DISTANCE PREDICTION FOR QUANTUM CODES

Anonymous authors
Paper under double-blind review

ABSTRACT

The distance of a quantum code is a fundamental metric that dictates its error-correction capability. While quantum low-density parity-check (qLDPC) families such as Bivariate Bicycle (BB) codes offer a promising pathway to low-overhead fault-tolerant quantum computing, computing their distance is challenging. Current exact solvers and even search heuristics that provide a distance upper bound suffer from severe computational bottlenecks, restricting the discovery of optimal codes. In this paper, we construct a dataset of BB codes with corresponding distance upper bounds obtained through high-budget heuristic searches. We introduce BB-Net, an attention-based neural network architecture for predicting the distance of BB codes. It achieves 96.39% accuracy on the reference distances in the test set, compared with 94.86% for state-of-the-art heuristics with a budget of up to 10^4 times more computational runtime. On a subset with certified exact distances, BB-Net achieves 98.94%. By relieving the distance-calculation bottleneck, BB-Net enables faster screening of large code spaces, particularly in ML-driven search pipelines, and thereby can accelerate the discovery of novel quantum error-correcting codes, an important capability for advancing scalable fault-tolerant quantum computing.

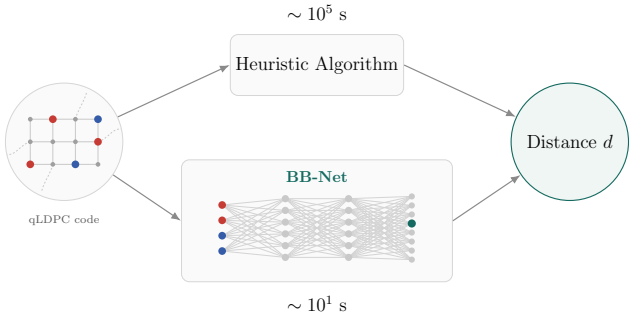

Figure 1: Distance estimation for bivariate bicycle codes through two approaches: computationally intensive heuristic search and orders of magnitude more efficient inference using BB-Net. Numbers refer to a full benchmark evaluation.

1 INTRODUCTION

Quantum computing offers the potential to solve computational problems that are intractable for classical methods (Steane, 1998; Preskill, 2012), with landmark algorithms such as Shor's factoring algorithm (Shor, 1994) and Grover's search algorithm (Grover, 1996) providing prominent examples of quantum computational advantage. Recent experimental advances across multiple quantum-computing platforms have further demonstrated this potential (Arute et al., 2019; Bharti et al., 2022; Bluvstein et al., 2024; Huang et al., 2022; Bennett & Brassard, 2014; Bao et al., 2023). However, realizing practical, large-scale quantum computation requires overcoming a fundamental challenge: the inherent fragility of physical qubits. Highly susceptible to decoherence and operational errors (Burnett et al., 2019; Etxezarreta Martinez et al., 2021), physical qubits cannot reliably preserve quantum information for the duration of complex calculations.

Quantum Error Correction (QEC) is therefore indispensable for bridging the gap between noisy physical hardware and reliable computation (Shor, 1995; Terhal, 2015). Recently, there has been sig-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

nificant interest in the development of quantum Low-Density Parity-Check (qLDPC) codes, which hold the promise of substantially reducing the resource overhead required for fault-tolerant quantum computation (Panteleev & Kalachev, 2021; Breuckmann & Eberhardt, 2021). Within this class, Bivariate Bicycle (BB) codes have emerged as a highly promising family, offering a practical pathway to high-threshold, low-overhead fault tolerance that maps efficiently to physical hardware constraints (Bravyi et al., 2024), and recently, perturbed bivariate bicycle (PBB) codes have been introduced as a non-CSS ansatz that augments the BB polynomial pairs (Cruz-Benito et al., 2026).

The efficacy of any QEC protocol is anchored by a fundamental figure of merit: its minimum distance, d. The distance of a QEC code is defined as the minimum number of physical errors required to cause an undetectable logical failure. It provides a rigorous, decoder-agnostic guarantee of a code's ability to suppress errors. Consequently, discovering codes with optimal distances is a primary objective when designing new fault-tolerant architectures. However, the minimum-distance problem for quantum stabilizer codes is computationally hard: the corresponding decision problem is NP-complete, and exact distance computation is therefore NP-hard (Kapshikar & Kundu, 2023; Grigorescu et al., 2025). The analogous hardness result for classical linear codes was established by Vardy (1997).

To evaluate code distance, researchers traditionally rely on exact solvers, such as Mixed Integer Linear Programming (MILP) (Punekar et al., 2010), or search heuristics such as QDistRnd (Pryadko et al., 2023), which provide an upper bound. Exact methods scale exponentially and become computationally intractable even for moderate code sizes. While heuristics offer a relative speedup compared to exact solvers, they remain far from real-time execution. Consequently, evaluating the thousands or millions of candidates encountered in searches over qLDPC code families remains severely constrained by distance-estimation cost (Webster et al., 2026). A common strategy is therefore to perform distance screening: limited computational budget is first used to filter candidates, after which computationally intensive heuristics or exact solvers are applied only to the most promising codes to tighten bounds or obtain certificates (Qian & Li, 2026; Yan et al., 2026; Cruz-Benito et al., 2026). However, the effectiveness of such pipelines remains limited by the runtime of existing distance-estimation methods, leaving large portions of the design space unexplored.

In this paper, we address this critical bottleneck by introducing BB-Net, an attention-based neural network architecture for both BB and PBB codes. We propose a compact tokenization scheme that exploits the algebraic structure of BB codes, enabling BB-Net to learn informative latent representations for distance prediction. By relieving this bottleneck, BB-Net enables fast evaluation of large code spaces, particularly in ML-driven search pipelines, and thereby can accelerate the discovery of novel QEC codes, an important capability for advancing scalable fault-tolerant quantum computing.

Our main contributions are summarized as follows:

- We frame distance estimation for BB codes as a machine-learning problem: a network is trained on labels from expensive heuristic searches and predicts them at a small fraction of the cost. To our knowledge, this is the first work to predict quantum-code distance directly with a learned model.
- We present a designated dataset of approximately 58,000 code examples of both CSS BB and non-CSS PBB types with a corresponding distance upper bound.
- We propose a highly compressed input representation for the codes and an attention-based sequence neural network architecture, BB-Net, for this task.
- We show that BB-Net's accuracy on the test benchmark is 96.39% compared to 94.86% by state-of-the-art heuristics with a budget of up to 10^4 more computational runtime, and it achieves 98.94% on certified exact distance subset test codes.

2 Related Work

Quantum-code distance can be computed or bounded by a variety of algebraic, combinatorial, optimization, and randomized techniques. Exact methods search systematically enough to certify that no logical operator exists below a given weight (Dumer et al., 2017). These include codeword-enumeration methods such as Brouwer–Zimmermann (Grassl, 2006; Hernando et al., 2026), error-enumeration methods such as bipartition matching and linked-cluster search (Dumer et al., 2014), and solver-based formulations using mixed-integer programming (Bestuzheva et al., 2021) or SAT/MaxSAT (Chen et al., 2026). Heuristic methods can provide a powerful alternative

and substantially faster than the exact methods, however, they only satisfy an upper bound to the distance. The methods include random-information-set such as QDistRnd (Pryadko et al., 2023), that uses random information-set sampling and Gaussian elimination to search for low-weight logical operators, evolutionary methods such as QDistEvol (Webster & Browne, 2025) that replaces random sampling with an evolutionary search over column permutations, syndrome-decoder-based distance searches using BP-OSD (Roffe et al., 2020), and Stim-based methods such as GEStim, CCStim, and UEStim that search for undetectable logical errors, targeting increasingly general error structures and detector degrees (Gidney, 2021).

Most machine-learning work on error-correcting codes addresses decoding: a model observes a noisy word or syndrome and predicts an error or corrected codeword Choukroun & Wolf (2022); Krenn et al. (2023); Wang & Tang (2024); deMarti iOlius et al. (2024); Varsamopoulos et al. (2018; 2019); Liu & Poulin (2019); Harper et al. (2020); Maskara et al. (2019); Meinerz et al. (2022). Some decoders are based on reinforcement learning Colomer et al. (2020); Sweke et al. (2021); Fitzek et al. (2020); Veeresh et al. (2024); Andreasson et al. (2019), others on GNN (Lange et al., 2025), transformers (Choukroun & Wolf, 2024; Bausch et al., 2024; Senior et al., 2025; Park et al., 2025; Zenati & Nachmani, 2026; Peled et al., 2026; Blue et al., 2026). Beyond decoding, recent works have been discovering new practical QEC codes with LLMs (Qian & Li, 2026; Yan et al., 2026; Cruz-Benito et al., 2026).

3 Background

3.1 Quantum Error-Correcting Codes

A classical linear code encodes k information bits into n binary symbols by imposing parity constraints through a binary matrix H (Hamming, 1950). Valid codewords satisfy $Hx^\top = 0$, while an error e produces the syndrome $s = He^\top$. Quantum error correction follows the same syndrome-based principle but within a two-level systems described by a state vector $|\psi\rangle$ (Shor, 1995; Wootters & Zurek, 1982), as a qubit can exist in coherent superpositions:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \qquad \text{where } \alpha, \beta \in \mathbb{C}, \quad |\alpha|^2 + |\beta|^2 = 1. \tag{1}$$

Quantum errors are expressed using the Pauli basis $\{I, X, Y, Z\}$, representing identity, bit-flip, combined bit-and-phase-flip, and phase-flip operations. An n-qubit Pauli error is a tensor product of these operators, and its weight is the number of qubits on which it acts nontrivially.

The stabilizer formalism (Gottesman, 1997) provides a convenient algebraic description of quantum codes. An $[[n, k, d]]$ stabilizer code is defined by a commuting set of Pauli operators within the Hilbert space $\mathcal{H}_2^n$ whose common $+1$ eigenspace forms the codespace. The stabilizers generate an Abelian group $\mathcal{S}$ that does not contain $-I$. A group with $n-k$ independent generators defines a 2^k-dimensional codespace and therefore encodes k logical qubits. Measuring the generators produces a binary syndrome indicating which stabilizers anticommute with an error, without revealing the encoded logical state. In CSS codes (Calderbank & Shor, 1996; Steane, 1996a;b), the stabilizers are separated into X and Z type checks described by binary matrices H_X and H_Z, which satisfy

$$H_X H_Z^\top = 0 \pmod 2. \tag{2}$$

This condition ensures that all X and Z type stabilizers commute. For an error represented by binary X and Z components (e_X, e_Z), the corresponding syndromes are $s_X = H_X e_Z^\top$ and $s_Z = H_Z e_X^\top$.

3.2 Quantum Code Distance Finding

Computing the distance of a quantum code amounts to finding the minimum-weight nontrivial logical operator. Let $\mathcal{N}(\mathcal{S})$ denote the set of Pauli operators that commute with every element of the stabilizer group $\mathcal{S}$. The distance is formally

$$d = \min_{L \in \mathcal{N}(\mathcal{S}) \setminus \mathcal{S}} \mathrm{wt}(L). \tag{3}$$

where $\mathrm{wt}(L)$ denotes the Pauli weight of L, i.e., the number of physical qubits on which L acts nontrivially with an X, Y, or Z operator rather than the identity. Thus, a logical operator preserves the codespace but acts nontrivially on the encoded information. A distance-d code detects every error of weight less than d and corrects every error of weight at most $\lfloor (d-1)/2 \rfloor$. For a CSS code, the X and Z distances can be computed independently, with the overall distance given by

$d = \min(d_X, d_Z)$. The decision version of the problem asks whether a nontrivial logical operator of weight at most a given threshold exists. This problem is NP-complete for general stabilizer codes, and the exact distance computation is therefore NP-hard (Kapshikar & Kundu, 2023). Distance-finding algorithms broadly trade computational cost for the strength of the guarantee they provide (Webster et al., 2026). A completed exact search certifies the distance, whereas an incomplete or heuristic search generally provides only a bound. During the search, such algorithms may maintain both a lower bound d_{lower} and an upper bound d_{upper}. When $d_{\text{lower}} = d_{\text{upper}}$, the distance is certified exactly. However, exact methods may require exhaustive runtimes and are frequently terminated before convergence on larger codes, in which case only partial bounds are available. Heuristic methods instead search for low-weight logical operators without certifying that no lower-weight operator exists. For a CSS code with parity-check matrices H_X and H_Z, a candidate Z-type witness $\mathbf{v} \in \mathbb{F}_2^n$ is accepted only if

$$H_X \mathbf{v}^\top = 0, \qquad \mathbf{v} \notin \text{rowspan}(H_Z), \qquad \text{wt}(\mathbf{v}) = \hat{d}, \tag{4}$$

and analogously an X-type witness must satisfy the corresponding conditions with H_X and H_Z interchanged. A heuristic result is considered invalid if any of these conditions fail.

3.3 Bivariate Bicycle Codes

Bivariate Bicycle (BB) codes are a family of quantum low-density parity-check (qLDPC) codes introduced in their present form by Bravyi et al. (2024). They use a two-dimensional cyclic structure and provide a compact algebraic description through sparse bivariate polynomials. BB codes are attractive because they combine sparse constant-weight stabilizer checks with comparatively high encoding rates, making them promising candidates for low-overhead fault-tolerant quantum memories.

A BB code is defined over the finite Abelian group $\mathbb{Z}_\ell \times \mathbb{Z}_m$ and can be represented using two sparse polynomials

$$A(x, y) = \sum_{(a,b)\in S_A} x^a y^b, \qquad B(x, y) = \sum_{(a,b)\in S_B} x^a y^b, \tag{5}$$

where S_A and S_B are sets of exponent pairs. The periodic relations $x^\ell = 1$ and $y^m = 1$ make the polynomial coordinates cyclic. Each monomial $x^a y^b$ corresponds to a two-dimensional cyclic shift, and the resulting $\ell m \times \ell m$ binary matrices A and B define the CSS parity check matrices

$$H_X = [AB], \qquad H_Z = [B^\top A^\top]. \tag{6}$$

The code therefore has block length $n = 2\ell m$. The construction typically use sparse weight-three polynomials, so each stabilizer has constant weight six. This gives BB codes their LDPC structure while retaining a highly regular algebraic organization. Different choices of (ℓ, m) and of the polynomial supports S_A and S_B generate different codes with different values of k and d, creating a large design space that must be searched efficiently. Our study also includes non-CSS Perturbed Bivariate Bicycle (PBB) (Cruz-Benito et al., 2026) codes, which extend this construction using four sparse bivariate polynomials (A, B, C, D) and do not admit the same separation into purely X and Z type parity-check matrices:

$$H = \begin{bmatrix} A & B & C & D \\ 0 & 0 & B^\top & A^\top \end{bmatrix}, \tag{7}$$

where the first stabilizer has both X-type (A,B) and Z-type (C,D) components and the second is purely Z-type.

4 Method

We propose BB-Net, an attention-based sequence architecture that consists of two complementary encoding branches: a polynomial encoder, which captures the local algebraic structure of the code specification, and a Tanner spectral encoder, which captures global structural properties of the corresponding Tanner graph. The code is first tokenized into a compact cyclic representation that preserves its underlying algebraic structure while avoiding the need to explicitly construct large parity-check matrices, and then proceeds through an embedding block. The latent representations produced by the two encoders are subsequently combined through a gate and passed to a distance-classification head, which predicts the code distance. Architecture is described in Figure 2.

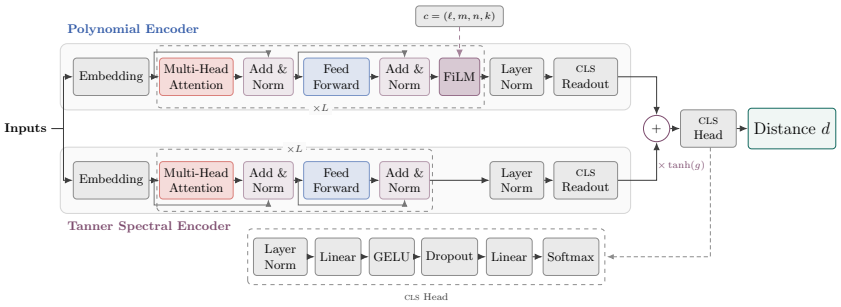

Figure 2: **BB-Net architecture**. A BB code is processed by two encoders; The polynomial encoder applies self-attention to monomial tokens and is conditioned on the code parameters through FiLM layers. In parallel, the Tanner spectral encoder processes Fourier-character tokens using attention as well. Each branch produces a code latent representation, which is combined through a learned gated residual connection. The fused representation then passed to a classification head.

4.1 Code Tokenization and Embedding

BB-Net constructs two complementary token representations from the sparse polynomial specification of each code: (i) polynomial tokens, which describe the local algebraic relations between monomials, and (ii) Tanner spectral tokens, which provide a global representation of the corresponding cyclic Tanner structure. Both representations are generated directly from the polynomial supports, without expanding the code into its full parity-check matrices. These token representations are subsequently mapped to continuous feature spaces by the encoder's embedding block, and learned CLS tokens are used as latent vector aggregators for the downstream classification task (Figure 3).

Polynomial tokens. Given the support sets S_A, S_B (and S_C, S_D for PBB codes), we first create one monomial token for every exponent pair appearing in the code. Each monomial token is associated with its polynomial identity,

$$\mathcal{T}_{\text{monomial}} = \{(P, a, b) : (a, b) \in S_P,\ P \in \{A, B, C, D\}\}. \tag{8}$$

For a monomial token (P, a, b), the component identity $P \in \{A, B, C, D\}$ is embedded by a one-hot vector. The a and b components are embedded as cyclic coordinates without imposing a linear ordering on the torus. For harmonic index $r = 1, \ldots, 8$, we compute

$$\boldsymbol{\phi}(a, b) = \left[\sin\left(\frac{2\pi r a}{\ell}\right), \cos\left(\frac{2\pi r a}{\ell}\right), \sin\left(\frac{2\pi r b}{m}\right), \cos\left(\frac{2\pi r b}{m}\right)\right]_{r=1}^{8} \in \mathbb{R}^{32}. \tag{9}$$

This allows the network to distinguish identical cyclic coordinates appearing in different components of the code (Tancik et al., 2020).

Tanner spectral tokens. The second representation exploits the translational symmetry of the BB Tanner graph. Tanner graphs provide the standard bipartite representation of sparse parity constraints (Tanner, 1981); for BB codes, the additional cyclic structure makes it possible to work with the associated group lift rather than explicitly materializing all Tanner-graph nodes and edges. Fourier-domain representations have similarly been used to analyze the structure of quasi-cyclic LDPC codes constructed from circulant matrices (Diao et al., 2012), while the spectra of group-based graph lifts can be decomposed through representations of the underlying group (Agarwal et al., 2019). Specifically, BB-Net represents the cyclic lift in the character domain. Since $G = \mathbb{Z}_\ell \times \mathbb{Z}_m$ is finite and Abelian, its dual group contains ℓm one-dimensional characters. For $(u, v) \in \mathbb{Z}_\ell \times \mathbb{Z}_m$,

$$\chi_{u,v}(a, b) = \exp\left[-2\pi i \left(\frac{ua}{\ell} + \frac{vb}{m}\right)\right]. \tag{10}$$

We construct one spectral token for every character $\chi_{u,v}$. For each character, the sparse polynomial supports are evaluated as

$$\widehat{P}(\chi_{u,v}) = \sum_{(a,b)\in S_P} \chi_{u,v}(a, b), \qquad P \in \{A, B, C, D\}. \tag{11}$$

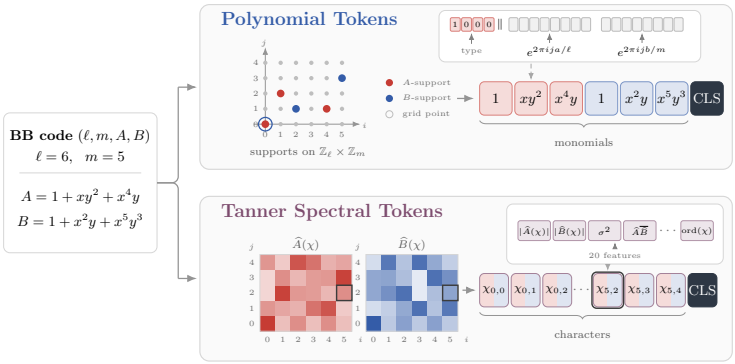

Figure 3: **BB-Net tokenization and embedding.** A BB code specified by sparse polynomials A and B is mapped to two complementary representations: (i) polynomial tokens encode monomial supports on $\mathbb{Z}_\ell \times \mathbb{Z}_m$ with the type (A,B) indicator as a one-hot vector and the first eight Fourier harmonics, represented by sine and cosine components along each lattice dimension, yielding a 32-dimensional positional representation. (ii) Tanner spectral tokens encode the responses of the full polynomials at each group character χ using 18-dimensional character features. Learned CLS tokens are used as latent vector aggregators for the downstream classification task.

Thus, each character indexes a particular Fourier mode of the cyclic lift, and the corresponding responses summarize how the local polynomial connections combine at that mode. Each character χ is embedded by a compact 18-dimensional feature vector whose entries are derived from the responses evaluated at that same character. These include normalized polynomial magnitudes such as $|\widehat{A}(\chi)|$ and $|\widehat{B}(\chi)|$, spectral-energy terms, cross-polynomial interactions such as $\widehat{A}(\chi)\widehat{B}(\chi)^*$, relative spectral differences, and invariant properties of the character itself. The spectral sequence contains $|G| = \ell m = \frac{n}{2}$ tokens, including the trivial character. A character is therefore identified by its cyclic invariants and by the responses of the complete polynomial specification at that character, rather than by an arbitrary flattened Tanner-node index.

4.2 Architecture

Each branch uses a stack of L blocks of multi-head self-attention (Vaswani et al., 2017) and feed-forward with residual connections and layer normalization. The polynomial branch is additionally conditioned on the code's information $\mathbf{c} = (\ell, m, n, k)$ with FiLM (Perez et al., 2018), an MLP that maps $\mathbf{c}$ to feature-wise scale and shift vectors. Through self-attention, the branch-specific CLS tokens aggregate the two token sequences into latent vectors, $\mathbf{h}_{\text{polynomial}}, \mathbf{h}_{\text{tanner}} \in \mathbb{R}^{d_{\text{model}}}$, which are subsequently combined through a learned residual gate: $\mathbf{h}_{\text{fuse}} = \mathbf{h}_{\text{polynomial}} + \tanh(\mathbf{g}) \odot \mathbf{h}_{\text{tanner}}$, where $\mathbf{g}$ is learned jointly with the network. This parameterization retains a direct polynomial path while allowing the model to use global spectral information.

The classification head follows layer normalization with a linear layer, GELU activation, dropout, another linear projection, and a softmax. Let $\mathcal{D} = \{d_{\min}, \ldots, d_{\max}\}$ denote the effective classes, the output layer's length is $d_{\max} - d_{\min} + 1$ and the model is trained with a categorical cross-entropy over these classes.

4.3 Canonicalization and Training Augmentation

Canonicalization. Shift canonicalization maps each input to a deterministic representative of its cyclic-shift equivalence class. Let $S_P \subseteq \mathbb{Z}_\ell \times \mathbb{Z}_m$ be the support of a polynomial and let $\mathbf{s} = (s_1, s_2)$. We define the translated support as

$$\tau_{\mathbf{s}}(S_P) = \{((a - s_1) \bmod \ell,\ (b - s_2) \bmod m) : (a, b) \in S_P\}\,. \tag{12}$$

This transformation is equivalent to multiplying the corresponding polynomial by the monomial $x^{-s_1} y^{-s_2}$, with the exponents evaluated modulo (ℓ, m), thus, inducing a cyclic row or column permutations and therefore does not change the code parameters, including its distance. For each polynomial, every monomial is used in turn as an anchor, and the support is cyclically shifted so that the anchor moves to the origin $(0, 0)$. The resulting exponent pairs are sorted, and the lexicographically smallest representation is selected as the canonical form (Feulner, 2009). For PBB codes we apply a single translation to all polynomials.

Augmentation. Each code is augmented using randomly sampled distance-preserving symmetries of its algebraic representation (Chen et al., 2020). Let $G = \mathbb{Z}_\ell \times \mathbb{Z}_m$. For a CSS BB code, we sample an automorphism $\varphi \in \mathrm{Aut}(G)$ (Hillar & Rhea, 2007) and apply it simultaneously to the exponent coordinates of both polynomial supports. Specifically, each monomial $x^a y^b$ is mapped to $x^{a'} y^{b'}$, where $(a', b') = \varphi(a, b)$. In addition, independent uniformly sampled cyclic translations are applied to S_A and S_B, and the two polynomial registers may be interchanged, that is, $(A, B) \longmapsto (B, A)$. For a non-CSS PBB code, the same sampled automorphism is applied simultaneously to all four polynomial supports. The register interchange is $(A, B, C, D) \longmapsto (B, A, D, C)$, and a single uniformly sampled cyclic translation is applied simultaneously to S_A, S_B, S_C, and S_D.

5 Dataset

We construct the corpus from CSS BB codes and non-CSS PBB extensions. We restrict the corpus to $n = 2\ell m \leq 360$ with $\ell, m \leq 30$. Candidate generation covers separable, coordinate-swapped, mixed-monomial, trinomial, and adversarial BB specifications together with valid PBB expansions. Then, candidate specifications are reduced modulo (ℓ, m), with coincident monomials canceled over $\mathbb{F}_2$. We keep only candidates with at least two surviving terms in A, B. for PBB codes, C and D must remain nonempty. We then exclude codes with zero logical dimension, $k = 0$. For PBB candidates, we further require the stabilizer generators to commute, verified through the symplectic commutation condition $H_X H_Z^\top + H_Z H_X^\top = 0 \pmod 2$. Finally, duplicate specifications are removed, including symmetry-equivalent.

To find the distance of each code, we apply the BP-OSD, QDistRnd, and QDistEvol heuristics, using a large computational budget of 10,000 trials per code. The smallest distance identified across the three methods is considered the target distance. Since these heuristic methods do not provide certificates, the resulting values are upper bounds on the true code distance. The final dataset contains 58,704 examples with upper-bound distances ranging from 2 to 25, and is partitioned into training, validation and test sets using a 80:10:10 split. We ran the exact solver MILP (Punekar et al., 2010) on all 5,870 test codes with a time limit of 2 hours per code; it certified the distance of 2,366 of them. On every certified code, the 10,000-trial heuristic label equals the certified distance.

6 Experimental Results

To evaluate BB-Net, we compare its performance against six distance-finding heuristic methods; QDistRnd (Pryadko et al., 2023), QDistEvol (Webster & Browne, 2025), BP-OSD (Roffe et al., 2020), GEStim, CCStim, and UEStim (Gidney, 2021), with a budget of up to 10^4 times more computational runtime for the benchmark (subsection 6.1). Then, we evaluate the methods on certified and high upper bound codes (subsection 6.2). AUC for distance screening is reported (subsection 6.3), and finally, we conduct ablation experiments to justify our architecture design (subsection 6.4). Experiments' configurations as well as more experimental analysis are in Appendix A.

6.1 Test Benchmark

Table 1 presents results on the test benchmark. To reliably identify the correct distance regime, which is particularly relevant for comparing candidates' codes, accuracy is measured using a tolerance of ± 2 around the reference distance. BB-Net achieves the highest accuracy for the referenced distances, reaching 96.39% with the lowest RMSE of 1.131. The heuristics methods require repeated stochastic searches, denoted as trials per code. BB-Net evaluates a code in 0.005 s, compared with 1.0 s for QDistEvol, 3.3 s for QDistRnd, and 19.1 s for BP-OSD. The Stim-based methods exhibit substantially lower accuracy and, in the case of GEStim, rarely return a solution. Overall, the results demonstrate that BB-Net achieves high accuracy with substantially better efficiency, enabling near-real-time distance estimation.

Table 1: Test benchmark across common heuristic methods (5,870 codes)

	QDistRnd	QDistEvol	BP-OSD	GEStim	CCStim	UEStim	BB-Net (ours)
Trials Per Code	200	200	200	1	1	3	1
Time Per Code [s] (↓)	3.3	1.0	19.1	1.2	1.2	13.1	**0.005**
Total Time [s] (↓)	19,383	6,057	112,275	6,940	7,164	76,775	**26.8**
RMSE (↓)	2.319	1.485	23.322	4.950	9.309	1.933	**1.131**
Accuracy (↑)	92.20%	94.86%	82.23%	0.02%	34.60%	61.23%	**96.39%**

6.2 Certified and High Upper Bound Codes

Of the 5,870 test codes, the MILP-based exact solver certified the code distance for 2,366 instances. The remaining instances were substantially more difficult to certify and generally exhibited larger heuristic upper bounds. None of the certified codes had a target distance greater than 14. We consequently report results separately for exact codes and for the upper bounds codes. On the certified subset (Table 2), BB-Net's accuracy is 98.94% compared to 100% for QDistEvol, QDistRnd, and BP-OSD. UEStim with 98.44%, CCStim with 64.03% and GEStim with 0.04%. On the unresolved upper bound subset (Table 3), BB-Net achieves an accuracy of 94.66% and an RMSE of 1.380, compared to 91.38% accuracy and an RMSE of 1.923 for the strongest heuristic baseline, QDistEvol. These results indicate that BB-Net is particularly effective on the high-targets upper bound distance instances for which both exact solvers and heuristics struggle. Accuracy across distances for the different subsets is presented in Figure 4. Selected challenging codes are presented in Table 7.

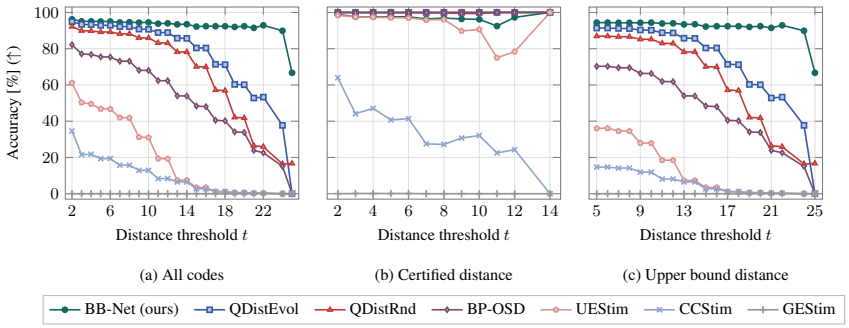

Figure 4: The x-axis denotes the minimum distance threshold t, and the y-axis reports the accuracy computed over codes whose distance satisfies $d \geq t$. Thus, moving to the right restricts the evaluation to increasingly high-distance codes. This was examined on three subsets: the whole benchmark, certified exact distances, and upper bound, unresolved distances.

Table 2: Certified distance codes, $d \leq 14$ (2,366 codes)

	QDistRnd	QDistEvol	BP-OSD	GEStim	CCStim	UEStim	BB-Net (ours)
Trials Per Code	200	200	200	1	1	3	1
Time Per Code [s] (↓)	5.0	1.6	3.8	1.0	1.0	2.1	**0.005**
Total Time [s] (↓)	11,887	3,883	9,032	2,406	2,407	4,898	**10.8**
RMSE (↓)	**0.000**	**0.000**	0.699	11.453	7.434	2.641	0.594
Accuracy (↑)	**100.00%**	**100.00%**	99.96%	0.04%	64.03%	98.44%	98.94%

Table 3: Upper bound distance codes (3,504 codes)

	QDistRnd	QDistEvol	BP-OSD	GEStim	CCStim	UEStim	BB-Net (ours)
Trials Per Code	200	200	200	1	1	3	1
Time Per Code [s] (↓)	2.1	0.6	29.5	1.3	1.4	20.5	**0.005**
Total Time [s] (↓)	7,496	2,175	103,243	4,535	4,758	71,877	**16.0**
RMSE (↓)	3.002	1.923	30.181	42.355	35.362	37.190	**1.380**
Accuracy (↑)	86.93%	91.38%	70.26%	0.00%	14.73%	36.10%	**94.66%**

6.3 Distance Screening

A common and crucial strategy in code-searching pipelines is distance screening: limited computational budget is first used to filter candidates above a certain potential distance, after which computationally intensive heuristics or exact solvers are applied only to the most promising codes to tighten bounds or obtain certificates. To evaluate BB-Net's suitability for distance screening, we compute ROC-AUC at distance thresholds t, using $P(d \geq t)$ as the ranking score and measuring how well codes above and below each threshold are separated (Table 4). BB-Net achieves slightly below QDistEvol and QDistRnd at intermediate thresholds and comparable at the highest ones, while requiring only a fraction of their inference time (reported in Table 1).

Table 4: AUC for distance screening over the test benchmark. Best value in each column in bold, second best underlined (higher is better).

	$d \geq 4$	$d \geq 8$	$d \geq 10$	$d \geq 12$	$d \geq 14$	$d \geq 16$	$d \geq 18$	$d \geq 20$	$d \geq 22$	$d \geq 24$
BB-Net	**1.000**	<u>0.998</u>	0.996	0.993	0.989	<u>0.989</u>	0.988	0.987	<u>0.990</u>	<u>0.991</u>
QDistEvol	**1.000**	**1.000**	**1.000**	**1.000**	<u>0.999</u>	**0.998**	**0.998**	**0.995**	**0.991**	**0.992**
QDistRnd	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.998**	<u>0.997</u>	<u>0.993</u>	**0.991**	0.990
BP-OSD	**1.000**	**1.000**	**1.000**	<u>0.997</u>	0.987	0.967	0.950	0.936	0.931	0.926
UEStim	0.996	0.996	<u>0.997</u>	0.992	0.974	0.944	0.917	0.900	0.890	0.883
GEStim	<u>0.998</u>	0.946	0.913	0.892	0.876	0.871	0.855	0.849	0.845	0.845
CCStim	0.991	0.973	0.934	0.924	0.908	0.909	0.895	0.889	0.884	0.893

6.4 Ablation

To validate the proposed architectural design choices, we conduct a comparison of model variants and present them in Table 5 and Figure 5. Specifically, we train them on the train set and evaluate their NLL on the validation set. We find that BB-Net with a tanner spectral branch, achieves NLL of 1.039 compare to 1.496 without it. Augmentation, shift-canonicalization and FiLM conditioning significantly contribute to the results, achieving NLL of 0.683. Unless stated otherwise, throughout the paper we use the term BB-Net to refer to the best-performing configuration.

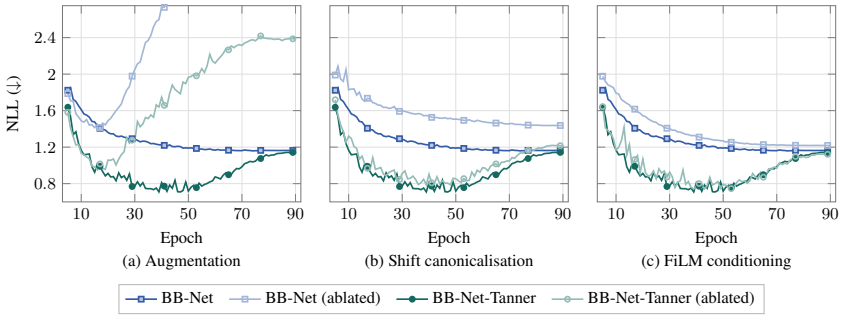

Figure 5: Bold curves denote the best configuration, including augmentation, shift canonicalisation, and FiLM conditioning; lighter curves show the corresponding model with the component named in each panel removed.

Table 5: Ablation study of BB-Net on the validation set. Best-epoch NLL is reported.

Component			NLL ↓	
Aug.	Shift-Canon	FiLM	BB-Net	BB-Net-Tanner
×	×	✓	1.496 ± 0.014	1.039 ± 0.019
✓	×	✓	1.436 ± 0.014	0.762 ± 0.012
×	✓	✓	1.409 ± 0.013	0.924 ± 0.026
✓	✓	×	1.217 ± 0.006	0.712 ± 0.025
✓	✓	✓	1.157 ± 0.001	$\mathbf{0.683 \pm 0.010}$

7 Conclusion

In this work, we reframed the problem of quantum-code distance computation as an ML task. We introduced BB-Net, an attention-based architecture for predicting the distance of qLDPC codes from the BB family. BB-Net combines polynomial and Tanner-spectral representations of the sparse algebraic structure of BB codes and is trained on a dedicated dataset of upper bound distance-labeled instances by high-budget heuristics. Our test benchmark experiments show that BB-Net recovers the referenced distances more closely than the same heuristics with a budget of up to 10^4 times more computational runtime, and that it succeeds on certified distances as well. These results establish learned distance prediction as a practical alternative to repeated computationally expensive distance estimation and enable substantially faster screening of candidate codes. By alleviating one of the primary computational bottlenecks in code search, BB-Net can accelerate the exploration and discovery of high-quality QEC codes, where large-scale searches currently rely heavily on costly heuristic distance evaluations.

AI USE STATEMENT

Generative AI tools were used to assist with writing and improving clarity, and to assist with the retrieval and discovery of finding relative work. The scientific ideas, methodology, experimental design, results, and conclusions were developed and validated by the authors. All parts where AI tools were involved were reviewed, edited, and verified by the authors.

REFERENCES

Naman Agarwal, Karthekeyan Chandrasekaran, Alexandra Kolla, and Vivek Madan. On the expansion of group-based lifts. *SIAM Journal on Discrete Mathematics*, 33(3):1338–1373, 2019.

Philip Andreasson, Joel Johansson, Simon Liljestrand, and Mats Granath. Quantum error correction for the toric code using deep reinforcement learning. *Quantum*, 3:183, 2019.

Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *nature*, 574(7779):505–510, 2019.

Jueming Bao, Zhaorong Fu, Tanumoy Pramanik, Jun Mao, Yulin Chi, Yingkang Cao, Chonghao Zhai, Yifei Mao, Tianxiang Dai, Xiaojiong Chen, et al. Very-large-scale integrated quantum graph photonics. *Nature Photonics*, 17(7):573–581, 2023.

Johannes Bausch, Andrew W Senior, Francisco JH Heras, Thomas Edlich, Alex Davies, Michael Newman, Cody Jones, Kevin Satzinger, Murphy Yuezhen Niu, Sam Blackwell, et al. Learning high-accuracy error decoding for quantum processors. *Nature*, 635(8040):834–840, 2024.

Charles H Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theoretical computer science*, 560:7–11, 2014.

Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper Van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, et al. The scip optimization suite 8.0. *arXiv preprint arXiv:2112.08872*, 2021.

Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S Kottmann, Tim Menke, et al. Noisy intermediate-scale quantum algorithms. *Reviews of Modern Physics*, 94(1):015004, 2022.

John Blue, Harshil Avlani, Zhiyang He, Liu Ziyin, and Isaac L Chuang. Machine learning decoding of circuit-level noise for bivariate bicycle codes. *Quantum*, 10:2149, 2026.

Dolev Bluvstein, Simon J Evered, Alexandra A Geim, Sophie H Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, et al. Logical quantum processor based on reconfigurable atom arrays. *Nature*, 626(7997):58–65, 2024.

Sergey Bravyi, Andrew W Cross, Jay M Gambetta, Dmitri Maslov, Patrick Rall, and Theodore J Yoder. High-threshold and low-overhead fault-tolerant quantum memory. *Nature*, 627(8005): 778–782, 2024.

Nikolas P Breuckmann and Jens Niklas Eberhardt. Quantum low-density parity-check codes. *PRX quantum*, 2(4):040101, 2021.

Jonathan J Burnett, Andreas Bengtsson, Marco Scigliuzzo, David Niepce, Marina Kudra, Per Delsing, and Jonas Bylander. Decoherence benchmarking of superconducting qubits. *npj Quantum Information*, 5(1):54, 2019.

A Robert Calderbank and Peter W Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2):1098, 1996.

Shuxiao Chen, Edgar Dobriban, and Jane H Lee. A group-theoretic framework for data augmentation. *Journal of Machine Learning Research*, 21(245):1–71, 2020.

Yu-Fang Chen, Seyed Mohammad Reza Jafari, and Ching-Yi Lai. Sat, maxsat, and smt for qldpc distance computation: A large-scale empirical study. *arXiv preprint arXiv:2606.12445*, 2026.

Yoni Choukroun and Lior Wolf. Denoising diffusion error correction codes. *arXiv preprint arXiv:2209.13533*, 2022.

Yoni Choukroun and Lior Wolf. A foundation model for error correction codes. In *International Conference on Learning Representations*, volume 2024, pp. 2153–2171, 2024.

Laia Domingo Colomer, Michalis Skotiniotis, and Ramon Muñoz-Tapia. Reinforcement learning for optimal error correction of toric codes. *Physics Letters A*, 384(17):126353, 2020.

Juan Cruz-Benito, Andrew W Cross, David Kremer, and Ismael Faro. Evolutionary discovery of bivariate bicycle codes with llm-guided search. *arXiv preprint arXiv:2606.02418*, 2026.

Antonio deMarti iOlius, Patricio Fuentes, Román Orús, Pedro M Crespo, and Josu Etxezarreta Martinez. Decoding algorithms for surface codes. *Quantum*, 8:1498, 2024.

Qiuju Diao, Qin Huang, Shu Lin, and Khaled Abdel-Ghaffar. A matrix-theoretic approach for analyzing quasi-cyclic low-density parity-check codes. *IEEE Transactions on Information Theory*, 58(6):4030–4048, 2012. doi: 10.1109/TIT.2012.2184834.

Ilya Dumer, Alexey A Kovalev, and Leonid P Pryadko. Numerical techniques for finding the distances of quantum codes. In *2014 IEEE International Symposium on Information Theory*, pp. 1086–1090. IEEE, 2014.

Ilya Dumer, Alexey A Kovalev, and Leonid P Pryadko. Distance verification for classical and quantum ldpc codes. *IEEE Transactions on Information Theory*, 63(7):4675–4686, 2017.

Josu Etxezarreta Martinez, Patricio Fuentes, Pedro Crespo, and Javier Garcia-Frias. Time-varying quantum channel models for superconducting qubits. *npj Quantum Information*, 7(1):115, 2021.

Thomas Feulner. The automorphism groups of linear codes and canonical representatives of their semilinear isometry classes. *Adv. Math. Commun.*, 3(4):363–383, 2009.

David Fitzek, Mattias Eliasson, Anton Frisk Kockum, and Mats Granath. Deep q-learning decoder for depolarizing noise on the toric code. *Physical Review Research*, 2(2):023230, 2020.

Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, 2021.

Daniel Gottesman. *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

Markus Grassl. Searching for linear codes with large minimum distance. In *Discovering Mathematics with Magma: Reducing the Abstract to the Concrete*, pp. 287–313. Springer, 2006.

Elena Grigorescu, Vatsal Jha, and Eric Samperton. On the hardness of approximating minimum distances of quantum codes. *arXiv preprint arXiv:2509.21469*, 2025.

Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, 1996.

Richard W Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

Robin Harper, Steven T Flammia, and Joel J Wallman. Efficient learning of quantum noise. *Nature Physics*, 16(12):1184–1188, 2020.

Fernando Hernando, Gregorio Quintana-Ortí, and Markus Grassl. Fast algorithms and implementations for computing the minimum distance of quantum codes. *ACM Transactions on Quantum Computing*, 7(2):1–19, 2026.

Christopher J Hillar and Darren L Rhea. Automorphisms of finite abelian groups. *The American Mathematical Monthly*, 114(10):917–923, 2007.

Hsin-Yuan Huang, Michael Broughton, Jordan Cotler, Sitan Chen, Jerry Li, Masoud Mohseni, Hartmut Neven, Ryan Babbush, Richard Kueng, John Preskill, et al. Quantum advantage in learning from experiments. *Science*, 376(6598):1182–1186, 2022.

Upendra Kapshikar and Srijita Kundu. On the hardness of the minimum distance problem of quantum codes. *IEEE Transactions on Information Theory*, 69(10):6293–6302, 2023.

Mario Krenn, Jonas Landgraf, Thomas Foesel, and Florian Marquardt. Artificial intelligence and machine learning for quantum technologies. *Physical Review A*, 107(1):010101, 2023.

Moritz Lange, Pontus Havström, Basudha Srivastava, Isak Bengtsson, Valdemar Bergentall, Karl Hammar, Olivia Heuts, Evert Van Nieuwenburg, and Mats Granath. Data-driven decoding of quantum error correcting codes using graph neural networks. *Physical Review Research*, 7(2): 023181, 2025.

Ye-Hua Liu and David Poulin. Neural belief-propagation decoders for quantum error-correcting codes. *Physical review letters*, 122(20):200501, 2019.

Nishad Maskara, Aleksander Kubica, and Tomas Jochym-O'Connor. Advantages of versatile neural-network decoding for topological codes. *Physical Review A*, 99(5):052351, 2019.

Kai Meinerz, Chae-Yeun Park, and Simon Trebst. Scalable neural decoder for topological surface codes. *Physical Review Letters*, 128(8):080505, 2022.

Pavel Panteleev and Gleb Kalachev. Quantum ldpc codes with almost linear minimum distance. *IEEE Transactions on Information Theory*, 68(1):213–229, 2021.

Seong-Joon Park, Hee-Youl Kwak, Sang-Hyo Kim, Yongjune Kim, and Jong-Seon No. Crossmpt: Cross-attention message-passing transformer for error correcting codes. In *International Conference on Learning Representations*, volume 2025, pp. 56710–56728, 2025.

Yotam Peled, David Zenati, and Eliya Nachmani. Neural minimum weight perfect matching for quantum error codes. *arXiv preprint arXiv:2601.00242*, 2026.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

John Preskill. Quantum computing and the entanglement frontier. *arXiv preprint arXiv:1203.5813*, 2012.

Leonid P Pryadko, Vadim A Shabashov, and Valerii K Kozin. Qdistrnd: A gap package for computing the distance of quantum error-correcting codes. *arXiv preprint arXiv:2308.15140*, 2023.

Mayur Punekar, Frank Kienle, Norbert Wehn, Akin Tanatmis, Stefan Ruzika, and Horst W Hamacher. Calculating the minimum distance of linear block codes via integer programming. In *2010 6th International Symposium on Turbo Codes & Iterative Information Processing*, pp. 329–333. IEEE, 2010.

Dongheng Qian and Tianyi Li. Multi-agent discovery of practical quantum ldpc codes. *arXiv preprint arXiv:2608.08996*, 2026.

Joschka Roffe, David R White, Simon Burton, and Earl Campbell. Decoding across the quantum low-density parity-check code landscape. *Physical Review Research*, 2(4):043423, 2020.

Andrew W Senior, Thomas Edlich, Francisco JH Heras, Lei M Zhang, Oscar Higgott, James S Spencer, Taylor Applebaum, Sam Blackwell, Justin Ledford, Akvilė Žemgulytė, et al. A scalable and real-time neural decoder for topological quantum codes. *arXiv preprint arXiv:2512.07737*, 2025.

Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pp. 124–134. Ieee, 1994.

Peter W Shor. Scheme for reducing decoherence in quantum computer memory. *Physical review A*, 52(4):R2493, 1995.

Andrew Steane. Multiple-particle interference and quantum error correction. *Proceedings: Mathematical, Physical and Engineering Sciences*, pp. 2551–2577, 1996a.

Andrew Steane. Quantum computing. *Reports on Progress in Physics*, 61(2):117–173, 1998.

Andrew M Steane. Error correcting codes in quantum theory. *Physical Review Letters*, 77(5):793, 1996b.

Ryan Sweke, Markus S Kesselring, Evert PL van Nieuwenburg, and Jens Eisert. Reinforcement learning decoders for fault-tolerant quantum computation. *Machine Learning: Science and Technology*, 2(2):025005, 2021.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

R Tanner. A recursive approach to low complexity codes. *IEEE Transactions on information theory*, 27(5):533–547, 1981.

Barbara M Terhal. Quantum error correction for quantum memories. *Reviews of Modern Physics*, 87(2):307–346, 2015.

Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, 1997.

S Varsamopoulos, B Criger, and K Bertels. Decoding small surface codes with feedforward neural networks. quantum science and technology, 3 (1), 015004, 2018.

Savvas Varsamopoulos, Koen Bertels, and Carmen Garcia Almudever. Comparing neural network based decoders for the surface code. *IEEE Transactions on Computers*, 69(2):300–311, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

K Veeresh, S Deepak, and T Srinivas. Rl-qec: Harnessing reinforcement learning for quantum error correction advancements. In *2024 International Conference on Trends in Quantum Computing and Emerging Business Technologies*, pp. 1–5. IEEE, 2024.

Zihao Wang and Hao Tang. Artificial intelligence for quantum error correction: A comprehensive review. *arXiv preprint arXiv:2412.20380*, 2024.

Mark Webster, Abraham Jacob, and Oscar Higgott. Distance-finding algorithms for quantum codes and circuits. *arXiv preprint arXiv:2603.22532*, 2026.

Mark A Webster and Dan E Browne. Engineering quantum error correction codes using evolutionary algorithms. *IEEE Transactions on Quantum Engineering*, 6:1–14, 2025.

William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886): 802–803, 1982.

Ge Yan, Shanchuan Li, Pengyue Ma, Qixin Zhang, Pingchuan Ma, Jianping Wang, Min-Hsiu Hsieh, and Yuxuan Du. Omniqec: discovering practical quantum error-correcting codes by an ai scientist. *arXiv preprint arXiv:2607.25865*, 2026.

David Zenati and Eliya Nachmani. Saq: Stabilizer-aware quantum error correction decoder. In *International Conference on Learning Representations*, volume 2026, pp. 22608–22633, 2026.

A Experimental Results

A.1 Training and Inference Configuration

All training experiments were conducted on an L40s GPU using PyTorch on the dataset we have created. Training hyperparametrs configuration are presented in Table 6. Both heuristics and BB-Net inference were conducted on a compute node equipped with two AMD EPYC 7763 64-Core processors, providing 128 physical CPU cores (256 hardware threads). Heuristics' invalid answers as defined in subsection 3.2 are also included in the results tables and affect their corresponding accuracy and RMSE.

In the ablation experiments, to match number of parameters when comparing BB-Net to BB-Net-Tanner, we set $L = 16$ for BB-Net. Both models have approximately 13M parameters.

Table 6: Hyperparameters used for training.

Params	Values
L	8
d_{model}	256
lr	3e-4
Dropout	0.1
Epochs	90
Warmup Epochs	9
Batch Size	256
Optimizer	AdamW
Weight Decay	0.01

Table 7: Selected test codes on which the heuristics overestimate the reference distance and BB-Net matches it. The reference is the minimum found by the same heuristics at 10,000 trials. GEStim, CCStim and UEStim failed to output a result.

(N, ℓ, m)	$A(x, y)$	$B(x, y)$	$C(x, y)$	$D(x, y)$	GT	BP-OSD	QDistEvol	QDistRnd	BB-Net (ours)
$(324, 18, 9)$	$x^2y^2 + x^9y + x^9y^8 + x^{13}y$	$x^4y^8 + x^8y + x^{12}y^4 + x^{16}y$	–	–	18	44	45	**18**	**18**
$(270, 27, 5)$	$y^2 + x^5 + x^{16}y + x^{26}y^3$	$x^8y^3 + x^{10}y^4 + x^{24}y^3 + x^{26}y^2$	–	–	25	35	38	37	**25**
$(342, 19, 9)$	$x^8y^2 + x^{16}y + x^{16}y^3$	$y^5 + x^2y^7 + x^{12}y^3$	$x^{12}y^5 + x^{12}y^7$	$x^6y^3 + x^6y^5$	24	Fail	36	26	**24**
$(348, 29, 6)$	$x^8y^2 + x^{13}y + x^{25}y^3$	$x^2y^5 + x^{12}y^4 + x^{16}y^3$	$x^{11}y^4 + x^{22}y$	$x^{18} + x^{25}y^3$	24	Fail	36	44	**24**
$(272, 17, 8)$	$xy^2 + x^4y^2 + x^8y^3 + x^9y^7$	$y^4 + x^7y^2 + x^8y^2 + x^{12}y^2$	–	–	25	37	36	28	**25**
$(360, 30, 6)$	$x^2 + x^7 + x^{19}y^5$	$x^5y^2 + x^{12}y^2 + x^{26}y$	–	–	20	30	30	22	**20**
$(360, 18, 10)$	$x^7y^8 + x^9y^9 + x^{14}y^2$	$y^6 + x^{10}y^8 + x^{14}y^5$	$x^5y^8 + x^{11}y^9$	$x^9 + x^{15}y^3$	24	Fail	34	42	**24**
$(336, 28, 6)$	$x^{10}y + x^{12}y^5 + x^{22}$	$x^7 + x^{19}y + x^{26}y^5$	$x^{15}y^2 + x^{25}y^3$	$x^5y^3 + x^{21}y^4$	20	Fail	30	28	**20**
$(336, 28, 6)$	$x^{12} + x^{16}y + x^{21}y^2$	$x^8y^5 + x^{11} + x^{27}y$	$x^{12} + x^{25}y^3$	$x^{11}y^3 + x^{24}y^3$	24	Fail	34	28	**24**
$(216, 18, 6)$	$x^4y + x^5 + x^7y + x^{15}y$	$xy + x^9y^2 + x^9y^3 + x^{13}y^2$	–	–	18	26	26	26	**18**
$(308, 22, 7)$	$x^9y + x^9y^4 + x^{12}y^2$	$x^{10}y + x^{16}y^6 + x^{18}y^5$	–	–	20	**20**	28	**20**	**20**
$(250, 25, 5)$	$x^3 + x^6y^3 + x^9y^3 + x^{13}y^2$	$x^6y^3 + x^{10}y + x^{22}y + x^{22}y^2$	–	–	25	31	33	31	**25**
$(360, 6, 30)$	$y^4 + x^3y^6 + x^4$	$y^8 + y^{19} + x^3$	–	–	24	26	32	32	**24**
$(324, 18, 9)$	$x^5y^5 + x^{16}y^7 + x^{17}y$	$x^8y^3 + x^{10}y^3 + x^{11}y^7$	$x^{11}y^3$	$x^8y^3 + x^{13}y^7$	20	Fail	28	36	**20**
$(294, 21, 7)$	$x^5 + x^7y^5 + x^9y^5$	$x^3y^4 + x^7y^5 + x^{11}y^4$	$x^5 + x^{11}y^3$	x^5y	22	Fail	30	28	**22**
$(350, 25, 7)$	$x^4y + x^{10}y^4 + x^{17}y^3$	$x^8y^4 + x^{18}y^3 + x^{19}y$	$x + x^{23}y^2$	$x^6y + x^{21}y^4$	24	Fail	32	42	**24**
$(350, 25, 7)$	$x^7y^6 + x^{15} + x^{18}y^2$	$x^5y^2 + x^{18}y^3 + x^{18}y^5$	$x^{23}y + x^{24}y^5$	$y^3 + x^{11}y^5$	22	Fail	30	36	**22**
$(312, 26, 6)$	$x^7y + x^7y^2 + x^{24}y^3$	$x^5y^4 + x^{10}y^5 + x^{17}$	$x^{14}y^3 + x^{17}y^2$	$x^{12} + x^{15}y^5$	24	Fail	32	34	**24**

A.2 Ablation Insights

We evaluate each ablation using several random seeds. Figure 5 reports the validation negative log-likelihood (NLL) as a function of the training epoch. The bold curves represent the complete configuration, which includes training augmentation, shift canonicalization, and FiLM conditioning, whereas the lighter curves represent variants in which the component indicated in each panel is removed. The Tanner spectral branch enables the model to reach a lower validation NLL, demonstrating the benefit of incorporating global spectral information. Training augmentation has the strongest regularizing effect: without it, both the polynomial-only and dual-branch models overfit rapidly and fail to reach their best attainable validation performance. Shift canonicalization and FiLM conditioning also provide consistent improvements, particularly for the polynomial-only model, where global spectral information is unavailable.

B Dataset statistics

Figure 6 presents the dataset's distributions. The dataset holds 58,704 codes, 83.9% of them CSS BB and 16.1% non-CSS PBB extensions, spanning block lengths $n = 18$ to 360, lattices $\ell, m \in [3, 30]$, $k = 2$ to 216 logical qubits and rates k/n between 0.006 and 0.750. All heuristics used to create the upper bound distance for this dataset were implemented by the codeDistance package (Webster et al., 2026).

Distributions. The distance distribution in Figure 6(a) is strongly right-skewed: the median is $d = 12$ against a maximum of 25. The distance of a code is defined to be its best available upper bound. Figure 6(b) describes the codes' length n distribution. Figure 6(c) shows the logical qubits in codes as a function of the block length.

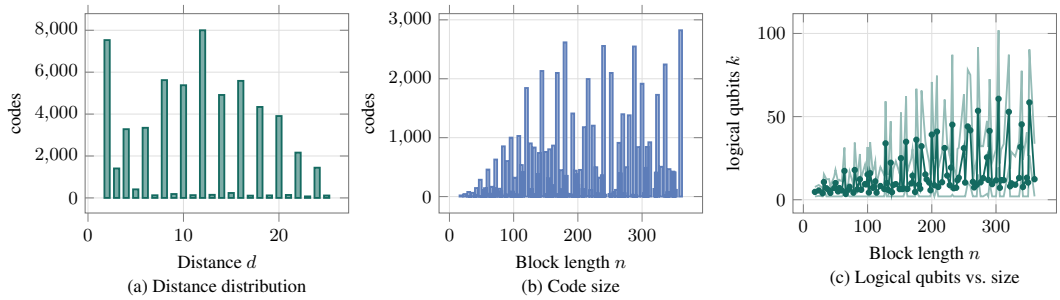

Figure 6: Corpus statistics. (a) distribution of the code distance; (b) distribution of block length $n = 2\ell m$; (c) mean logical dimension k per block length, with $\pm$ one standard deviation. Throughout, the distance of a code without an exact solution is its best available upper bound.

C Feature Construction and Embedding

The two BB-Net branches operate on different raw feature spaces. Polynomial tokens combine a learned polynomial-type representation with cyclic Fourier coordinates, whereas Tanner spectral tokens contain explicitly computed character-response features. Each raw token is projected to the dimension of d_{model} before self-attention. The following is a description of the full feature computation on the spectral branch:

Notation. For integers p, q we write $\gcd(p, q)$ for their greatest common divisor, the largest integer dividing both, and $\operatorname{lcm}(p, q) = pq/\gcd(p, q)$ for their least common multiple, the smallest positive integer that both divide. For a complex number z, $\overline{z}$ is its complex conjugate, $|z|$ its modulus, and $\operatorname{Re}(z), \operatorname{Im}(z)$ its real and imaginary parts. The symbol $\mathbf{1}[\,\cdot\,]$ denotes the indicator function, equal to 1 when the enclosed condition holds and 0 otherwise; $p \mid q$ reads "p divides q". Finally, $\|$ denotes concatenation of vectors.

Character responses. For each character $\chi = \chi_{u,v}$, define the response of polynomial P as

$$z_P(\chi) = \widehat{P}(\chi) = \sum_{(a,b)\in S_P} \exp\left[-2\pi i \left(\frac{ua}{\ell} + \frac{vb}{m}\right)\right], \qquad w_P = |S_P|, \tag{13}$$

for $P \in \{A, B, C, D\}$, where w_P is the number of monomials in P.

Its normalized magnitude is

$$a_P(\chi) = \begin{cases} |z_P(\chi)|/w_P, & w_P > 0, \\ 0, & w_P = 0, \end{cases} \tag{14}$$

so that $0 \le a_P(\chi) \le 1$. We also compute the joint energy, cross-spectrum, and relative spectral difference:

$$s_2(\chi) = \frac{|z_A(\chi)|^2 + |z_B(\chi)|^2}{w_A^2 + w_B^2}, \quad q_{AB}(\chi) = \frac{z_A(\chi)\overline{z_B(\chi)}}{w_A w_B}, \quad \delta_{AB}(\chi) = \frac{|z_A(\chi) - z_B(\chi)|}{w_A + w_B}. \tag{15}$$

All three are normalized by polynomial weight so that they remain comparable across codes with different numbers of monomials.

Character order. The order of $\chi_{u,v}$ is

$$o_\ell(u) = \frac{\ell}{\gcd(u,\ell)}, \;\; o_m(v) = \frac{m}{\gcd(v,m)}, \;\; o_\chi = \operatorname{lcm}\big(o_\ell(u), o_m(v)\big). \tag{16}$$

The feature $\mathbf{1}[2 \mid o_\chi]$ equals 1 when o_χ is even. The feature $\mathbf{1}[(u,v) = (0,0)]$ flags the trivial character, at which $z_P(\chi) = w_P$ for every P.

Spectral feature vector. For readability, we specify the 18-dimensional spectral feature vector in three groups. All response-dependent quantities below are evaluated at χ.

The first group collects the per-polynomial magnitudes and the pairwise response statistics:

$$\mathbf{s}_\chi^{(1)} = \big[a_A,\, a_B,\, a_C,\, a_D,\, a_A^2,\, a_B^2,\, s_2,\, \mathrm{Re}(q_{AB}),\, \mathrm{Im}(q_{AB}),\, |q_{AB}|\big] \in \mathbb{R}^{10}. \tag{17}$$

The second group sharpens the near-degenerate regime:

$$\mathbf{s}_\chi^{(2)} = \big[\delta_{AB},\, e^{-8s_2},\, e^{-8(1-\min(a_A,a_B))}\big] \in \mathbb{R}^3, \tag{18}$$

The third group contains invariants of the character itself, independent of the polynomial supports:

$$\mathbf{s}_\chi^{(3)} = \left[\frac{o_\chi}{\ell m},\, \frac{\log(1+o_\chi)}{\log(1+\ell m)},\, \frac{1}{o_\chi},\, \mathbf{1}[2 \mid o_\chi],\, \mathbf{1}[(u,v) = (0,0)]\right] \in \mathbb{R}^5. \tag{19}$$

The complete feature vector is

$$\mathbf{s}_\chi = \mathbf{s}_\chi^{(1)} \,\|\, \mathbf{s}_\chi^{(2)} \,\|\, \mathbf{s}_\chi^{(3)} \in \mathbb{R}^{18}. \tag{20}$$