Decoding Small Quantum Codes as a Classification Problem: A Comparison of Analytic and Learned Decoders

Sharoon Ilyas[1]

[1]Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Abstract

This paper treats quantum error correction decoding as a classification problem. The measured syndrome provides the input features, and the logical error class is the output label. We compare an exact lookup table (maximum-a-posteriori), two analytic methods (minimum weight and maximum likelihood), and two learned models (logistic regression and a small multilayer perceptron). We apply every method to small stabilizer codes under bit-flip noise. Because these codes are small enough to enumerate fully, we compute exact logical accuracy instead of estimating it with Monte-Carlo sampling. Under homogeneous noise, minimum-weight and maximum-likelihood decoding have the same logical accuracy on the tested repetition codes and on Surface-17 at $p = 0.15$. Under checkerboard-biased noise on Surface-17, maximum likelihood reaches 0.8886 accuracy, compared with 0.8881 for weighted minimum-error decoding. Logistic regression stays below the optimum because the syndrome map is not linearly separable. In contrast, multilayer perceptrons with at least eight hidden ReLU units reach the exact optimum in a five-seed ablation. These results provide a reproducible benchmark that connects quantum decoding with statistical classification and supports further work on data-mining methods for error correction.

Keywords

quantum error correction, decoding, classification, statistical models, neural networks, semirings

1. Introduction

Quantum error correction (QEC) protects fragile quantum information by encoding one logical qubit across several physical qubits. The system repeatedly measures parity checks, called *stabilizers*. These measurements produce a binary *syndrome* that signals possible errors. A classical algorithm called a *decoder* reads the syndrome and predicts the required logical error class. Decoding quality directly affects whether the encoded information survives, so decoding is a central challenge in fault-tolerant quantum computing.

This task has the same basic form as classification. The syndrome is a fixed-length binary feature vector, the correct outcome is a logical error class, and the decoder maps the features to a label. Neural-network decoders are already an active research area. The same framework can also describe classical analytic decoders, which lets us compare analytic and learned methods directly.

We focus on small stabilizer codes: repetition codes of lengths three and five under independent bit-flip noise. Their small size lets us enumerate every possible error. We can therefore compute logical accuracy exactly instead of estimating it by random sampling. Removing Monte-Carlo uncertainty makes it possible to state precisely when the decoders agree, when they disagree, and why.

This paper makes the following contributions:

- We express small-code decoding as classification and use one evaluation harness to measure analytic and learned decoders with the same exactly computed logical accuracy (Sections 2 and 3).
- We compare minimum-weight and maximum-likelihood aggregation under homogeneous and biased noise. The biased Surface-17 experiment produces a strict accuracy gap. We use semiring language only to interpret the two endpoints (Sections 3 and 4.4).
- We compare logistic regression with multilayer perceptrons through learning curves, explicit syndrome-level errors, and a hidden-width ablation. These results show where each model family succeeds or fails (Section 4).

ITAT'26: Information Technologies -- Applications and Theory, September 25--29, 2026, Biele Karpaty, Slovakia

sharoon.ilyas94@gmail.com (S. Ilyas)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-09

CEUR-WS.org/Vol-4323/paper2.pdf

The individual methods are well known. This paper compares them directly and reproducibly within a single classification framework for a data-mining audience. It does not propose a new decoding algorithm.

2. Problem setup

Encoding and error detection: We work over the binary field $\mathbb{F}_2$ and consider CSS stabilizer codes that encode one logical qubit. We study only X (bit-flip) errors. Therefore $H \equiv H_Z \in \mathbb{F}_2^{m\times n}$ contains the Z-type stabilizer checks that detect those errors; it is not the full quantum check matrix. A binary vector $e \in \mathbb{F}_2^n$ records the data qubits on which an X error occurred. The failed checks form the *syndrome*:

$$s(e) = He \bmod 2 \in \mathbb{F}_2^m. \tag{1}$$

Several errors can produce the same syndrome. A decoder must therefore choose among all errors that match the observed syndrome.

Logical equivalence: Two errors with the same syndrome differ by a vector v in the kernel of H_Z, written $\ker(H_Z)$. Because all arithmetic is over $\mathbb{F}_2$, this means $H_Z v = 0$: the difference produces no syndrome. Multiplying an X error by an X-type stabilizer does not change the encoded state, whereas multiplying it by a logical X does. We choose a representative logical-Z operator $L \in \mathbb{F}_2^n$. Its overlap parity with an X error assigns the error's *logical class*:

$$\ell(e) = L \cdot e \bmod 2 \in \{0, 1\}. \tag{2}$$

Because L commutes with every X stabilizer and anticommutes with logical X, this parity distinguishes the two classes for a one-logical-qubit code. Decoding is therefore binary classification from syndrome to logical class. A correction succeeds when the decoder predicts the true class; otherwise, a logical failure occurs.

Repetition codes: The length-n repetition code uses $H \in \mathbb{F}_2^{(n-1)\times n}$. Row i compares neighboring bits i and $i+1$, for $i = 1, \ldots, n-1$. We choose a logical-Z representative L with one nonzero entry in its first position. We study $n = 3$ and $n = 5$. These are the smallest useful examples of the syndrome/class structure, and all of their errors can be enumerated ($2^n \leq 32$).

Noise model: Each physical bit flips independently with probability $p < \frac{1}{2}$. If an error e has Hamming weight $w = |e|$, then

$$P(e) = p^w (1-p)^{n-w}. \tag{3}$$

Every flipped bit contributes a factor p, and every unchanged bit contributes a factor $1-p$. Since $P(e)$ depends only on w, we can group errors by syndrome and logical class. For biased noise, qubit i instead flips with probability p_i:

$$P_{\text{bias}}(e) = \prod_{i=1}^{n} p_i^{e_i} (1-p_i)^{1-e_i}. \tag{4}$$

This probability depends on the full error pattern, not only its weight. The exact evaluation therefore keeps each enumerated error separately.

Decoding as classification: The input feature is a syndrome $s \in \mathbb{F}_2^m$, and the target label is a logical class $\ell \in \{0, 1\}$. A decoder is a function $d : \mathbb{F}_2^m \to \{0, 1\}$. We measure its *logical accuracy* as

$$\mathrm{Acc}(d) = \sum_{e\,:\,d(s(e))=\ell(e)} P(e), \tag{5}$$

the total probability of all errors classified correctly. For the small codes in this study, we evaluate the sum over all 2^n errors. The result is exact and requires no stochastic test sampling.

3. Decoder methodologies

We evaluate five decoding strategies from three families. Every strategy is measured with the exact logical accuracy in (5).

3.1. Lookup-table (MAP) decoder

For a small code, exhaustive enumeration can produce an optimal reference table. For each syndrome, the table stores the logical class with the largest posterior probability:

$$d_{\mathrm{LUT}}(s) = \arg\max_{\ell \in \{0,1\}} \sum_{e\,:\,s(e)=s,\ \ell(e)=\ell} P(e). \tag{6}$$

This maximum-a-posteriori (MAP) decoder is optimal by construction. We use it as the reference for the other methods. A complete table is practical here only because the tested codes have few syndromes.

3.2. Analytic scoring decoders and their deformation

The analytic decoders differ in how they combine errors within a logical class. Define the cost of one flip as $\omega = \log \frac{1-p}{p} > 0$. Taking the negative logarithm of (3) gives

$$-\log P(e) = -w \log p - (n-w)\log(1-p) \tag{7}$$

$$= w \cdot \log \frac{1-p}{p} - n \log(1-p) = w \cdot \omega + \text{const}. \tag{8}$$

Thus, under homogeneous noise, the lowest-weight error is also the most likely individual error. Under biased noise (4), we use the negative log-probability of the full error pattern as a weighted cost.

Minimum-weight decoder: For each logical class, minimum weight finds the lightest compatible error. It then chooses the class with the smaller minimum:

$$d_{\mathrm{MW}}(s) = \arg\min_{\ell} \min_{e\,:\,s(e)=s,\ \ell(e)=\ell} w(e). \tag{9}$$

In this small-code setting, it plays the role of the minimum-weight matching methods commonly used for surface codes.

Maximum-likelihood decoder: Maximum likelihood adds the probabilities of all compatible errors in each logical class:

$$d_{\mathrm{ML}}(s) = \arg\max_{\ell} \sum_{e\,:\,s(e)=s,\ \ell(e)=\ell} P(e). \tag{10}$$

It gives the same decision as the MAP lookup table. We list it separately because it performs the probability sum directly and forms one endpoint of the interpretation below.

Semiring interpretation: Minimum weight uses tropical aggregation $(\min, +)$, whereas maximum likelihood uses ordinary probability aggregation $(+, \times)$. The *soft minimum* gives a compact interpolation:

$$\mathrm{softmin}_T(c_1, \ldots, c_k) = -T \log \sum_i e^{-c_i/T}, \tag{11}$$

where $c_i = w_i \cdot \omega$. As $T \to 0$, it keeps only the minimum cost. At $T = 1$, it equals the negative logarithm of the summed class probability, up to a class-independent constant. Minimizing it therefore gives the ML decision. We do not test intermediate temperatures. Equation (11) only explains the relationship between the two reported endpoints; it is not a new decoder or a separate experiment.

3.3. Learned classifiers

We also treat decoding as supervised learning. We sample errors from (3) and create labelled pairs $(s(e), \ell(e))$. We train two standard models:

- **Logistic regression**, a linear classifier on the binary syndrome features;
- **Multilayer perceptron (MLP)**, with one hidden layer of 32 ReLU units.

For $n = 5$, only $2^{n-1} = 16$ syndromes can be produced by the $2^n = 32$ possible errors. The 32-unit hidden layer is therefore not a compression bottleneck; it is a high-capacity baseline for this small domain. Section 4 reports a width ablation. Widths of at least eight reach the optimum in this experiment, while narrower models are less accurate and more sensitive to random seeds. This threshold is not a scaling claim for larger codes.

After training, each model defines a map $\mathbb{F}_2^m \to \{0, 1\}$. We apply that map to every one of the 2^n errors and sum the probabilities of the correctly classified cases. Learned and analytic decoders are therefore compared on the same exact scale.

4. Experiments and results

Protocol. For each repetition code ($n \in \{3, 5\}$) and each physical error rate p on a 15-point grid from 0.02 to 0.30, we compute the exact logical accuracy (5) of all five decoders. At each p, the learned models train on 2000 sampled syndromes. We evaluate their final accuracy by full enumeration. This removes Monte Carlo evaluation error for each fitted rule, but not variation from training samples or model initialization. The main comparison uses the fixed seeds in Appendix A; the learning curves and width ablation average over five paired data/model seeds. The Surface-17 experiments also use full enumeration, with either homogeneous noise or the qubit-dependent probabilities in (4). The code uses the standard nine-data-qubit Surface-17 stabilizers of O'Brien et al. [2].

4.1. Analytic decoders agree on repetition codes

Table 1 gives exact accuracies for the distance-5 repetition code. The three analytic decoders appear in one column because they make the same logical decision for every syndrome. Their accuracy is therefore equal at every tested error rate. Figure 1 shows the same result for both the $n = 3$ and $n = 5$ codes: the maximum-likelihood and minimum-weight curves overlap exactly.

The reason is simple. For these repetition codes, the logical class with the lightest compatible error also has the largest total probability. Minimum weight and probability summation therefore choose the same class. These tests instead ask whether a learned model can reproduce that analytic rule. They do not show an advantage from degeneracy, where the sum of many errors in one class beats the lightest error in another. Section 4.4 first shows the same equality for homogeneous Surface-17 noise, then separates the methods with biased noise.

Table 1
Exact logical accuracy on the distance-5 repetition code. The lookup, minimum-weight, and ML decoders agree and are shown once. LR = logistic regression; MLP = multilayer perceptron trained on 2000 samples at each p. The table shows seven representative points from the 15-point grid used in Figure 2.

p	Analytic	LR	MLP
0.06	0.9980	0.9896	0.9924
0.10	0.9914	0.9720	0.9850
0.14	0.9780	0.9467	0.9780
0.18	0.9563	0.9145	0.9563
0.22	0.9256	0.8761	0.9256
0.26	0.8857	0.8324	0.8857
0.30	0.8369	0.7840	0.8369

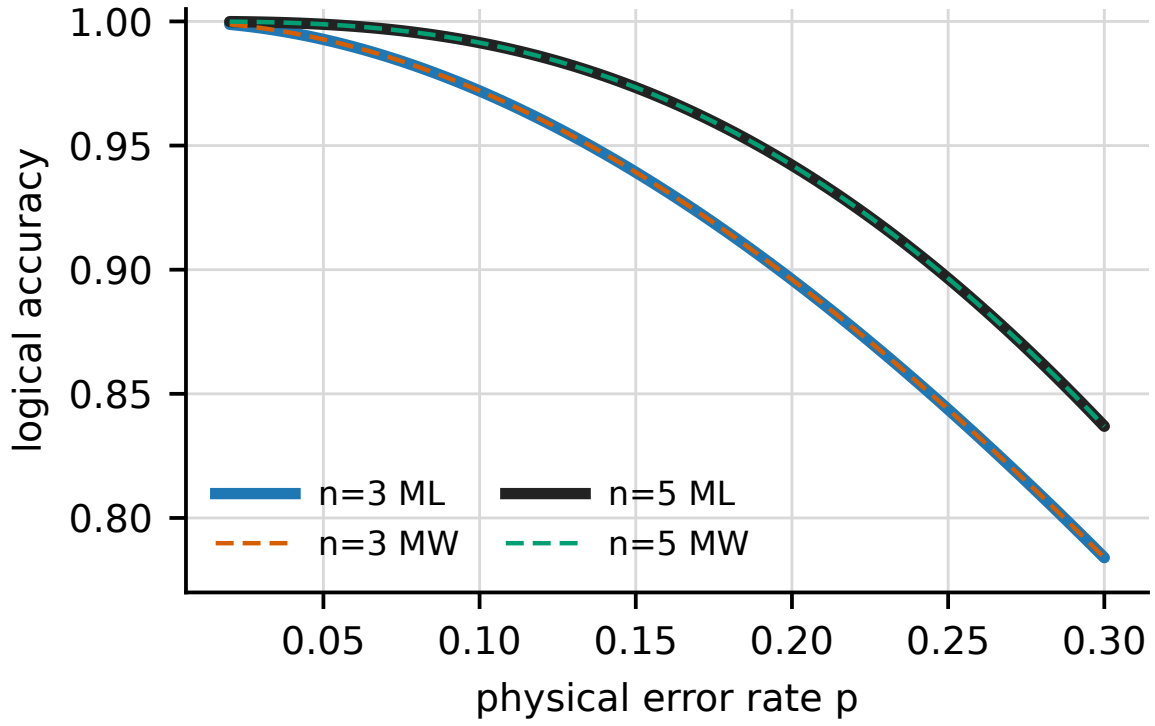


Figure 1: Maximum-likelihood and minimum-weight decoding have identical logical accuracy on the $n = 3$ and $n = 5$ repetition codes. Each dashed curve lies directly over its corresponding solid curve.

4.2. A linear classifier cannot represent the boundary

Logistic regression stays below the analytic optimum (Table 1, Figure 2). The limitation is structural: the optimal map from raw syndrome bits to logical classes is not linear. Logistic regression cannot represent it exactly, regardless of how much training data it receives. This small example shows why model choice matters: the given syndrome features are not linearly separable by logical class.

At $p = 0.15$, with 2000 samples and seed 7, LR agrees with MAP on 13 of the 16 syndrome inputs. Table 2 lists the three disagreements. A standard 2×2 confusion matrix would not show which syndrome vectors caused them.

Table 2
All LR syndrome-level errors for the distance-5 repetition code at $p = 0.15$ (2000 samples, seed 7). Mass is the exact probability of each syndrome, summed over both logical classes.

Syndrome	MAP class	LR class	Mass
0100	1	0	0.016256
1010	0	1	0.016256
1110	1	0	0.016256

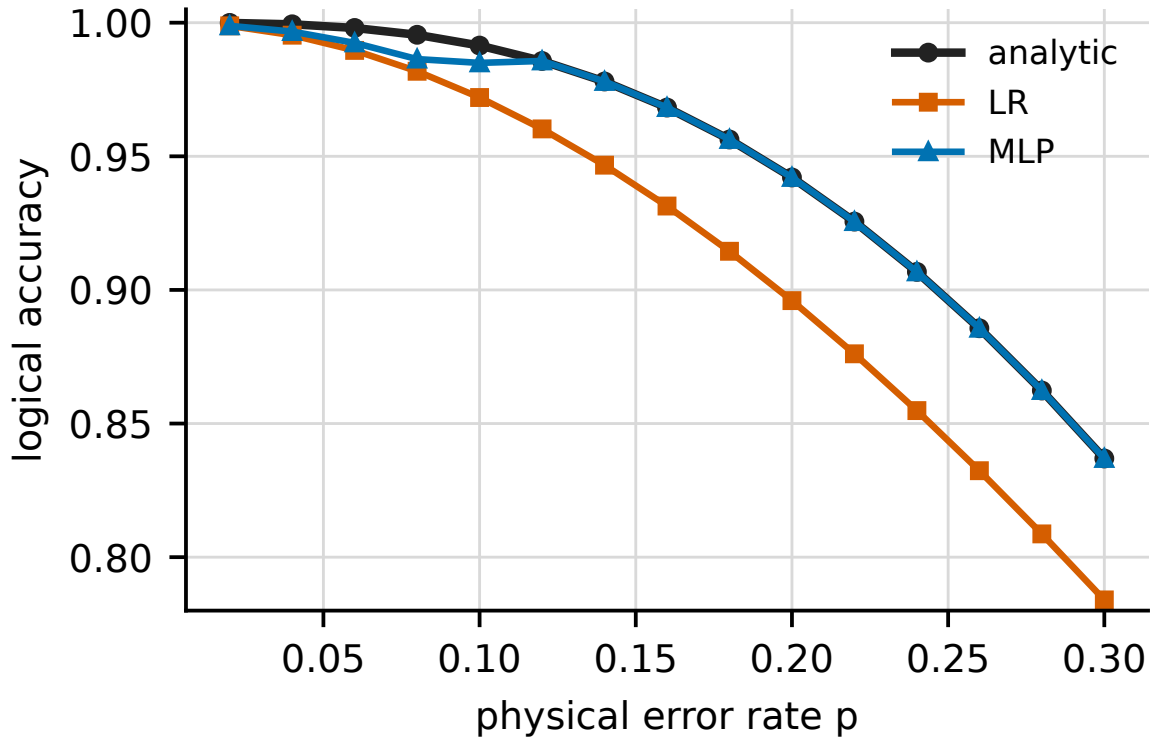


Figure 2: Logical accuracy on the distance-5 repetition code. The analytic decoders coincide, and the trained MLP follows them closely. Logistic regression remains lower because the raw syndrome classes are not linearly separable.

4.3. A small MLP reaches the optimum with enough data

With enough training data, the MLP matches the analytic optimum at most error rates (Table 1). Figure 3 shows both learned models at $p = 0.15$. Across five paired data/model seeds, MLP-32 improves from 0.9260 ± 0.0266 with 50 samples to the maximum-likelihood optimum of 0.9734 with 2000 samples. LR improves from 0.8523 ± 0.0046 but levels off at 0.9393. More data cannot remove its linear-boundary limitation.

Only 16 distinct syndrome inputs are possible. Larger training sets therefore reuse these inputs while adding more sampled errors and their logical-class labels. This resembles multiple passes over a finite input set. The required sample counts still depend on the code, error rate, architecture, optimizer, and seed protocol.

Figure 4 fixes $p = 0.15$ and the training size at 2000. Two hidden units reach 0.9191 ± 0.0348, and four reach 0.9620 ± 0.0102. Widths 8, 16, 32, and 64 all reach 0.9734 with no observed spread across five seeds. The result is stable above eight units in this experiment, but it is not independent of width.

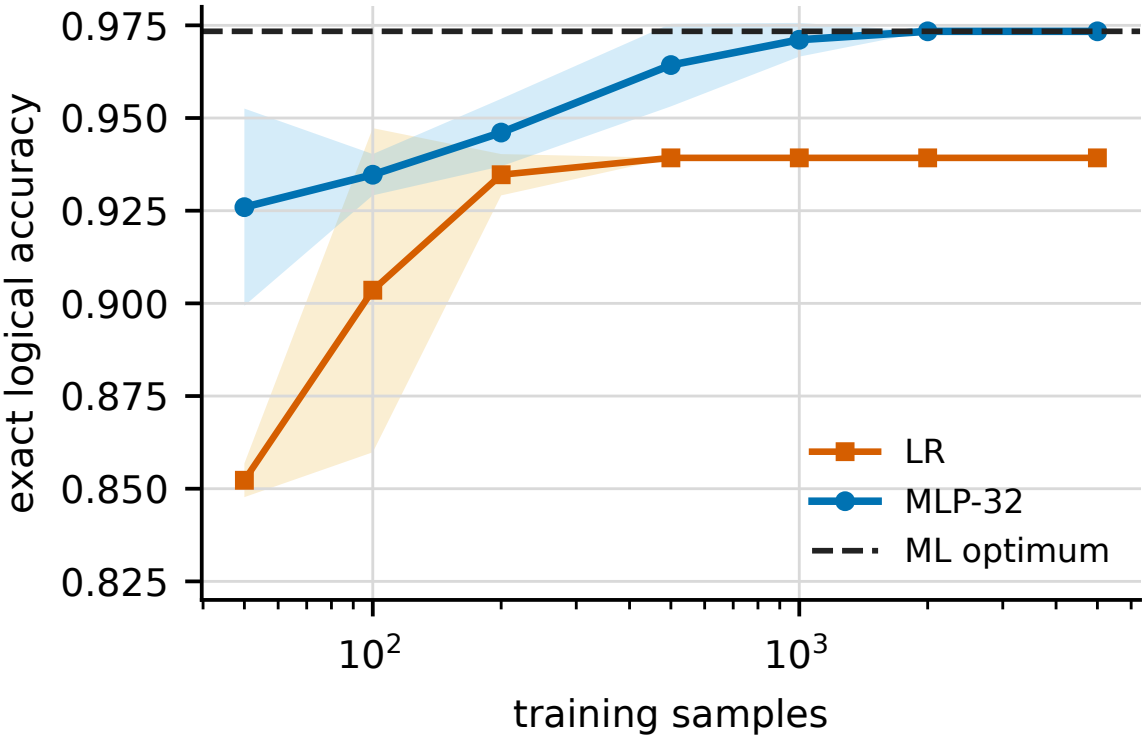


Figure 3: Learning curves for LR and MLP-32 at $p = 0.15$ on the distance-5 repetition code (mean $\pm$ standard deviation over five paired data/model seeds). MLP-32 reaches the ML optimum; LR levels off below it.

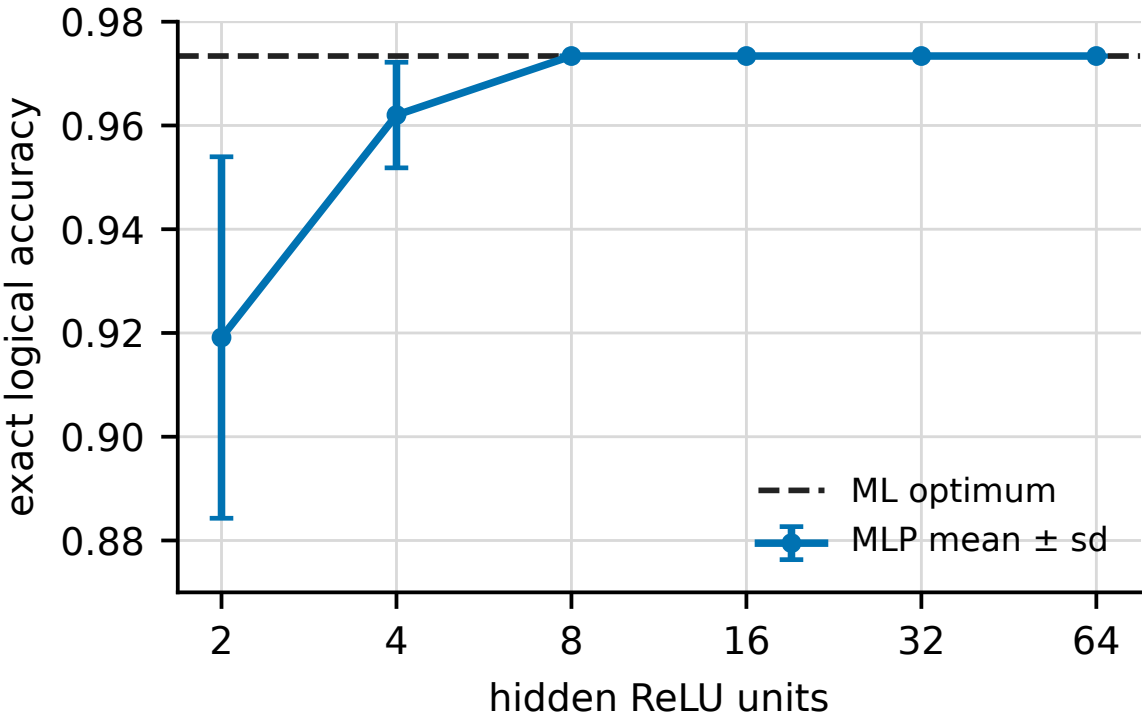


Figure 4: MLP hidden-width ablation at $p = 0.15$ with 2000 samples (mean $\pm$ standard deviation over five paired data/model seeds). Widths of at least eight units reach the exact ML optimum.

4.4. Surface-17: equal under homogeneous noise, different under bias

Surface-17 contains 9 data qubits and 8 measurement ancillas. The code-capacity simulation places errors only on the data qubits. It uses the four Z-type checks that detect bit flips; the four X-type checks would detect phase flips. This gives 16 possible syndromes. Under homogeneous noise at $p = 0.15$, ML and MW make the same decision for every syndrome and have exact accuracy 0.783126. The minimum weights of the two logical classes are unequal for every syndrome, so this agreement does not depend on a tie-breaking rule.

For the biased experiment, checkerboard qubits $\{1, 3, 5, 7, 9\}$ receive $p_{\text{high}} = 0.15$, and the other four receive $p_{\text{low}} = 0.03$. Qubits are numbered row by row on the 3×3 patch. Weighted MW chooses the class containing the most likely individual error. ML instead sums every error probability in each class. Table 3 shows an exact gap of 0.000445 and two different syndrome decisions.

Table 3
Exact Surface-17 accuracy under representative homogeneous and checkerboard-biased noise.

Channel	ML	Weighted MW	Gap	Different decisions
All $p_i = 0.15$	0.783126	0.783126	0	0
$p_{\text{high}} = 0.15$, $p_{\text{low}} = 0.03$	0.888567	0.888121	0.000445	2

4.5. Decode cost and practical trade-offs

Table 4
Asymptotic costs for m syndrome bits, n data qubits, h hidden units, and N_s enumerated errors compatible with a syndrome.

Decoder	Construction	Stored state	Per syndrome
MAP lookup	$O(2^n)$	$O(2^m)$	$O(1)$
Enumerated MW/ML	$O(2^n)$	$O(2^n)$	$O(N_s)$
LR	training dependent	$O(m)$	$O(m)$
One-layer MLP	training dependent	$O(mh + h)$	$O(mh + h)$

Table 4 separates the one-time cost of building or training a decoder from the cost of decoding one syndrome. For these small codes, a precomputed lookup table is already fast because decoding requires only one table lookup. A learned decoder therefore has no speed advantage here. For larger codes, however, a trained model may be more compact: its parameters can represent the decoding rule without storing or searching a large error table.

4.6. Summary of the comparison

The experiments support four conclusions. First, homogeneous noise can hide the difference between aggregation rules in the tested cases, while biased Surface-17 noise gives ML a small but strict measured advantage over weighted MW. Second, LR keeps three syndrome-level errors because its boundary is linear. Third, MLP capacity matters: widths of at least eight reach the optimum here, but narrower models do not do so reliably. Fourth, learned inference replaces explicit error-table storage with a parameterized computation, although a precomputed MAP lookup is still faster for these tiny domains.

5. Related work

Decoding is often treated as an inference problem. Maximum-likelihood decoding is closely linked to statistical-mechanics partition functions [1, 3, 4]. For surface codes, minimum-weight perfect matching remains a standard practical method [1]. The tropical $(\min, +)$ semiring and the ordinary $(+, \times)$

semiring also provide an algebraic way to distinguish minimum-weight from maximum-likelihood aggregation [5].

Machine-learned decoding is another broad research area. The cited examples include stochastic neural, feedforward, and recurrent-transformer decoders, including a model trained with quantum-hardware data [6, 7, 8]. Torlai and Melko present a stochastic neural decoder for a toric code [6]. Varsamopoulos et al. describe small-surface-code decoding as classification and report competitive accuracy and hardware-oriented inference time [7]. Bausch et al. use a recurrent transformer that outperforms algorithmic baselines on hardware data and on simulations up to distance 11 [8].

This paper finds that an expressive nonlinear classifier can reach the analytic decoding optimum in the tested small-code setting. This result is consistent with the larger-scale work described above, which also motivates bounded inference time after training. The LR and width ablations in this paper further show that model capacity and training coverage matter. The experiments do not reproduce the large-code, repeated-syndrome, or hardware-noise advantages of those methods. Instead, they provide exact diagnostics for a small domain, including details that aggregate logical-error rates can hide. Roffe provides a broader survey of the field [9].

6. Limitations and outlook

This study is deliberately narrow: repetition codes and Surface-17 under code-capacity bit-flip noise. The checkerboard-biased channel shows a strict maximum-likelihood advantage, but it is only one transparent pattern of unequal error rates. It does not imply a universal gap size. The natural next tests are larger code distances, calibrated device-specific bias, and circuit-level noise.

Code-capacity noise also assumes perfect syndrome measurements. A more realistic study would include measurement errors and repeated rounds of syndrome extraction. The learned models here are intentionally generic; models that use the code lattice should perform better. The width ablation shows robustness only for this 16-syndrome domain and does not explain how network capacity should scale with code distance.

Within these limits, exact evaluation gives a useful baseline. The same definition and implementation can test somewhat larger codes and richer noise models, but full enumeration scales as $O(2^n)$ and is not a practical large-code decoder.

Declaration on Generative AI

During this work, the author used Anthropic Claude to refine the initial research plan, find learning resources, and organize notes according to the paper guidelines. The author used OpenAI Codex to simplify and reorganize prose, check mathematical and numerical consistency, implement validation and reproduction scripts, generate plots, format LaTeX, and troubleshoot compilation. The author reviewed, edited, and approved the final content and accepts full responsibility for the publication.

References

[1] Eric Dennis, Alexei Kitaev, Andrew Landahl, John Preskill, Topological quantum memory, Journal of Mathematical Physics 43 (2002) 4452–4505. doi:10.1063/1.1499754.

[2] T. E. O'Brien, B. Tarasinski, L. DiCarlo, Density-matrix simulation of small surface codes under current and projected experimental noise, npj Quantum Information 3 (2017) 39. doi:10.1038/s41534-017-0039-x.

[3] Sergey Bravyi, Martin Suchara, Alexander Vargo, Efficient algorithms for maximum likelihood decoding in the surface code, Physical Review A 90 (2014) 032326. doi:10.1103/PhysRevA.90.032326.

[4] Christopher T. Chubb, Steven T. Flammia, Statistical mechanical models for quantum codes with correlated noise, Annales de l'Institut Henri Poincaré D 8 (2021) 269–321. doi:10.4171/AIHPD/105.

[5] Diane Maclagan, Bernd Sturmfels, Introduction to Tropical Geometry, Graduate Studies in Mathematics 161, American Mathematical Society, 2015. doi:10.1090/gsm/161.

[6] Giacomo Torlai, Roger G. Melko, Neural decoder for topological codes, Physical Review Letters 119 (2017) 030501. doi:10.1103/PhysRevLett.119.030501.

[7] Savvas Varsamopoulos, Ben Criger, Koen Bertels, Decoding small surface codes with feedforward neural networks, Quantum Science and Technology 3 (2018) 015004. doi:10.1088/2058-9565/aa955a.

[8] Johannes Bausch, Andrew W. Senior, Francisco J. H. Heras, Thomas Edlich, Alex Davies, Michael Newman, Cody Jones, Kevin Satzinger, Murphy Yuezhen Niu, Sam Blackwell, George Holland, Dvir Kafri, Juan Atalaya, Craig Gidney, Demis Hassabis, Sergio Boixo, Hartmut Neven, Pushmeet Kohli, Learning high-accuracy error decoding for quantum processors, Nature 635 (2024) 834–840. doi:10.1038/s41586-024-08148-8.

[9] Joschka Roffe, Quantum error correction: an introductory guide, Contemporary Physics 60 (2019) 226–245. doi:10.1080/00107514.2019.1667078.

A. Reproducibility and validation details

The analytic results use exact enumeration and therefore have no sampling error. The learned models use the settings in Table 5; the learning curve averages over five seeds.

Table 5
Reproducibility parameters.

Parameter	Value
Codes	repetition $n \in \{3, 5\}$; rotated surface $[[9, 1, 3]]$
Noise channel	homogeneous p; checkerboard bias $(0.15, 0.03)$
p grid	15 points, linspace$(0.02, 0.30)$
Accuracy	exact (sum over all 2^n errors)
Main training-data seed	`numpy default_rng(7)`
Training samples (main)	2000 per p
Logistic regression	`sklearn`, `max_iter=1000`
MLP architecture	1 hidden layer, 32 ReLU units
MLP solver	`sklearn MLPClassifier`, `max_iter=2000`
MLP seed (main)	`random_state=0`
Learning-curve sizes	$\{50, 100, 200, 500, 1000, 2000, 5000\}$
Learning/width seeds	$\{0, 1, 2, 3, 4\}$ (data and model)
Widths tested	$\{2, 4, 8, 16, 32, 64\}$ ReLU units
Numerics	`numpy 2.4.6`; `scipy 1.17.1`
ML library	`scikit-learn 1.9.0`
Plotting	`matplotlib 3.11.1`

The Surface-17 check layout follows O'Brien et al. [2]. It contains nine data qubits and eight measurement ancillas. We check the layout with `listing_surface.py`: the X and Z checks commute, each check matrix has rank 4, the code has $k = 9 - 4 - 4 = 1$, the representative logical operators anticommute, and both logical distances are 3. Data qubits are numbered row by row. The complete CSS check matrices are

$$H_Z = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}, \qquad H_X = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Representative logical supports are $\mathrm{supp}(\overline{X}) = \{3, 6, 9\}$ and $\mathrm{supp}(\overline{Z}) = \{1, 2, 3\}$. For bit-flip decoding, H_Z produces 16 syndromes and overlap with $\overline{Z}$ labels the two logical classes. No syndrome has equal minimum class weights. In the biased experiment, qubits $\{1, 3, 5, 7, 9\}$ have $p_i = 0.15$, while $\{2, 4, 6, 8\}$ have $p_i = 0.03$. Exact enumeration gives accuracies 0.888567 for ML and 0.888121 for weighted MW.

B. Core exact-evaluation code

Listing 1 gives the central evaluation routine. It enumerates every possible error and weights it with its exact channel probability.

Listing 1: Exact logical accuracy by full enumeration.

```python
from itertools import product

import numpy as np
from scipy.special import logsumexp


def channel_probabilities(p, n):
    """Return one physical flip probability per data qubit."""
    probabilities = np.asarray(p, dtype=float)
    if probabilities.ndim == 0:
        probabilities = np.full(n, float(probabilities))
    if probabilities.shape != (n,):
        raise ValueError(f"expected one probability or {n} probabilities")
    if np.any((probabilities <= 0) | (probabilities >= 1)):
        raise ValueError("flip probabilities must lie strictly between 0 and 1")
    return probabilities


def exact_accuracy(H, L, n, decode, p):
    """Exact logical accuracy: sum P(e) over all errors the decoder
    classifies into the correct logical class. No Monte-Carlo sampling."""
    probabilities = channel_probabilities(p, n)
    lp, lq = np.log(probabilities), np.log1p(-probabilities)
    terms = []
    for bits in product((0, 1), repeat=n):
        e = np.array(bits, dtype=np.int8)
        s = tuple((H @ e) % 2)
        if decode(s) == int((L @ e) % 2):     # correct logical class?
            terms.append(float(e @ lp + (1 - e) @ lq))  # log P(e)
    return float(np.exp(logsumexp(terms))) if terms else 0.0
```