The Expressivity Boundary of Probabilistic Circuits: A Comparison with Large Language Models

Zhiyu Zhao[1*], **Xuejie Liu**[2,3*], **Muhan Zhang**[2], **Anji Liu**[1]
[1]School of Computing, National University of Singapore
[2]Institute for Artificial Intelligence, Peking University
[3]School of Intelligence Science and Technology, Peking University
zhiyu.zhao@nus.edu.sg, xjliu@stu.pku.edu.cn,
muhan@pku.edu.cn, anjiliu@comp.nus.edu.sg

Abstract

Probabilistic Circuits (PCs) are deep generative models that support exact and efficient probabilistic inference. Yet in autoregressive language modeling, PCs still lag behind Transformer-based large language models (LLMs), suggesting an important expressivity gap. In this work, we compare PCs and LLMs under a unified autoregressive formulation. First, an output bottleneck: PCs parameterize predictions as convex combinations in probability space, which struggles to represent the sharp distributions typical of language; adopting a logit-space parameterization substantially narrows this gap. Second, a context-encoding bottleneck: we prove that structured-decomposable PCs can achieve high separation rank on vtree-aligned partitions, but show, both theoretically and empirically, that this capacity is limited to partitions aligned with the fixed routing structure, leading to severe degradation when the data exhibits heterogeneous dependency topologies. We further prove that decomposable PCs are strictly more expressive than structured-decomposable ones, though effectively optimizing them remains an open challenge.

1 Introduction

Generative models have achieved remarkable success in learning high-dimensional distributions and producing realistic samples across data modalities [11, 9, 10]. Despite this progress, most state-of-the-art neural generative models are optimized primarily for sampling and scoring a narrow family of queries (e.g., next-token probabilities in autoregressive LLMs [34, 4, 35]), while many downstream settings require richer probabilistic reasoning such as exact marginalization, conditioning, and structured inference.

Probabilistic Circuits (PCs) [5] are a class of deep generative models that occupy a distinct point in this landscape: by design, they support *exact* and *efficient* probabilistic queries over their learned distributions, including arbitrary marginal and conditional probability computations. This tractability offers a sharp contrast to dominant neural paradigms, where inference is often approximate or restricted to model-native queries [33, 36, 12]. As a result, PCs have enabled strong results in inference-demanding applications such as controlled generation [38, 42], lossless (or near-lossless) compression [17], and reinforcement learning [22], where repeated conditional queries and marginalizations are central to the task. However, on standard generative modeling benchmarks, PCs still lag behind modern neural generators, suggesting an important open question: *what are the root causes of the expressivity gap between tractable PCs and neural sequence models?*

[*]Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Prior work has developed theoretical characterizations of PC expressiveness and succinctness, typically analyzing PCs in isolation or comparing subclasses within the tractable modeling family [7, 26]. In contrast, this paper takes a different route: we directly compare PCs against the dominant neural paradigm in sequence modeling, namely Transformer-based Large Language Models (LLMs). To enable a head-to-head analysis, we cast both model classes under a unified autoregressive view as *context-conditioned encoders*: given a prefix $x_{<t}$, both models instantiate a mapping $\theta_t = \mathrm{Encoder}(x_{<t})$ that produces a predictive distribution over the next token X_t. This unified lens allows us to investigate the expressivity gap through two structural bottlenecks: (i) the *output bottleneck*, i.e., how the internal representation is translated into a distribution over the vocabulary, and (ii) the *context encoding bottleneck*, i.e., how flexibly the model routes and fuses information from the prefix.

Our analysis reveals that the gap is not explained by a single factor, but instead decomposes into two mismatches between standard PCs and Transformers.

1) The Output Mismatch. Transformers generate logits and apply a softmax, naturally accommodating the sparse, peaked next-token distributions of language. Standard PCs operate directly in probability space, and we find that adopting a logit-style output parameterization can substantially narrow the gap.

2) The Routing Mismatch. Self-attention enables Transformers to route information dynamically based on the input. In contrast, structured PCs use a fixed routing pattern; we find that this rigidity can significantly hurt performance on data with heterogeneous dependency structures, yet when the data's dependency topology aligns with the circuit's structure, the gap can be markedly reduced, a finding that is not obvious a priori. We further explore relaxing structured decomposability to mitigate the fixed routing limitation of PCs.

Taken together, we characterize an expressivity gap between PCs and LLMs through a combination of theoretical and empirical analysis. Specifically, we attribute this gap to two structural bottlenecks, namely the output bottleneck and the context-encoding bottleneck. These findings also yield concrete implications for future work on improving PC expressiveness. *

2 Preliminaries

In this section, we review the generative modeling frameworks of Transformer-based Large Language Models (LLMs) and Probabilistic Circuits (PCs). While both model classes aim to represent a joint distribution over a sequence of T discrete (categorical) variables $\mathbf{X} = \{X_1, \ldots, X_T\}$, they differ fundamentally in their computational structures and probabilistic semantics. We therefore first introduce the basic syntax and semantics of each model class, and then present a unified perspective in Section 3 to support a direct comparison of their expressive power.

Large Language Models LLMs are typically defined as autoregressive models that decompose the joint distribution $p(\boldsymbol{x})$ into a product of conditional distributions:

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p(x_t | \boldsymbol{x}_{<t}), \tag{1}$$

where $\boldsymbol{x}_{<t} = \{x_1, \ldots, x_{t-1}\}$ denotes the prefix context. The core of a modern LLM is the Transformer backbone f_{LLM}, which serves as a powerful context encoder. Specifically, for each token $\boldsymbol{x}_t$, the model computes a vector of parameters $\boldsymbol{\theta}_t$ (e.g., logits) that defines the categorical distribution over the vocabulary:

$$\boldsymbol{\theta}_t = f_{\mathrm{LLM}}(\boldsymbol{x}_{<t}), \quad p(x_t | \boldsymbol{x}_{<t}) = \mathrm{Softmax}(\boldsymbol{\theta}_t). \tag{2}$$

From a probabilistic perspective, the Transformer's expressiveness stems from its ability to learn complex dependencies in f_{LLM}, allowing the conditional distribution $p(x_t | \boldsymbol{x}_{<t})$ to depend on the entire history in a highly flexible manner.

Probabilistic Circuits Probabilistic Circuits [5] represent a joint distribution through a rooted Directed Acyclic Graph (DAG). Each node n in a PC explicitly represents a probability distribution

*Code is available at: `https://github.com/zhiyu-zhao-ucas/Understanding-the-Expressivity-Boundary-of-PCs`

p_n over a subset of variables $\mathbf{X}_n \subseteq \mathbf{X}$, known as its scope. A PC defines this distribution by starting from input nodes and recursively combining them through inner nodes. Specifically, input nodes are the leaves of the DAG and define primitive distributions over their scope $\mathbf{X}_n$. Inner nodes represent distributions formed by their children and are of two types: product nodes, which represent the factorized distribution of their children, and sum nodes, which represent a weighted mixture of their children's distributions. Formally, the distribution $p_n(\boldsymbol{x})$ at node n is defined recursively as:

$$p_n(\boldsymbol{x}) := \begin{cases} g_n(\boldsymbol{x}) & n \text{ is an input node,} \\ \prod_{c \in \mathrm{ch}(n)} p_c(\boldsymbol{x}) & n \text{ is a product node,} \\ \sum_{c \in \mathrm{ch}(n)} \omega_{n,c}\, p_c(\boldsymbol{x}) & n \text{ is a sum node,} \end{cases} \tag{3}$$

where $\mathrm{ch}(n)$ denotes the children of node n, $g_n(\boldsymbol{x})$ is a univariate distribution (e.g., Categorical) over a variable $X \in \mathbf{X}$, and $\omega_{n,c} \geq 0$ are learnable weights such that $\sum_{c \in \mathrm{ch}(n)} \omega_{n,c} = 1$. Compared to LLMs, a defining advantage of PCs is that they support efficient exact and efficient inference for a large class of probabilistic queries, including arbitrary marginalization and conditional probability queries [36]. This tractability is enabled by structural constraints imposed on the circuit, most notably *decomposability*, where the children of every product node have disjoint scopes, effectively simulating fully factorized distributions.

3 A Unified Autoregressive Perspective

As established in the previous section, LLMs and PCs represent two distinct modeling frameworks: while LLMs rely on neural function approximators to learn autoregressive factors, PCs directly model the joint distribution by hierarchically composing simple distributions into complex ones. To facilitate a direct comparison, we observe that the tractable nature of PCs allows them to be equivalently cast in an autoregressive form, since PCs can compute arbitrary conditional probabilities efficiently, including the exact sequence of autoregressive factors defined in Equation 1.

As a result, the PC's inference procedure for computing these autoregressive factors induces a similar context-conditioned mapping as LLMs, i.e., $\boldsymbol{\theta}_t = f_{\mathrm{PC}}(\boldsymbol{x}_{<t})$, which is analogous to f_{LLM} and also transforms a prefix into the predictive parameters of the distribution over X_t. In this sense, while a PC represents a joint distribution, it can be operationalized as an autoregressive computation graph. This allows us to directly compare the expressiveness of the Transformer's forward pass f with the PC's conditional inference h, treating both as alternative mechanisms for generating next-token distributions.

With the unified perspective of $\boldsymbol{\theta}_t = \mathrm{Encoder}(\boldsymbol{x}_{<t})$ established, where f_{LLM} and f_{PC} both serve as context-conditioned encoders, we can now investigate the sources of the expressiveness gap between both models.

In Transformer-based LLMs, the predictive parameters are typically produced by a linear output projection:

$$\boldsymbol{\theta}_t^{\mathrm{LLM}} = \mathbf{W}^{\mathrm{LLM}} \mathbf{e}^{\mathrm{LLM}}(\boldsymbol{x}_{<t}), \qquad p^{\mathrm{LLM}}(X_t \mid \boldsymbol{x}_{<t}) = \mathrm{Softmax}\big(\boldsymbol{\theta}_t^{\mathrm{LLM}}\big), \tag{4}$$

where $\mathbf{e}^{\mathrm{LLM}}(\boldsymbol{x}_{<t}) \in \mathbb{R}^d$ is the final hidden representation of the context $\boldsymbol{x}_{<t}$, and $\mathbf{W}^{\mathrm{LLM}} \in \mathbb{R}^{|\mathcal{V}| \times d}$. Since the output space is high-dimensional, with size $|\mathcal{V}|$, while the hidden dimension d is typically much smaller, the attainable logit vectors are constrained by a low-dimensional projection. This dimensionality mismatch characterizes the output bottleneck in Transformer-based language models.

PCs face an analogous bottleneck during conditional inference. Unlike Transformer hidden units, which carry unconstrained continuous activations, PC nodes have explicit probabilistic semantics: their values correspond to probabilities. Furthermore, since sum nodes represent mixture distributions, each sum node can be associated with a categorical latent variable selecting among its children [29]. When evaluating the conditional distribution of X_t, we identify a categorical latent variable Z whose states correspond to the parent nodes of the input leaves for X_t. The resulting conditional is a mixture over latent states:

$$p^{\mathrm{PC}}(x_t \mid \boldsymbol{x}_{<t}) = \sum_{z \in \mathrm{val}[Z]} p^{\mathrm{PC}}(z \mid \boldsymbol{x}_{<t})\, p^{\mathrm{PC}}(x_t \mid z). \tag{5}$$

This mixture can be written in matrix form. Let $\mathbf{e}^{\mathrm{PC}}(\boldsymbol{x}_{<t}) \in \Delta^k$ denote the posterior vector over the $k = |\mathrm{val}[Z]|$ latent states, with entries $\mathbf{e}^{\mathrm{PC}}_j(\boldsymbol{x}_{<t}) = p^{\mathrm{PC}}(z_j \mid \boldsymbol{x}_{<t})$. Let $\mathbf{W}^{\mathrm{PC}} \in \mathbb{R}_+^{|\mathcal{V}| \times k}$ be the emission matrix, where $[\mathbf{W}^{\mathrm{PC}}]_{i,j} = p^{\mathrm{PC}}(X_t = v_i \mid z_j)$. Each column of $\mathbf{W}^{\mathrm{PC}}$ is a categorical distribution over the vocabulary. Therefore,

$$\boldsymbol{\theta}^{\mathrm{PC}}_t := p^{\mathrm{PC}}(X_t \mid \boldsymbol{x}_{<t}) = \mathbf{W}^{\mathrm{PC}} \mathbf{e}^{\mathrm{PC}}(\boldsymbol{x}_{<t}) \in \Delta^{|\mathcal{V}|}. \tag{6}$$

This vectorized form makes the analogy with the Transformer output layer in Equation 4 explicit. In both cases, a low-dimensional context representation is mapped to a vocabulary-sized predictive object. However, the two mappings operate in different spaces. In Transformers, $\boldsymbol{\theta}^{\mathrm{LLM}}_t$ is an unconstrained logit vector, which is subsequently normalized by a softmax. In PCs, $\boldsymbol{\theta}^{\mathrm{PC}}_t$ is already the probability vector itself: both the mixture coefficients $\mathbf{e}^{\mathrm{PC}}(\boldsymbol{x}_{<t})$ and the columns of $\mathbf{W}^{\mathrm{PC}}$ lie in probability simplices. Consequently, the PC prediction is constrained to lie in the convex hull of k basis distributions. Thus, when $k \ll |\mathcal{V}|$, PCs exhibit an output bottleneck analogous to that of LLMs in dimensionality, but geometrically more restrictive because the bottleneck operates in probability space rather than logit space.

Context Encoding Bottleneck The second source of the gap is the capacity of the encoders f and h to capture dependencies within the prefix $\boldsymbol{x}_{<t}$. The self-attention mechanism of Transformers (f_{LLM}) provides a highly flexible functional encoding. It allows every variable in the context to interact non-linearly, enabling the model to learn complex, long-range dependencies without structural overhead. By contrast, to support tractable and exact inference, PCs must adhere to structural constraints (e.g., decomposability). These constraints act as a rigid structural inductive bias, limiting the "flow" of information and the complexity of interactions that can be encoded in f_{PC}. We will investigate how these structural biases fundamentally limit the PC's ability to represent the dense dependency structures that Transformers capture with ease.

4 The Output Bottleneck

As introduced in Section 3, PCs and LLMs share a common structural output bottleneck: a d-dimensional latent representation is projected into a $|V|$-dimensional space to parameterize the predictive distribution θ. However, the key distinction lies in the geometric domain of this mapping. Transformer-based LLMs perform the projection in logit space, whereas PCs operate directly in probability space via convex combinations of basis distributions.

This difference suggests a potential representational disparity. Natural language next-token distributions are often low-entropy and can exhibit sharply peaked behavior given the context. In such cases, representing distributions as convex combinations in probability space may impose stronger geometric constraints than unconstrained linear mappings in logit space. To systematically investigate the impact of this "output bottleneck", we conduct controlled experiments across two model families: Hidden Markov Models (HMMs) [5], as a representative class of PCs, and Transformers.

Controlled Model Variants. An HMM can be viewed as a specific PC architecture where the latent variable Z denotes the state, and the output mapping W_{PC} corresponds to the emission matrix. In this setup, the embedding size d is equivalent to the number of latent states. To isolate the effect of the output bottleneck, we introduce a Logit-HMM variant. Instead of the standard convex combination, the Logit-HMM extracts a feature vector $\mathbf{e}_t = \log p(z_t \mid \mathbf{x}_{\leq t})$, representing the log-posterior over latent states. This feature is then passed through a linear layer $\mathbf{W} \in \mathbb{R}^{|V| \times d}$ to generate logits, which are normalized via $\mathrm{Softmax}$ to produce the final distribution.

To provide a parallel intervention within the Transformer family, we introduce a Prob-Transformer variant that replaces the standard logit-space output layer with a probability-space parameterization. Specifically, instead of producing logits followed by a Softmax, the model predicts a mixture over a set of basis distributions, where the mixture coefficients are directly derived from the hidden representation. This enforces a convex combination structure analogous to PCs, while preserving the same context encoder (see Appendix D). For the output-bottleneck experiments in this section, we use 2-layer Transformers.

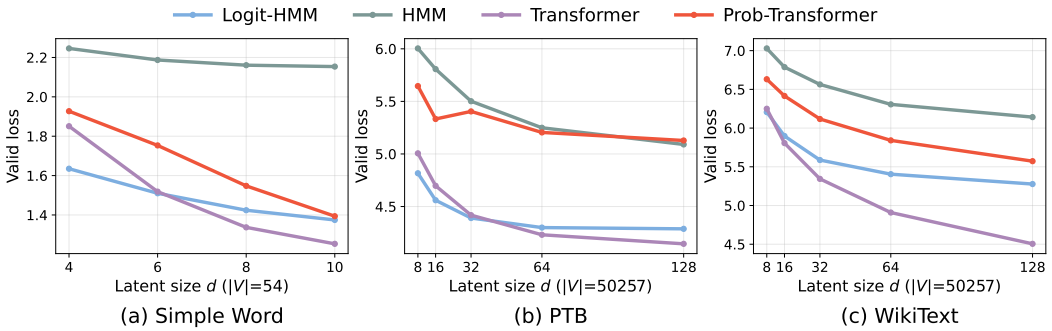

Figure 1: **Impact of output parameterization on modeling performance.** Lower validation loss is better. Across all datasets, logit-space variants improve over their probability-space counterparts within the same model family: Logit-HMM improves over HMM, and Transformer improves over Prob-Transformer. This supports the hypothesis that probability-space convex combinations impose a stronger geometric constraint than logit-space projections under a limited latent dimension.

We now evaluate these four model variants across datasets with different output scales to test whether the probability-space parameterization imposes a stronger output bottleneck than logit-space prediction. See Appendices E and F for parameter counts, architectural and training hyperparameters.

Controlled Sparse Prediction. To directly validate our hypothesis, we first use a simple word-level dataset of English words and roots [24] with a character-level tokenizer, where the task is to predict the final character from the prefix. This setting induces sparse, sharply peaked next-token distributions, since a prefix often nearly determines the final token. Because the vocabulary is small ($|\mathcal{V}| = 54$), we sweep smaller latent sizes $d \in \{4, 6, 8, 10\}$ so that $d \ll |\mathcal{V}|$ and the output bottleneck remains nontrivial. As shown in Figure 1 (a), the logit-space variants consistently outperform their probability-space counterparts under the same latent capacity. This indicates that even in a small-vocabulary setting, probability-space convex combinations are less efficient at representing sharp predictive distributions than logit-space projections.

Real-World Language Modeling. To further validate our hypothesis on more realistic data, we extend the evaluation to two standard language modeling benchmarks: the Penn Treebank (PTB) [25], using $\sim$20K training sequences of length 64, and a larger-scale WikiText-103 [27] subset with 200K training sequences of length 128. All models are trained using the standard autoregressive language modeling objective. We adopt the GPT-2 tokenizer [34] in both settings, yielding a vocabulary size of $|\mathcal{V}| = 50{,}257$, and sweep $d \in \{8, 16, 32, 64, 128\}$.

In these settings, the output bottleneck becomes more pronounced since low-dimensional latent representations must be projected into a much larger vocabulary space. The results in Figure 1 (b) and (c) again show a consistent within-family pattern: logit-space variants outperform their probability-space counterparts across both datasets. On PTB, the Logit-HMM achieves performance comparable to the Transformer, suggesting that improving the output parameterization alone can substantially narrow the gap in this setting. On the larger WikiText subset, while the Logit-HMM remains substantially stronger than the standard HMM and also outperforms the Prob-Transformer, a gap with the Transformer persists. This indicates that once the probability-space output bottleneck is alleviated, additional factors such as context encoding become increasingly important at larger scale.

Parameter Matching and Optimizer Robustness. To account for differences in parameter counts, we further conduct approximately parameter-matched comparisons across three PC architectures: HMM, Monarch, and HCLT. The logit-space variants consistently outperform their probability-space counterparts on both PTB and WikiText-103 (Appendix G). We also evaluate a parameter-matched HMM pair using four different optimizers, finding that the logit-space advantage persists across all tested optimizers (Appendix H).

5 The Context Encoding Bottleneck

We now turn to the second source of the expressiveness gap: each encoder's capacity to fuse information across context variables. A model that fails to capture rich dependencies within $\boldsymbol{x}_{<t}$will produce poor conditionals. We therefore ask: how richly can each architecture combine information from distinct subsets of $\boldsymbol{x}_{<t}$?

In a Transformer, information fusion is carried out by self-attention. Attention weights are computed as a function of the input, allowing the model to dynamically rewire which positions interact and how strongly. This input-dependence lets the same architecture instantiate distinct effective interaction graphs depending on the context, capturing complex, long-range dependencies without prior structural commitment. Unlike Transformers, PCs commit to a fixed interaction structure at design time. Cross-variable fusion occurs only at product nodes, whose children have pairwise disjoint scopes; sum nodes operate over children with identical scopes and thus do not directly combine information across different variables. Dependencies are instead captured indirectly: sum nodes assign different mixture weights to different factorization patterns, so variable interactions are modeled collectively through the global mixture structure. To make this fixed structure precise, we focus on structured-decomposable PCs [12, 6, 23], where all scope splits are governed by a fixed binary tree called a *vtree*.

Definition 1 (Vtree)**.** [32, 12] A *vtree* $\mathcal{T}$ over variables $\mathbf{X}$ is a full binary tree whose leaves are in bijection with $\mathbf{X}$. Each internal node $v \in \mathcal{T}$ is associated with a scope $S_v \subseteq \mathbf{X}$, defined recursively as $S_v = S_v^\ell \cup S_v^r$, where S_v^ℓ and S_v^r are the scopes of its left and right children respectively, with $S_v^\ell \cap S_v^r = \varnothing$. The root node has scope $\mathbf{X}$.

A PC is *structured-decomposable* with respect to $\mathcal{T}$ if, at every product node n with scope S_n, the scopes of its children correspond exactly to (S_v^ℓ, S_v^r) for the unique vtree node v such that $S_v = S_n$. This means that the vtree fully enumerates the set of all possible variable interactions, *making the expressiveness constraints imposed by the fixed structure explicit and analyzable.*

To quantify and compare the richness of variable fusing in PCs and LLMs, we need a formal measure of *how much a function can couple information across a partition of its inputs*. We adopt *separation rank* as our primary complexity measure.

Definition 2 (Separation Rank)**.** Let $\mathbf{X}$ be a set of discrete variables, and let (A, B) be a partition of $\mathbf{X}$ (so $A \cup B = \mathbf{X}$ and $A \cap B = \varnothing$). For a scalar function $f(\boldsymbol{x}_A, \boldsymbol{x}_B)$, the separation rank of f with respect to (A, B) is defined as

$$\operatorname{sep}(f; A, B) := \min\Big\{R \in \mathbb{N}_+ : \exists\{g_r\}_{r=1}^R, \{h_r\}_{r=1}^R, \text{such that } f(\boldsymbol{x}_A, \boldsymbol{x}_B) = \sum_{r=1}^R g_r(\boldsymbol{x}_A)\, h_r(\boldsymbol{x}_B)\Big\}.$$

Intuitively, $\operatorname{sep}(f; A, B)$ is the minimum number of rank-one cross terms needed to express f across the cut (A, B). A separation rank of 1 corresponds to a fully factored function $f(\boldsymbol{x}_A, \boldsymbol{x}_B) = g(\boldsymbol{x}_A)h(\boldsymbol{x}_B)$, meaning the two sides of the partition contribute independently. A high separation rank, on the other hand, indicates that the function encodes many linearly independent interaction patterns between A and B, and hence cannot be decomposed without a large number of terms.

Existence of High-Rank Partitions. Prior work on Transformer expressiveness has established that a Transformer of embedding dimension d and depth $L < \log_3 d$ achieves the following separation rank [40]: $\log \operatorname{sep}(f_{\text{LLM}}; A, B) = \widetilde{O}(d \log d)$. A naive expectation is that the fixed routing structure of structured-decomposable PCs fundamentally limits their separation rank, resulting in much weaker variable interactions compared to Transformers across all partitions.

Surprisingly, we show that for any structured-decomposable PC, there always exists a partition of the variables along which the circuit achieves separation rank.

Theorem 1 (Informal)**.** *Fix a vtree $\mathcal{T}$ over T variables and a maximum interface dimension d. There exists a partition $(A^\star, B^\star)$ aligned with $\mathcal{T}$ such that every structured-decomposable PC respecting $\mathcal{T}$ satisfies*

$$\log \operatorname{sep}(f_{\text{PC}}; A^\star, B^\star) = \widetilde{O}(T \log d).$$

Moreover, under the variable-domain assumption stated in Lemma A.4, this capacity is attainable up to logarithmic factors: there exists a structured-decomposable PC respecting $\mathcal{T}$, with maximum

Table 1: Performance of structured-decomposable PCs with two different vtree layouts and a Transformer baseline on the local-copy and induction-style prefix-copy datasets. The standard layout is aligned with local dependencies and the shifted-induction layout is aligned with nonlocal prefix-copy dependencies. A matched layout achieves performance close to the Transformer, while a mismatched layout degrades substantially, consistent with Theorem 1 and Proposition 1.

	`Shifted-induction PC Model`	`Standard Model`	`Transformer`
Adjacent Data	6.283±0.002	3.158±0.002	**3.018±0.000**
Distant Data	3.144±0.000	6.297±0.001	**3.018±0.000**

interface dimension d, for which

$$\log \operatorname{sep}(f_{\mathrm{PC}}; A^\star, B^\star) = \Omega(T \log d).$$

Thus, $\widetilde{\Theta}(T \log d)$ characterizes the maximum separation-rank capacity permitted by a fixed vtree on a suitable aligned partition. Formal definitions and proofs are provided in Appendix A.

Theorem 1 shows that, for any fixed vtree, there exists a suitable partition on which a structured-decomposable PC can attain separation rank exponential in the sequence length T.

6 The Cost of Fixed Routing

Theorem 1 shows that PCs can achieve high separation rank on suitable partitions. However, this capacity is realized only when the vtree aligns with the data's dependency topology.

We now demonstrate that this alignment requirement is a fundamental limitation: PCs suffer severe performance degradation under structural mismatch (Sec. 6.1), and real-world data, where different samples exhibit different dependency topologies, inherently demands the sample-dependent routing that only Transformers provide (Sec. 6.2).

6.1 Structural Selectivity in Probabilistic Circuits

We construct a controlled setting where the dependency topology of the target distribution can be explicitly manipulated, which allows us to directly test PC performance under matched and mismatched vtree layouts against a Transformer baseline.

Setup. We use two synthetic autoregressive datasets, each emphasizing a different dependency topology (details in Appendix B.2). For PCs, we use a structured-decomposable model with a balanced binary vtree, fixing the architecture and varying only the *leaf order*. We compare a *standard* layout (original token order, aligned with local dependencies) and a *shifted-induction* layout (pairing each prefix position with its copy target, aligned with induction-style dependencies). Each layout is matched to one dataset and mismatched to the other. We compare against a standard causal Transformer trained on the same next-token objective; all models are evaluated by token-level NLL on held-out data. Full experimental details are in Appendix B.

Result. Table 1 shows that when the vtree layout is aligned with the data's dependency topology, the PC achieves NLL close to the Transformer. This is consistent with Theorem 1: the aligned vtree places the relevant variable interactions along high-rank cuts, allowing the PC to capture them as effectively as a Transformer of comparable size. However, this high-rank capability is not universal. Formally, the high separation rank from Theorem 1 is attainable only on a restricted family of vtree-aligned partitions; any degeneracy in the circuit's local computations yields strictly smaller rank (Proposition 1 in the appendix). The mismatched layout in Table 1 confirms this: NLL increases substantially, indicating that the relevant partition falls outside the high-rank family.

6.2 The Necessity of Topology Alignment

The previous experiment shows that a single vtree cannot simultaneously align with multiple dependency topologies. In practice, natural data rarely exhibits a uniform dependency pattern. For

Table 2: Validation NLL of Transformers with different attention routing structures trained on the mixed-topology dataset. Vanilla denotes a standard causal Transformer with unconstrained self-attention. Adj Attn and Dist Attn denote Transformers with fixed tree-structured attention masks aligned with local and induction-style dependencies respectively. The vanilla Transformer maintains strong performance across both validation sets, while fixed-routing variants degrade on the topology mismatched with their attention layout, demonstrating that structural commitment to a fixed routing pattern incurs a performance cost under mixed dependency topologies regardless of model family.

Mask	`vanilla`	`adjacent`	`distant`
synthetic	**3.077±0.057**	3.312±0.461	3.154±0.016
PTB	**3.818±0.008**	3.827±0.007	3.871±0.006

instance, language modeling requires both local phrase-level interactions and long-range structural dependencies such as subject-verb agreement across clauses. We further investigate a broader question: is this limitation specific to the PC framework, or does it reflect a more general cost of fixing the routing structure regardless of model family?

To isolate the effect of structural commitment, we compare three Transformers trained on a mixed-topology dataset $\mathcal{D}_{\text{mix}}$ combining local-copy and induction-style tasks: (i) a standard causal Transformer with unconstrained attention, (ii) an *adjacent*-layout variant whose attention masks follow a fixed tree aligned with local dependencies, and (iii) a *distant*-layout variant aligned with induction-style dependencies. Both fixed-routing variants use $\log_2 T$ layers matching the tree depth. We evaluate on both the synthetic mixed-topology dataset $\mathcal{D}_{\text{mix}}$ and PTB to test whether the cost of fixed routing persists on real data.

As shown in Table 2, the vanilla Transformer achieves substantially lower NLL than both fixed-routing variants. This confirms that the cost of structural commitment is not specific to PCs: no single fixed topology can simultaneously achieve high separation rank for both local and nonlocal dependencies, whereas input-dependent attention dynamically routes information according to each sample's dependency structure.

Real-World Evaluation. We further evaluate the effect of fixed routing on PTB and WikiText-103, comparing standard Transformers with variants using fixed tree-structured attention masks. As shown in Table 4, the standard Transformer consistently achieves lower validation NLL than both fixed-routing variants on both datasets. We also include stronger PC architectures, HCLT and Monarch, which outperform the balanced-tree PC baselines. These results extend our observations to natural-language modeling.

7 Relaxing Structured Decomposability

The preceding experiments show that fixed routing is a fundamental bottleneck. In structured-decomposable PCs, this rigidity originates from the vtree, which fixes all scope splits at design time. A natural question is whether we can improve routing flexibility by relaxing structured decomposability to the broader class of decomposable PCs, while retaining the tractability guarantees of the PC framework. We first provide theoretical justification: decomposable PCs are strictly more expressive than structured-decomposable ones.

Theorem 2 (Limitation of structured-decomposable PCs, informal)**.** *There exists a function C on n variables that admits a polynomial-size decomposable PC, but does not admit any polynomial-size structured-decomposable PC. See Theorem A.5 for a formal statement.*

This inspires us to investigate empirically how much this relaxation matters in practice. We evaluate two regimes: a controlled synthetic setting and two real language modelling benchmarks (PTB [25] and UD-ATIS [13]. Full details are in Appendix B.3.

Synthetic Setting. We combine the standard and shifted-induction PCs from the previous experiments via a shared sum node at the root. The resulting model is decomposable but not structured-decomposable, as its branches impose conflicting scope-splitting patterns.

Table 3: Comparison of mixture training strategies (test NLL; lower is better). *Best Single*: the best-performing single structured-decomposable PC selected by validation. *Init Mix*: each component is a structured-decomposable PC pre-trained under a distinct vtree layout, combined via a sum node. *Scratch Mix*: a decomposable mixture with the same architecture trained end-to-end. Bold indicates the better mixture variant.

Dataset	`Best Single`	`Init Mix`	`Scratch Mix`
$\mathcal{D}_{\text{mix}}$ (synthetic)	6.219±0.005	**5.431±0.018**	6.117±0.030
UD-ATIS	6.975±0.035	6.779±0.012	**6.755±0.031**
PTB	**6.527±0.000**	**6.527±0.000**	6.531±0.000

Real Datasets. For PTB and UD-ATIS, we adopt a *specialist initialization* strategy: we independently train structured-decomposable components, each under a distinct vtree layout and trained on the subset of data whose dependency topology best matches that layout. We compare this against a *scratch mixture* trained end-to-end with matched parameter budget.

Results. On the synthetic $\mathcal{D}_{\text{mix}}$, the fused decomposable PC achieves the lowest NLL among all PC variants, validating Theorem 2: relaxing structured decomposability allows the model to combine complementary routing structures, recovering expressiveness that no single vtree can provide. However, the gains diminish as task complexity increases (Table 3). On UD-ATIS, the mixture still improves over the best single-vtree PC, but the margin is smaller than on the synthetic task. On PTB, the initialized mixture barely improves over the single-vtree baseline. This points to a key open challenge: while the theoretical expressiveness gap between decomposable and structured-decomposable PCs is clear, current methods for constructing and optimizing decomposable PCs do not yet scale effectively enough to close the gap with Transformers on real-world distributions.

8 Related Work

Expressivity of Probabilistic Circuits. Extensive research has analyzed the theoretical capacity of PCs. Martens and Medabalimi [26] proved fundamental limitations of decomposable and complete SPNs by showing that certain tractable distributions cannot be represented by such SPNs without exponential size. De Colnet and Mengel [7] provided a systematic succinctness map for arithmetic circuits under combinations of structural restrictions including decomposability, determinism, and structured decomposability, proving unconditional exponential separations between several circuit classes. In a complementary direction, it has been shown that, in the context of SPNs, consistency does not yield an exponential succinctness advantage over decomposability, as any complete and consistent SPN can be converted into a decomposable one with only a polynomial increase in size [31]. Other theoretical perspectives include sample complexity limits [2], relationships between circuit classes [37], and the hardness of distribution approximation [14]. Particularly relevant to our analysis is the formal connection between PCs and structured tensor factorizations [23]. Related distinctions have also been studied in Boolean knowledge compilation, where structured decomposability requires decomposable circuits to respect a fixed vtree [32]. Our Theorem 2 establishes a related separation in the arithmetic setting of probabilistic circuits, using an explicit construction in which two functions admit compact structured representations under different vtrees but not under any common vtree.

Theoretical Analysis of Transformers. Prior work has extensively analyzed the expressivity of Large Language Models, identifying a "softmax bottleneck" where the low-dimensional latent space restricts the rank of the predicted output distribution [39, 41]. However, recent work has shown that this bottleneck may not significantly limit the probabilities assigned to the most likely tokens [3]. Beyond expressivity, Godey and Artzi [8] further showed that this bottleneck also impedes optimization, as the rank-deficient output projection suppresses the vast majority of the gradient signal during backpropagation. Furthermore, theoretical studies have quantified the context-encoding capacity of self-attention using separation rank, demonstrating that Transformers can dynamically maintain high-rank interactions across arbitrary input partitions [15, 40].Our work bridges these analyses with tractable modeling, directly contrasting the Transformer's low-rank logit-space outputs and dynamic routing with the probability-space mixtures of PCs.

Empirical Advancements in Probabilistic Circuits. Recent work has significantly scaled Probabilistic Circuits to complex, high-dimensional distributions. A major line of this research focuses on structural innovations [1, 19, 23, 43, 20, 21]. However, these architectures predominantly rely on strict structured decomposability to guarantee efficient inference. Beyond structural design, the empirical success of modern PCs is further driven by continuous advancements in parameter learning [33, 30, 18] and systems frameworks [28, 16].

9 Conclusion and Limitations

In this work, we established a unified autoregressive framework to directly compare the expressive power of PCs and LLMs. Our analysis reveals two structural bottlenecks underlying the expressivity gap. First, an output bottleneck: PCs parameterize predictions as convex combinations in probability space, which struggles to represent sharp distributions; adopting a logit-space parameterization substantially narrows this gap. Second, a context encoding bottleneck: we proved that structured-decomposable PCs can achieve high separation rank, but only on vtree-aligned partitions, and that this limitation is fundamental to structured decomposability, as decomposable PCs are provably more expressive. These findings suggest two concrete directions for improving tractable probabilistic models: exploring output parameterizations beyond probability-space convex combinations, and developing learning algorithms that can effectively exploit the broader class of decomposable PCs. However, our analysis has limitations. Our separation rank results characterize worst-case expressivity gaps, and the context encoding bottleneck is primarily validated on synthetic datasets with controlled dependency topologies. Moreover, while decomposable PCs are provably more expressive than structured-decomposable ones, realizing this advantage in practice remains an open challenge that we leave for future work.

Acknowledgments and Disclosures

This work was funded in part by the National University of Singapore under its Start-up Grant (Award No: SUG-251RES2505) and the Ministry of Education, Singapore, under the Academic Research Fund Tier 1 (FY2026) (Grant ID: T1 251RES2610). XL and MZ acknowledge support by the NSFC (Grant Number 62550138 and 62561160152). The authors declare no competing interests.

References

[1] Tameem Adel, David Balduzzi, and Ali Ghodsi. Learning the structure of sum-product networks via an svd-based algorithm. In *UAI*, pages 32–41, 2015.

[2] Ishaq Aden-Ali and Hassan Ashtiani. On the sample complexity of learning sum-product networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4508–4518. PMLR, 2020.

[3] Ronen Basri and David Jacobs. The softmax bottleneck does not limit the probabilities of the most likely tokens. In *The Fourteenth International Conference on Learning Representations*, 2026.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. *UCLA. URL: http://starai.cs.ucla.edu/papers/ProbCirc20.pdf*, 2020.

[6] Meihua Dang, Antonio Vergari, and Guy Van den Broeck. Strudel: Learning structured-decomposable probabilistic circuits. In *International Conference on Probabilistic Graphical Models*, pages 137–148. PMLR, 2020.

[7] Alexis de Colnet and Stefan Mengel. A compilation of succinctness results for arithmetic circuits. In *18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*, 2021.

[8] Nathan Godey and Yoav Artzi. Lost in backpropagation: The lm head is a gradient bottleneck. *arXiv preprint arXiv:2603.10145*, 2026.

[9] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[12] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *KR*, 2014.

[13] Aslı Kuzgun, Neslihan Cesur, and Olcay Taner Yıldız. UD English-ATIS. `https://github.com/UniversalDependencies/UD_English-Atis`, 2022. Universal Dependencies v2.9. License: CC BY-SA 4.0.

[14] John Leland and YooJung Choi. On the hardness of approximating distributions with tractable probabilistic models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[15] Yoav Levine, Noam Wies, Daniel Jannai, Dan Navon, Yedid Hoshen, and Amnon Shashua. The inductive bias of in-context learning: Rethinking pretraining example design. *arXiv preprint arXiv:2110.04541*, 2021.

[16] Anji Liu, Kareem Ahmed, and Guy Van Den Broeck. Scaling tractable probabilistic circuits: a systems perspective. In *Proceedings of the 41st International Conference on Machine Learning*, pages 30630–30646, 2024.

[17] Anji Liu, Stephan Mandt, and Guy Van den Broeck. Lossless compression with probabilistic circuits. In *International Conference on Learning Representations*, 2021.

[18] Anji Liu, Zilei Shao, and Guy Van den Broeck. Rethinking probabilistic circuit parameter learning. *arXiv preprint arXiv:2505.19982*, 2025.

[19] Anji Liu and Guy Van den Broeck. Tractable regularization of probabilistic circuits. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, dec 2021.

[20] Anji Liu, Honghua Zhang, and Guy Van den Broeck. Scaling up probabilistic circuits by latent variable distillation. *arXiv preprint arXiv:2210.04398*, 2022.

[21] Xuejie Liu, Anji Liu, Guy Van den Broeck, and Yitao Liang. Understanding the distillation process from deep generative models to tractable probabilistic circuits. In *International Conference on Machine Learning*, pages 21825–21838. PMLR, 2023.

[22] Xuejie Liu, Anji Liu, Guy Van den Broeck, and Yitao Liang. A tractable inference perspective of offline rl. *Advances in Neural Information Processing Systems*, 37:70953–70980, 2024.

[23] Lorenzo Loconte, Antonio Mari, Gennaro Gala, Robert Peharz, Cassio de Campos, Erik Quaeghebeur, Gennaro Vessio, and Antonio Vergari. What is the relationship between tensor factorizations and circuits (and how can we exploit it)? *arXiv preprint arXiv:2409.07953*, 2024.

[24] Loren Lugosch. pytorch_hmm: A PyTorch implementation of Hidden Markov Models. `https://github.com/lorenlugosch/pytorch_HMM`, 2020.

[25] Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[26] James Martens and Venkatesh Medabalimi. On the expressive efficiency of sum product networks. *arXiv preprint arXiv:1411.7717*, 2014.

[27] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[28] Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Pranav Subramani, Nicola Di Mauro, Pascal Poupart, and Kristian Kersting. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks. *arXiv preprint arXiv:1901.03704*, 2019.

[29] Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the latent variable interpretation in sum-product networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(10):2030–2044, 2016.

[30] Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, jul 2020.

[31] Robert Peharz, Sebastian Tschiatschek, Franz Pernkopf, and Pedro Domingos. On theoretical properties of sum-product networks. In *Artificial Intelligence and Statistics*, pages 744–752. PMLR, 2015.

[32] Knot Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. In *AAAI*, volume 8, pages 517–522, 2008.

[33] Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE, 2011.

[34] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[35] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[36] Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, dec 2021.

[37] Benjie Wang and Guy Van den Broeck. On the relationship between monotone and squared probabilistic circuits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 21026–21034, 2025.

[38] Gwen Yidou Weng, Benjie Wang, and Guy Van den Broeck. Trace back from the future: A probabilistic reasoning approach to controllable language generation. *arXiv preprint arXiv:2504.18535*, 2025.

[39] Noam Wies, Yoav Levine, Daniel Jannai, and Amnon Shashua. Which transformer architecture fits my data? a vocabulary bottleneck in self-attention. In *International Conference on Machine Learning*, pages 11170–11181. PMLR, 2021.

[40] Jiechao Yang and Yong Liu. Etas: Zero-shot transformer architecture search via network trainability and expressivity. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6780–6795, 2024.

[41] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*, 2017.

[42] Honghua Zhang, Meihua Dang, Nanyun Peng, and Guy Van den Broeck. Tractable control for autoregressive language generation. In *International Conference on Machine Learning*, pages 40932–40945. PMLR, 2023.

[43] Honghua Zhang, Meihua Dang, Benjie Wang, Stefano Ermon, Nanyun Peng, and Guy Van Den Broeck. Scaling probabilistic circuits via monarch matrices. In *International Conference on Machine Learning*, pages 74804–74818. PMLR, 2025.

A Proofs

A.1 Preliminary Definitions

For a structured-decomposable PC with vtree $\mathcal{T}$ over variables $X = \{X_1, \ldots, X_T\}$, we introduce the following definitions used throughout the proofs.

Local state dimension. For each vtree node v, let $\mathcal{H}_v$ denote the space of functions exported from the subtree rooted at v toward the rest of the circuit. We define the *local state dimension*

$$d_v := \dim(\mathcal{H}_v).$$

In a normalized, non-redundant implementation, d_v coincides with the interface width $|I_v|$; in general, $d_v \leq |I_v|$.

Frontier sets. Given a partition $X = A \sqcup B$, a vtree node v is called *A-pure* if $U_v \subseteq A$, *B-pure* if $U_v \subseteq B$, and *mixed* otherwise. We define the maximal pure frontier sets as:

- $\mathcal{C}_A$: the set of all A-pure nodes whose parent is mixed;
- $\mathcal{C}_B$: the set of all B-pure nodes whose parent is mixed.

Inductive rank quantity. Define $R_A(v)$ as the minimal integer R such that there exist functions $\phi_1^v, \ldots, \phi_R^v : \mathcal{X}_{U_v \cap A} \to \mathbb{R}$ with the property that for every $f \in \mathcal{H}_v$, there exist coefficient functions $\alpha_1^f, \ldots, \alpha_R^f : \mathcal{X}_{U_v \cap B} \to \mathbb{R}$ satisfying

$$f(x_{U_v}) = \sum_{i=1}^{R} \phi_i^v(x_{U_v \cap A})\, \alpha_i^f(x_{U_v \cap B}).$$

Intuitively, $R_A(v)$ is the minimum number of basis functions on the A-side needed to represent all exported functions from subtree v, with B-dependent coefficients.

A.2 Proof of Theorem 1

Lemma A.1. For any vtree node v with children L and R:

1. If v is A-pure, then $R_A(v) \leq d_v$.
2. If v is B-pure, then $R_A(v) = 1$.
3. If v is mixed, then $R_A(v) \leq R_A(L) \cdot R_A(R)$.

Proof. **Case 1** (v is A-pure). Since $U_v \cap B = \varnothing$, all functions in $\mathcal{H}_v$ depend only on $x_{U_v \cap A} = x_{U_v}$. Let $\{\phi_i^v\}_{i=1}^{d_v}$ be any basis of $\mathcal{H}_v$. Then every $f \in \mathcal{H}_v$ can be written as

$$f(x_{U_v}) = \sum_{i=1}^{d_v} c_i^f\, \phi_i^v(x_{U_v}),$$

where $c_i^f \in \mathbb{R}$ are constants, which we view as trivial functions on the empty B-side. Hence $R_A(v) \leq d_v$.

Case 2 (v is B-pure). Since $U_v \cap A = \varnothing$, every $f \in \mathcal{H}_v$ is independent of x_A. Hence every such function admits the separation representation

$$f(x_{U_v}) = 1 \cdot f(x_{U_v}),$$

where the first factor is the single constant function on the A-side. Therefore $R_A(v) = 1$.

Case 3 (v is mixed). Let $r_L = R_A(L)$ and $r_R = R_A(R)$. By definition, there exist basis functions $\{\phi_i^L\}_{i=1}^{r_L}$ on $U_L \cap A$ and $\{\phi_j^R\}_{j=1}^{r_R}$ on $U_R \cap A$ such that for any $f_L \in \mathcal{H}_L$ and $f_R \in \mathcal{H}_R$:

$$f_L = \sum_{i=1}^{r_L} \phi_i^L\, \alpha_i^{f_L}, \qquad f_R = \sum_{j=1}^{r_R} \phi_j^R\, \beta_j^{f_R}.$$

Since the PC is structured-decomposable, any product gate at v combines a function $f_L \in \mathcal{H}_L$ with a function $f_R \in \mathcal{H}_R$ over disjoint scopes. Their product satisfies:

$$\begin{aligned} f_L \cdot f_R &= \left(\sum_i \phi_i^L \alpha_i^{f_L}\right)\left(\sum_j \phi_j^R \beta_j^{f_R}\right) \\ &= \sum_{i=1}^{r_L}\sum_{j=1}^{r_R} \underbrace{(\phi_i^L \phi_j^R)}_{\text{depends on } U_v \cap A} \underbrace{(\alpha_i^{f_L} \beta_j^{f_R})}_{\text{depends on } U_v \cap B}. \end{aligned}$$

The tensor-product functions $\{\phi_i^L \phi_j^R\}$ span the A-side of any product term, yielding at most $r_L r_R$ basis functions. Sum gates take linear combinations of functions with the same scope and do not increase the span dimension. Therefore all functions in $\mathcal{H}_v$ are spanned by these $r_L r_R$ tensor-product basis functions, giving

$$R_A(v) \le R_A(L) \cdot R_A(R). \qquad \square$$

Lemma A.2. Let y be the output function of a structured-decomposable PC over variables X, with vtree $\mathcal{T}$. For any partition $X = A \sqcup B$,

$$\operatorname{sep}(y;\, A, B) \;\le\; \min\left\{ \prod_{v \in \mathcal{C}_A} d_v,\; \prod_{u \in \mathcal{C}_B} d_u \right\}.$$

Proof. Since $y \in \mathcal{H}_r$ (the output function belongs to the root's exported function space), the definition of $R_A(r)$ directly gives a separation representation of y with $R_A(r)$ terms, hence

$$\operatorname{sep}(y;\, A, B) \le R_A(r).$$

Applying Lemma A.1 recursively from the root down to the frontier $\mathcal{C}_A \cup \mathcal{C}_B$, at each mixed node the bound multiplies, and at each pure node it terminates. This gives:

$$\begin{aligned} R_A(r) \;&\le\; \prod_{v \in \mathcal{C}_A} R_A(v) \cdot \prod_{u \in \mathcal{C}_B} R_A(u) \\ &\le\; \prod_{v \in \mathcal{C}_A} d_v \cdot \prod_{u \in \mathcal{C}_B} 1 \;=\; \prod_{v \in \mathcal{C}_A} d_v. \end{aligned}$$

By the symmetric argument (defining $R_B(v)$ analogously and swapping the roles of A and B), we also obtain

$$\operatorname{sep}(y;\, A, B) \le \prod_{u \in \mathcal{C}_B} d_u.$$

Taking the minimum of the two bounds completes the proof. $\square$

Lemma A.3. For any non-root vtree node v, let $A = U_v$ and $B = X \setminus U_v$. Let $\{\phi_1, \ldots, \phi_{d_v}\}$ be a basis for $\mathcal{H}_v$, and write

$$y(x_A, x_B) = \sum_{i=1}^{d_v} \phi_i(x_A)\, \alpha_i(x_B).$$

Define the *effective transfer rank* $r_v := \dim \operatorname{span}\{\alpha_1, \ldots, \alpha_{d_v}\}$. Then

$$\operatorname{sep}(y;\, A, B) = r_v \;\le\; d_v.$$

Proof. **Upper bound** ($\operatorname{sep} \le r_v$). Let $\{\beta_1, \ldots, \beta_{r_v}\}$ be a basis for $\operatorname{span}\{\alpha_i\}$, so that $\alpha_i = \sum_{j=1}^{r_v} c_{ij} \beta_j$. Substituting:

$$y(x_A, x_B) = \sum_{j=1}^{r_v} \underbrace{\left(\sum_{i=1}^{d_v} c_{ij}\, \phi_i(x_A)\right)}_{\text{depends only on } A} \beta_j(x_B),$$

which is a decomposition into r_v rank-one terms. Hence $\operatorname{sep}(y; A, B) \le r_v$.

Lower bound ($\text{sep} \geq r_v$). Suppose for contradiction that $\text{sep}(y; A, B) = m < r_v$, so that

$$y(x_A, x_B) = \sum_{k=1}^{m} g_k(x_A)\, h_k(x_B).$$

Let ℓ_i be the linear functional on H_v satisfying $\ell_i(\phi_j) = \delta_{ij}$, and extend ℓ_i linearly to the ambient function space containing the g_k's. Then, from any separation decomposition

$$y = \sum_{k=1}^{m} g_k(x_A) h_k(x_B),$$

applying ℓ_i to the x_A-side gives

$$\alpha_i(x_B) = \sum_{k=1}^{m} \ell_i(g_k) h_k(x_B).$$

This implies $\text{span}\{\alpha_i\} \subseteq \text{span}\{h_1, \ldots, h_m\}$, so $r_v \leq m$, a contradiction. Therefore $\text{sep}(y; A, B) \geq r_v$.

Combining both bounds gives $\text{sep}(y; A, B) = r_v$, and $r_v \leq d_v$ holds by construction. □

Corollary A.3.1 (Exact attainment of the upper bound). If the transfer at node v is full-rank, i.e., $r_v = d_v$, then

$$\text{sep}(y;\, U_v,\, X \setminus U_v) = d_v,$$

and the frontier-product upper bound is exactly attained on the vtree-induced cut $(U_v,\, X \setminus U_v)$.

Combine above results, we have the following result. For any structured-decomposable PC with vtree $\mathcal{T}$, there exists a partition $(A^\star, B^\star)$ of the variables that is aligned with $\mathcal{T}$ such that

$$\begin{aligned}\log \text{sep}(y; A^\star, B^\star) \leq & \min\{ \sum_{v \in \mathcal{C}_{A^\star}} \log d_v,\ \sum_{v \in \mathcal{C}_{B^\star}} \log d_v\} \\ = & \widetilde{O}(T \log d),\end{aligned}$$

Lemma A.4 (Attainability of the fixed-vtree capacity). Let $\mathcal{T}$ be any binary vtree over T variables, and suppose that each variable has at least d possible values. Then there exists a partition $(A^\star, B^\star)$ and a structured-decomposable PC respecting $\mathcal{T}$, with maximum interface dimension at most d, such that

$$\text{sep}(f_{\text{PC}}; A^\star, B^\star) = d^{\lfloor T/2 \rfloor}.$$

Consequently,

$$\log \text{sep}(f_{\text{PC}}; A^\star, B^\star) = \Omega(T \log d).$$

Proof. We construct a collection of $m = \lfloor T/2 \rfloor$ disjoint pairs of leaves recursively along the vtree.

For each vtree node v, process its subtree bottom-up while maintaining at most one unmatched leaf. A leaf is initially unmatched. At an internal node with children L and R, first retain all pairs constructed recursively inside the two child subtrees. If both children contain an unmatched leaf, pair these two leaves and leave no unmatched leaf at v. If exactly one child contains an unmatched leaf, propagate that leaf upward. Thus every subtree contains disjoint completed pairs and at most one unmatched leaf.

At the root, this procedure produces $m = \lfloor T/2 \rfloor$ disjoint pairs $(a_1, b_1), \ldots, (a_m, b_m)$. For each pair, place a_i in $A^\star$ and b_i in $B^\star$. If T is odd, assign the single remaining unmatched variable arbitrarily to one side.

For each pair define

$$q_i(x_{a_i}, x_{b_i}) = \frac{1}{d} \sum_{z=1}^{d} \mathbf{1}[x_{a_i} = z] \mathbf{1}[x_{b_i} = z],$$

and let

$$f_{\text{PC}}(x) = \prod_{i=1}^{m} q_i(x_{a_i}, x_{b_i}).$$

This function admits a structured-decomposable representation respecting $\mathcal{T}$ with maximum interface dimension d. Indeed, under the recursive pairing construction, every subtree contains at most one unmatched variable. If a subtree contains no unmatched variable, it exports only the product of the factors already completed inside that subtree and therefore requires a one-dimensional interface. If it contains one unmatched variable, it exports at most d functions, indexed by the possible state z of that variable. When two unmatched variables meet at their first common vtree node, their d states are combined to form the corresponding factor q_i, after which the resulting subtree again has a one-dimensional interface. Hence no interface dimension exceeds d.

Across the partition $(A^\star, B^\star)$, each factor q_i has matrix $\frac{1}{d} I_d$ and therefore rank d. Since the pairs involve disjoint variables, the matricization of f_{PC} across $(A^\star, B^\star)$ is, up to permutation of rows and columns, $\bigotimes_{i=1}^{m} \frac{1}{d} I_d$. Its rank is therefore $d^m = d^{\lfloor T/2 \rfloor}$. This proves the claim. □

Proof of Theorem 1. For the upper bound, Lemma A.2 gives, for every partition (A, B),

$$\operatorname{sep}(f_{\mathrm{PC}}; A, B) \le \min\left\{ \prod_{v \in \mathcal{C}_A} d_v,\ \prod_{v \in \mathcal{C}_B} d_v \right\}.$$

Since $d_v \le d$ and each frontier contains at most T nodes,

$$\log \operatorname{sep}(f_{\mathrm{PC}}; A, B) \le T \log d = O(T \log d).$$

In particular, this bound holds for the partition $(A^\star, B^\star)$ constructed in Lemma A.4.

Conversely, Lemma A.4 constructs, for every fixed vtree $\mathcal{T}$, a structured-decomposable PC respecting $\mathcal{T}$, with maximum interface dimension d, such that

$$\log \operatorname{sep}(f_{\mathrm{PC}}; A^\star, B^\star) = \lfloor T/2 \rfloor \log d = \Omega(T \log d).$$

Therefore the maximum separation-rank capacity permitted by a fixed vtree, on a suitable aligned partition, is

$$\tilde{O}(T \log d).$$

□

A.3 Proof of Strict Non-Attainment of the Frontier Bound

Proposition 1 (Strict non-attainment of the frontier bound)**.** *Let* $U_A := \prod_{v \in \mathcal{C}_A} d_v$. *If at least one of the following conditions holds, then* $\operatorname{sep}(y; A, B) < U_A$*:*

1. ***Non-full tensor growth at a mixed node:*** *there exists a mixed node* t *with children* L, R *such that* $R_A(t) < R_A(L) \cdot R_A(R)$*;*
2. ***Root output does not span the full root basis:*** $\operatorname{sep}(y; A, B) < R_A(r)$.

Proof. From the recursive bound established in the proof of Lemma A.2, we have the chain of inequalities:

$$\begin{aligned} \operatorname{sep}(y;\, A, B) \ \le\ R_A(r) \ &\le \prod_{v \in \mathcal{C}_A} R_A(v) \cdot \prod_{u \in \mathcal{C}_B} R_A(u) \\ &= \prod_{v \in \mathcal{C}_A} d_v \ =\ U_A, \end{aligned}$$

where the equality follows from $R_A(v) = d_v$ for $v \in \mathcal{C}_A$ and $R_A(u) = 1$ for $u \in \mathcal{C}_B$. Under either condition, an inequality in the recursive argument is strict. Since all rank factors are positive, this yields $\operatorname{sep}(y; A, B) < U_A$. □

Proposition 1 complements Corollary A.3.1: while the corollary identifies when the bound is exactly attained (full-rank transfer at a vtree-induced cut), the proposition identifies three structural sources of slack. In practice, condition (1) arise naturally when parameters are degenerate or when the circuit has redundant components; condition (2) reflects the additional gap between the root's internal representation and the scalar output y.

A.4 Proof of Theorem 2

Proposition 2 (No universal polynomial restructuring)**.** *Assume* $\mathrm{FP} \neq \#\mathrm{P}$. *Then there is no polynomial-time algorithm* $\mathcal{R}$ *with the following property:*

for every polynomial-size structured-decomposable and deterministic PC P *and every target vtree* T*, the algorithm* $\mathcal{R}(P, T)$ *outputs an equivalent polynomial-size structured-decomposable and deterministic PC respecting* T*.*

Proof. Suppose, for contradiction, that such an algorithm $\mathcal{R}$ exists.

Let P and Q be two polynomial-size structured-decomposable and deterministic PCs over the same variable set X, possibly respecting different vtrees. Let T_Q be the vtree respected by Q. Apply $\mathcal{R}$ to restructure P into an equivalent structured-decomposable and deterministic PC

$$P' := \mathcal{R}(P, T_Q)$$

that respects the same vtree T_Q.

Therefore, by Theorem B.2 [36], their product $P' \cdot Q$ can be computed in time polynomial in $|P'|$ and $|Q|$. Since P' is equivalent to P, this computes the product $P \cdot Q$ in polynomial time.

However, [36] show that computing the product of two structured-decomposable and deterministic circuits as a decomposable circuit is $\#\mathrm{P}$-hard. Thus we would obtain a polynomial-time algorithm for a $\#\mathrm{P}$-hard problem, implying $\mathrm{FP} = \#\mathrm{P}$, a contradiction.

Hence no such universal polynomial-time restructuring algorithm exists. □

Theorem A.5 (Limitation of structured-decomposable PCs). Let $\mathbf{Y}$ be a set of variables, and let s be an additional binary variable. Let

$$A, B : \mathcal{X}_{\mathbf{Y}} \to \mathbb{R}_{\geq 0}$$

be two functions such that:

1. A admits a polynomial-size structured-decomposable PC;
2. B admits a polynomial-size structured-decomposable PC;
3. there is no vtree T on $\mathbf{Y}$ such that both A and B admit polynomial-size structured-decomposable PCs respecting T.

Define

$$F(\boldsymbol{y}, s) = \mathbf{1}[s = 1]\, A(\boldsymbol{y}) + \mathbf{1}[s = 0]\, B(\boldsymbol{y}).$$

Then F admits a polynomial-size decomposable PC, but does not admit a polynomial-size structured-decomposable PC.

Proof. We first show that F admits a polynomial-size decomposable PC. Construct a circuit with two product nodes,

$$G_1 = \mathbf{1}[s = 1] \cdot A(\boldsymbol{y}), \qquad G_2 = \mathbf{1}[s = 0] \cdot B(\boldsymbol{y}),$$

and a sum node at the root computing

$$F(\boldsymbol{y}, s) = G_1 + G_2.$$

Each product node is decomposable, since one child depends only on s and the other depends only on $\mathbf{Y}$, so their scopes are disjoint. Hence the whole circuit is decomposable. Its size is polynomial.

Now suppose, for contradiction, that F admits a polynomial-size structured-decomposable PC C respecting some vtree $\mathcal{T}$ over $\mathbf{Y} \cup \{s\}$.

Condition on the selector variable. Setting $s = 1$ yields

$$F(\boldsymbol{y}, 1) = A(\boldsymbol{y}),$$

and setting $s = 0$ yields

$$F(\boldsymbol{y}, 0) = B(\boldsymbol{y}).$$

Therefore, the conditioned circuits

$$C\mid_{s=1} \qquad \text{and} \qquad C\mid_{s=0}$$

compute A and B, respectively.

Moreover, conditioning preserves structured decomposability and does not increase circuit size. After restricting s, both conditioned circuits respect the same restricted vtree $\mathcal{T}|_{\mathbf{Y}}$ on $\mathbf{Y}$.

Hence both A and B admit polynomial-size structured-decomposable PCs respecting one common vtree $\mathcal{T}|_{\mathbf{Y}}$, contradicting assumption (3).

Therefore F does not admit a polynomial-size structured-decomposable PC. □

Remark A.6. We construct A_k and B_k that each have polynomial-size structured-decomposable representations, but *no common vtree admits polynomial-size representations of both*.

The construction is obtained from a $k \times k$ grid. Let $x_{i,j} \in \{0, 1\}$, and define

$$A_k(x) = \prod_{i=1}^{k} \prod_{j=1}^{k-1} (x_{i,j} \vee x_{i,j+1}), \qquad B_k(x) = \prod_{i=1}^{k-1} \prod_{j=1}^{k} (x_{i,j} \vee x_{i+1,j}).$$

A_k contains only the horizontal constraints and B_k only the vertical constraints. Each consists of k disjoint paths and therefore admits an $O(k^2)$-size structured-decomposable representation, using respectively a row-aligned and a column-aligned vtree.

We now show that they cannot both admit polynomial-size representations under any common vtree. Suppose otherwise, and let T be such a vtree. Since two structured PCs respecting the same vtree are compatible, their product would have a polynomial-size structured-decomposable representation:

$$A_k B_k = \prod_{u,v \in E(G_k)} (x_u \vee x_v) =: \Phi_k.$$

However, Φ_k requires size $2^{\Omega(k)}$ under every vtree. To see this directly, any vtree contains a node inducing a balanced partition $S \mid \bar{S}$, with both sides containing $\Omega(k^2)$ variables. Any such partition of the $k \times k$ grid crosses $\Omega(k)$ grid edges. Since the grid has constant degree, these edges contain an induced matching

$$(a_1, b_1), \ldots, (a_m, b_m), \qquad m = \Omega(k),$$

with $a_i \in S$ and $b_i \in \bar{S}$. Setting all other variables to 1 restricts Φ_k to

$$H_m(a, b) = \prod_{i=1}^{m} (a_i \vee b_i).$$

Across the partition $a \mid b$, its function matrix is

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{\otimes m},$$

whose rank is $2^m = 2^{\Omega(k)}$. Hence the separation rank of Φ_k across this vtree cut is at least $2^{\Omega(k)}$. A structured-decomposable PC respecting T has separation rank at most its number of interface nodes, and thus at most its circuit size.

Therefore every structured representation of Φ_k respecting T has size $2^{\Omega(k)}$, a contradiction.

B Experimental Details

B.1 Model Architectures

Balanced-tree PC. All PC models are structured-decomposable with respect to a full balanced binary tree vtree over the N sequence positions, where N is required to be a power of two. The tree is constructed by repeatedly halving the sequence: at each level, adjacent pairs of nodes are merged by alternating sum and product layers, until a single root node remains. Concretely, the architecture consists of $\log_2 N$ stages, each comprising a `SumLayer` followed by a `ProductLayer`, and a final `SumLayer` at the root with a learned mixture weight vector. Each node maintains C channels throughout. The leaf nodes are parameterized as categorical distributions over the vocabulary, implemented as a `CategoricalLeaf` module that outputs per-position, per-channel log-probabilities.

Tree layouts. The vtree structure is controlled by a *leaf permutation*, which maps tree positions to original sequence indices. We use two layouts:

Standard layout. The identity permutation

$$\pi_{\text{std}}(i) = i, \quad i = 0, \ldots, N-1,$$

so that tree position i corresponds to original token i. Under this layout, adjacent tokens are placed as siblings at the lowest level of the tree, making the model naturally aligned with local dependency patterns.

Shifted-induction layout. Designed to align the tree with induction- style prefix-copy dependencies, where token t in the second half copies token $t - N/2$ in the first half. Let $H = N/2$. We construct sibling pairs of the form $(\text{pair_id},\ H + \text{pair_id})$, placing each prefix token together with its corresponding copy in a shared low-level subtree. The pairs are then ordered according to a bit-reversal permutation to balance the tree depth across all positions. Formally, let $\{r_0, \ldots, r_{H-1}\}$ be the bit-reversal of $\{0, \ldots, H-1\}$ using $\lfloor \log_2 N \rfloor - 1$ bits. The leaf permutation is defined as

$$\pi_{\text{ind}}(2k) = r_k, \qquad \pi_{\text{ind}}(2k+1) = H + r_k,$$

where $k = 0, \ldots, H-1$, so that tree positions $(2k, 2k+1)$ are siblings corresponding to the original token pair $(r_k,\ H + r_k)$.

In both cases, the future mask used for autoregressive marginalization is derived directly from the permutation: tree position p is treated as a future position at prediction step t if and only if $\pi^{-1}(p) \geq t$.

Transformer The Transformer baseline in Section 6 is a decoder-only causal Transformer. It uses standard causal self-attention without any tree-structured attention mask. The model has 5 layers, 4 attention heads, a hidden size of 256, dropout 0.1, context length 32, and approximately 4.09M trainable parameters. It is trained with an autoregressive next-token cross-entropy objective.

B.2 Experimental Details for Synthetic Data

Local-copy dataset. Each sequence of length N is constructed by sampling $N/2$ tokens independently and uniformly from the vocabulary $\{1, \ldots, V\}$ and repeating each token in the immediately following position:

$$x = (a_1, a_1, a_2, a_2, \ldots, a_{N/2}, a_{N/2}), \quad a_i \sim \text{Uniform}(\{1, \ldots, V\}).$$

Each even-position token is a copy of the token immediately before it, $x_{2i} = x_{2i-1}$, whereas each odd-position token is sampled independently of the preceding context.

Induction-style prefix-copy dataset. Each sequence of length N is constructed by sampling a prefix of length $N/2$ independently and uniformly from $\{1, \ldots, V\}$ and appending an exact copy:

$$x = (a_1, \ldots, a_{N/2}, a_1, \ldots, a_{N/2}), \quad a_i \sim \text{Uniform}(\{1, \ldots, V\}).$$

Every token in the second half repeats the token $N/2$ positions earlier, $x_t = x_{t-N/2}$ for $t > N/2$, whereas the tokens in the first half are mutually independent.

Training objective. Both datasets are trained with the same autoregressive objective. The Transformer minimizes the next-token cross-entropy averaged over the $N-1$ targets $x_2, \ldots, x_N$. The PC models minimize the autoregressive negative log-likelihood $-\sum_{t=1}^{N} \log p(x_t \mid x_{<t})$, averaged over all N positions, including x_1.

Mixed-topology dataset. The mixed training distribution is constructed by combining the two datasets above with equal probability: each sample is drawn from the local-copy generator with probability 0.5 and from the induction-style prefix-copy generator with probability 0.5. The validation set is drawn from the same mixed distribution, and we report the per-token NLL averaged over it.

B.3 Experimental Details for PTB and UD-ATIS

PTB. We use the Penn Treebank [25] for autoregressive language modeling with packed token segments of length 32 and a vocabulary of 10,000 tokens. We train 9 structure specialists and compare frozen specialist-initialized, unfrozen specialist-initialized, and from-scratch late mixtures. All specialists and mixtures use 8 channels, batch size 64, learning rate 10^{-4}, and weight decay 10^{-4}; specialists are trained for 30 epochs.

UD-ATIS. We use the UD English ATIS treebank for sentence-level autoregressive language modeling, filtering sentences to lengths between 4 and 32 tokens with a vocabulary of 2,000 tokens. We train 6 structure specialists and compare frozen specialist-initialized, unfrozen specialist-initialized, and from-scratch late mixtures. All specialists and mixtures use 4 channels, batch size 64, learning rate 10^{-4}, and weight decay 10^{-4}; models are trained for 3 epochs.

Structures. The structure denotes a fixed vtree layout over token positions. We consider several families of layouts: balanced trees, left- and right-branching chains, local block trees, strided or dilated chains, boundary-to-center or center-to-boundary chains, and half-pairing layouts. Local block layouts first compose nearby positions within contiguous blocks, while strided and dilated layouts group non-adjacent positions. The half-pairing layouts pair position i with position $N/2 + i$, which is intended to favor repeated or induction-like dependencies across the two halves of a sequence.

Each specialist uses a fixed vtree over token positions. PTB uses nine layouts covering left/right chains, local block trees, strided parity-based chains, outside-in ordering, and half-sequence pairing. UD-ATIS uses six specialist slots drawn from dilated chain, inside-out chain, right chain, and shifted induction. These structures provide different inductive biases over local, directional, long-range, and repeated-position dependencies.

B.4 Hardware Configuration

All experiments are reproducible with single-process, single-GPU runs. Our hardware configuration consists of one NVIDIA RTX PRO 6000 Blackwell Server Edition GPU (98 GB memory) and an AMD EPYC 9555 CPU (256 logical cores, 1.0 TiB RAM).

C Real-Data Evaluation of Routing Constraints

We extend the fixed-routing analysis to real language modeling on PTB and WikiText-103. Alongside the standard Transformer, we evaluate Transformers with adjacent- and distant-layout tree attention masks, as well as balanced-tree PCs with different vtree layouts and two stronger PC architectures, Logit-HCLT and Logit-Monarch. The results of this additional experimental suite are reported in Table 4. Logit-HCLT and Logit-Monarch outperform Logit-balanced-tree PCs, while combining two vtree layouts yields modest gains. Although architectural differences prevent isolating the effect of routing, these results demonstrate the importance of considering stronger PC baselines.

D Prob-Transformer Parameterization

To isolate the effect of the output space independently from the context encoder, we construct a Prob-Transformer variant that replaces the standard logit-space output layer with a probability-space parameterization.

Model formulation. Given a prefix $\mathbf{x}_{\leq t}$, the Transformer encoder produces a hidden representation

$$\mathbf{h}_t \in \mathbb{R}^d.$$

Instead of mapping $\mathbf{h}_t$ to logits, we directly parameterize the next-token distribution as a convex combination of basis distributions:

$$p(x_{t+1} \mid \mathbf{x}_{\leq t}) = \sum_{i=1}^{r} c_i(\mathbf{x}_{\leq t})\, B_i,$$

Table 4: Validation average-token NLL on real-language benchmarks (lower is better). These are results from an additional experimental suite and should not be directly compared numerically with earlier tables using potentially different training or preprocessing configurations. The sequence length is 64.

Model	PTB	WikiText-103
Balanced-tree PC (standard)	6.041	7.115
Balanced-tree PC (shifted induction)	6.337	7.416
Balanced-tree PC (fused)	6.014	7.058
Logit-HCLT	4.723	6.022
Logit-Monarch	4.469	5.454
Fixed-routing Transformer (adjacent)	4.015	4.896
Fixed-routing Transformer (distant)	4.000	4.939
Standard Transformer	**3.961**	**4.842**

where:

- $c(\mathbf{x}_{\leq t}) \in \Delta^r$ is a vector of mixture coefficients,
- $\{B_i\}_{i=1}^r$, with $B_i \in \Delta^{|V|}$, are learnable basis distributions over the vocabulary,
- r is the mixture rank.

Mixture coefficient parameterization. We derive the mixture coefficients directly from the hidden representation via a positive normalization:

$$c_i(\mathbf{x}_{\leq t}) = \frac{\mathrm{softplus}(h_{t,i})}{\sum_{j=1}^r \mathrm{softplus}(h_{t,j})}.$$

In our implementation, we set $r = d$, tying the mixture rank to the embedding dimension. This mirrors the Logit-HMM construction, where the latent state dimension serves as the feature space.

This formulation enforces that the predictive distribution lies in the convex hull of $\{B_i\}_{i=1}^r$, thereby introducing a probability-space output bottleneck analogous to that of PCs. Importantly, the context encoder remains unchanged, allowing us to isolate the effect of the output parameterization while keeping the representational capacity of the encoder fixed.

E Model Size

Parameter scaling. Let V denote the vocabulary size and d the latent or embedding dimension. In our PTB and WikiText experiments, $V = 50{,}257$, so the dominant contribution to the parameter count comes from vocabulary-sized projections.

The parameter counts for the four model families scale approximately as:

$$\begin{aligned}
\text{HMM}(d) &= d^2 + Vd + d \;\approx\; Vd, \\
\text{Logit-HMM}(d) &= d^2 + 2Vd + V + d \;\approx\; 2Vd, \\
\text{Transformer}(d) &\approx 2Vd + 24d^2 + O(Ld), \\
\text{Prob-Transformer}(d) &\approx 2Vd + 24d^2 + O(Ld).
\end{aligned}$$

Discussion. For the Transformer and Prob-Transformer, we use $n_{\text{layer}} = 2$. The $24d^2$ term arises from the two Transformer blocks, while the $2Vd$ term comes from the input embedding and the output vocabulary projection.

The Prob-Transformer introduces an additional $d \times V$ probabilistic basis, making its parameter count comparable to that of a Transformer without weight tying. Similarly, the Logit-HMM includes both an emission matrix and a logit projection, leading to two vocabulary-sized components. In contrast, the vanilla HMM contains only a single $d \times V$ emission matrix, making it roughly a factor of two smaller.

Concrete sizes. To make the scaling concrete, we instantiate the formulas using the PTB and WikiText configuration, where $|\mathcal{V}| = 50{,}257$ and $d \in \{8, 16, 32, 64, 128\}$. The resulting parameter counts are:

d	8	16	32	64	128
HMM	0.40M	0.80M	1.61M	3.22M	6.45M
Logit-HMM	0.85M	1.66M	3.27M	6.49M	12.93M
Transformer	0.81M	1.62M	3.24M	6.54M	13.27M
Prob-Transformer	0.81M	1.62M	3.24M	6.54M	13.27M

Takeaway. Except for the vanilla HMM, the other three models have broadly comparable parameter counts at the same d. This is because they all contain two vocabulary-sized $d \times V$ components, whereas the HMM contains only one. As a result, comparisons among Logit-HMM, Transformer, and Prob-Transformer isolate differences in output parameterization rather than model capacity.

F Training Details

Overview. We evaluate our hypotheses across three datasets: **Simple Word**, **Penn Treebank (PTB)**, and **WikiText-103**. We compare four model families: **HMM**, **Logit-HMM**, **Transformer**, and **Prob-Transformer**. For Simple Word, we sweep the latent or embedding dimension over $d \in \{4, 6, 8, 10\}$; for PTB and WikiText, we sweep $d \in \{8, 16, 32, 64, 128\}$. All Transformer-based models use a 2-layer architecture. We use 2 attention heads for Simple Word and 4 attention heads for PTB and WikiText.

To maintain consistency across experiments, we adopt standardized optimization parameters for each model family. We use the AdamW optimizer with a constant learning rate and the standard autoregressive training objective. The following table summarizes the primary optimization hyperparameters.

Table 5: Common optimization hyperparameters across model families.

Hyperparameter	**HMM**	**Logit-HMM**	**Transformer**	**Prob-Transformer**
Learning Rate	0.01	5×10^{-4}	1×10^{-4}	5×10^{-4}
Weight Decay	0.0	5×10^{-6}	0.1	0.1
Dropout	0.0	0.0	0.1	0.1

Dataset-Specific Configurations. Across all datasets, we use early stopping with a patience of 5 epochs and report results based on the best validation loss. We adjust the maximum number of epochs, batch size, sequence length, and attention heads according to dataset scale.

Simple Word. We use character-level tokenization for a last-token prediction task. The vocabulary size is $|\mathcal{V}| = 54$, and we sweep $d \in \{4, 6, 8, 10\}$. Transformer-based models use 2 layers and 2 attention heads. All models are trained with a maximum of 100 epochs, early stopping patience 5, and batch size 64.

PTB. We use GPT-2 tokenization and optimize the full-sequence autoregressive objective with sequence length $L = 64$ on 17,736 training sequences. The vocabulary size is $|\mathcal{V}| = 50{,}257$, and we sweep $d \in \{8, 16, 32, 64, 128\}$. Transformer-based models use 2 layers and 4 attention heads. All models are trained with a maximum of 200 epochs, early stopping patience 5, and batch size 32.

WikiText. We use GPT-2 tokenization with sequence length $L = 128$ on a subset of 200K training sequences. The vocabulary size is $|\mathcal{V}| = 50{,}257$, and we sweep $d \in \{8, 16, 32, 64, 128\}$. Transformer-based models use 2 layers and 4 attention heads. All models are trained with a maximum of 150 epochs, early stopping patience 5, and an effective batch size of 128.

G Parameter-Matched Baselines for the HMM vs. Logit-HMM Comparison

We provide parameter-matched comparisons between probability-space and logit-space parameterizations across three PC architectures: HMM, Monarch, and HCLT, on two real-world language modeling benchmarks, PTB and WikiText-103.

For HMMs, the parameter count is dominated by the $N \times |V|$ emission matrix, while Logit-HMMs introduce an additional $|V| \times N$ output projection. Thus, at the same latent size N, a Logit-HMM has approximately twice as many parameters as a standard HMM. We therefore compare probability-space models with latent size $2N$ against logit-space models with latent size N. For the other PC architectures, we select approximately parameter-matched configurations. Here, L denotes the sequence length used in each experiment.

All results are reported as validation average per-token negative log-likelihood (NLL), where lower is better.

G.1 HMM vs. Logit-HMM

Table 6: Parameter-matched comparison of HMM and Logit-HMM on PTB ($L = 64$). Values in parentheses indicate the latent size N. Lower NLL is better.

$\approx$ Params	HMM (N)	Logit-HMM (N)
0.80M	5.81 (16)	**4.82** (8)
1.61M	5.50 (32)	**4.56** (16)
3.22M	5.25 (64)	**4.39** (32)
6.45M	5.09 (128)	**4.30** (64)

Table 7: Parameter-matched comparison of HMM and Logit-HMM on WikiText-103 ($L = 128$). Values in parentheses indicate the latent size N. Lower NLL is better.

$\approx$ Params	HMM (N)	Logit-HMM (N)
0.80M	6.79 (16)	**6.21** (8)
1.61M	6.56 (32)	**5.90** (16)
3.22M	6.31 (64)	**5.59** (32)
6.45M	6.14 (128)	**5.40** (64)

G.2 Other PC Architectures

We extend the comparison to two structurally distinct PC architectures, Monarch and HCLT, to investigate whether the logit-space advantage generalizes beyond HMMs.

G.2.1 Monarch

Table 8: Approximately parameter-matched comparison of vanilla Monarch and Logit-Monarch on PTB ($L = 64$). Values in parentheses indicate the latent size N. Lower NLL is better.

$\approx$ Params	Vanilla (N)	Logit (N)
1.7M	5.59 (36)	**4.65** (16)
3.4M	5.35 (64)	**4.47** (36)
6.9M	5.38 (144)	**4.44** (64)
13M	5.24 (256)	**4.43** (144)

Table 9: Approximately parameter-matched comparison of vanilla Monarch and Logit-Monarch on WikiText-103 ($L = 64$). Values in parentheses indicate the latent size N. Lower NLL is better.

$\approx$ Params	Vanilla (N)	Logit (N)
1.7M	6.37 (36)	**5.72** (16)
3.4M	6.18 (64)	**5.45** (36)
6.9M	6.11 (144)	**5.31** (64)
13M	5.93 (256)	**5.30** (144)

G.2.2 HCLT

Table 10: Parameter-matched comparison of vanilla HCLT and Logit-HCLT on PTB ($L = 64$). Values in parentheses indicate the latent size N. Lower NLL is better.

$\approx$ Params	Vanilla (N)	Logit (N)
0.8M	5.64 (16)	**5.09** (8)
1.67M	5.45 (32)	**4.87** (16)
3.4M	5.31 (64)	**4.72** (32)
7M	5.20 (128)	**4.62** (64)

Table 11: Parameter-matched comparison of vanilla HCLT and Logit-HCLT on WikiText-103 ($L = 64$). Values in parentheses indicate the latent size N. Lower NLL is better.

$\approx$ Params	Vanilla (N)	Logit (N)
0.8M	6.82 (16)	**6.41** (8)
1.67M	6.66 (32)	**6.16** (16)
3.4M	6.45 (64)	**6.02** (32)
7M	6.36 (128)	**5.95** (64)

G.3 Analysis

Across all tested parameter budgets, datasets, and architectures, logit-space models consistently outperform their probability-space counterparts. These results demonstrate that the advantage of logit-space output parameterization is not specific to HMMs, but also appears in other PC architectures, including Monarch and HCLT.

The consistent improvements under approximately matched parameter budgets provide further empirical evidence that probability-space convex combinations impose a stronger output bottleneck than logit-space parameterizations.

H Potential Optimization Confound

To investigate whether the logit-space advantage is sensitive to the choice of optimizer, we conduct an additional experiment on a parameter-matched pair: HMM ($N = 64$, 3.2M parameters) and Logit-HMM ($N = 32$, 3.3M parameters), evaluated on WikiText-103 ($L = 64$).

We retrain each model with four optimizers: Adam, RMSProp with and without momentum, and SGD with momentum. For each optimizer, we sweep the learning rate and report the best validation average per-token NLL.

Table 12: Effect of optimizer choice on parameter-matched HMM and Logit-HMM models on WikiText-103 ($L = 64$). We report the best validation average per-token NLL across learning rates (lower is better). Δ denotes the difference between probability-space and logit-space NLL.

Optimizer	HMM	Logit-HMM	Δ (prob $-$ logit)
Adam	6.34	**5.67**	0.67
RMSProp (no momentum)	6.32	**5.69**	0.63
RMSProp (momentum 0.9)	6.33	**5.66**	0.67
SGD (momentum 0.9)	7.07	**6.06**	1.01

As shown in Table 12, the logit-space model outperforms its probability-space counterpart under every tested optimizer, with improvements ranging from 0.63 to 1.01 nats. The advantage persists with non-adaptive SGD and momentum-free RMSProp, indicating that the logit-space gain is robust across the tested optimizers.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims presented in the abstract and introduction accurately reflect the scope and contributions of our work.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: We explicitly discuss limitations in Section 9.

   Guidelines:

   - The answer [N/A] 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: We provide the full set of assumptions and complete proofs in Section A.

Guidelines:

- The answer [N/A] 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: The paper includes all necessary information for reproduction in the Appendix.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: We release the full codebase to facilitate reproducibility.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all the necessary details in Appendix.

   Guidelines:

   - The answer [N/A] 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: We report the mean and standard deviation.

   Guidelines:

   - The answer [N/A] 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: We report computer resources in Appendix B.4.

   Guidelines:

   - The answer [N/A] 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: We have reviewed and fully adhered to the NeurIPS Code of Ethics throughout the research process.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: This work is primarily methodological and does not have direct societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not involve the release of data or models that have a high risk for misuse.

    Guidelines:

    - The answer [N/A] 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: [Yes]

    Justification: All assets used in this work, including datasets, code, and models, are publicly available. We have properly credited the original creators and ensured that all licenses and terms of use are respected.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.