiv preprint arXiv:1705.00857*, 2017.
[9] C. Chamberland and P. Ronagh, "Deep neural decoders for near term fault-tolerant experiments," *arXiv preprint arXiv:1802.06441*, 2018.
[10] J. Bausch, A. W. Senior, F. J. H. Heras *et al.*, "Learning high-accuracy error decoding for quantum processors," *Nature*, vol. 635, pp. 834–840, 2024.
[11] Google Quantum AI and Collaborators, "Quantum error correction below the surface code threshold," *Nature*, vol. 638, pp. 920–926, 2025.
[12] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker, "Machine-learning-assisted correction of correlated qubit errors in a topological code," *Quantum*, vol. 2, p. 48, 2018.
[13] X. Ni, "Neural network decoders for large-distance 2D toric codes," *Quantum*, vol. 4, p. 310, 2020.
[14] K. Meinerz, C.-Y. Park, and S. Trebst, "Scalable neural decoder for topological surface codes," *Physical Review Letters*, vol. 128, p. 080505, 2022.
[15] S. Gicev, L. C. L. Hollenberg, and M. Usman, "A scalable and fast artificial neural network syndrome decoder for surface codes," *Quantum*, vol. 7, p. 1058, 2023.
[16] M. Lange, P. Havström, B. Srivastava, I. Bengtsson, V. Bergentall, K. Hammar, O. Heuts, E. van Nieuwenburg, and M. Granath, "Data-driven decoding of quantum error correcting codes using graph neural networks," *Physical Review Research*, vol. 7, p. 023181, 2025.
[17] B. M. Varbanov, M. Serra-Peralta, D. Byfield, and B. M. Terhal, "Neural network decoder for near-term surface-code experiments," *Physical Review Research*, vol. 7, p. 013029, 2025.
[18] Google Quantum AI, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, pp. 676–681, 2023.
[19] C. Gidney, "Stim: a fast stabilizer circuit simulator," *Quantum*, vol. 5, p. 497, 2021.
[20] Google Quantum AI, "Data for "suppressing quantum errors by scaling a surface code logical qubit"," Zenodo, 2022.
[21] V. Agrawal and M. Goswami, "Learning coarse-graining transformations in the 2D Ising model: A physics-informed approach to neural network interpretability," Research Square preprint, 2025.

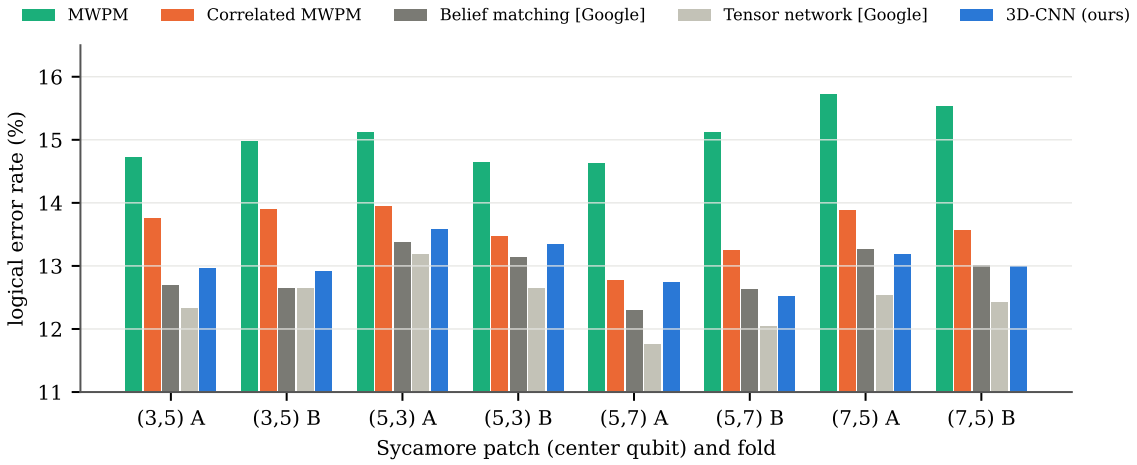


Fig. 6. Real-hardware LER per Sycamore patch and cross-validation fold. The CNN beats MWPM and correlated MWPM on every fold; Google's belief-matching and tensor-network decoders remain stronger.

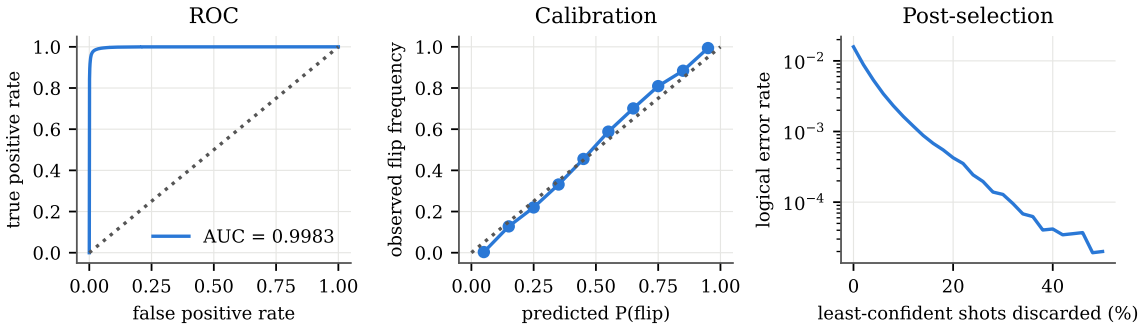


Fig. 7. CNN ensemble at d=5, p=0.6% (2×10^5 shots): ROC curve, reliability diagram and LER after discarding the least-confident fraction of shots.

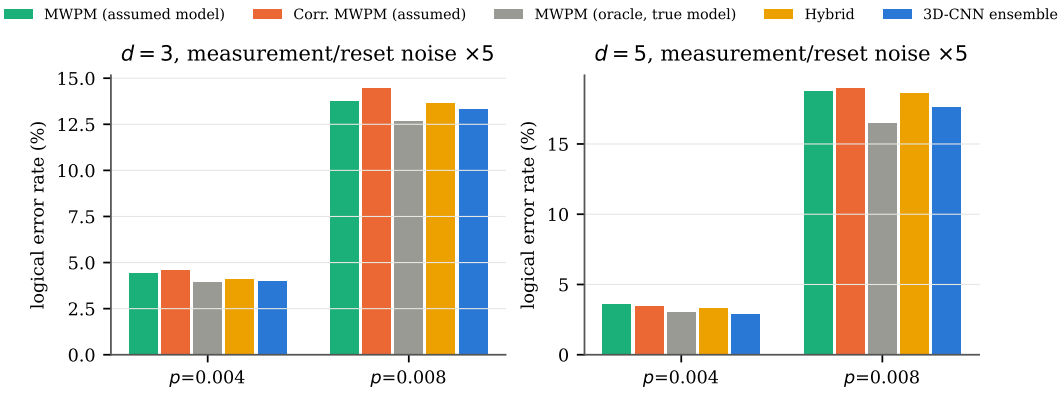


Fig. 8. Out-of-distribution test: measurement/reset noise 5× stronger than in training. Grey: MWPM with oracle knowledge of the true noise.