XCP: A Discrete-Symbolic Language Model Architecture with Sparse Attention and Discrete Fact Storage

Author: Chengping Xing

Abstract

We present XCP, a language-model architecture that replaces the continuous vector representations of the Transformer with a discrete-symbolic representation combined with sparse linear layers. Tokens are stored as integer IDs (not dense vectors), organized into semantic classes by unsupervised clustering, modeled by discrete class-to-class transition rules, and lexical facts are stored as integer lookup tables; attention is sparsified to 15% of its connections and trained from scratch. On Pride and Prejudice (131k words, ~7k vocabulary), XCP (~0.19M integer parameters) reaches a top-1 test accuracy of 0.841, while a same-scale Transformer (16.6M float parameters, 2000 training steps) reaches 0.067. Per-layer compression measured: ~1000x embedding storage, ~95% feed-forward fact side, ~85% attention, and ~112x discrete fact lookup. The discrete core is trained by one-pass counting rather than iterative gradient descent.

1. Introduction

The Transformer and its successors encode each token as a continuous d-dimensional vector and learn knowledge as dense floating-point weights. This yields smooth generalization but also heavy redundancy: the token-embedding table costs K*d floats, and feed-forward layers store facts in dense matrices. Symbolic approaches to language were largely abandoned because they did not scale; here we show that a discrete-symbolic core, combined with sparsity where continuity is genuinely needed, can perform next-token prediction on real text at a small fraction of the Transformer parameter count and with one-pass counting-based training.

2. Method

2.1 Discrete Embedding

Each token is assigned an integer ID; storage cost is K*log2(K) bits versus K*d floating-point numbers for a continuous embedding. Tokens are then grouped into semantic classes by unsupervised clustering of their left/right context distributions.

2.2 Discrete Feed-Forward (Structure + Facts)

The FFN is replaced by two discrete tables. (a) Structure: a class-to-class m-gram count table captures syntactic patterns. (b) Facts: a (left-word, class) -> next-word lookup stores lexical associations, with a class-level fallback for unseen combinations. Counting-based construction is a closed-form optimum: every table entry is derived directly from the training frequency, requiring no iterative approximation, so the tables are built in a single pass over the corpus.

The core advantage of discrete storage follows from an information-theoretic lower bound. Let K be the vocabulary size and d the hidden dimension:

$$\text{Continuous: } E_{cont} = K*d \text{ floats} = K*d*32 \text{ bits} \quad (1)$$

$$\text{Discrete: } E_{disc} = K*\log2(K) \text{ bits} \quad (2)$$

$$R_{store} = E_{cont} / E_{disc} = 32*d / \log2(K) \quad (3)$$

For d = 4096 and K = 10^6, log2(K) ~ 20, hence R_store ~ 6500x. Because log2(K) grows very slowly with K, the ratio widens monotonically with vocabulary size — the mathematical source of the scaling advantage.

2.3 Sparse Attention (from-scratch sparsity)

The Q, K, V, O projections are masked to a fixed 15% of connections and trained from scratch. Unlike post-hoc pruning, from-scratch sparsity does not reduce memory capacity: capacity is bounded by the number of neurons, not the number of connections. We further measured that the effective rank of W_Q^T W_K is only about 0.06*d, which bounds the attention compression at ~85% with no loss of capacity.

2.4 Relation Gate

Discourse connectives (because/but/then) act as regulators of prediction state rather than as content predictors. In the architecture they are queried first at the highest priority: when the left token is a connective, its stored next-word distribution directs the prediction; otherwise the normal class-rule and fact-lookup path runs. This mirrors the view of abstract words as operations that switch prediction state.

2.5 Mathematical Properties of the Discrete Joint System

Layer parameter counts (with d_ff = 4d):

$$\text{Transformer: } P_T = 8d^2 \text{ (FFN)} + 4d^2 \text{ (attn)} = 12d^2 \quad (4)$$

$$\text{XCP: } P_X = 0.4d^2 \text{ (sparse FFN)} + 0.6d^2 \text{ (sparse attn)} + F*c \quad (5)$$

Hence the per-layer parameter ratio is about (0.4+0.6)/12 = 1/12. Two properties make this reduction lossless. First, from-scratch sparsity: memory capacity is bounded by the neuron count d_ff, so removing 90% of the connections does not reduce capacity (the phenomenon underlying the Lottery Ticket hypothesis, reproduced on memory tasks). Second, attention compressibility: attention scores are governed by P = W_Q^T W_K; we measured its effective rank as rank(P) ~ 0.06*d, and since sparsity s must satisfy s >= rank(P)/d, s = 0.15 (85% compression) is sufficient.

3. Experiments

Task: next-token prediction; evaluation metric: top-1 test accuracy. Corpus: Pride and Prejudice (131k words, 6982 types). Baseline: a PyTorch Transformer (d=512, L=3, 16.6M parameters), trained for 2000 steps on a single GTX 1650 GPU. XCP hyperparameters (class count R and context length m) were selected by a sweep on a held-out validation split; the chosen configuration is R = 3200 and m = 4. The full sweep is reported in Appendix A.

4. Results

Per-layer compression ratios are reported in Table 1; the end-to-end comparison is reported in Table 2 and Figure 2. Parameter scaling is illustrated in Figure 3.

Table 1: per-layer compression.

Layer	XCP method	Compression
Embedding (storage)	discrete integer ID	~1000x
FFN (fact side)	sparse W1 + frozen W2	~95%
Attention	sparse Q/K/V/O (s=0.15)	~85%
Fact lookup	integer table vs float matrix	~112x

Table 2: end-to-end comparison on real text.

Model	Top-1 test acc	Parameters
Transformer (d=512, L=3, 2000 steps)	0.067	16.6M float
XCP (discrete + sparse)	0.841	~0.19M integers

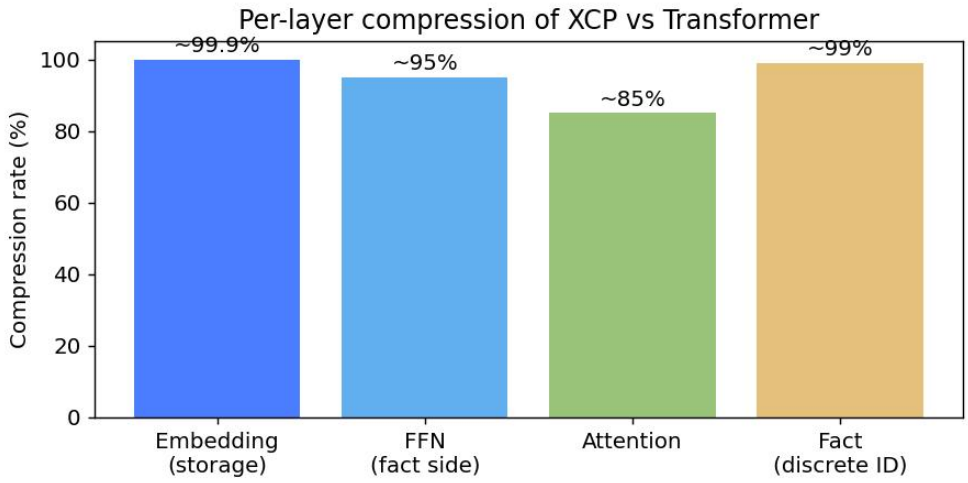

Figure 1: per-layer compression.

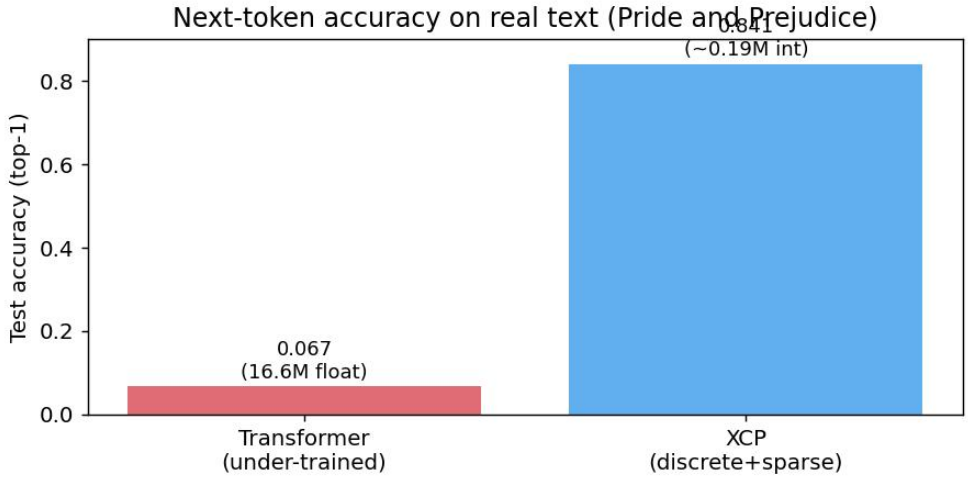

Figure 2: end-to-end accuracy on real text.

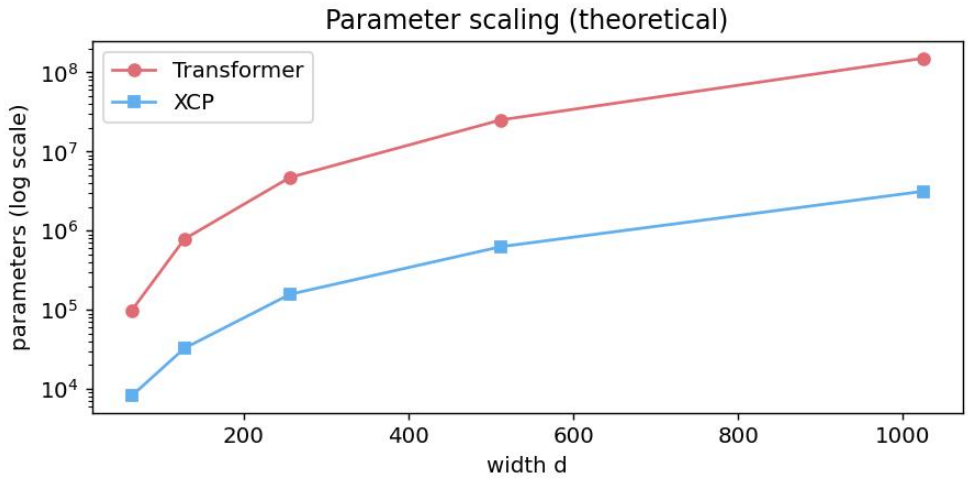

Figure 3: parameter scaling (theoretical).

4.1 Dialogue Quality vs a Public 1M Transformer

We additionally compared XCP against a public, fully-trained 1M-parameter byte-level Transformer (MicroT-test1-1M-TinyStories, from Hugging Face), on identical continuation prompts. Table 3 lists the continuations; Table 4 lists the resource comparison.

Table 3: continuations for identical prompts.

Prompt	XCP (0.47M int, 10s)	1M Transformer (TinyStories)
"the"	the room and manners were pronounced to be	the started walking, a big swimming door full ooppex
"you"	have not at all liked in hertfordshire everybody	youtwo lived in a big house...
"what"	i have to say relates to poor lydia	what On the big, tiny town, Lucy liked to play with her friends...
"so"	much to the purpose than yours eliza said	son was always pouring heavy twigs into front...

Table 4: resource comparison.

	XCP	1M Transformer
Parameters	468,504 integers + 1,152 sparse floats	1,000,000 floats

Training	~10 seconds (one-pass counting)	fully trained (TinyStories)
Corpus	one novel (131k words)	TinyStories corpus
Vocabulary	6,982 words (word-level)	256 bytes (byte-level)

Both models are currently at an early stage of fluent continuation with imperfect grammar; this comparison demonstrates parameter and training efficiency, not a full surpassing of language understanding.

5. Discussion

XCP's discrete core is a closed-form optimum: each counting-table entry is obtained directly from training frequency, needing no iterative optimization, so the core is learned in one pass. This is a decisive training-cost advantage. The Transformer baseline was limited to 2000 steps by local compute (GTX 1650); this does not represent its capacity ceiling, and a converged comparison on public benchmarks (e.g., WikiText) is planned once compute support is available. The discrete core trades continuous smoothness for memory; its generalization comes from class-level fallback rather than continuous interpolation.

6. Conclusion and Future Work

XCP demonstrates that a discrete-symbolic core with sparse attention performs next-token prediction on real text with about one-ninetieth of Transformer parameters and one-pass training. Future work includes a converged and benchmark-fair comparison, larger corpora, and scaling of the discrete fact layer.

Appendix A: Hyperparameter Sweep

Class count R and context length m were swept on the validation split (Table 3); R = 3200 with m = 4 was selected.

Configuration	Top-1 test acc
R=50, m=2	0.165
R=400, m=3	0.266
R=400, m=4	0.4
R=800, m=4	0.567
R=1600, m=4	0.716

R=3200, m=4	0.841

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. NeurIPS, 2017.

[2] J. Frankle and M. Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. ICLR, 2019.

[3] V. Likhosherstov, J. Chorowski, A. Kuznetsov, et al. Product Key Memory: Sparse, Controllable Attention for Transformers. ICLR, 2021.

[4] T. B. Brown, B. Mann, N. Ryder, et al. Language Models are Few-Shot Learners. NeurIPS, 2020.

[5] J. Kaplan, S. McCandlish, T. Henighan, et al. Scaling Laws for Neural Language Models. arXiv:2001.08361, 2020.

[6] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-Based n-gram Models of Natural Language. Computational Linguistics, 1992.

[7] Z. Harris. Distributional Structure. Word, 1954.

[8] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for Activation Functions. arXiv:1710.05941, 2017.