PARAMETER-EFFICIENT TRANSFORMER ARCHITECTURES FOR ERROR CORRECTION CODE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Error correction coding is a cornerstone of reliable communication systems, ensuring data integrity over noisy channels. The Error Correction Code Transformer (ECCT) and its successors have demonstrated that Transformer-based neural decoders can outperform classical decoding methods for short and medium linear block codes. However, existing Transformer-based decoders suffer from high parameter counts and computational costs, both of which scale linearly with model depth, constraining their deployment scenarios and performance potential. In this work, we propose two architectures to tackle the parameter efficiency challenge. First, we introduce the Shared Error Correction Code Transformer (SECCT), which employs cross-layer parameter sharing, effectively collapsing the model to a single Transformer layer that is applied repeatedly. By preserving decoding depth through iteration rather than stacked distinct layers, SECCT improves the bit error rate (BER) over the published ECCT baseline on all 39 code–signal-to-noise ratio (SNR) settings we evaluate while using less than 20% of the parameters of that baseline. The price is only a modest increase in computational cost due to additional forward passes. This provides a viable solution with performance guarantees for memory-constrained edge deployment scenarios. Second, we propose the Hybrid Error Correction Code Transformer (HECCT), which employs Grouped-Query Attention (GQA) and a bottleneck feed-forward structure to simplify each Transformer layer. HECCT achieves lower parameter count and computational cost than the published ECCT baseline, while delivering better BER performance, establishing a new Pareto-optimal frontier in the accuracy–efficiency trade-off. Extensive experiments on 13 codes from five families validate the effectiveness of both architectures, which provide a reliable and efficient solution for next-generation communication systems.

1 INTRODUCTION

The evolution of wireless communication toward 6G imposes unprecedented demands on reliability, latency, and energy efficiency Jiang et al. (2021); Giordani et al. (2020); Zhang & Tong (2024). At its core lies error correction coding, a foundational technique established by Shannon Shannon (1948): 5G New Radio (NR) adopts Low-Density Parity-Check (LDPC) codes for data and Polar codes for control channels, while classical Bose–Chaudhuri–Hocquenghem (BCH) codes serve specialized scenarios.

Decoding remains code-specific—LDPC codes use belief propagation (BP) Gallager (1962), Polar codes use successive cancellation (SC) Arikan (2009), and BCH codes use algebraic hard-decision decoding Bose & Ray-Chaudhuri (1960)—forcing custom hardware and limiting flexibility in multi-standard systems. Unified decoding has therefore been pursued through ordered statistics decoding (OSD) Fossorier & Lin (1995), whose complexity grows exponentially with code dimension, and through AI-driven methods, from early neural decoders Nachmani et al. (2016); Gruber et al. (2017); Kim et al. (2018) and preprocessing Bennatan et al. (2018) to the Error Correction Code Transformer (ECCT) Choukroun & Wolf (2022) and its foundation-model successor (FECCT) Choukroun & Wolf (2024).

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.

Despite this progress, a critical challenge remains: parameter efficiency. In existing Transformer-based decoders, parameter count and computational cost scale *linearly* with depth, restricting deployment on memory-constrained edge devices and capping deeper architectures.

Measuring the linear Centered Kernel Alignment (CKA) between the outputs of every layer pair and every head pair of a trained 6-layer ECCT, we find both highly similar and the attention projections markedly low-rank. This redundancy implies that a small parameter set can fulfil many layers and heads, motivating two architectures that decode more accurately with fewer parameters (Section 3.2).

SECCT (Shared Error Correction Code Transformer) applies cross-layer parameter sharing, collapsing the decoder into one Transformer layer applied repeatedly. Iterating a single layer rather than stacking distinct ones, it beats the published 6-layer ECCT baseline (L=6, d=128) on all 39 code–SNR settings while storing under 20% of its parameters. HECCT (Hybrid Error Correction Code Transformer) instead simplifies each layer, combining Grouped-Query Attention with a bottleneck feed-forward structure, and surpasses the same baseline at lower parameter count and computational cost.

Experiments on 13 codes from five families, with ablations on iteration depth, KV-head count, and feed-forward network (FFN) expansion, validate both designs (Section 3).

2 Related Work

The code-specific nature of classical decoding has motivated two families of learned decoders: model-free decoders apply generic networks directly to the channel output Gruber et al. (2017); Kim et al. (2018), while model-based decoders unfold a classical algorithm and parameterize it, beginning with the weighted Tanner-graph belief propagation of Nachmani et al. Nachmani et al. (2016) and extended by hyper-graph Nachmani & Wolf (2019) and autoregressive Nachmani & Wolf (2021) variants. A complementary line applies learning as preprocessing through the syndrome-based framing of Bennatan et al. Bennatan et al. (2018), whose codeword invariance is inherited by every Transformer-based decoder that follows.

Choukroun and Wolf Choukroun & Wolf (2022) then introduced the Error Correction Code Transformer (ECCT), the first Transformer to decode a code end-to-end, embedding the channel magnitude and binary syndrome and injecting the Tanner graph into self-attention through a two-hop binary mask. Their foundation-model successor (FECCT) Choukroun & Wolf (2024) adds code-agnostic embeddings and a smooth distance-based mask modulation for zero-shot generalization to unseen codes. Neither addresses the parameter cost of an individual decoder for a given code.

A parallel line of work in natural language processing reduces the parameter cost of Transformers along axes that transfer directly to ECC decoding. ALBERT Lan et al. (2020) shares one Transformer layer across all depths, and the Universal Transformer Dehghani et al. (2019) reuses one recurrent block at every step. Grouped-query attention (GQA) Ainslie et al. (2023) shares key–value projections within groups of heads, exposing a knob that trades key–value parameters against accuracy, with multi-query attention (MQA) Shazeer (2019) as its extreme. Finally, the feed-forward expansion is over-provisioned: a narrower one costs almost nothing in quality Vaswani et al. (2017), and bottleneck adapters Houlsby et al. (2019) show that a small intermediate width suffices.

3 Proposed Model

3.1 Preliminaries

3.1.1 Linear Block Codes

A binary linear block code $\mathcal{C}(n, k)$ maps a message $x \in \{0,1\}^k$ to a codeword $c \in \{0,1\}^n$ via a generator matrix $\mathbf{G} \in \{0,1\}^{k\times n}$: $c = x\mathbf{G}$. The code is equivalently defined by its parity-check matrix $H \in \{0,1\}^{(n-k)\times n}$ with $\mathbf{G}H^T = 0$ over GF(2). The Tanner graph has n variable nodes and $(n-k)$ check nodes, with an edge (i, j) whenever $H_{ji} = 1$. Our experiments span several widely used code families—including BCH, LDPC, Polar, MacKay, and CCSDS (Consultative Committee

for Space Data Systems) codes Bose & Ray-Chaudhuri (1960); Gallager (1962); Arikan (2009)—with the concrete (n, k) configurations listed in Section 4.1.

3.1.2 Channel Model and Modulation

We consider BPSK modulation over an AWGN channel. The modulated signal is $x_s = 1 - 2c$ (mapping $\{0, 1\}$ to $\{\pm 1\}$). The received signal is $y = x_s + n$ with $n \sim \mathcal{N}(0, \sigma^2)$. The noise variance is $\sigma^2 = (2R \cdot 10^{E_b/N_0/10})^{-1}$ where $R = k/n$.

Following ECCT Choukroun & Wolf (2022), we adopt the multiplicative noise model $y = x_s \cdot z$ with $z \in \mathbb{R}^n$. The decoding task predicts $\tilde{z} = \text{bin}(\text{sign}(z))$, and the decoded bits are $\hat{x} = \text{bin}(\text{sign}(y \cdot \tilde{z}))$.

3.1.3 Input Preprocessing and Output Postprocessing

Following Bennatan et al. Bennatan et al. (2018) and ECCT Choukroun & Wolf (2022), the input is:

$$\tilde{y} = [|y|, s(y)], \tag{1}$$

where $|y| \in \mathbb{R}^n$ is the channel magnitude and $s(y) = H \cdot \text{bin}(\text{sign}(y)) \in \{0, 1\}^{n-k}$ is the binary syndrome. The concatenated vector is linearly projected to a d-dimensional embedding. The output predicts $\tilde{z}$, and the decoded bits are recovered as $\hat{x} = \text{bin}(\text{sign}(y \cdot \tilde{z}))$. This ensures codeword invariance, preventing overfitting to training codewords Bennatan et al. (2018).

3.1.4 Transformer Decoding Framework

The ECCT baseline Choukroun & Wolf (2022) consists of L Transformer layers. Each layer applies masked multi-head self-attention with a Tanner-graph-derived binary mask $g(H)$, where H is the parity-check matrix: $g(H)_{ij} = -\infty$ if nodes i and j are not within two-hop Tanner graph distance, and 0 otherwise, giving $A_H = \text{Softmax}((QK^T + g(H))/\sqrt{d_k})V$; Figure 1 shows the resulting mask for BCH(31,16). The attention output then passes through a feed-forward network $\text{FFN}(X) = W_2 \cdot \text{GELU}(W_1 X + b_1) + b_2$ with $W_1 \in \mathbb{R}^{d_f \times d}$, $W_2 \in \mathbb{R}^{d \times d_f}$ and the standard expansion $d_f = 4d$, and residual connections with layer normalization surround each sublayer.

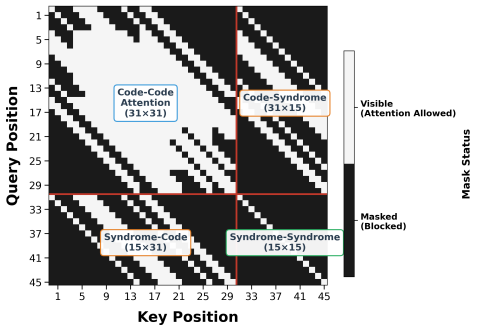


Figure 1: Self-attention mask for BCH(31,16): a 46×46 mask over $n = 31$ code-bit and $n - k = 15$ syndrome positions (white: visible, black: masked).

3.2 Motivating Analysis: Functional Redundancy in ECCT

Before introducing our architectures, we dissect a trained 6-layer ECCT decoder (d=128, N_h=8 heads) on Polar(64,32) to locate the redundancy that parameter-efficient designs can exploit. Four measurements are performed on real channel inputs: cross-layer and cross-head functional similarity, the singular-value spectrum of the attention projections, and the effective rank of the FFN bottleneck activation.

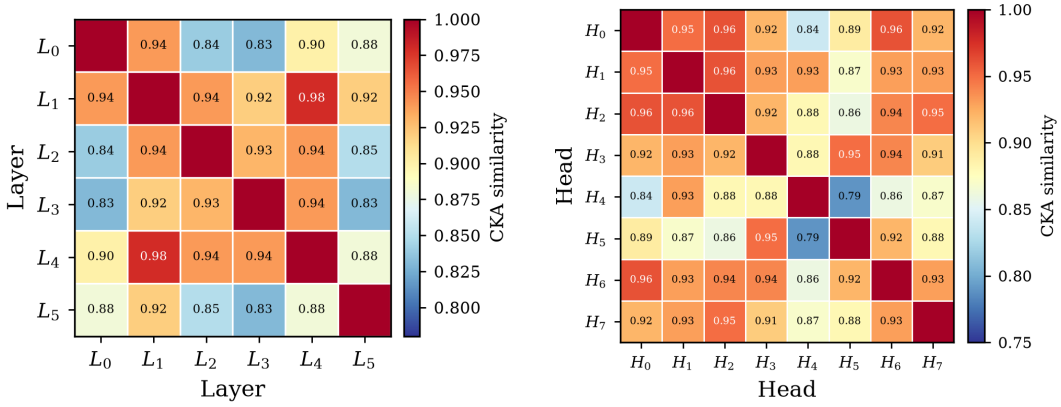
Layers and heads turn out to be functionally redundant. Computing the linear Centered Kernel Alignment (CKA) Kornblith et al. (2019) between the output representations of every layer pair, averaged over 1,000 input groups, gives the matrix of Figure 2(a): off-diagonal CKA values range from 0.83 to 0.98 with a mean of 0.90, peaking at 0.98 between layers L_1 and L_4. The same holds

across heads at layer L_0. Capturing each head's context $\mathrm{ctx} = \mathrm{AttnW} \cdot V$ *before* the shared output projection W_O, where the N_h=8 heads are still separable, and computing pairwise CKA over 2,000 codewords $\times$ 10 groups gives Figure 2(b), with a mean off-diagonal value of 0.91 at layer L_0. The appendix extends both measurements to the remaining five layers. This redundancy across both layers and heads indicates that a small set of parameters can serve the role of many layers and heads, which is the basis for the shared layer of SECCT and for the small number of KV groups in HECCT.

The attention projections and the FFN activations both exhibit capacity redundancy. Figure 3(a) shows how the energy of the 128×128 key and value projections W_K, W_V at layer L_0 is distributed over the singular directions. Using the energy-based rank r_{90}—the smallest r such that the top r directions capture 90% of $\|W\|_F^2$—we obtain $r_{90} = 63$ (W_K) and 62 (W_V) out of 128: roughly *half* of the directions carry 90% of the energy, indicating that the KV projections are not fully utilized.

The Bottleneck FFN rests on a different quantity: how many dimensions the FFN intermediate layer *actually* uses on real codewords, which is measured on the activation rather than the weights. Taking the L_1 bottleneck activation $h = \mathrm{GELU}(W_1 \cdot \mathrm{LN}(x)) \in \mathbb{R}^{d_f}$, flattening it over 2,000 random codewords, and eigendecomposing its covariance gives Figure 3(b) for the default configuration (expansion 4.0, $d_f = 512$). In sharp contrast to the attention projections, here $r_{90}(h) = 67$ with the same 90%-energy criterion applied to the activation covariance: only 13% of the capacity carries 90% of the activation energy, and the first 64 directions alone already hold 89.4% of it. The energy decays continuously rather than being cut off at any point, so r_{90} marks a *soft* effective-dimensionality boundary rather than a hard pruning point. The standard FFN is thus far wider than the representational dimensionality it actually uses.

Together, these measurements indicate that functional and capacity redundancy is present in the trained decoder, and that compressing it is a promising route to a smaller parameter budget.



(a) cross-layer CKA

(b) cross-head CKA (layer L_0)

Figure 2: Functional redundancy inside a trained 6-layer ECCT (d=128) on Polar(64,32): (a) inter-layer and (b) inter-head linear CKA.

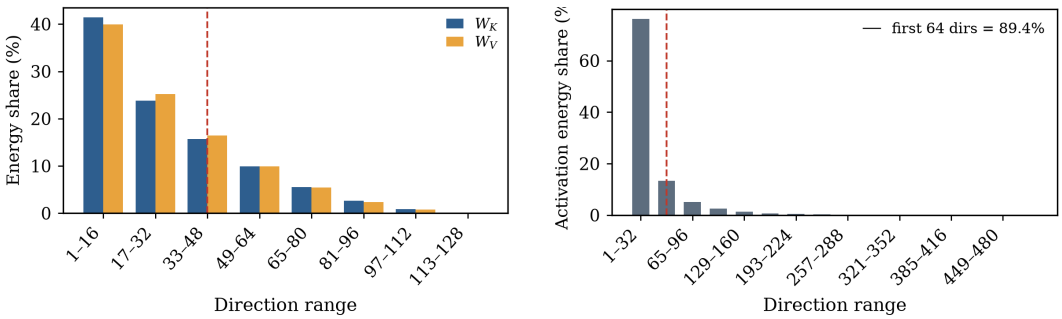
3.3 SECCT: Shared Error Correction Code Transformer

SECCT replaces the L independent Transformer layers of ECCT with a *single shared layer* applied iteratively L times, adopting the cross-layer parameter sharing of ALBERT Lan et al. (2020) (Section 2) and illustrated in Figure 4. It uses one Transformer encoder layer $\mathcal{T}$ with parameters θ, applied L times:

$$X^{(l+1)} = \mathcal{T}(X^{(l)}; \theta), \quad \forall l \in \{0, 1, \ldots, L-1\}, \tag{2}$$

where $X^{(0)}$ is the input embedding and $X^{(L)}$ is the final representation. SECCT reuses the identical transformation at every iteration, as in the Universal Transformer Dehghani et al. (2019), but sharing *all* parameters including attention projections, FFN weights, and normalization parameters.

Unlike ECCT, whose parameter count grows as $O(L \cdot d^2)$ with depth L, SECCT stores only a *single* shared layer; its parameter count is $|\theta_{\text{SECCT}}| = |\theta| + |\theta_{\text{embed}}| + |\theta_{\text{output}}| = O(d^2 + (n-k)d + nd)$, *independent of* L. SECCT can therefore be iterated arbitrarily many times to emulate an arbitrarily



(a) attention-projection energy

(b) FFN bottleneck activation

Figure 3: Capacity redundancy in the trained ECCT on Polar(64,32): (a) energy of W_K, W_V over singular directions and (b) FFN activation over direction ranges; dashed lines mark $r_{90} = 63/62$ and $r_{90}(h) = 67$.

deep network at constant parameter and memory cost $O(d^2)$. As in ECCT Choukroun & Wolf (2022), a single LayerNorm is placed between Transformer layers at the midpoint of the decoder depth.

3.4 HECCT: Hybrid Error Correction Code Transformer

HECCT simplifies each Transformer layer instead of sharing it: we replace the standard attention and feed-forward sublayers by grouped-query attention Ainslie et al. (2023) and a bottleneck FFN, as illustrated in Figure 4. The first change reduces the cost of the key–value projections. With $N_h = 8$ heads of dimension $d_k = 16$ and $d = 128$, each head's context is computed by

$$\text{head}_i = \text{Attention}(XW_i^Q, XW_g^K, XW_g^V), \tag{3}$$

for head i in group $g = \lfloor i/G \rfloor$, where $W_i^Q \in \mathbb{R}^{d\times d_k}$ is head-specific and $W_g^K, W_g^V \in \mathbb{R}^{d\times d_k}$ are shared within group g; the ECCT binary mask $g(H)$ is applied to the attention scores as before. Grouping the $N_h = 8$ heads into $G = 2$ groups reduces the key–value projections from $2d^2$ to $0.5d^2$ parameters per layer, while retaining head-specific queries.

The second change shrinks the feed-forward sublayer. The standard $4\times$ expansion ($d_f = 4d$) is known to be over-provisioned: Vaswani et al. Vaswani et al. (2017) found that a $2\times$ expansion costs only 0.4 BLEU, and bottleneck adapters Houlsby et al. (2019) show that a reduce–nonlinearity–expand module of small intermediate width suffices for representation learning. Section 3.2 provides direct evidence for this task: in the trained ECCT the intermediate activation of the full-width ($d_f = 512$) FFN carries 90% of its energy in only about 13% of the directions ($r_{90} = 67$), so its effective representational dimensionality is far below its width. We therefore shrink the FFN intermediate dimension directly, rather than inserting an external adapter:

$$\text{FFN}_{\text{bn}}(X) = W_2 \cdot \text{GELU}(W_1 X + b_1) + b_2, \tag{4}$$

with $W_1 \in \mathbb{R}^{d'_f \times d}$, $W_2 \in \mathbb{R}^{d\times d'_f}$, and $d'_f < 4d$, where d is the model dimension d_{model} and d'_f the bottleneck dimension $d_{\text{bottleneck}}$ of Figure 4; the uncompressed width of the standard FFN is written d_f throughout.

The complete HECCT layer combines both changes, so that each of the L layers reduces the attention and feed-forward cost while keeping the ECCT masking mechanism and residual structure; as in SECCT, a single LayerNorm is placed between Transformer layers at the midpoint of the decoder depth.

4 Experiments

4.1 Experimental Setup

We evaluate on 13 codes from five families—Polar, LDPC, MacKay, CCSDS and BCH, with (n, k) pairs as listed in Table 1—covering code lengths from 49 (LDPC(49,24)) to 128 and code rates

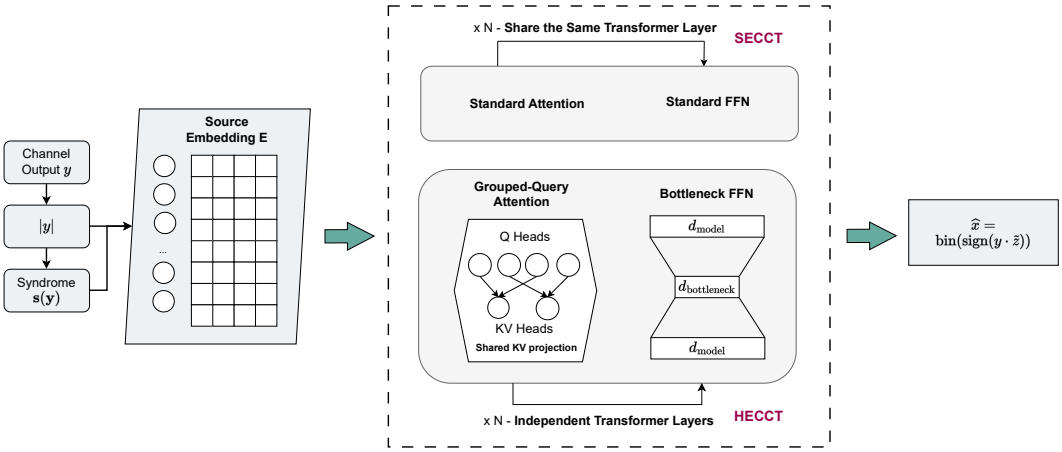


Figure 4: Overview of SECCT and HECCT. SECCT (top) replaces the L independent layers by one shared layer iterated L times; HECCT (bottom) adds GQA and a Bottleneck FFN, whose intermediate dimension is reduced from d_{model} to $d_{\text{bottleneck}}$.

from 0.49 to 0.75. The parity-check matrices follow the benchmark of Choukroun & Wolf (2022). For each code we generate 128,000 training samples per epoch, transmitted over an additive white Gaussian noise (AWGN) channel with binary phase-shift keying (BPSK) modulation. At test time, each E_b/N_0 point is evaluated on fresh random codewords until more than 100 frame errors have been collected or a cap of 10^9 codewords is reached; at least 10^5 codewords are simulated at each point.

All our models use d=128 and N_h=8 heads (d_k=16). SECCT (S1) applies a single shared Transformer layer iterated N_{dec}=12 times with full-width attention and FFN (expansion 4.0), where the iteration depth N_{dec} is the number of layers L of Section 3; HECCT (S2) uses N_{dec}=12 *independent* layers with GQA (G=2) and a Bottleneck FFN (expansion 0.5), where the expansion ratio is defined as d'_f/d. All models are trained with Adam (cosine-decayed learning rate 10^{-4} to 7×10^{-5}, batch size 128, cross-entropy loss). Deeper decoders need more training epochs to converge, as is also the case in the ECCT Choukroun & Wolf (2022) and FECCT Choukroun & Wolf (2024) papers: ECCT trains its 6-layer models for 1,000 epochs but its 10-layer models for 1,500 epochs, while FECCT keeps the same setting for its shallow configurations. HECCT follows this pattern, since its twelve layers are independent and every layer must be fitted separately; it is therefore trained for 1,500 epochs. SECCT needs no such increase, at any depth: all iterations share one parameter set, so the single stored layer receives as many updates per epoch as the whole of an equally deep unshared decoder, and 1,000 epochs suffice for all SECCT models.

4.2 Main Results

4.2.1 Comparison with Published Baselines

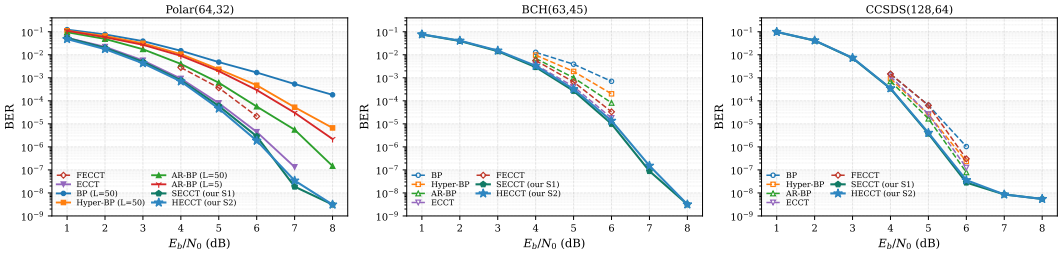
Table 1 compares the decoding performance of our two architectures with the BP-based decoders and with the published ECCT and FECCT models on 13 codes at $E_b/N_0 = 4, 5, 6$ dB; all values are $-\ln(\text{BER})$, so higher is better. Results for belief propagation (BP), hyper-graph BP, autoregressive BP, ECCT and FECCT are quoted from their original publications Nachmani et al. (2016); Nachmani & Wolf (2019; 2021); Choukroun & Wolf (2022; 2024). Each code occupies three rows, and the best result per SNR is set in **bold** with the second best underlined. Although we outperform FECCT in BER, FECCT retains a capability that none of the other decoders compared here, including our own, possesses: zero-shot inference on unseen codes. Adopting the training architecture of FECCT, the schemes we propose could in principle inherit this zero-shot potential, which we regard as a direction for future work.

To visualise the performance gain of the proposed schemes over the other decoders, Figure 5 shows the BER performance curves of the different models on three representative codes (Polar(64,32), BCH(63,45) and CCSDS(128,64)). SECCT and HECCT are markedly better than all baselines.

Table 1: Performance comparison of the different methods at $E_b/N_0 = 4, 5, 6$ dB across 13 codes. Hyper-BP and AR-BP denote hyper-graph BP Nachmani & Wolf (2019) and autoregressive BP Nachmani & Wolf (2021), respectively.

Code	E_b/N_0	BP	Hyper-BP	AR-BP	ECCT	FECCT	SECCT	HECCT
Polar(64,32)	4 dB	4.26	4.59	5.57	6.99	5.88	7.28	**7.46**
	5 dB	5.38	6.10	7.43	9.44	7.91	9.65	**9.83**
	6 dB	6.50	7.69	9.82	12.32	10.76	13.04	**13.20**
Polar(64,48)	4 dB	4.74	4.92	5.41	6.36	6.06	6.39	**6.47**
	5 dB	5.94	6.44	7.19	8.46	8.21	8.59	**8.60**
	6 dB	7.42	8.39	9.30	11.09	10.90	11.10	**11.13**
Polar(128,64)	4 dB	4.10	4.52	4.84	5.92	-	5.99	**6.42**
	5 dB	5.11	6.12	6.78	8.64	-	8.84	**9.48**
	6 dB	6.15	8.25	9.30	12.18	-	12.60	**13.44**
Polar(128,86)	4 dB	4.49	4.95	5.39	6.31	5.53	**6.77**	6.69
	5 dB	5.65	6.84	7.37	9.01	7.90	**9.95**	9.61
	6 dB	6.97	9.28	10.13	12.45	11.29	**13.67**	13.15
Polar(128,96)	4 dB	4.61	4.94	5.27	6.31	-	6.54	**6.56**
	5 dB	5.79	6.76	7.44	9.12	-	9.43	**9.45**
	6 dB	7.08	9.09	10.20	12.47	-	**12.85**	12.84
LDPC(49,24)	4 dB	6.23	6.23	6.58	5.79	-	**6.73**	6.71
	5 dB	8.19	8.54	9.39	8.13	-	**9.65**	9.64
	6 dB	11.72	11.95	12.39	11.40	-	**13.38**	13.29
LDPC(121,60)	4 dB	4.82	5.22	5.22	5.01	-	**5.74**	5.70
	5 dB	7.21	8.29	8.31	7.99	-	**9.29**	9.19
	6 dB	10.87	13.00	13.07	12.78	-	**14.84**	14.77
LDPC(121,70)	4 dB	5.88	6.39	6.45	6.19	-	**7.10**	7.06
	5 dB	8.76	9.81	10.01	9.89	-	**11.56**	11.51
	6 dB	13.04	14.04	14.77	15.58	-	**17.35**	17.33
LDPC(121,80)	4 dB	6.66	6.95	7.22	7.07	-	**8.12**	8.08
	5 dB	9.82	10.68	11.03	10.96	-	**12.81**	12.67
	6 dB	13.98	15.80	15.90	16.25	-	**18.38**	18.32
MacKay(96,48)	4 dB	6.84	7.19	7.43	7.23	-	**8.33**	8.25
	5 dB	9.40	10.02	10.65	10.42	-	**12.16**	12.05
	6 dB	12.57	13.16	14.65	14.12	-	15.78	**15.83**
CCSDS(128,64)	4 dB	6.55	6.99	7.25	6.77	6.52	**7.93**	7.88
	5 dB	9.65	10.57	10.99	10.55	9.67	**12.45**	12.39
	6 dB	13.78	15.27	16.36	15.90	15.01	17.67	**17.68**
BCH(63,36)	4 dB	4.03	4.29	4.57	4.86	4.53	**5.02**	**5.02**
	5 dB	5.42	5.91	6.39	6.65	6.38	**6.86**	6.84
	6 dB	7.26	8.01	8.92	9.10	9.10	**9.43**	9.38
BCH(63,45)	4 dB	4.36	4.64	4.97	5.60	5.18	**5.87**	5.74
	5 dB	5.55	6.27	6.90	7.79	7.32	**8.11**	8.00
	6 dB	7.26	8.51	9.41	10.93	10.31	**11.47**	11.21

Table 2 places the complexity and accuracy of the three methods on Polar(64,32) side by side; MACs are multiply-accumulate operations per decoded codeword. SECCT stores a single shared layer instead of twelve independent ones, so its stored parameters amount to only 18% of those of the published 6-layer ECCT baseline (L=6, d=128; -82.0%) and, since the same layer is reused at every iteration, its weight memory at inference stays at that level; the price is a higher computational cost, as its MACs per codeword rise to $2.0\times$ the baseline ($+100\%$). HECCT reduces both quantities (parameters -40.5%, MACs -25.9%) while exceeding the baseline in $-\ln(\text{BER})$ at all three SNRs ($+0.47/+0.39/+0.88$). FECCT is not included in this comparison: it does not report detailed complexity figures, and no public code is available to reproduce them. According to its original description, ECCT carries about 15% more parameters than FECCT Choukroun & Wolf (2024); on



(a) Polar(64,32) (b) BCH(63,45) (c) CCSDS(128,64)

Figure 5: BER versus E_b/N_0 for three representative codes, comparing SECCT and HECCT with the baselines.

that basis the schemes we propose should still require fewer parameters than FECCT. Together these figures delimit the applicability of the two architectures: SECCT targets memory-constrained edge deployment, whereas HECCT suits settings constrained in both memory and compute.

Table 2: Efficiency and accuracy on Polar(64,32); the 4/5/6 dB columns are $-\ln(\text{BER})$. Parentheses give the change relative to the ECCT baseline and the absolute Δ within a row.

Method	**Params**	**MACs / codeword**	**4 dB**	**5 dB**	**6 dB**
ECCT (L=6, d=128)	1.21M (—)	127.4M (—)	6.99	9.44	12.32
SECCT	217K ($-82.0\%\downarrow$)	254.8M ($+100.0\%\uparrow$)	7.28 (+0.29)	9.65 (+0.21)	13.04 (+0.72)
HECCT	0.72M ($-40.5\%\downarrow$)	94.4M ($-25.9\%\downarrow$)	7.46 (+0.47)	9.83 (+0.39)	13.20 (+0.88)

4.3 Ablation Studies

We conduct dedicated ablations on Polar(64,32) to quantify the contribution of each design choice: the SECCT iteration depth, the number of KV head groups in GQA, and the FFN expansion ratio of the Bottleneck structure.

4.3.1 Iteration Depth of SECCT

A key advantage of SECCT is that the number of iterations N_{dec} can be adjusted at deployment time without changing the stored parameters. We evaluate $N_{\text{dec}} \in \{6, 9, 12, 15, 18\}$ with full-width attention (8 KV heads) and FFN expansion 4.0.

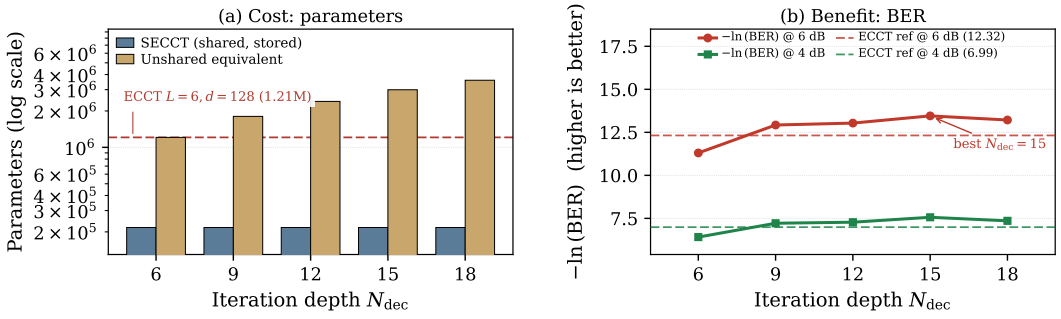


Figure 6: SECCT iteration-depth ablation on Polar(64,32). (a) Stored parameters versus an equally deep *unshared* decoder; the red dashed line marks the published ECCT baseline. (b) $-\ln(\text{BER})$ at 4 and 6 dB versus depth.

Figure 6(a) compares the stored parameters of SECCT with those of an equally deep unshared decoder: the former is constant at 217,409 regardless of depth, whereas the latter would require 1.21M to 3.59M parameters, so SECCT stores only 6–18% of the memory of an equally deep unshared

model and this ratio falls further as depth grows. Figure 6(b) shows that performance improves with depth, and every depth $N_{\text{dec}} \geq 9$ surpasses the published ECCT baseline by at least $+0.22$ in $-\ln(\text{BER})$ at 4 dB (N_{dec}=9; the gap grows with depth).

4.3.2 Number of KV Head Groups (GQA)

Table 3 ablates the number of KV head groups at a fixed depth of 12 layers and FFN expansion 0.5, with G=8 (full-width multi-head attention, MHA) as the reference. Reducing the number of KV groups leaves the result essentially unchanged and, at G=2, even improves it slightly at 5 and 6 dB while falling by 0.01 at 4 dB, consistent with the head-redundancy analysis of Section 3.2. Only full multi-query attention (MQA, G=1) degrades performance markedly. Grouped-query attention therefore cuts the KV projections by $4\times$ at a change of at most 0.01 in $-\ln(\text{BER})$.

Table 3: Ablation of the number of KV head groups G in GQA on Polar(64,32) (12 layers, FFN expansion 0.5); G=8 is the full-width MHA reference. Values are $-\ln(\text{BER})$.

Config	G	**FF**	**Params**	**4 dB**	**5 dB**	**6 dB**
G=8 (full-width MHA)	8	0.5	1,016,769	7.4750	9.8264	13.1806
G=4	4	0.5	818,625	7.4632	9.7980	13.1806
G=2	2	0.5	719,553	7.4609	9.8274	13.1985
G=1 (MQA)	1	0.5	670,017	7.3667	9.6781	12.9367

4.3.3 FFN Expansion Ratio (Bottleneck)

Table 4 ablates the FFN expansion ratio at G=2 and 12 layers.

Table 4: Ablation of the Bottleneck FFN expansion ratio on Polar(64,32) (G=2, 12 layers). Values are $-\ln(\text{BER})$.

Expansion	d'_f	**Params**	**MACs**	**4 dB**	**5 dB**	**6 dB**
0.25	32	620,865	85.3M	7.3101	9.5975	12.9817
0.5	64	719,553	94.4M	7.4609	9.8274	13.1985
1.0	128	916,929	113.3M	7.3874	9.6283	12.7107
2.0	256	1,311,681	151.0M	7.5620	9.7944	12.8200
4.0	512	2,101,185	226.5M	7.6548	9.9117	12.9231

Reducing the expansion ratio from 4.0 to 0.5 cuts the parameter count substantially while the accuracy is largely retained. Compressing further to 0.25 reduces the parameters by a further 13.7% but costs 0.15 in $-\ln(\text{BER})$ at 4 dB relative to 0.5, a larger accuracy drop than that between 4.0 and 0.5. This is consistent with the measurement of Section 3.2 that the activation energy of a full-width FFN concentrates in a few directions ($r_{90}(h) = 67$, only 13% of its capacity), so the intermediate layer can be narrowed without discarding information that the decoder relies on.

5 Conclusion

In this paper, we introduced SECCT and HECCT, two parameter-efficient Transformer architectures for error correction code decoding. SECCT stores about 18% of the parameters of the published 6-layer ECCT baseline (L=6, d=128) while surpassing it on all 39 code–SNR settings, whereas HECCT combines Grouped-Query Attention (G=2) with a Bottleneck Feed-Forward structure for higher BER at 40% fewer parameters and 26% fewer MACs. Experiments on 13 codes with ablations on iteration depth, KV-head count, and FFN expansion validate both architectures, demonstrating that parameter-efficient natural language processing techniques transfer to ECC decoding. Future work includes extending these architectures to longer code lengths.

AI Assistance Disclosure

In this work, we used generative AI tools to correct grammatical errors and to polish the writing of the manuscript. We have not used generative AI tools for any other purpose: in particular, we did not use them to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, support qualitative or thematic data analysis, interpret results, or assist with translation. We have reviewed all AI-assisted work: every AI-edited passage was checked against the authors' original text, and all equations, tables, figures and numerical results were verified by the authors and are reported from our own runs. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. In *Proc. 2023 Conf. Empir. Methods Nat. Lang. Process. (EMNLP'23)*, pp. 4895–4901, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298.

Erdal Arikan. Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels. *IEEE Trans. Inf. Theory*, 55(7):3051–3073, July 2009. ISSN 0018-9448.

Amir Bennatan, Yoni Choukroun, and Pavel Kisilev. Deep learning for decoding of linear codes-a syndrome-based approach. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1595–1599. IEEE, 2018.

Raj Chandra Bose and Dwijendra Kumar Ray-Chaudhuri. On a Class of Error Correcting Binary Group Codes. *Inf. Control*, 3(1):68–79, March 1960. ISSN 0019-9958.

Yoni Choukroun and Lior Wolf. Error Correction Code Transformer. In *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS'22)*, volume 35, pp. 38695–38705, New Orleans, LA, USA, November 2022.

Yoni Choukroun and Lior Wolf. A Foundation Model for Error Correction Codes. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 2153–2171, Vienna, Austria, April 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/093b793f0af498e8fa218eb5445daeb0-Paper-Conference.pdf.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal Transformers. In *International Conference on Learning Representations*, May 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Marc PC Fossorier and Shu Lin. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on Information Theory*, 41(5):1379–1396, 1995.

Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1): 21–28, 1962.

Marco Giordani, Michele Polese, Marco Mezzavilla, Sundeep Rangan, and Michele Zorzi. Toward 6G Networks: Use Cases and Technologies. *IEEE Commun. Mag.*, 58(3):55–61, March 2020. ISSN 0163-6804.

Tobias Gruber, Sebastian Cammerer, Jakob Hoydis, and Stephan ten Brink. On deep learning-based channel decoding. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6. IEEE, 2017.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-Efficient Transfer Learning for NLP. In *Proc. 36th Int. Conf. Mach. Learn. (ICML'19)*, pp. 2790–2799. PMLR, 2019.

Wei Jiang, Bin Han, Mohammad Asif Habibi, and Hans Dieter Schotten. The Road Towards 6G: A Comprehensive Survey. *IEEE Open J. Commun. Soc.*, 2:334–366, February 2021. ISSN 2644-125X.

Hyeji Kim, Yihan Jiang, Ranvir Rana, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Communication algorithms via deep learning. In *Sixth International Conference on Learning Representations (ICLR)*, 2018.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*, pp. 11574–11591, Addis Ababa, Ethiopia, April 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Eliya Nachmani and Lior Wolf. Hyper-Graph-Network Decoders for Block Codes. In *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS'19)*, pp. 1767–1777, 2019.

Eliya Nachmani and Lior Wolf. Autoregressive Belief Propagation for Decoding Block Codes. *arXiv preprint arXiv:2103.11780*, 2021.

Eliya Nachmani, Yair Beery, and David Burshtein. Learning to Decode Linear Codes Using Deep Learning. In *2016 54th Annual Allerton Conf. on Communication, Control, and Computing*, pp. 341–346, Monticello, IL, USA, September 2016.

Claude E. Shannon. A Mathematical Theory of Communication. *Bell Syst. Tech. J.*, 27(3):379–423, July 1948. ISSN 0005-8580.

Noam Shazeer. Fast Transformer Decoding: One Write-Head Is All You Need. *arXiv preprint arXiv:1911.02150*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, and Illia Polosukhin. Attention Is All You Need. In *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS'17)*, volume 30, pp. 5998–6008, Long Beach, CA, USA, December 2017.

Hao Zhang and Wei Tong. Channel Coding for 6G Extreme Connectivity—Requirements, Capabilities, and Fundamental Tradeoffs. *IEEE BITS Inf. Theory Mag.*, 3(1):1–12, March 2024. ISSN 2692-4080.