Transformers Learn to Compress Variable-order Markov Chains in-Context

Anonymous Author(s)
Affiliation
Address
email

Abstract

In recent years, large language models (LLMs) have demonstrated impressive in-context learning (ICL) capability. However, it is still unclear how the underlying transformers accomplish it, especially in more complex scenarios. Toward this goal, several recent works studied how transformers learn fixed-order Markov chains (FOMC) in context, yet natural languages are more suitably modeled by variable-order Markov chains (VOMC), i.e., context trees (CTs). In this work, we study the ICL of VOMCs by viewing language modeling as a form of data compression and focusing on small alphabets and low-order VOMCs. This perspective allows us to leverage mature compression algorithms, such as the context-tree weighting (CTW) algorithm as a baseline, which is Bayesian optimal for a class of priors. We empirically observe that the performance of transformers is not very sensitive to the number of layers, and even a two-layer transformer can learn in context quite well, tracking closely the performance of CTW. We provide a construction with $D + 2$ layers that can mimic the CTW algorithm accurately for VOMCs of maximum order D. One distinction from the FOMC setting is that a counting mechanism plays an important role in this setting.

1 Introduction

Large language models (LLMs) are capable of completing various tasks (Kasneci et al., 2023; Wu et al., 2023; Thirunavukarasu et al., 2023; Wei et al., 2022). The transformer model (Vaswani et al., 2017), the key behind current prevailing LLMs, is known to have strong in-context learning (ICL) capabilities, and concrete ICL results for transformers have been established for some simple tasks (Garg et al., 2022; Von Oswald et al., 2023; Bai et al., 2024; Ahn et al., 2024). Despite these results, the mechanism for transformers to learn in context is still not fully understood, especially when the scenario is complex or the sequences have memories. Toward this goal, several recent works studied how transformers can learn fixed-order Markov chains (FOMCs) either in training or in-context (Makkuva et al., 2024; Edelman et al., 2024), where insightful observations and theoretical results were obtained. The FOMC is however a poor match for natural languages, for which variable-order Markov chains (VOMCs), also known as context tree (CT) models (Rissanen, 1983; Willems et al., 1995), are often viewed as a more suitable model (Begleiter et al., 2004).

To this end, we study the ICL of transformers on VOMCs from the perspective of compression, motivated by a recent work connecting language models and data compression (Delétang et al., 2023). We therefore use compression rates in a fixed context window as our main evaluation metric. This allows us to use several well-known compression algorithms, particularly the context weighting (CTW) algorithm (Willems et al., 1995), as a baseline. The CTW algorithm is Bayesian optimal under certain priors, which gives us a fundamental lower bound in such settings. Appendix A gives a more detailed discussion on related works.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

We first train a set of shallow transformers of various numbers of layers for VOMCs of various maximum orders, and empirically observe that the performance of transformers is not very sensitive to the number of layers, and even a two-layer transformer can learn in context quite well. We then answer the question of whether transformers can mimic the CTW algorithm. For this purpose, we first propose an alternative representation of CTW next token prediction, based on which a transformer construction with $D+2$ layers is given, that can mimic CTW accurately for VOMCs of maximum order D. This establishes a fundamental capability of transformers for ICL-VOMC. The alternative representation enjoys an intuitive interpretation as blending probability estimates along a path on the context tree.

Main Contributions: (i) We believe that ours is the first study of ICL for VOMC and we demonstrate that transformers can indeed (numerically) learn to compress VOMC in-context, close to optimal CTW algorithm for appropriate CTW-prior. (ii) we give an explicit $D+2$-layer transformer construction to imitate CTW, based on a novel Bayesian optimal next token prediction representation, which can be of independent interests.

2 Preliminaries

2.1 The Transformer Model

Transformer interacts with sequential data, e.g., $x_1^N = (x_1, \ldots, x_N)$, where token x_i is a symbol from an alphabet (a.k.a. vocabulary) $\mathcal{A}$ with $A = |\mathcal{A}|$. Each token x_i is embedded into $\mathbf{h}_i^{(1)} \in \mathbb{R}^E$ by integrating the information of its value x_i and position i, where E is the embedding dimension.

We introduce an L-layer decoder-only transformer model. Each layer of the transformer takes matrix $\mathbf{H}^{(\ell)} = [\mathbf{h}_1^{(\ell)}, \mathbf{h}_2^{(\ell)}, \ldots, \mathbf{h}_N^{(\ell)}]$, where $\mathbf{h}_i^{(\ell)} \in \mathbb{R}^E$, as its input and applies the multi-head attention layer operation and the feed-forward layer operation, and the output of the layer is the input to the next layer, denoted as $\mathbf{H}^{(\ell+1)}$. The decoder-only multi-head attention layer with $M^{(\ell)}$ heads is

$$\mathbf{a}_i^{(\ell)} = \mathrm{MHA}\left(\mathbf{h_i}, \mathbf{H}; \{W_{O,m}^{(\ell)}, W_{Q,m}^{(\ell)}, W_{K,m}^{(\ell)}, W_{V,m}^{(\ell)}\}_{m=1}^{M^{(\ell)}}\right) \triangleq W_O^{(\ell)} \left[\mathbf{b}_{1,i}^{(\ell)}; \mathbf{b}_{2,i}^{(\ell)}; \ldots; \mathbf{b}_{M^{(\ell)},i}^{(\ell)}\right], \quad (1)$$

where $\{W_{Q,m}^{(\ell)}, W_{K,m}^{(\ell)}, W_{V,m}^{(\ell)}\}_{m=1}^{M^{(\ell)}}$ are the $E^{(\ell)} \times E$ query matrices, key matrices, and value matrices[1] at the ℓ-th layer and m is the index of the attention head, respectively, $W_O^{(\ell)}$ is the $E \times M^{(\ell)} E^{(\ell)}$ output mapping matrix, and $\mathbf{b}_m^{(\ell)}$ is the output of the m-th attention head at this layer defined as

$$\mathbf{b}_{m,i}^{(\ell)} = (W_{V,m}^{(\ell)}[\mathbf{h}_1^{(\ell)}, \mathbf{h}_2^{(\ell)}, \ldots, \mathbf{h}_i^{(\ell)}]) \cdot \mathrm{softmax}((W_{K,m}^{(\ell)}[\mathbf{h}_1^{(\ell)}, \mathbf{h}_2^{(\ell)}, \ldots, \mathbf{h}_i^{(\ell)}])^\top (W_{Q,m}^{(\ell)} \mathbf{h}_i^{(\ell)})), \quad (2)$$

where we used ";" to indicate vertical matrix concatenation and "," to indicate horizontal matrix concatenation. The attention layer has a residual connection, and the attention output together with the residual connection also goes through a feedforward layer with a residual connection

$$\mathbf{h}_i^{(\ell+1)} = \mathrm{FF}(\mathbf{a}_i; W_1^{(\ell)}, W_2^{(\ell)}) = W_1^{(\ell)} \sigma(W_2^{(\ell)}(\mathbf{a}_i^{(\ell)} + \mathbf{h}_i^{(\ell)})) + (\mathbf{a}_i^{(\ell)} + \mathbf{h}_i^{(\ell)}), \quad (3)$$

where σ is a non-linear activation function (e.g., ReLU or sigmoid). The output of the last (L-th) transformer layer $\mathbf{H}^{(L+1)}$ goes through a linear then softmax unit to predict the probability of generating the next symbol in vocabulary $\mathcal{A}$ based on the past observations:

$$\hat{\mathbf{p}}_{i+1} = \mathrm{softmax}(W_O^{(L+1)} \mathbf{h}_i^{(L+1)}) \in \Delta_A, \quad i = 1, \ldots, N-1, \quad (4)$$

where Δ_A is the probability simplex on $\mathcal{A}$. The model is illustrated in Appendix C.

2.2 Context Tree Models (Variable-Order Markov Chains)

Variable-order Markov chains (VOMCs), also known as context tree (CT) models, have been studied extensively in the data compression literature (Rissanen, 1983; Willems et al., 1995; Begleiter et al., 2004). String $s = (x_{1-l}, x_{2-l}, \ldots, x_0)$ is a suffix of the string $s' = (x'_{1-l'}, x'_{2-l'}, \ldots, x'_0)$, if $0 \le l \le l'$ and $x_{-i} = x'_{-i}$ for $i = 0, 1, \ldots, l-1$; e.g., (a, b, c, b) is suffix of (a, c, a, a, b, c, b).

A CT source is specified by a suffix set $\mathcal{S}$ and the associated probability distributions. The suffix set is a collection of strings $s(k)$, $k = 1, \ldots, |\mathcal{S}|$, which needs to be proper and complete: The set is

[1] In practice, embedding dimension E is divisible by the number of heads $M^{(\ell)}$ and $E = M^{(\ell)} E^{(\ell)}$.

proper if no string in $\mathcal{S}$ is a suffix of any other string; it is complete if each semi-infinite sequence $(\ldots, x_{n-1}, x_n)$ has a unique suffix that belongs to $\mathcal{S}$, denoted as $\beta_{\mathcal{S}}(\ldots, x_{n-1}, x_n)$. Associated with each suffix $s \in \mathcal{S}$, there is a probability mass function $p_s \in \Delta_{\mathcal{A}}$. A CT has maximum order D if any suffix in $\mathcal{S}$ has has length at most D. Given a semi-infinite sequence $(\ldots, x_{n-1}, x_n)$, the next symbol x_{n+1} is generated randomly according to the distribution $p_{\beta_{\mathcal{S}}(\ldots,x_{n-1},x_n)}$. An example CT is in Fig. 5 in the appendix. For each suffix set $\mathcal{S}$, there is a unique tree T with suffix set $\mathcal{S}$ being its leaves $\mathcal{L}(T)$, and a CT can thus be represented by $(T, \{p_s\}_{s\in\mathcal{L}(T)})$.

2.3 Bayesian Context Tree Weighting Compression Algorithm

Once the underlying CT is estimated accurately, arithmetic coding (AC) can be used to compress the sequence efficiently. The likelihood of a sequence x_1^i given x_{1-D}^0 for a CT with parameter $(T, \{p_s\}_{s\in\mathcal{L}(T)})$ is $P_{T,\{p_s\}}(x_1^i|x_{1-D}^0) = \prod_{j=1}^i p_{\beta_{\mathcal{L}(T)}(x_{j-D},\ldots,x_{j-1})}(x_j) = \prod_{s\in\mathcal{L}(T)} \prod_{a\in\mathcal{A}} p_s(a)^{\mathbf{n}_{i,s}(a)}$, where $\mathbf{n}_{i,s}$ is the *counting vector* associated with suffix s that

$$\mathbf{n}_{i,s}(a) := \text{number of times symbol } a \in \mathcal{A} \text{ follows suffix } s \text{ in sequence } (x_1, \ldots, x_i). \tag{5}$$

Willems et al. (1995) proposed the context tree weighting (CTW) algorithm for CT sources. CTW estimates the probability of the sequence x_1^n by the auxiliary parameters p^e, p^w's as follows.

1. For each $s \in \mathcal{A}^*$ with $|s| \le D$, compute $p_{n,s}^e = \frac{\Gamma(\sum_{a\in\mathcal{A}}\alpha(a))}{\Gamma(\sum_{a\in\mathcal{A}}(\mathbf{n}_s(a)+\alpha(a)))} \prod_{q\in\mathcal{A}} \frac{\Gamma(\mathbf{n}_s(a)+\alpha(a))}{\Gamma(\alpha(a))}$, where $\mathbf{n}_s$ is the counting vector $\mathbf{n}_{i,s}$ with $i = n$, and $\Gamma(\cdot)$ is the Gamma function.
2. From nodes in the D-th level to the 0-th level (i.e., root), iteratively compute

$$p_{n,s}^w := \begin{cases} p_{n,s}^e, & \text{if } |s| = D, \\ \lambda p_{n,s}^e + (1-\lambda)\prod_{q\in\mathcal{A}} p_{n,qs}^w, & \text{otherwise,} \end{cases} \tag{6}$$

   where qs is the string by appending symbol $q \in \mathcal{A}$ before the suffix s.

Kontoyiannis et al. (2022) took the Bayesian view towards this procedure under a CTW prior. CTW prior π_{CTW} is a Bayesian CT prior over the trees in $\mathcal{T}(D) := \{\text{full } A\text{-ary tree with depth at most } D\}$ and the transition distributions $p_s \in \Delta_{\mathcal{A}}$. Specifically, $\pi_{\text{CTW}}(T, (p_s)_{s\in\mathcal{L}(T)}) = \pi_D(T)\prod_{s\in\mathcal{L}(T)} \pi_p(p_s)$ with

$$\pi_D(T) = (1-\lambda)^{(|\mathcal{L}(T)|-1)/(A-1)}\lambda^{|\mathcal{L}(T)|-|\mathcal{L}_D(T)|}, \qquad \pi_p(p_s) = \text{Dir}(p_s; \{\alpha(a)\}_{a\in\mathcal{A}}).$$

$\pi_D(\cdot)$ represents a bounded branching process with stopping probability λ for each node; and $\mathcal{L}_D(T)$ is the leaves of T with depth D. The next token probability p_s follows a Dirichlet prior parameterized by $\{\alpha(a)\}$. Kontoyiannis et al. (2022) showed that the $p_{n,()}^w$ at root computed by CTW equals to the Bayesian predicted probability under CTW prior, i.e., $p_{n,()}^w = P_{\pi_{\text{CTW}}}(x_1^n|x_{1-D}^0) = \sum_{T\in\mathcal{T}(D)} \int P_{T,\{p_s\}}(x_1^n|x_{1-D}^0)\pi(T,\{p_s\})\Big(\prod_{s\in\mathcal{L}(T)} \mathrm{d}p_s\Big)$. AC can be applied via sequentially calculating the predictive next token probability as $P_{\pi_{\text{CTW}}}(x_{i+1}|x_{1-D}^i) = \frac{P_{\pi_{\text{CTW}}}(x_1^{i+1}|x_{1-D}^0)}{P_{\pi_{\text{CTW}}}(x_1^i|x_{1-D}^0)}$.

3 Transformers Learn In-context of VOMCs

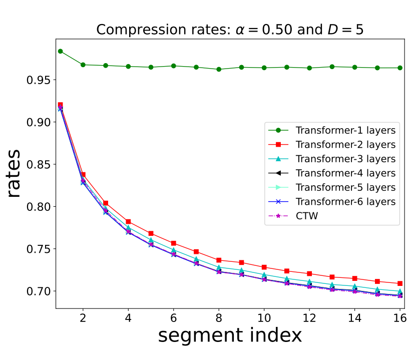


Figure 1: Transformer vs CTW

We choose ternary alphabet $|\mathcal{A}| = 3$, and pretrain a transformer of context window size N on data sequences of length-N generated using CTs randomly sampled from a CTW prior π_{CTW} parameterized by $\alpha = 0.5$, $\lambda = 0.15$ and a fixed maximum tree depth D, illustrated in Fig. 7 in Appenidx D. The training loss is the canonical next-token prediction cross-entropy loss. During the inference, given a source sequence of length-N generated from an unknown VOMC with the order at most D, can the transformer compress this sequence efficiently, i.e., at a compression rate close to the optimal rate?

In Fig. 1, we show the performance comparisons between trained transformers with various numbers of layers and the reference CTW algorithm. Experimental details are in Appendix D.

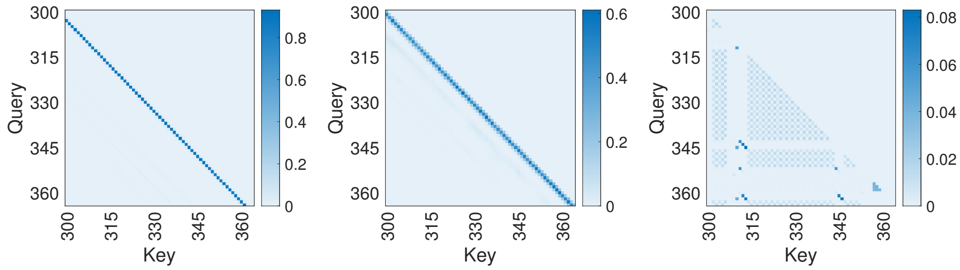


Figure 2: Partial attention heatmaps for different attention heads.

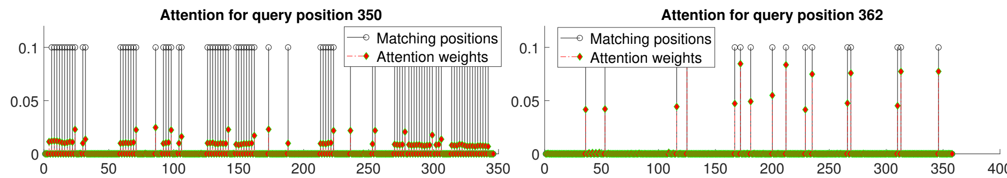


Figure 3: Suffix locations and attention weights in the second type of pattern at two query positions.

We observe that almost all trained transformers, except that with a single layer, track the performance of the CTW algorithm fairly closely. The overall performance does improve as the number of layers increases in general; see Table 1 in the Appendix D.1 for numerical comparisons. Nevertheless, the improvements with increased numbers of layers are relatively small. Even transformers with two layers appear to learn in context quite well.

4 Theoretical Interpretations and Empirical Evidences

4.1 Analysis of Attention Maps

To understand why and how the trained transformers perform comparable CTW, we first analyzed the attention maps of the trained transformers where two distinguished patterns emerge. One pattern is solely relative-position dependent. In the left two panels of Fig. 3, we observe off-diagonal stripes for these two attention heads, which are a few positions below the main diagonal. They can be a single off-diagonal or a collection of several off-diagonals. This indicates that the query position is attending positions at a few fixed but close distances ahead of itself. This pattern usually appears in the first or second layers of the transformers. Combining with the suffix structure in compression algorithms such as CTW, such an attention pattern suggests the suffix is being copied into the current query position for subsequent processing.

Another pattern, shown in the third panel has more sophisticated spotty patterns, and the attention appears to depend more explicitly on the current token features instead of the position alone, and they usually appear in the second layer or above in the transformers. Taking query positions 350 and 362 for the attention head shown in the third panel of Fig. 2, we plot in Fig. 3 the positions in the data sequence that match their suffixes of length-3 using the stem plots with a black circle on top, and the attention values as the red stems with the diamonds on top. This attention pattern suggests that it is collecting information for those positions with the matched suffix of a fixed length.

4.2 Capability and capacity of transformer via construction

Given a sequence x_1^n generated according to a $\mathrm{CT}(T, \{p_s\})$ sampled from the CTW-prior π_{CTW} parameterized by (D, λ, α), we propose a novel representation for computing the predictive probability $P_{\pi_{\text{CTW}}}(x_{n+1}|x_{1-D}^n)$ in the following theorem. It is based on the weighted average of the next token prediction probability vector of each potential suffix $s_{n,l} := x_{n-l+1}^n$ of length $l = 0, 1, \ldots, D$. The proof of Theorem 1 is in Appendix E.1.

Theorem 1. *The predicted probability can be computed as*

$$P_{\pi_{CTW}}(x_{n+1}|x_1^n) = \sum_{l=0,\ldots,D} \omega_{n,l} \cdot \mathbf{p}_{n,s_{n,l}}(x_{n+1}), \tag{7}$$

where $\mathbf{p}_{n,s_{n,l}}(a) = \frac{\alpha(a)+\mathbf{n}_{n,s_{n,l}}(a)}{\sum_q(\alpha(q)+\mathbf{n}_{n,s_{n,l}}(q))}$*; and* $\omega_{n,\cdot} \in \Delta_{D+1}$ *with* $\ln(\omega_{n,l}) - \ln(\omega_{n,l-1}) = \ln(1-\lambda) - \mathbb{I}_{l=D}\ln(\lambda) + \ell^e_{n,s_{n,l}} - \ell^e_{n,s_{n,l-1}} + \sum_{q\in\mathcal{A}} \ell^w_{n,qs_{n,l-1}} - \ell^w_{n,s_{n,l}}$*, where* $\ell^e_{n,s} = \ln(p^e_{n,s})$*,* $\ell^w_{n,s} = \ln(p^w_{n,s})$*.*

As illustrated in Fig. 4, each suffix $s_{n,l}$, e.g., $s_{n,0} = (), s_{n,2} = ba$, can potentially be the true suffix of the underlying CT dynamics, i.e., $s_{n,l} \in \mathcal{L}(T)$; and $\mathbf{p}_{n,s_{n,l}}$ is in fact the Bayesian optimal next token prediction given $s_{n,l} \in \mathcal{L}(T)$. The weights $\omega_{n,l}$ assign credibility that $s_{n,l}$ is the true suffix. Theorem 1 suggests that the weights are based on both the information in the suffix path such as $p^e_{s_{n,s_{n,l}}}$ as well as the information from their siblings $p^w_{n,qs_{n,l-1}}$ (siblings are in triangles in Fig. 4). The information of counting vector $\mathbf{n}_{n,s}$ plays a vital role since $\mathbf{p}_{n,s}$, $p^e_{n,s}$, e.t.c. are all functions of $\mathbf{n}_{n,s}$.

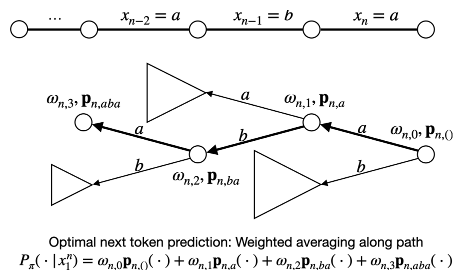


Figure 4: Illustration of Theorem 1

4.3 Transformer construction: Approximating CTW

We provide a construction of $(2 + D)$-layer transformer with sufficient representation power in the FF layer that can essentially approximate CTW, demonstrating the capacity of the transformer. The first two layers are motivated by the attention map patterns observed in Section 4.1, which we show their capabilities of capturing the important counting vector statistics suggested by Theorem 1. The last D layers are induction layers imitating the CTW procedure.

We consider the initial embedding is one-hot, with additional scratch pad elements initialized as zeros and a positional embedding, i.e., $\mathbf{h}_i^{(1)} = (\mathbf{x}_i; \mathbf{0}; \mathbf{pos}_i)$ where $\mathbf{x}_i \in \mathbb{R}^A$ is the one-hot (column vector) embedding of x_i , $\mathbf{pos}_i = (1, \cos(i\pi/N), \sin(i\pi/N))^\top$ is a positional embedding, and the remaining $(E - A - 3)$ elements being zero. The proofs of this section are in Appendix E.2.

We begin with the first layer, which is referred to as a *finite-memory context-extension layer*.

Theorem 2. *There is an M-headed transformer layer that can perform finite-memory context extension, defined by the following output, with the initial one-hot embedded input $\mathbf{H}^{(1)}$:*

$$\mathbf{h}_i^{(2)} = (\mathbf{s}_{i,M+1}; \mathbf{0}; \mathbf{pos}_i), \tag{8}$$

where $\mathbf{s}_{i,M+1} = (\mathbf{x}_i; \dots; \mathbf{x}_{i-M})$ is the vector version of the M-length suffix $s_{i,M+1} = x_{i-M}^i$.

This layer copies and stacks M past embedded symbols to the current position i. It utilizes the positional encoding $\mathbf{pos}_i$ via rotation and matching the corresponding positions.

The second layer is referred to as the *statistics collection layer*, which takes a sequence of vectors $\mathbf{h}_i^{(2)}$, $i = 1, \dots, N$, defined in (8) as its input. To rigorously specify the function of this layer, we define the forward and backward statistics vectors at position i,

$$\mathbf{g}_{i,s}(a) = \frac{\mathbf{n}_{i,s}(a)}{\sum_{q \in \mathcal{A}} \mathbf{n}_{i,s}(q)}, \qquad \mathbf{g}^{\leftarrow}_{i-1,s}(a) = \frac{\sum_{q \in \mathcal{A}} \mathbf{n}_{i,as}(q)}{\sum_{q \in \mathcal{A}} \mathbf{n}_{i,s}(q)}, \qquad \forall a \in \mathcal{A}, \tag{9}$$

where $\mathbf{n}_{i,s}$ is the counting vector defined in (5), and $\sum_{q \in \mathcal{A}} \mathbf{n}_{i,s}(q)$ is the number of appears of the string s in the sequence x_1^{i-1}. In plain words, with $|s| = k - 1$ they are the empirical probability of the next and previous token associated with the suffix s in the k-gram statistics seen before x_i. For both $\mathbf{g}_{i,s}$ and $\mathbf{g}^{\leftarrow}_{i-1,s}$, if the suffix s has not appeared in data x_1^{i-1}, it can be initialized arbitrarily as a vector in the probability simplex.

Theorem 3. *There is an M'-head attention layer, where $M' \leq M + 1$, that can perform statistics collection, defined by the following output, with $\mathbf{H}^{(2)}$ in (8) as its input:*

$$\mathbf{a}_i^{(2)} = (\mathbf{s}_{i,M+1}; \mathbf{g}_{i,M'}; \mathbf{g}^{\leftarrow}_{i-1,M'}; \mathbf{0}; \mathbf{pos}_i), \tag{10}$$

where $\mathbf{g}_{i,M'} := (\mathbf{g}_{i,s_{i,0}}; \dots; \mathbf{g}_{i,s_{i,M'-1}})$ and $\mathbf{g}^{\leftarrow}_{i-1,M'} = (\mathbf{g}^{\leftarrow}_{i-1,s_{i,0}}; \dots; \mathbf{g}^{\leftarrow}_{i-1,s_{i,M'-1}})$.

This functional layer essentially collects k-gram statistics for various lengths of $k = 1, 2, \dots, M'$. For example, when $k = 3$, it collects the normalized frequency associated with the suffix (x_{n-1}, x_n).

For ICL of FOMCs, two-layer transformers collecting forward statistics $\mathbf{g}_{i,M'}$ with $M' = D + 1$ is sufficient (Edelman et al., 2024). However, for the ICL-VOMC task, the underlying CT structure is unknown, therefore, collecting such simple statistics is no longer sufficient. As indicated in Theorem 1, the information of counting statistics $\mathbf{n}_{i,s_{i,l}}$ is important to the performance of prediction since the weights heavily depend on $\mathbf{n}_{i,s}(a)$. Yet due to the softmax function of the attention layer, only (normalized) probabilistic vector can be obtained instead of the exact count. With the backward statistics $\mathbf{g}^{\leftarrow}_{i,s}$, $\mathbf{n}_{i,s_{i,l}}$ can be derived as $\mathbf{n}_{i,s_{i,l}}(a) = \frac{\mathbf{n}_{i,s_{i,l}}(a)}{\sum_{q\in\mathcal{A}}\mathbf{n}_{i,s_{i,l}}(q)} \frac{\sum_{q\in\mathcal{A}}\mathbf{n}_{i,s_{i,l}}(q)}{\sum_{q\in\mathcal{A}}\mathbf{n}_{i,s_{i,l-1}}(q)} \cdots \frac{\sum_{q\in\mathcal{A}}\mathbf{n}_{i,s_{i,1}}(q)}{\sum_{q\in\mathcal{A}}\mathbf{n}_{i,s_{i,0}}(q)} \left(\sum_{q\in\mathcal{A}}\mathbf{n}_{i,s_{i,0}}(q)\right) = \mathbf{g}_{i,s_{i,l}}(a)\left(\prod_{j=0}^{l-1}\mathbf{g}^{\leftarrow}_{i-1,s_{i,j}}(x_{i-j})\right) i$, by the information contained in vector $\mathbf{a}_i^{(2)}$.

Taking $M = M' - 1 = D$ and a sufficiently wide FF layer in the second transformer layer, we have

$$\mathbf{h}_i^{(3)} = (\mathbf{s}_{i,D}; \mathbf{p}_{i,D}; \mathbf{l}^e_{i,D}; \ell^w_{i,s_{i,D}}; \mathbf{0}; \mathbf{pos}_i), \tag{11}$$

where $\mathbf{p}_{i,D} = (\mathbf{p}_{i,s_{i,0}}; \ldots; \mathbf{p}_{i,s_{i,D}})$ and $\mathbf{l}^e_{i,D} = (\ell^e_{i,s_{i,0}}; \ldots; \ell^e_{i,s_{i,D}})$, by universal approximation.

To fulfill the Bayesian optimal prediction, we introduce the following *CTW induction layer* that iteratively computes $\ell^w_{i,s}$ on the suffix path and their siblings, and also the weight difference $\delta_{i,l} := \ln(\omega_{i,l}) - \ln(\omega_{i,l-1})$ for $l = d, D-1, \ldots, 1$. The desired embedding for $\ell = 3, 4, \ldots, 3+D$ is

$$\mathbf{h}_i^{(\ell)} = (\mathbf{s}_{i,M^{(1)}+1}; \mathbf{p}_{i,D}; \mathbf{l}^e_{i,D}; \delta_{i,D}; \delta_{i,D-1}; \ldots; \delta_{i,D-\ell+4}; \ell^w_{i,s_{i,D+3-\ell}}; \mathbf{0}; \mathbf{pos}_i). \tag{12}$$

Theorem 4. *There exists a A-head transformer layer that can perform the induction: Takes $\mathbf{H}^{(\ell)}$ in (12) as input and outputs $\mathbf{H}^{(\ell+1)}$. And the final output layer taking $\mathbf{H}^{(D+3)}$ as input can output the A-dimensional Bayesian optimal next token prediction vector $P_{\pi_{CTW}}(\cdot|x^n_{1-D}) = \sum_{l=0,\ldots,D} \omega_{n,l}\mathbf{p}_{n,s_{n,l}}$.*

Although transformers with sufficient FF layers can theoretically compute the optimal prediction as CTW, empirically, transformers of $2 + D$ layers perform slightly worse in our experiments. This is likely due to the less-than-perfect pretraining optimization and the limited representation capability of finite-width FF layers with ReLU activation. We also note that the proposed transformer construction may not be the only way to mimic CTW, however, we believe the first two layers do capture important universal features. We provide supporting evidence empirically in the sequel.

Hybrid transformer with two-layer construction We construct hybrid versions of transformers, with details given in Appendix D.1.2. We train a two-layer transformer with a constructed $\mathbf{a}_i^{(2)}$ followed by a trainable FF layer (the FF layer in the second layer of the transformer) and an output layer, and compare the impacts different choices of $\mathbf{a}_i^{(2)}$. We replace the backward statistics $\mathbf{g}^{\leftarrow}_{i,1,M'}$ and $\mathbf{pos}_i$ with $\{\mathbf{n}_{n,s_{n,l}}\}_{l=0}^D$ and i in $\mathbf{a}_i^{(2)}$ in Eq. (10), and notice its performance is almost the same as the one using $\mathbf{a}_i^{(2)}$ in Eq. (10), and their performances are close to that of canonical 2-layer transformer. Moreover, the performances get worse if the statistics like $\{\mathbf{n}_{n,s_{n,l}}\}_{l=0}^D$ and i are further removed. Thus such couting statistics are necessary and essential for the ICL of VOMC sources. More discussions and experiments with 4-layer transformers with one or two constructed layers are in Appendix D.1.2.

5 Conclusion

We considered the in-context learning of transformers for VOMC sources. By drawing a close analogy of ICL and Bayesian universal compression, we leverage the CTW as a baseline. Experimentally, we observe the performances of the trained transformers are close to that of CTW even with just two layers under CTW priors. To understand the mechanism of transformers' ICL ability, we analyzed the attention maps and extracted two likely mechanisms. We then construct the finite-memory context extension layer, and the statistics collection layer, corresponding to these two mechanisms, respectively. The latter collects both the forward and backward statistics, which are vital as theoretically demonstrated by a novel representation of the CTW optimal next-token prediction. We also provide empirical evidence that the statistics collected by the constructed second layer, in particular the counting statistics, are indeed necessary.

Although we empirically showed transformers can perform ICL-VOMC tasks and constructed an idealized transformer to mimic the CTW algorithm, it is not clear whether a trained transformer will indeed utilize the upper layer mechanisms. Extending the existing approach (Edelman et al., 2024) to answer this question appears quite difficult, given the complexity of the constructed transformer and the underlying VOMCs; this is part of ongoing investigations.

References

Ahn, K., Cheng, X., Daneshmand, H., and Sra, S. (2024). Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36.

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. (2022). What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*.

Akyürek, E., Wang, B., Kim, Y., and Andreas, J. (2024). In-context language learning: Arhitectures and algorithms. *arXiv preprint arXiv:2401.12973*.

Bai, Y., Chen, F., Wang, H., Xiong, C., and Mei, S. (2024). Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in neural information processing systems*, 36.

Begleiter, R., El-Yaniv, R., and Yona, G. (2004). On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, 22:385–421.

Bietti, A., Cabannes, V., Bouchacourt, D., Jegou, H., and Bottou, L. (2024). Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems*, 36.

Burrows, M. (1994). A block-sorting lossless data compression algorithm. *SRS Research Report*, 124.

Cleary, J. and Witten, I. (1984). Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402.

Cleary, J. G. and Teahan, W. J. (1997). Unbounded length contexts for PPM. *The Computer Journal*, 40(2_and_3):67–75.

Dai, D., Sun, Y., Dong, L., Hao, Y., Ma, S., Sui, Z., and Wei, F. (2022). Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *arXiv preprint arXiv:2212.10559*.

Delétang, G., Ruoss, A., Duquenne, P.-A., Catt, E., Genewein, T., Mattern, C., Grau-Moya, J., Wenliang, L. K., Aitchison, M., Orseau, L., et al. (2023). Language modeling is compression. *arXiv preprint arXiv:2309.10668*.

Edelman, B. L., Edelman, E., Goel, S., Malach, E., and Tsilivis, N. (2024). The evolution of statistical induction heads: In-context learning Markov chains. *arXiv preprint arXiv:2402.11004*.

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. (2022). What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598.

Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al. (2023). ChatGPT for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274.

Kirsch, L., Harrison, J., Sohl-Dickstein, J., and Metz, L. (2022). General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*.

Kontoyiannis, I. (2023). Context-tree weighting and Bayesian context trees: Asymptotic and non-asymptotic justifications. *IEEE Transactions on Information Theory*.

Kontoyiannis, I., Mertzanis, L., Panotopoulou, A., Papageorgiou, I., and Skoularidou, M. (2022). Bayesian context trees: Modelling and exact inference for discrete time series. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(4):1287–1323.

Li, Y., Ildiz, M. E., Papailiopoulos, D., and Oymak, S. (2023). Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, pages 19565–19594. PMLR.

Makkuva, A. V., Bondaschi, M., Girish, A., Nagle, A., Jaggi, M., Kim, H., and Gastpar, M. (2024). Attention with Markov: A framework for principled analysis of transformers via Markov chains. *arXiv preprint arXiv:2402.04161*.

Moffat, A. (1990). Implementing the PPM data compression scheme. *IEEE Transactions on communications*, 38(11):1917–1921.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. (2022). In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*.

Pasco, R. (1976). *Source coding algorithms for fast data compression*. PhD thesis, Stanford University.

Rajaraman, N., Bondaschi, M., Ramchandran, K., Gastpar, M., and Makkuva, A. V. (2024). Transformers on markov data: Constant depth suffices. *arXiv preprint arXiv:2407.17686*.

Reddy, G. (2023). The mechanistic basis of data dependence and abrupt learning in an in-context classification task. *arXiv preprint arXiv:2312.03002*.

Rissanen, J. (1983). A universal data compression system. *IEEE Transactions on information theory*, 29(5):656–664.

Rissanen, J. and Langdon, G. G. (1979). Arithmetic coding. *IBM Journal of research and development*, 23(2):149–162.

Rissannen, J. (1976). Generalized kraft's inequality and arithmetic coding. *IBM Journal on Research Development*, 20.

Sadakane, K., Okazaki, T., and Imai, H. (2000). Implementing the context tree weighting method for text compression. In *Proceedings DCC 2000. Data Compression Conference*, pages 123–132. IEEE.

Shkarin, D. (2002). Ppm: One step to practicality. In *Proceedings DCC 2002. Data Compression Conference*, pages 202–211. IEEE.

Thirunavukarasu, A. J., Ting, D. S. J., Elangovan, K., Gutierrez, L., Tan, T. F., and Ting, D. S. W. (2023). Large language models in medicine. *Nature medicine*, 29(8):1930–1940.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. (2023). Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Willems, F. M. (1998). The context-tree weighting method: Extensions. *IEEE Transactions on Information Theory*, 44(2):792–798.

Willems, F. M., Shtarkov, Y. M., and Tjalkens, T. J. (1995). The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664.

Willems, F. M., Shtarkov, Y. M., and Tjalkens, T. J. (1996). Context weighting for general finite-context sources. *IEEE transactions on information theory*, 42(5):1514–1520.

Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., and Mann, G. (2023). Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. (2021). An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Ziv, J. and Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343.

Ziv, J. and Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE transactions on Information Theory*, 24(5):530–536.

A Related Work

There have been many efforts in studying the ICL capabilities of transformers. A significant recent development is the elucidation of the connection to gradient descent, particularly for linear regression tasks (Von Oswald et al., 2023; Akyürek et al., 2022; Dai et al., 2022; Ahn et al., 2024). Li et al. (2023) formulated the ICL problem as a multi-task learning problem and considered ICL for several simple problem settings for which the authors provide risk bounds for ICL of supervised learning algorithms in these problem settings. Kirsch et al. (2022) viewed the ICL problem as a meta-learner and studied the relation between tasks and model sizes.

Olsson et al. (2022) studied the induction head, i.e., the forming of small k-gram attention in LLMs. Reddy (2023) studied the balance between ICL and in-weights learning, and observed the abrupt emergence of the induction head corresponds to the emergence of ICL. The induction head was generalized to the statistical induction head in (Edelman et al., 2024) mainly to study bigrams. We adopted it but further allowed more statistical induction heads for more suffixes to be included together, in the first two layers of the idealized transformer.

There have also been efforts to study transformers and learning of Markov chains. Xie et al. (2021) viewed ICL as a Bayesian inference problem, where a latent concept determines an HHM, and the observations from the HHM can lead to the identification of the hidden concept. They studied the eventual ICL capability, i.e., when the number of in-context examples goes to infinity. The work in (Bietti et al., 2024) allowed a fixed-order Markov chain to switch to a new deterministic mode, and the authors study the training behavior of the corresponding ICL task with this mode transition. Akyürek et al. (2024) made a comprehensive empirical comparison of various language models on random finite automata, and showed that the transformer performs the best among these models. Makkuva et al. (2024) studied the loss landscape during transformer training on sequences generated from a single fixed-order Markov chain, using a single-layer transformer. Their study does not consider ICL. More recently Rajaraman et al. (2024) considered ICL of FOMCs with single-head transformers, and provided a construction to show that it is possible to use a single attention head to capture longer memory in the sequence. The work most relevant to us is (Edelman et al., 2024), where ICL of a fixed-order Markov chain was considered, and the training behavior was studied both empirically and theoretically, and the forming of induction heads in a two-layer network was demonstrated. All these existing work assumed fixed-order Markov models or fixed-order HHMs, usually with orders kept at 1 or 2; moreover, they almost all focus on the emergence of the induction heads during training or the training landscape. Our study is different firstly in the variable-order nature of the Markov models, and secondly the focus on the on-time ICL performance instead of the training landscape and behavior.

Lossless data compression has a long history, with many different algorithms being developed over the years. The most popular general-purpose compression algorithms are perhaps the Lempel-Ziv compression algorithms (Ziv and Lempel, 1977, 1978) and their variants, which belong to dictionary-based compression algorithms. These algorithms do not explicitly maintain any probabilistic models, and their efficiency comes from maintaining an efficiency dictionary of sequences that have been seen before, and to be matched with future sequences. More powerful compression algorithms usually maintain probability models explicitly, which are then plugged into an AC module (Rissannen, 1976; Pasco, 1976; Rissanen and Langdon, 1979) for efficient compression. The most well-known classes of algorithms in this category is the context-tree weighting algorithm (Willems et al., 1995; Begleiter et al., 2004; Kontoyiannis, 2023) and prediction by partial matching (Cleary and Witten, 1984). The former enjoys a strong theoretical guarantee, particularly on binary sources (Willems et al., 1995), but has some difficulty in its practical implementation (Willems, 1998; Willems et al., 1996; Sadakane et al., 2000; Begleiter et al., 2004), particularly for large alphabet sizes and sequential data. The latter is based more on heuristics, and has been improved and extended in various ways (Cleary and Teahan, 1997; Moffat, 1990; Shkarin, 2002). Methods based on probabilistic modeling are usually more resource-extensive, though they have gained more popularity recently due to the increased availability of computing resources. The evaluation given in (Begleiter et al., 2004) suggests that CTW and PPM are the two most powerful compression algorithms in practice. There are other compression algorithms such as those based on the Burrows-Wheeler transformation (Burrows, 1994) which does not explicitly maintain a probabilistic model, but are also not dictionary-based.

B An Example CT

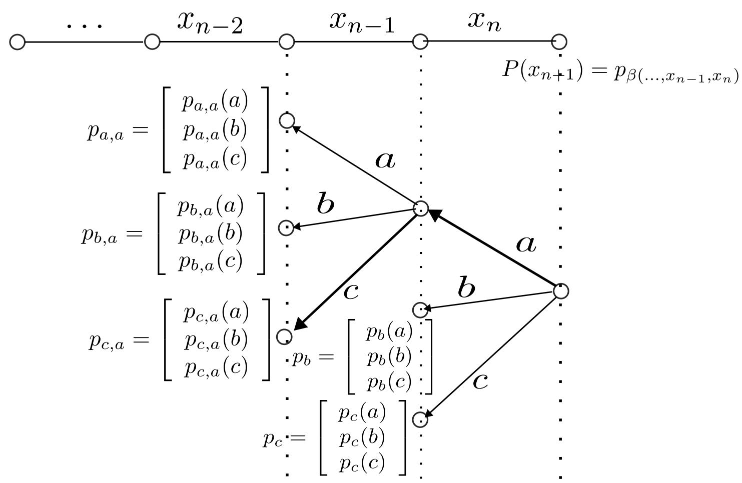


Figure 5: A CT in the alphabet $\mathcal{A} = \{a, b, c\}$ with suffix set $\mathcal{S} = \{(b), (c), (a, a), (b, a), (c, a)\}$ and the associated probability distributions. If $(\ldots, x_{n-1}, x_n) = (\ldots, c, a)$, then the probability distribution for the next symbol x_{n+1} is $p_{c,a}$.

C Transformer Architecture

The transformer considered in this is illustrated in Fig. 6.

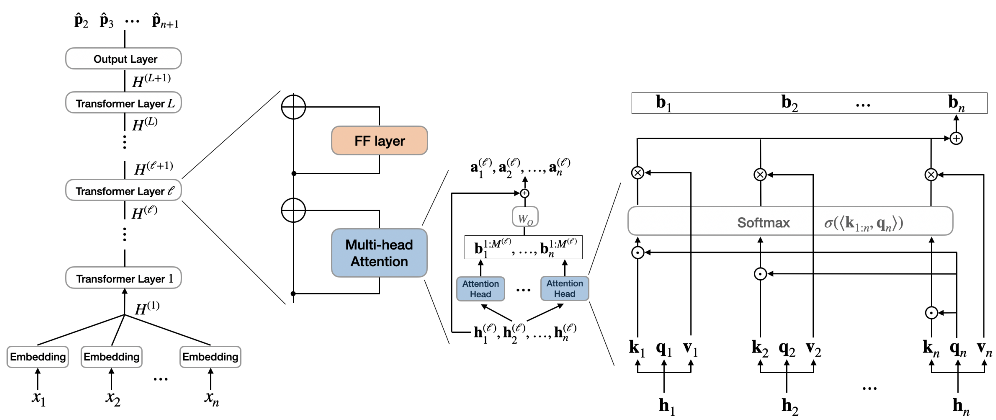


Figure 6: Transformer model

D Pretraining Details

We choose the alphabet size to be $|\mathcal{A}| = 3$ in the experiments. For training, we randomly generate $K = 20000$ CTs of various depths (maximum order $D \leq 5$), and then for each CT leaf, we generate a probability distribution. Two different ways of generating these probability distributions are taken: the first approach is use the Dirichlet distribution to sample such distributions, and the second approach is to randomly select some of the elements in the alphabet to have probability zero, and the others with random values. Different values of the Dirichlet parameter are tested but only the results do not appear to be sensitive to the choice. For each CT, a source sequence of certain length (e.g., $N_k = 5120$) is produced. The context window N can vary, but in most cases, we set it at 512 (except when $D = 5$, we set it to be 1536 to allow sufficient data collection in context). Each source sequence is segmented into $\lfloor N_k/N \rfloor$ training sequence.

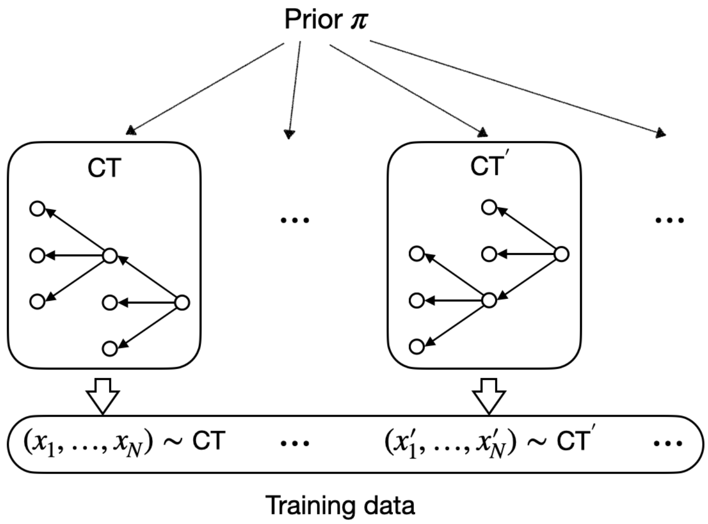


Figure 7: Training data collection

	TF-1	TF-2	TF-3	TF-4	TF-5	TF-6	CTW
CTs $D = 3$	0.9368	0.7297	0.7265	0.7220	0.7245	0.7258	0.7165
CTs $D = 4$	0.9667	0.7831	0.7818	0.7759	0.7791	0.7774	0.7603
CTs $D = 5$	0.9661	0.7569	0.7490	0.7440	0.7437	0.7438	0.7400

Table 1: Average compression rates in the context window by transformers and CTW, where the CTs are sampled from the CTW-prior. The context window and embedding dimension for CTs of $D = 5$ are $N = 1536$ and $E = 128$, while for others it are $N = 512$ and $E = 64$.

During testing, we randomly generate multiple (2048 in our experiments) new CTs of varying depths using the same procedure, and for each CT, a sequence of length $N_k = 5120$ are generated, and then again segmented into a length of the context window for testing.

The transformer model is implemented using Pytorch, and trained using the AdamW optimizer with the default parameters. A100/T100 GPUs are used for training. Training a model requires roughly 4 to 6 hours. Batch size is set at 512, and the maximum epoch is set at 100 with early termination allowed after 15 epochs of no improvement. Testing was performed on a local workstation with a GeForce GTX 1660 Ti GPU card.

D.1 Additional experimental results

In Table 1, we further provide the average compression rates over the whole context window for CTs of different orders; we refer to the transformers as TF. For CTs with lower order, the transformer embedding dimension is set at 64 instead of 128.

D.1.1 Transformers vs. CTW under Non-CTW-Priors

The CTW algorithm is known to be Bayesian optimal when the CTs are generated from a CTW-prior. When the CTs do not follow those priors, can learning-based transformers perform better than CTWs? We empirically observe that in such settings, transformers indeed have advantages. The training data are generated by using CTs of different maximum orders, where the orders are chosen uniformly at random between 1 and 3. Moreover, the probability vector is not generated from the Dirichet prior, but from a distribution that for each CT leaf, randomly assigns one of the element in the alphabet to have zero-probability. We test on sequences generated from CTs produced from the same distribution as in the training setting. We assume the CTW takes the default (non-informative prior) parameters of $\alpha = 0.5$, and the same tree branch stopping parameter $\lambda = 0.15$ as taken in the testing sequence CTs.

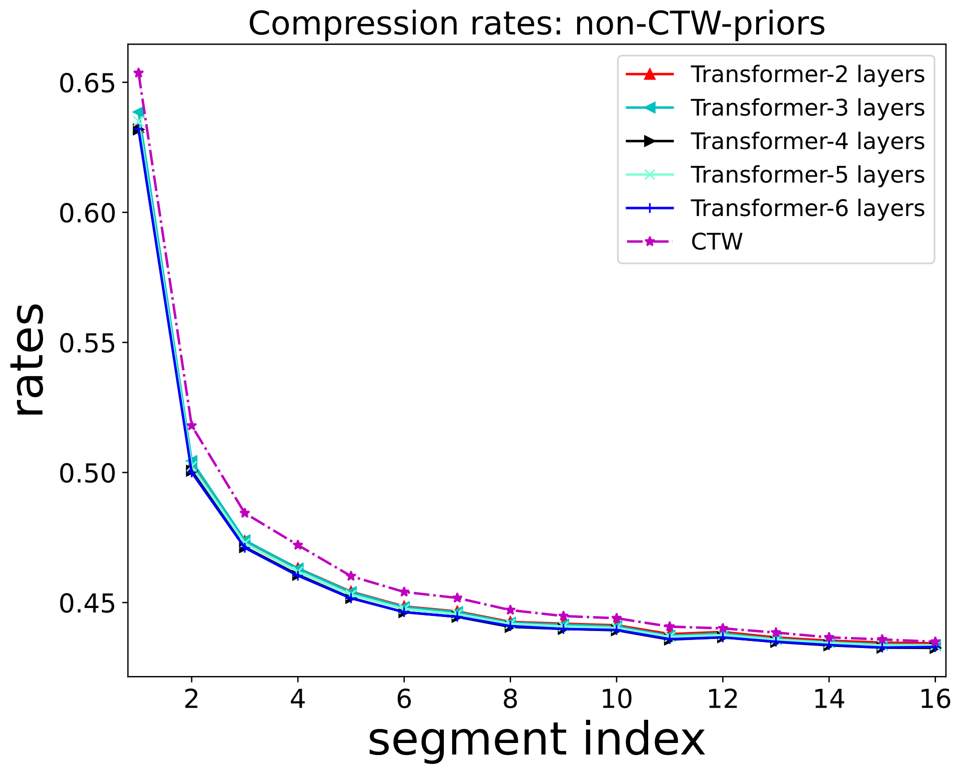


Figure 8: Transformers vs. CTW

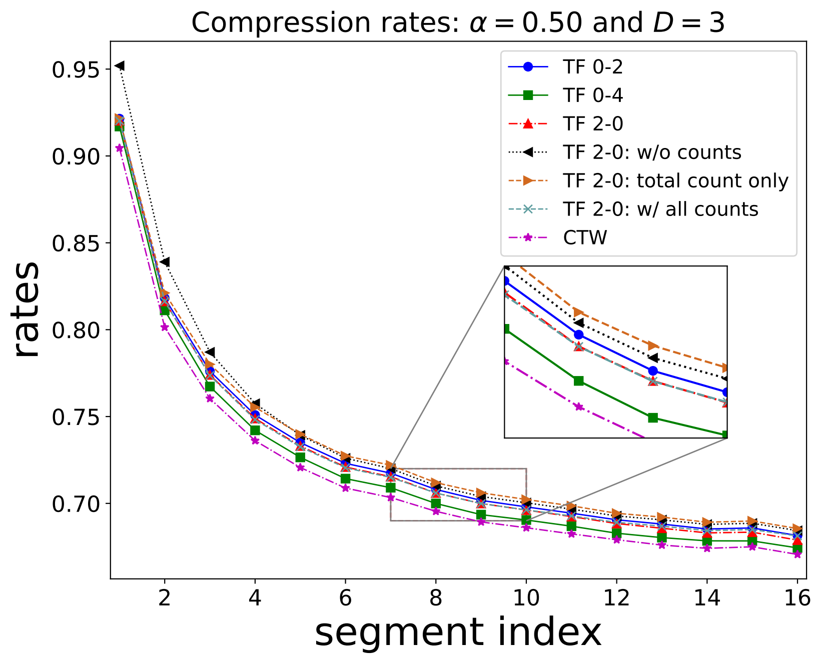
As can be observed in Fig. 8, the CTW algorithm is no longer optimal, and trained transformers can perform considerably better. In fact, even transformers with 2 layers can outperform the CTW algorithm in this setting, and more layers usually lead to further improved performance, albeit the improvement is less significant.

D.1.2 Hybrid transformer

We conduct experiments on the hybrid versions of transformers. Let "TF 0-2" denote the canonical 2-layer transformer; "TF 1-1" denote the transformer consisted of a constructed layer with output $\mathbf{h}_i^{(1)}$ (8), and a trainable transformer layer and a output layer taking $\mathbf{H}^{(1)}$ as input; and denote by "TF 2-0" the transformer with 2 constructed layer with output $\mathbf{a}_i^{(2)}$ in (10), followed by a trainable FF layer (the FF layer in the second layer of the transformer) and an output layer.

We first study the key statistics behind the strong performance of two-layer transformers, as shown in the left panel in Fig. 9. Compared to "TF 2-0" which is the constructed layers given previously, the version "TF 2-0 w/o counts" does not contain $\mathbf{g}^{\leftarrow}_{i-1,M'}$ or $\mathbf{pos}_i$ in $\mathbf{a}_i^{(2)}$; the version "TF 2-0 total counts only" does not contain $\mathbf{g}^{\leftarrow}_{i-1,M'}$ in $\mathbf{a}_i^{(2)}$ and $\mathbf{pos}_i$ is replaced by the total count i; "TF 2-0 w/ all counts" replaces $\mathbf{g}^{\leftarrow}_{i-1,M'}$ and $\mathbf{pos}_i$ with $\{\mathbf{n}_{n,s_{n,l}}\}_{l=0}^{D}$ and i. Even though their performances are rather clustered, we can make the following observations: 1) The performances degrade as more counting information is removed from the representation, and the counting information is clearly very important, 2) The performances of "TF 2-0" and "TF 2-0 w/ all counts" almost match exactly, indicating the main purpose of the backward statistics $\mathbf{g}^{\leftarrow}_{i-1,M'}$ is to extract the counts, and 3) The performance of the original 2-layer transformer is similar to that of the constructed "TF 2-0" and "TF 2-0: w/ all counts" that those without less counting information.

We further study hybrid transformers with the first one or two being the constructed layers. As shown in the right panel of Fig. 9, transformers with 2 total layers and 4 total layers form two clusters,



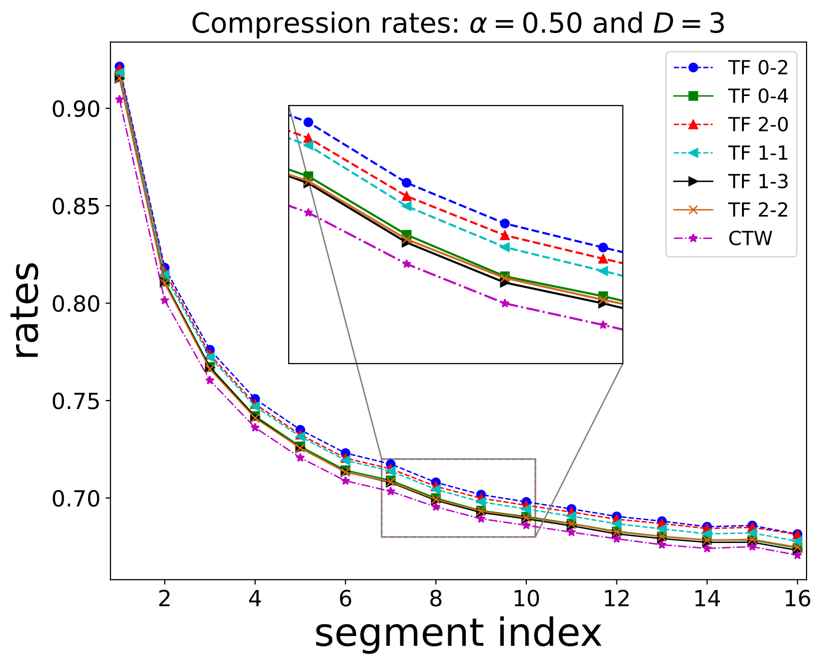


Figure 9: Hybrid Transformers: Effects of accumulative suffix counts and synthetic layers

which provides strong evidence that the constructed layers are indeed replacing the first two layers of the original transformers in a functional manner. Moreover, the performances of transformers with a single constructed layer, such as "TF 1-1" and "TF 1-3", are slightly better than those with two constructed layers, such as "TF 2-0" and "TF 2-2", likely due to the flexibility in the remaining trainable transformer layers. Interestingly, for two layer transformers, the hybrid versions can perform even better than the original transformer "TF 0-2", which we believe is because the latter is having difficulty extracting the exact statistics as those more readily available in the constructed layers.

E Proofs of The Theorems for CT Sources

E.1 A New Representation for Bayesian Next Token Prediction

We aim to predict the next token x_{n+1} based on the observations $x_{1-D}^n = (x_{1-D}, \dots, x_n)$ via a transformer-friendly formula. Note that x_{1-D}^0 is a place holder or dummy initialization sequence, which does not contain any information of $(T, \{p_s\})$.

Theorem 5 (Restate Theorem 1)**.** *The predicted probability can be computed as*

$$P_{\pi_{CTW}}(x_{n+1}|x_{1-D}^n) = \sum_{l=0,\dots,D} \omega_{n,l} \cdot \mathbf{p}_{n,s_{n,l}}(x_{n+1}), \tag{13}$$

where $\mathbf{p}_{n,s_{n,l}}(a) = \frac{\alpha(a)+\mathbf{n}_{n,s_{n,l}}(a)}{\sum_q(\alpha(q)+\mathbf{n}_{n,s_{n,l}}(q))}$*; and* $\omega_{n,\cdot} \in \Delta_{D+1}$ *with* $\ln(\omega_{n,l}) - \ln(\omega_{n,l-1}) = \ln(1-\lambda) - \mathbb{I}_{l=D}\ln(\lambda) + \ell^e_{n,s_{n,l}} - \ell^e_{n,s_{n,l-1}} + \sum_{q\in\mathcal{A}} \ell^w_{n,qs_{n,l-1}} - \ell^w_{n,s_{n,l}}$*, where* $\ell^e_{n,s} = \ln(p^e_{n,s})$*,* $\ell^w_{n,s} = \ln(p^w_{n,s})$*.*

Note that $p^e_{n,s}, p^w_{n,s}$ can be efficiently calculated by the CTW procedure, and compared to calculate $\frac{P_{\pi_{\text{CTW}}}(x_1^{n+1}|x_{1-D}^0)}{P_{\pi_{\text{CTW}}}(x_1^n|x_{1-D}^0)}$ for each x_{n+1} the extra computation besides the CTW procedure is A times larger than that by Eq (7). As illustrated in Fig. 4, the weighted average formula in Eq (7) gives a natural interpretation for the Bayesian optimal next token predicted probability. Each suffix along the root the leaf path $s_{n,0} - s_{n,1} - \cdots - s_{n,D}$ can potentially be the true suffix, i.e., $s_{n,l} \in \mathcal{L}(T)$, and $\mathbf{p}_{n,s_{n,l}}$ is in fact the Bayesian optimal next token prediction given $s_{n,l} \in \mathcal{L}(T)$.

The weights $\omega_{n,l}$'s are based on stopping probability λ, the information in the potential suffix path such as $p^e_{s_{n,s_{n,l}}}$ as well as the information from their siblings $p^w_{n,qs_{n,l-1}}$. We can interpret $p^e_{n,s}$ as the evidence (unnormalized likelihood) that $s \in \mathcal{L}(T)$, and $p^w_{n,s}$ as the evidence that $s \in T$, i.e., the underlying tree covers node s. Theorem 1 indicates that more weights are assigned to $s_{n,l}$ than $s_{n,l-1}$, i.e., $\omega_{n,l} > \omega_{n,l-1}$, if λ is smaller (i.e., node $s_{n,l-1}$ is more likely to branch and thus less likely to be a leaf node), $p^e_{n,s_{n,l}} - p^e_{n,s_{n,l-1}}$ is larger (i.e., $s_{n,l}$ has more evidence than $s_{n,l-1}$) and $\sum_{q\in\mathcal{A}} \ell^w_{n,qs_{n,l-1}} - \ell^w_{n,s_{n,l}}$ is larger (i.e., $s_{n,l}$'s siblings have more evidence to explain the data and thus $s_{n,l-1}$ is less likely to be a leaf node).

Proof of Theorem 5. Recall $s_{i,l} = (x_{i-l+1}, \dots, x_i)$ is the suffix at position i of length l. We omit D by writing $\mathcal{T} = \mathcal{T}(D)$ when D is clear from the context. Define partition $\{\mathcal{T}_{s_{n,l}}\}_{0\le l\le D}$, that $\mathcal{T}_s = \{T \in \mathcal{T} : s \in \mathcal{L}(T)\}$ is the set of trees with leaf s. The predicted probability can then be computed as

$$\begin{aligned}
P_{\pi_{\text{CTW}}}(x_{n+1}|x_{1-D}^n) &= \sum_{T\in\mathcal{T}} \int p(x_{n+1}|T, \{p_s\}, x_{1-D}^n)\pi(T, \{p_s\}|x_{1-D}^n)\Big(\prod_{s\in\mathcal{L}(T)} \mathrm{d}p_s\Big) \\
&= \sum_{l=0,\dots,D} \sum_{T\in\mathcal{T}_{s_{n,l}}} \int p_{s_{n,l}}(x_{n+1})\pi(T, \{p_s\}|x_{1-D}^n)\Big(\prod_{s\in\mathcal{L}(T)} \mathrm{d}p_s\Big) \\
&= \sum_{l=0,\dots,D} \sum_{T\in\mathcal{T}_{s_{n,l}}} \int p_{s_{n,l}}(x_{n+1})\pi(T|x_{1-D}^n)\pi(p_{s_l}|T, x_{1-D}^n)\mathrm{d}p_{s_l} \\
&= \sum_{l=0,\dots,D} \sum_{T\in\mathcal{T}_{s_{n,l}}} \pi_D(T|x_{1-D}^n) \int p_{s_{n,l}}(x_{n+1})\pi(p_{s_l}|T, x_{1-D}^n)\mathrm{d}p_{s_l} \\
&= \sum_{l=0,\dots,D} \Big(\sum_{T\in\mathcal{T}_{s_{n,l}}} \pi_D(T|x_{1-D}^n)\Big)\Big(\int p_{s_{n,l}}(x_{n+1})\pi(p_{s_l}|T, x_{1-D}^n)\mathrm{d}p_{s_l}\Big) \\
&= \sum_{l=0,\dots,D} \omega_{n,l} \cdot \mathbf{p}_{n,s_{n,l}}(x_{n+1}),
\end{aligned} \tag{14}$$

where the last equality is by the definition that

$$\omega_{n,l} = \sum_{T\in\mathcal{T}_{s_{n,l}}} \pi_D(T|x_{1-D}^n), \tag{15}$$

and the optimal prediction probability given suffix $s_{n,l}$ is

$$\mathbf{p}_{n,s_{n,l}}(a) = \frac{\alpha(a) + \mathbf{n}_{n,s_{n,l}}(a)}{\sum_{q\in\mathcal{A}}(\alpha(q) + \mathbf{n}_{n,s_{n,l}}(q))}, \tag{16}$$

since for any $T \in \mathcal{T}_{s_l}$, the posterior of p_s follows Dirichlet distribution

$$\pi(p_{s_l}|T, x_{1-D}^n) = \mathrm{Dir}(\theta_{s_l}; \alpha + \mathbf{n}_{n,s_{n,l}}), \tag{17}$$

with posterior mean $\mathbb{E}[p_{s_l}|T, x_{1-D}^n] \in \Delta_\mathcal{A}$ and $\propto \alpha + \mathbf{n}_{n,s_{n,l}}$.

It remains that whether the parameters $\omega_{n,l}$ is easy to compute or not. The following theorem shows that these parameters $\omega_{n,l}$ can be computed easily via p_s^w and p_s^e based on x_{1-D}^n without the knowledge of x_{n+1}.

Since the length of data n is fixed and clear from the context, let $\underline{x} = x_{1-D}^n$ be the sequence, and we omit n in the subscript of $p_{n,s}^e$, $p_{n,s}^w$ and $s_{n,l}$ for simplicity.

For any model $T \in \mathcal{T}(D)$, the posterior probability $\pi(T|\underline{x})$ is given by:

$$\pi_D(T|\underline{x}) = \frac{\pi_D(T)P_\pi(\underline{x}|T)}{P_\pi(\underline{x})} = \frac{\pi_D(T)\prod_{s\in\mathcal{L}(T)} p_s^e}{p_{()}^w}, \tag{18}$$

where the denominator $P_\pi^*(\underline{x}) = p_{()}^w$ is the prior predictive likelihood computed by CTW, and the numerator is by $P_\pi(\underline{x}|T) = \prod_{s\in\mathcal{L}(T)} p_s^e$ in (Kontoyiannis et al., 2022, Lemma 2.2). Since $\omega_l = \sum_{T\in\mathcal{T}_{s_l}} \pi(T|\underline{x})$ by definition, we have for any $l = 1, 2, \ldots, d$,

$$\frac{\omega_l}{\omega_{l-1}} = \frac{\sum_{T'\in\mathcal{T}_{s_l}} \pi_d(T'|x)}{\sum_{T\in\mathcal{T}_{s_{l-1}}} \pi_d(T|x)} = \frac{\sum_{T'\in\mathcal{T}_{s_l}} \pi_d(T')\prod_{s\in\mathcal{L}(T')} p_s^e}{\sum_{T\in\mathcal{T}_{s_{l-1}}} \pi_d(T)\prod_{s\in\mathcal{L}(T)} p_s^e}. \tag{19}$$

Note that tree in $\mathcal{T}_{s_l}$ and trees in $\mathcal{T}_{s_{l-1}}$ share similarities. For any $T \in \mathcal{T}_{s_{l-1}}$, let $\mathcal{T}_{s_l;T} = \{T' \in \mathcal{T}_{s_l} : \mathcal{L}(T) \subset \mathcal{L}(T') \cup \{s_{l-1}\}\}$ be the set of trees that differs from T only at subtree $\mathrm{sub}(T'; s_l) :=$ {subtree of T' with root at s}.

Take any $l = 1, 2, \ldots, D-1$. For any $T \in \mathcal{T}_{s_{l-1}}$ and $T' \in \mathcal{T}_{s_l;T}$. Based on the definition of $\pi_D = (1-\lambda)^{(|\mathcal{L}(T)|-1)/(A-1)}\lambda^{|\mathcal{L}(T)|-|\mathcal{L}_D(T)|}$, it is not hard to verify that

$$\begin{aligned}
\frac{\pi_D(T')}{\pi_D(T)} &= \frac{\pi_{D-l+1}(\mathrm{sub}(T'; s_{l-1}))}{\pi_{D-l+1}(\mathrm{sub}(T; s_{l-1}))} \\
&= \frac{(1-\lambda)\pi_{D-l}(\mathrm{sub}(T'; s_l))\prod_{s_l'\in\mathrm{sib}(s_l)} \pi_{D-l}(\mathrm{sub}(T'; s_l'))}{\lambda} \\
&= (1-\lambda)\prod_{s_l'\in\mathrm{sib}(s_l)} \pi_{D-l}(\mathrm{sub}(T'; s_l')),
\end{aligned}$$

where $\mathrm{sib}(s_{l+1}) = \{qs_l : q \in \mathcal{A} \text{ and } qs_l \neq s_{l+1}\}$ is set of siblings of s_{l+1}. We can interpret the ratio as follows. T' and T only differs at the $\mathrm{sub}(T'; s_{l-1})$ and $\mathrm{sub}(T; s_{l-1})$. Since T' branch at node s_{l-1}, we thus have the numerator in the second equation, where $(1-\lambda)$ corresponds to the branching and then compute for the subtrees. Note that T stops branching at s_{l-1} and T' stops branching at s_l, then $\pi_{D-l+1}(\mathrm{sub}(T; s_{l-1})) = \pi_{D-l}(\mathrm{sub}(T'; s_l)) = \lambda$ equals to the stopping probability.

Given any suffix s with $|s| \leq D$, it has been shown in (Kontoyiannis et al., 2022, Proof of Theorem 3.1) that for any $l \leq D$,

$$p_s^w = \sum_{U\in\mathcal{T}(D-l)} \pi_{D-l}(U) \prod_{u\in\mathcal{L}(U)} p_{us}^e, \tag{20}$$

where $\mathcal{T}(D-l)$ is the set of trees with maximum depth $D-l$ and π_{D-l} is the prior for bounded branching process with maximum depth $D-l$. We thus have

$$\frac{\sum_{T'\in\mathcal{T}_{s_l;T}}\pi_D(T')\prod_{s\in\mathcal{L}(T')}p_s^e}{\pi_D(T)\prod_{s\in\mathcal{L}(T)}p_s^e} = \frac{\sum_{T'\in\mathcal{T}_{s_l;T}}\pi_D(T')\prod_{s\in\mathcal{L}(T')}p_s^e}{\pi_D(T)\prod_{s\in\mathcal{L}(T)}p_s^e} \tag{21}$$

$$= \sum_{T'\in\mathcal{T}_{s_l;T}}\frac{\pi_D(T')}{\pi_D(T)}\frac{\prod_{s\in\mathcal{L}(T')\setminus\mathcal{L}(T)}p_s^e}{p_{s_{l-1}}^e} \tag{22}$$

$$= \sum_{T'\in\mathcal{T}_{s_l;T}}\left((1-\lambda)\prod_{s_l'\in\mathrm{sib}(s_l)}\pi_{D-l}(\mathrm{sub}(T';s_l'))\right)\left(\frac{p_{s_l}^e\prod_{s_l'\in\mathrm{sib}(T;s_l)}\prod_{s\in\mathcal{L}(\mathrm{sub}(T';s_l'))}p_s^e}{p_{s_{l-1}}^e}\right) \tag{23}$$

$$= (1-\lambda)\frac{p_{s_l}^e}{p_{s_{l-1}}^e}\sum_{T'\in\mathcal{T}_{s_l;T}}\left(\prod_{s_l'\in\mathrm{sib}(s_l)}\pi_{D-l}(\mathrm{sub}(T';s_l'))\right)\left(\prod_{s_l'\in\mathrm{sib}(T;s_l)}\prod_{s\in\mathcal{L}(\mathrm{sub}(T';s_l'))}p_s^e\right) \tag{24}$$

$$= (1-\lambda)\frac{p_{s_l}^e}{p_{s_{l-1}}^e}\sum_{T'\in\mathcal{T}_{s_l;T}}\left(\prod_{s_l'\in\mathrm{sib}(s_l)}\pi_{D-l}(\mathrm{sub}(T';s_l'))\prod_{s\in\mathcal{L}(\mathrm{sub}(T';s_l'))}p_s^e\right) \tag{25}$$

$$= (1-\lambda)\frac{p_{s_l}^e}{p_{s_{l-1}}^e}\prod_{s_l'\in\mathrm{sib}(s_l)}\left(\sum_{U\in\mathcal{T}(D-l)}\pi_{D-l}(U)\prod_{u\in\mathcal{L}(U)}p_{us_l'}^e\right) \tag{26}$$

$$= \frac{(1-\lambda)p_{s_l}^e\prod_{a\neq s_l\setminus s_{l-1}}p_{as_{l-1}}^w}{p_{s_{l-1}}^e}. \tag{27}$$

Similarly, for any $T\in\mathcal{T}_{s_{D-1}}$ and $T'\in\mathcal{T}_{s_D;T}$, $\frac{\pi_D(T')}{\pi_D(T)}=\frac{1-\lambda}{\lambda}$, and we have

$$\frac{\omega_D}{\omega_{D-1}} = \frac{(1-\lambda)p_{s_d}^e\prod_{a\neq s_d\setminus s_i}p_{as_{D-1}}^w}{\lambda p_{s_{D-1}}^e}, \tag{28}$$

in the same manner. The proof can then be concluded by taking logarithm on both hands. $\square$

E.2 Construction of Transformer for CTW

To make the presentation clear, in the following we separate the layers by their functionality and present them separately. Recall that

$$\mathbf{a}_i^{(\ell)} = \mathrm{MHA}\left(\mathbf{h_i},\mathbf{H};\{W_{O,m}^{(\ell)},W_{Q,m}^{(\ell)},W_{K,m}^{(\ell)},W_{V,m}^{(\ell)}\}_{m=1}^{M^{(\ell)}}\right) \triangleq W_O^{(\ell)}\left[\mathbf{b}_{1,i}^{(\ell)};\mathbf{b}_{2,i}^{(\ell)};\ldots;\mathbf{b}_{M^{(\ell)},i}^{(\ell)}\right],$$

where $\{W_{Q,m}^{(\ell)},W_{K,m}^{(\ell)},W_{V,m}^{(\ell)}\}_{m=1}^{M^{(\ell)}}$ are the $E^{(\ell)}\times E$ query matrices, key matrices, and value matrices and $W_O^{(\ell)}$ is the $E\times M^{(\ell)}E^{(\ell)}$ output mapping matrix. For simplicity of presentation, we take $E^\ell = E$ and $W_O^\ell = [\mathbf{I};\mathbf{I};\ldots;\mathbf{I}]$. It is not hard to see the following constructions can be applied to much smaller $E^{(\ell)}$ while taking W_O as a permutation matrix.

We have omitted the dimensionality of several zero matrices when they are obvious from the context. The first and second layer constructions are illusatred in Fig. 10.

E.2.1 Finite-memory context-extension layer

We begin with the first layer, which is referred to as a finite-memory context-extension layer.

Theorem 6 (Restatement of Theorem 2)**.** *There is an M-headed transformer layer that can perform finite-memory context-extension, defined by the following output, with the initial one-hot embedded input $\mathbf{H}^{(1)}$:*

$$\mathbf{h}_i^{(2)} = (\mathbf{s}_{i,M+1};\mathbf{0};\mathbf{pos}_i) = (\mathbf{x}_i;\mathbf{x}_{i-1};\ldots;\mathbf{x}_{i-M};\mathbf{0};\mathbf{pos}_i), \tag{29}$$

where $\mathbf{s}_{i,M+1} = (\mathbf{x}_i;\ldots;\mathbf{x}_{i-M})$ is the vector version of the M-length suffix $s_{i,M+1} = x_{i-M}^i$.

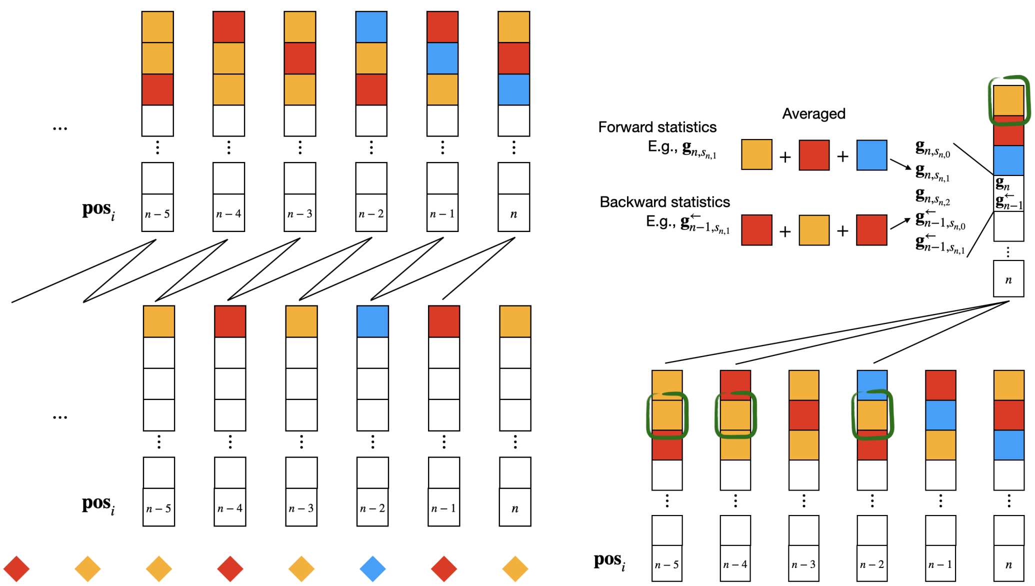


Figure 10: Transformer construction for $D = 2$. The left figure illustrates the first layer – finite-memory context-extension layer, which append the previous D tokens. The right figure demonstrate the MHA of the second layer – statistics collection layer, which extracts forward and backward statistics based on the matched suffix.

Proof of Theorem 2. The input of the of the first layer is a initial one-hot embedded input with positional embedding $\mathbf{H}^{(1)}$, where its n-th column is

$$\mathbf{h}_i^{(1)} = (\mathbf{x}_i; \mathbf{0}; \mathbf{pos}_i) \in \mathbb{R}^E, \tag{30}$$

where positional encoding

$$\mathbf{pos}_i = (1; \cos(i\pi/N); \sin(i\pi/N)), \tag{31}$$

with C being the maximum context size.

The multi-head attention in the first layer is consisted of $M^{(1)} = D$ heads parameterized by $(W_{Q,m}^{(1)}, W_{K,m}^{(1)}, W_{V,m}^{(1)})_{m=1,2,\ldots,M^{(1)}}$. Specifically, for $m = 1, 2, \ldots, M^{(1)}$,

$$W_{Q,m}^{(1)} = \begin{pmatrix} \mathbf{0} & \mathrm{Rot}(m) \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad W_{K,m}^{(1)} = \begin{pmatrix} \mathbf{0} & c\mathbf{I}^{2\times 2} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad W_{V,m}^{(1)} = \begin{pmatrix} \mathbf{0}^{mA\times A} & \mathbf{0} \\ \mathbf{I}^{A\times A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{32}$$

where $\mathrm{Rot}(m) = \begin{pmatrix} \cos(m\pi/N) & \sin(m\pi/N) \\ -\sin(m\pi/N) & \cos(m\pi/N) \end{pmatrix}$ is a rotation matrix that rotates clockwise by an angle of $m\pi/C$, and $c \in \mathbb{R}_+$ is a temperature factor. The query, key, and value after the mapping are

$$W_{Q,m}^{(1)}\mathbf{h}_n^{(1)} = \begin{pmatrix} \mathbf{pos}_{n-m} \\ \mathbf{0} \end{pmatrix}, \quad W_{K,m}^{(1)}\mathbf{h}_i^{(1)} = c\begin{pmatrix} \mathbf{pos}_i \\ \mathbf{0} \end{pmatrix}, \quad W_{V,m}^{(1)}\mathbf{h}_i^{(1)} = \begin{pmatrix} \mathbf{0}^{mA\times 1} \\ \mathbf{x}_i \\ \mathbf{0} \end{pmatrix}. \tag{33}$$

Take $c = \infty$ or sufficiently large. It is seen that the m-th head essentially copies the m-th earlier symbol to stack at the $(m+1)$-th position below the original symbol $\mathbf{x}_i$. Together with the residual link, the attention layer gives exactly the $\mathbf{h}_i^{(2)}$ shown in (34) while the feedforward network in this layer can be set as all zero.

$$\mathbf{h}_i^{(2)} = (\mathbf{x}_i; \mathbf{x}_{i-1}; \mathbf{x}_{i-2}; \mathbf{x}_{i-M^{(1)}}; \mathbf{0}; \mathbf{pos}_i) = (\mathbf{s}_{i,M^{(1)}+1}; \mathbf{0}; \mathbf{pos}_i), \tag{34}$$

where $\mathbf{s}_{i,l} = (\mathbf{x}_i; \mathbf{x}_{i-1}; \cdots; \mathbf{x}_{i-l+1})$ is the one-hot embedded version of suffix $s_{i,l} = (x_{i-l+1}, \ldots, x_{i-1}, x_i)$. $\square$

E.2.2 Statistics collection layer

Theorem 7 (Restatement of Theorem 3)**.** *There is an M'-head attention layer, where $M' \leq M+1$, that can perform statistics collection, defined by the following output, with $\mathbf{H}^{(2)}$ in (8) as its input:*

$$\mathbf{a}_i^{(2)} = (\mathbf{s}_{i,M+1}; \mathbf{g}_{i,M'}; \mathbf{g}^{\leftarrow}_{i-1,M'}; \mathbf{0}; \mathbf{pos}_i), \tag{35}$$

where $\mathbf{g}_{i,M'} := (\mathbf{g}_{i,s_{i,0}}; \dots; \mathbf{g}_{i,s_{i,M'-1}})$ and $\mathbf{g}^{\leftarrow}_{i-1,M'} = (\mathbf{g}^{\leftarrow}_{i-1,s_{i,0}}; \dots; \mathbf{g}^{\leftarrow}_{i-1,s_{i,M'-1}})$.

Proof of Theorem 3. To make the proof self-contained, we first recall some key notations. The second layer is referred to as the statistics collection layer, which uses a sequence of vectors $\mathbf{h}_i^{(2)}$, $i = 1, 2, \dots, N$, defined in (8) as its input, restated as follows.

$$\mathbf{h}_i^{(2)} = (\mathbf{s}_{i,M+1}; \mathbf{0}; \mathbf{pos}_i), \tag{36}$$

where $\mathbf{s}_{i,M+1} = (\mathbf{x}_i; \dots; \mathbf{x}_{i-M})$. To rigorously specify the function of this layer, recall the definition of the k-gram statistics vector $\mathbf{g}_{i,s}$, which in plain words, is the empirical probability distribution of the next token associated with the suffix s for a sequence x_1^i. Mathematically, for a suffix s whose length is $k-1$ and the current position i,

$$\mathbf{g}_{i,s}(a) = \frac{\mathbf{n}_{i,s}(a)}{\sum_{q\in\mathcal{A}} \mathbf{n}_{i,s}(q)} \quad \forall a \in \mathcal{A}, \tag{37}$$

where $\mathbf{n}_{i,s}$ is the counting vector defined in (5).

The k-gram backward statistics vector $\mathbf{g}^{\leftarrow}_{i-1,s}$ is defined similarly, which is the empirical probability distribution of the previous token associated with the suffix s for data x_1^{i-1}, and mathematically

$$\mathbf{g}^{\leftarrow}_{i-1,s}(a) = \frac{\sum_{q\in\mathcal{A}} \mathbf{n}_{i,as}(q)}{\sum_{q\in\mathcal{A}} \mathbf{n}_{i,s}(q)} \quad \forall a \in \mathcal{A}, \tag{38}$$

where $\sum_{q\in\mathcal{A}} \mathbf{n}_{i,s}(q)$ is the number of appears of the sub-string s in the sequence x_1^{i-1}.

The multi-head attention in the second layer is consisted of $M^{(2)} = M' \leq M^{(1)} + 1 = M+1$ heads parameterized by $(W^{(2)}_{Q,m}, W^{(2)}_{K,m}, W^{(2)}_{V,m})_{m=0,1,2,\dots,M^{(2)}-1}$. Specifically, for $m = 1, 2, \dots, M^{(2)}-1$,

$$W^{(2)}_{Q,m} = \begin{pmatrix} \mathbf{I}^{(m-1)A\times(m-1)A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \; W^{(2)}_{K,m} = \begin{pmatrix} \mathbf{0}^{(m-1)A\times A} & c\mathbf{I}^{(m-1)A\times(m-1)A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{39}$$

$$W^{(2)}_{V,m} = \begin{pmatrix} \mathbf{0}^{(M^{(1)}+m)A\times A} & \mathbf{0} \\ \mathbf{I}^{A\times A} & \mathbf{0} \\ \mathbf{0}^{(M^{(2)}-1)A\times A} & \mathbf{0} \\ \mathbf{0}^{A\times A} & [\mathbf{0}^{A\times(m-1)A}, \mathbf{I}^{A\times A}, \mathbf{0}] \\ \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{40}$$

The corresponding query, key, and value vectors after the mapping are

$$W^{(2)}_{Q,m}\mathbf{h}_n^{(2)} = \begin{pmatrix} \mathbf{s}_{n,m-1} \\ \mathbf{0} \end{pmatrix}, \quad W^{(2)}_{K,m}\mathbf{h}_i^{(2)} = c\begin{pmatrix} \mathbf{s}_{i-1,m-1} \\ \mathbf{0} \end{pmatrix}, \quad W^{(2)}_{V,m}\mathbf{h}_i^{(2)} = \begin{pmatrix} \mathbf{0}^{(M^{(1)}+m)A\times 1} \\ \mathbf{x}_i \\ \mathbf{0}^{(M^{(2)}-1)A\times 1} \\ \mathbf{x}_{i-m} \\ \mathbf{0} \end{pmatrix}.$$

For $m = M^{(2)}$, $W^{(2)}_{Q,m}, W^{(2)}_{K,m}$ are of the same structure, while $W^{(2)}_{V,m}$ does not contains that $\mathbf{I}^{A\times A}$ in that $[\mathbf{0}^{A\times(m-1)A}, \mathbf{I}^{A\times A}, \mathbf{0}]$ block, and thus $W^{(1)}_{V,m}\mathbf{h}_i^{(1)}$ does not have $\mathbf{x}_{i-m}$.

It is not hard to see that taking $c \to \infty$ gives

$$(\mathbf{s}_{i,M^{(1)}+1}; \mathbf{g}_{i,M^{(2)}-1}; \mathbf{g}^{\leftarrow}_{i-1,M^{(2)}-1}; \mathbf{0}; \mathbf{pos}_i) = [\mathrm{MHA}(\mathbf{H}^{(2)}) + \mathbf{H}^{(2)}]_i, \tag{41}$$

where

$$\mathbf{g}_{i,M'} = (\mathbf{g}_{i,s_{i,0}}; \dots; \mathbf{g}_{i,s_{i,M'-1}})$$
$$\mathbf{g}^{\leftarrow}_{i-1,M'} = (\mathbf{g}^{\leftarrow}_{i-1,s_{i,0}}; \dots; \mathbf{g}^{\leftarrow}_{i-1,s_{i,M'-1}}).$$

$\square$

Note that the counting vector can be obtained via

$$\mathbf{n}_{i,s_{i,l}}(a) = \frac{\mathbf{n}_{i,s_{i,l}}(a)}{\sum_{q\in\mathcal{A}} \mathbf{n}_{i,s_{i,l}}(q)} \frac{\sum_{q\in\mathcal{A}} \mathbf{n}_{i,s_{i,l}}(q)}{\sum_{q\in\mathcal{A}} \mathbf{n}_{i,s_{i,l-1}}(q)} \cdots \frac{\sum_{q\in\mathcal{A}} \mathbf{n}_{i,s_{i,1}}(q)}{\sum_{q\in\mathcal{A}} \mathbf{n}_{i,s_{i,0}}(q)} \left(\sum_{q\in\mathcal{A}} \mathbf{n}_{i,s_{i,0}}(q)\right) \tag{42}$$

$$= \mathbf{g}_{i,s_{i,l}}(a) \left(\prod_{j=0}^{l-1} \mathbf{g}^{\leftarrow}_{i-1,s_{i,j}}(x_{i-j})\right) \cdot i, \tag{43}$$

by the information contained in vector $(\mathbf{s}_{i,M^{(1)}+1}; \mathbf{g}_{i,M^{(2)}-1}; \mathbf{g}^{\leftarrow}_{i-1,M^{(2)}-1}; \mathbf{0}; \mathbf{pos}_i)$.

Since $p^e_{i,s_{i,l}}$ and $\mathbf{p}_{i,s_{i,l}}$ in (16) are functions of $\mathbf{n}_{i,s_{i,l}}$, we can then obtain (approximate) the following output by a sufficiently wide FF layer that

$$\mathbf{h}^3_i = (\mathbf{s}_{i,M^{(1)}+1}; \mathbf{p}_{i,D}; \mathbf{l}^e_{i,D}; \ln(p^w_{i,s_{i,D}}); \mathbf{0}; \mathbf{pos}_i), \tag{44}$$

where $\mathbf{l}^e_{i,D}$ contains the logarithm of p^e along the path from root () to $(x_{i-d+1}, \ldots, x_i)$, and $\mathbf{p}_{i,D}$ stacks the optimal prediction given suffices $s_{i,0}, \ldots, s_{i,D}$, i.e.,

$$\mathbf{l}^e_{i,D} = (\ell^e_{i,s_{i,0}}; \ell^e_{i,s_{i,1}}; \ldots; \ell^e_{i,s_{i,D}}) = (\ln(p^e_{i,s_{i,0}}); \ln(p^e_{i,s_{i,1}}); \ldots; \ln(p^e_{i,s_{i,D}})), \tag{45}$$

$$\mathbf{p}_{i,D} = (\mathbf{p}_{i,s_{i,0}}; \mathbf{p}_{i,s_{i,1}}; \ldots; \mathbf{p}_{i,s_{i,D}}), \tag{46}$$

and $\ln(p^w_{i,s_{i,D}}) = \ln(p^e_{i,s_{i,D}})$ with suffix $|s_{i,D}| = D$. These quantities can be extracted, since they are functions of the statistics collected from $\mathbf{a}^{(2)}_i$.

This functional layer essentially collects k-gram statistics for various lengths of $k = 1, 2, \ldots, M^{(2)}$ via multi-head attention and then process the the statistics for follow-up optimal scheme.

E.2.3 Inductive CTW layer

Recall the input and the expected outputs of the inductive CTW layer that

$$\mathbf{h}^{(\ell)}_i = (\mathbf{s}_{i,M^{(1)}+1}; \mathbf{p}_{i,D}; \mathbf{l}^e_{i,D}; \delta_{i,D}; \delta_{i,D-1}; \ldots; \delta_{i,D-\ell+4}; \ell^w_{i,s_{i,D+3-\ell}}; \mathbf{0}; \mathbf{pos}_i), \tag{47}$$

for $\ell = 3, 4, \ldots, 3+D$, where $\delta_{i,l} := \ln(\omega_{i,l}) - \ln(\omega_{i,l-1})$ for $l = d, D-1, \ldots, 1$ are the the weight difference, and we take $M^{(1)} = D$.

Theorem 8 (Restatement of Theorem 4)**.** *There exists a A-head transformer layer that can perform the induction: Takes $\mathbf{H}^{(\ell)}$ in (47) as input and outputs $\mathbf{H}^{(\ell+1)}$. And the final readout layer taking $\mathbf{H}^{(D+2)}$ as input can output the A-dimensional Bayesian optimal next token prediction vector $P_{\pi_{\text{CTW}}}(\cdot|x^n_{1-D}) = \sum_{l=0,\ldots,D} \omega_{n,l}\mathbf{p}_{n,s_{n,l}}$.*

Proof of Theorem 4. For any fixed $\ell = 3, 4, \ldots, 2+D$, we specify the construction for the ℓ-th transformer layer. It contains A heads and for each $m = 1, 2, \ldots, A$, the Q, K, V matrices are

$$W^{(\ell)}_{Q,m} = \begin{pmatrix} \mathbf{I}^{(D+1-\ell)A\times(D+1-\ell)A} & \mathbf{0} \\ \mathbf{0} & [\mathbf{e}_m, \mathbf{0}^{A\times 2}] \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}^{2\times 2} \end{pmatrix}, \; W^{(\ell)}_{K,m} = \begin{pmatrix} c\mathbf{I}^{(D+2-\ell)A\times(D+2-\ell)A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & c\mathbf{I}^{2\times 2} \end{pmatrix},$$

$$W^{(\ell)}_{V,m} = \begin{pmatrix} \mathbf{0}^{(\text{place}_\ell+m)\times(\text{place}_\ell+m)} & \mathbf{0} \\ [\mathbf{0}^{1\times(\text{place}_\ell-1)}, 1] & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix},$$

where $\mathbf{e}_m$ is the A-dimensional one-hot vector at position m, and $\text{place}_\ell = (M^{(1)}+D+2)A+D+\ell-1$ is index of element $\ell^w_{i,s_{i,D+3-\ell}}$ in $\mathbf{h}^{(\ell)}_i$. The corresponding query, key, and value vectors after the mapping are

$$W^{(\ell)}_{Q,m}\mathbf{h}^{(\ell)}_n = \begin{pmatrix} \mathbf{s}_{n,D+1-\ell} \\ \mathbf{e}_m \\ \mathbf{0} \\ \mathbf{pos}_n \end{pmatrix}, \quad W^{(\ell)}_{K,m}\mathbf{h}^{(\ell)}_i = c\begin{pmatrix} \mathbf{s}_{i,D+2-\ell} \\ \mathbf{0} \\ \mathbf{pos}_i \end{pmatrix}, \quad W^{(\ell)}_{V,m}\mathbf{h}^{(\ell)}_i = \begin{pmatrix} \mathbf{0}^{(\text{place}_\ell+m)\times 1} \\ \ell^w_{i,s_{i,D+3-\ell}} \\ \mathbf{0} \end{pmatrix}.$$

At position n, the query of m-head will select the latest (due to positional embedding) position with suffix $[\mathbf{s}_{n,D+1-\ell};\mathbf{e}_m]$, and append its ℓ^w at the end. It is not hard to see that taking $c \to \infty$ gives

$$\begin{aligned}\mathbf{a}_i^{(\ell)} &= [\mathrm{MHA}(\mathbf{H}^{(2)}) + \mathbf{H}^{(2)}]_i \\ &= (\mathbf{s}_{i,D+1};\mathbf{p}_{i,D};\mathbf{l}^e_{i,D};\delta_{i,D};\delta_{i,D-1};\ldots;\delta_{i,D+4-\ell};\ell^w_{i,s_{i,D+3-\ell}};[\ell^w_{i,qs_{i,D+2-\ell}}]_{q\in\mathcal{A}};\mathbf{0};\mathbf{pos}_i)\end{aligned}$$

Recall $\ln(\omega_{n,l})-\ln(\omega_{n,l-1}) = \ln(1-\lambda)-\mathbb{I}_{l=D}\ln(\lambda)+\ell^e_{n,s_{n,l}}-\ell^e_{n,s_{n,l-1}}+\sum_{q\in\mathcal{A}}\ell^w_{n,qs_{n,l-1}}-\ell^w_{n,s_{n,l}}$ by Theorem 1. $\delta_{i,D+3-\ell} = \ln(\omega_{i,D+3-\ell}) - \ln(\omega_{i,D+2-\ell})$ can be computed by $\mathbf{a}_i^{(\ell)}$ and thus $\mathbf{h}_i^{(\ell+1)}$ can be approximated via the FF layer following the ℓ-th multi-head attention layer.

The final layer approximate an A-dimensional vector

$$P_{\pi_{\text{CTW}}}(\cdot|x^n_{1-D}) = \sum_{l=0,\ldots,D} \omega_{n,l}\cdot\mathbf{p}_{n,s_{n,l}}(\cdot), \tag{48}$$

by an FF layer taking input

$$\mathbf{h}_n^{(D+3)} = (\mathbf{s}_{n,M^{(1)}+1};\mathbf{p}_{n,D};\mathbf{l}^e_{n,D};\delta_{n,D};\ldots;\delta_{n,1};\mathbf{0};\mathbf{pos}_i). \tag{49}$$

The proof is now complete. □

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes] .

   Justification: Contributions and claims are clearly stated and match the theory and experimental results of the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes] .

   Justification: See the conclusion.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes] .

   Justification: Yes, the assumptions and proofs are clearly stated.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes] .

Justification: Yes, the experimental setup is described in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No] .

Justification: Source code will be released after paper acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes] .

   Justification: Yes, details are given in the appendix

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes] .

   Justification: Yes, error bars are included in the result

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes] .

   Justification: The compute resources are described in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes] .

   Justification: Nothing to justify.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA] .

    Justification: It is mainly a technical study without foreseeable social impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA] .

Justification: The model is small, and is not expected to lead to any risk.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA] .

Justification: Nothing to report

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA] .

Justification: Nothing to report

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA] .

Justification: Nothing to report

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer:[NA] .

Justification: Nothing to report

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.