WHAT'S IN A TREE ? HIERARCHICAL n-GRAM MODELS AS CONTROLLED APPROXIMATORS OF LANGUAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Hierarchical models have been proposed as synthetic settings for studying language. However, the relevance to realistic language corpora is under-explored. In this work, we introduce tree-grams, autoregressive generative n-gram models that model the conditional probability tensor $p(x_n \mid x_{1:n-1})$ through a hierarchy of order-2 and order-3 merges augmented with normalization layers, residual connections, and MLPs. At fixed n, a rank hyper-parameter, d, controls model capacity, yielding an interpretable and systematically scalable family of approximators of language. Tree-grams are shown to out-perform existing n-gram baselines for higher n, while remaining competitive with deep neural sequence models. We demonstrate the utility of such controlled approximations by using tree-grams as generative models. In particular, we show that hyper-parameter transfer under μP can be made more robust when narrower models are tuned on simpler data. In addition, tree-grams allow us to investigate the effect of pretraining and finetuning distributions. We show that performance increasingly worsens as the complexity of the finetuning distribution increases. However, the pre-training distribution complexity has little effect in a capacity-limited regime, while there is a non-monotonic effect in a horizon-limited regime. Tree-grams are interpretable by construction. In particular, terminal nodes in different layers have receptive fields whose information can be independently recovered through linear probes and activation patching. When allowed to route across candidate topologies, we observe emergent sparsity where tree-grams produce concentrated path probabilities fit by a power law with a generalized exponential cutoff.

1 INTRODUCTION

Language is hypothesized to be hierarchically structured (Chomsky, 1956; 1957; Everaert et al., 2015; Hauser et al., 2002; Goodfellow et al., 2016). Its statistical structure has been studied using classical and modern language models, including Markov-chain and n-gram models (Shannon, 1951; Brown et al., 1992; Bengio et al., 2003a). Synthetic, controlled datasets with hierarchical structure have also been proposed to model language, including Markov chains and context-free grammars (Chomsky, 1956; Hopcroft & Ullman, 1979; Klein & Manning, 2004; Kim et al., 2019). In particular, (Cagnetta et al., 2024) propose a hierarchical generative model with latent nodes generating random leaves. Tree-like generative models can successfully represent compositional dependencies and improve language modeling (Dyer et al., 2016; Kuncoro et al., 2018); however, learning linguistically meaningful tree structures directly from data remains difficult (Klein & Manning, 2004; Kim et al., 2019; Hewitt & Manning, 2019). It is also important to understand what tree-like properties are present in real data, which can be accomplished by creating trainable versions of such models.

In this work, we propose such an architecture. In particular, we propose **tree-grams**, which are hybrid n-gram models based on hierarchy and local connectivity. Tree-grams combine order-2/order-3 tensor approximation with normalization layers, residual connections, and MLPs to approximate the order-n transition tensor $p(x_n|x_{1:n-1})$. The n controls the *horizon* of the model, which fixes the block design up to a choice of topology. A hyper-parameter d controls model *capacity*. A summary of our contributions is provided below.

Summary of contributions

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

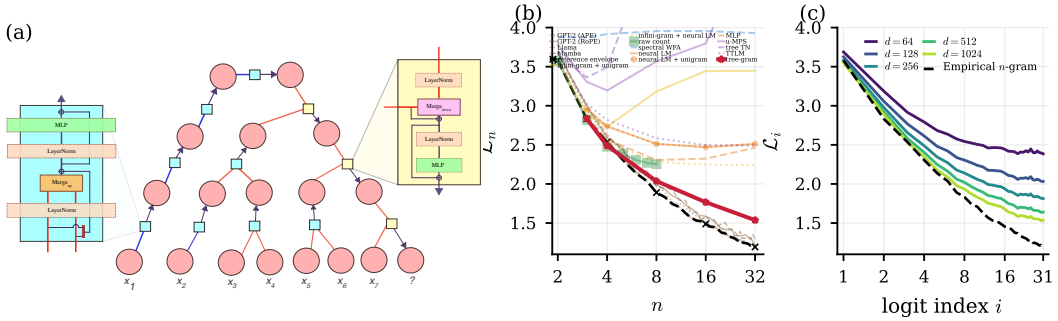

Figure 1: **(a)** The tree-gram architecture. Tree-grams consist of hybrid blocks which are used to merge intermediate nodes. Red edges are rank-3 merges ( ), blue edges denote rank-2 merges ( ). In the upward merge, the LayerNorm and MLPs are shared across the first $n^{(\ell)} - 1$ nodes for the entire layer. The last node outputs logits for next-token prediction. **(b)** Scaling with n: cross-entropy of the next-token conditional given exactly $n-1$ tokens of context. Tree-grams (in red) out-perform tested n-gram estimators for large n but fall short of deep neural sequence models tested within our compute budget. **(c)** Scaling with d: Per-token loss as a function of the rank d. We observe a smooth approach to the empirical lower envelope of the available references as d is increased.

- We present tree-grams as generative models for controlled approximation of language. The approximation is controlled through a horizon-limited mode or a capacity-limited mode. Tree-grams, in contrast with Chomsky grammars, learn latent feature representations from data. We train tree-grams on TinyStories as proof of concept.
- Tree-grams match the empirical lower bound of cross-entropy from classical smoothed counting models at $n = 2, 3$ and outperform tested n-gram estimators at longer orders. They match deep neural references through approximately $n = 8$, with a monotonically decreasing gap at longer context with an increase in the width d of the model.
- We demonstrate the utility of such controlled approximations by analyzing hyper-parameter (HP) transfer and finetuning performance. We show that HP transfer is more robust when the proxy is trained on lower-complexity synthetic data. We also investigate the effect of distributional differences between pre-training and finetuning corpora. The performance gap is shown to depend monotonically on the complexity of the finetuning distribution. On the other hand, performance gaps are nearly independent of generator capacity in the tested capacity-limited regime, but depend non-monotonically in the horizon-limited regime.
- We demonstrate that tree-grams have interpretable substructures. Linear probes can recover effective receptive fields of different nodes in a particular topology, which is absent in a different topology as well as transformers. We also observe concentrated topology-routing distributions when tree-grams are allowed to choose topologies at each layer through linear routers.

1.1 Background and Related Works

Hierarchical models have been used to study synthetic datasets resembling language. (Chomsky, 1956; Chomsky & Schützenberger, 1963; Yao et al., 2021; Cagnetta et al., 2024). Various explanations have been put forward to portray the relevance of such data to language (Thurner et al., 2015; Futrell & Hahn, 2026; Ding et al., 2016). However, all such fix hidden features based on semantic or syntactic structures. Tree-grams are introduced as generative models where the hidden node features are directly learnt from data. In addition, it is useful to have explicit hyper-parameters that control approximation of the full corpus. Such an approximation can be made by either controlling the generative model capacity or the horizon (Cagnetta et al., 2026), provided the model is expressive enough. While traditional n-gram estimators (Chen & Goodman, 1996; Parikh et al., 2014a; Liu et al., 2025) can control horizon, the sample complexity, compute budget or memory requirements are often too high. Tensor-train and MPS models provide compact, trainable, low-rank parame-

terizations of sequence distributions (Pestun & Vlassopoulos, 2017; Miller et al., 2021; Su et al., 2024), but suffer from similar problems when scaling vocabulary size or bond dimension. Zhang et al. (2019) derive a sequential tensor-space language model connecting tensor decompositions to n-gram conditionals; tree-grams instead use multiscale order-2/3 merges with residual–MLP blocks. We introduce tree-grams in light of the above considerations. Detailed background and related works are presented in Appendix B.

2 THE TREE-GRAM ARCHITECTURE

We implement tree-grams as a hierarchical architecture, with layers indexed by ℓ. The number of nodes in each layer is defined via a recursive relation

$$n_{(\ell)} = \lfloor n_{(\ell-1)}/2 \rfloor + 1; \quad n_{(0)} \equiv n. \tag{1}$$

The n here refers to the n in n-gram. The recurrence is applied till $n_{(d_n)} = 2$, where $d_n = \lceil \log_2(n-1) \rceil$ represents the depth of the network.

We label different topologies for a tree-gram with τ. A given topology decides the connections between nodes in successive layers. In particular, it decides whether a node in an upper layer has a single child ( ) or two children ( ). We denote this set of connections for a layer ℓ by c_i^ℓ. The connections decide the $\mathrm{Merge}_{\text{up}}$ operation which is defined as follows:

$$h_i^{(\ell)} = \mathrm{Merge}_{\text{up}}(z) = \begin{cases} W_i^{(\ell-1)} z_{c_i^\ell}^{(\ell-1)} & (\text{ connection}), \\ W_i^{(\ell-1)} \left(z_{c_{i,L}^\ell}^{(\ell-1)} \odot z_{c_{i,R}^\ell}^{(\ell-1)} \right) & (\text{ connection}), . \end{cases} \tag{2}$$

In the above notation, $c_{i,L(R)}^\ell$ denotes the left (right) child of the parent node whose children are denoted by c_i^ℓ. Here, $z_j^{(\ell)} \in \mathbb{R}^d$, $W_j^{(\ell-1)} \in \mathbb{R}^{d\times d}$, and $\odot$ is elementwise multiplication. The order-3 merge builds on the order-3 tensor approximation introduced by (Loconte et al., 2025). The composite upward nodes occurs as follows:

$$\bar{z}_i^{(\ell-1)} = \mathrm{LN}_{\ell,\text{proj}}(z_i^{(\ell-1)}), \quad q_i^{(\ell-1)} = \mathrm{Merge}\left(\bar{z}_i^{(\ell-1)}\right) + \frac{\mu}{2} z_i^{\text{res},\ell-1}, \quad z_i^\ell = \mu q_i^{(\ell-1)} + \mathrm{MLP}_\ell(\mathrm{LN}_{\ell,\text{mlp}}(q_i^{(\ell-1)})). \tag{3}$$

where $z_i^{\text{res},\ell-1} = \bar{z}_i$ if i has a single connection, and $z_i^{\text{res},\ell-1} = \bar{z}_{c_{i,L}^\ell} + \bar{z}_{c_{i,R}^\ell}$ if it has a double connection. The LayerNorm (LN) and MLP are shared across nodes in a layer, while the tree-gram weight depends on the parent position. Unless stated otherwise, $\mu = 1$, the MLP uses GELU and hidden width $4d$, and all reported tree-gram runs use learned positional embeddings to break the commutativity of the merge. The shared token embedding defines the initial layer hidden nodes

$$z_i^{(0)} = \mathrm{Emb}(x_i) + p_i \in \mathbb{R}^d. \tag{4}$$

The downward pass occurs along the terminal nodes at each layer, with the same residual/MLP pattern as Eq. 3. The only difference is that $\mathrm{Merge}_{\text{down}}$ operation is modified as

$$h_i^{(\ell)} = \mathrm{Merge}_{\text{down}}(z) = \begin{cases} (W_i^{(\ell+1)})^T z_{c_i^\ell}^{(\ell+1)} & (\text{ connection}), \\ \left(z_{c_{i,L}^\ell}^{(\ell+1)} \odot (W_i^{(\ell+1)})^T z_{c_{i,R}^\ell}^{(\ell+1)} \right) & (\text{ connection}), . \end{cases} \tag{5}$$

The terminal node in the bottom layer serves as the predictive node, where an embedding-weight-tied softmax head maps it to next-token probabilities. Tree-grams perform autoregressive prediction by unfolding each sequence into its $n-1$ left-padded prefixes, which activate different subtrees (Fig. 1). Appendix C provides further details on the architecture, while the supplementary material provides a minimal training script.

Unless otherwise stated, we train on a 529M-token subset of TinyStories, a synthetic language corpus (Eldan & Li, 2023), using a custom BPE tokenizer with vocabulary size $V = 8192$ (Cagnetta et al., 2026). Details on optimization are in Appendix F.

3 RESULTS

3.1 COMPARISON TO EXISTING BASELINES

We show in Figure 1 that n and d yield increasingly complicated models of language, as evidenced by the monotonic decrease in the cross-entropy between the trained tree-grams' data and the real language corpora. In addiiton, we compare our models to several independent baselines for estimating n-gram distributions. The comparisons of these baselines with tree-grams for varying context size n is presented in Figure 1. Note that baseline names are mentioned in bold and italicized text to compare across the figure and the main text.

We compare tree-grams to a wide range of different classes of n-gram estimators. These include count-based n-gram estimators ***(raw count)*** (Chen & Goodman, 1996; Parikh et al., 2014b), unbounded suffix array calculations ***(infinigram)*** (Liu et al., 2025), linear models like spectral automaton ***(spectral WFA)*** (Balle et al., 2014), tensor train approximations ***(tree TN, u-MPS, TTLM)*** (Miller et al., 2021; Su et al., 2024), MLP-based neural models ***(MLP, neural LM, neural LM + unigram)*** (Castro et al., 1999; Bengio et al., 2003b; Blat et al., 2005) and deep neural sequence models ***(GPT-2 (APE), GPT-2 (RoPE), Llama, Mamba)*** (Cagnetta et al., 2026). Except for deep neural sequence models, tree-grams match the predicted lower bound n-gram entropy at lower n and outperform them at longer n (Figure 1). Well-trained deep sequence models provide a stronger estimate of n-gram statistics at large n (Nguyen, 2024; Krishnan et al., 2023; Cagnetta et al., 2026). Panel (c) in Figure 1 shows a monotonic convergence with increasing tree-gram capacity. A detailed description and comparison of these baselines may be found in Appendix D.

3.2 COMPARISON WITH TRANSFORMERS

In order to understand more carefully the performance gap between deep sequential neural models, we compare tree-grams with transformers of different depth with the same *parameter count*, the same *context length* and trained with the same *number of optimization steps*, thereby highlighting the difference in their architectural biases. For each tree-gram target, we take a μP decoder-only transformer of depth $L \in \{1, 2, 3, 4, 5, 6, 10, 12\}$. Two regimes appear at short context and long context. Tree-grams perform better than the shallow matched transformers and match the raw-count references at small n. This pattern holds until approximately $n = 8$. At longer context

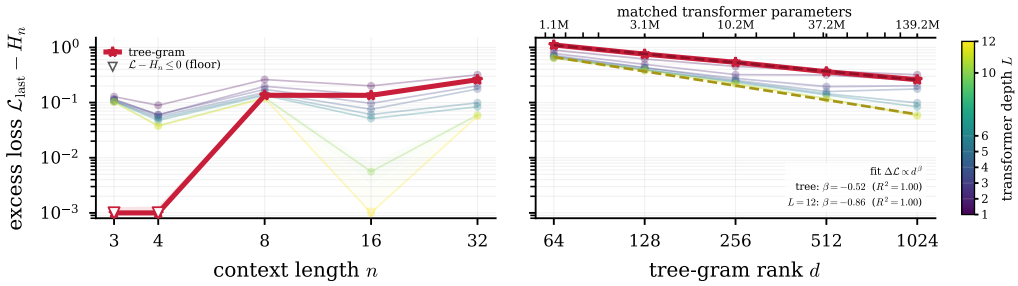

Figure 2: Last-token loss of the fixed tree-gram (red) against μP transformers matched to it in both parameter count, context and optimization steps, with varying depth L. **(left)** Scaling with tree-gram order n at $d = 1024$: tree-grams perform better at small n before a gap opens up for longer context lengths. **(right)** Against tree-gram rank d at $n = 32$. Transformers plateau earlier at smaller depth. Tree-gram loss continues to decrease with d over the tested range. Hwoever, it's slope is less steep as compared to that of the deepest transformer. H_n is the (empirical) n-gram entropy.

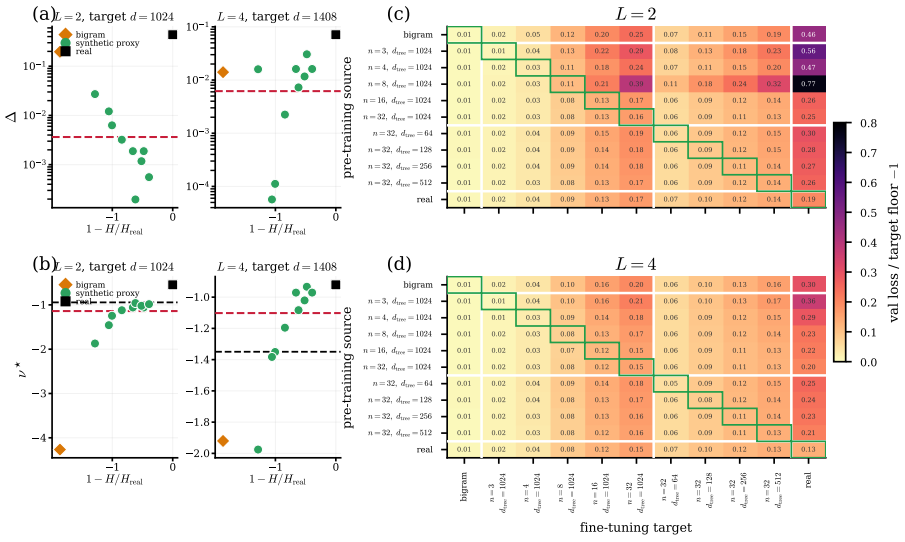

Figure 3: **(a), (b)** Hyper-parameter (HP) transfer from narrow proxies trained on controlled synthetic approximations of the target corpus. The excess loss Δ is the difference between the loss of a wide model (d=1024 at L=2 and d=1408 at L=4 in at the proxy's optimal learning rate $\eta^\star$ and the wide model's own HP-tuned minimum. Tuning on different distributions gives different optimal learning rate $\eta^\star$. Almost all tuning with simpler proxy data gives a lower excess loss than if tuned on real data. In addition, we see a *monotonic* drift of the optimal learning rate with data complexity. H is the entropy of the synthetic data. **(c), (d)** Finetuning on controlled distributions after pre-training. We plot the relative difference between the final validation loss achieved after finetuning and the Bayes floor of the target distribution. There is a clear monotonic worsening of performance with increasing source distribution complexity. We observe two regimes. In the *capacity-limited regime*, finetuning performance is the same regardless of the rank d that generates the underlying data distribution. In the *horizon-limited regime*, however, finetuning performance shows a non-monotonic behavior, the performance being the worst at an intermediate n. The effect is more pronounced for $L = 2$ than $L = 4$.

lengths, the tree-gram is unable to match deep transformers. The right panel shows that the excess loss monotonically decreases with d. While shallower transformers plateau quickly, tree-grams monotonically decrease. However, the deepest transformers have a steeper slope than that of tree-grams. This emphasizes the importance of depth in transformers. It is worth highlighting that the above result could simply be due to inefficient optimization for tree-grams. However, this already provides evidence for the conjecture that the inductive biases of tree-grams are compatible with the structure of language. The per-position curves behind these endpoints are in Fig. 7 (Appendix I), where the tree-gram tracks the transformer family for the first few positions and peels away at long context, the same signature as in Fig. 1.

4 Tree-grams as Controlled Approximation of Language

We use the tree-grams as a source of synthetic data, with n and d controlling the degree of approximation of the synthetic corpus to the real data on which the tree-grams are trained. We note that there are two distinct regimes in the distribution of the generated data from tree-grams— capacity-limited regime (controlled by the rank d) and horizon-limited regime (controlled by the context window n). In Appendix K we present a qualitative analysis of the decoded text generated by tree-grams from the same prompt. Increasing n (at sufficiently large d) increases coherence from word-level ($n = 3$) to sentence-level ($n = 32$). On the other hand, increasing d increases both coherence *and* quality of the generated text.

Beyond qualitative analyses, tree-grams provide finite-window and finite-capacity approximations of a text corpus. We demonstrate the scientific utility of such approximations by experimenting on

hyper-parameter transfer under μP and effect of pre-training and fine-tuning distribution shifts on fine-tuning performance.

4.1 HYPERPARAMETER TRANSFER

The common recipe for hyper-parameter transfer is to tune hyperparameters on smaller models on the *same* data, with a principled transfer to large models (Yang et al., 2022; Kalra & Barkeshli, 2026). More formally, the log optimal learning rate $\nu^\star \equiv \log_2 \eta^\star$ when trained on a model with width w and on a distribution $\mathcal{D}$, is assumed to have the dependence (Ghosh et al., 2025; Hayou, 2026)

$$\nu^\star(w, \mathcal{D}) = \nu^\star(\mathcal{D}) + g_{\mathcal{D}}(w) \tag{6}$$

where $g_{\mathcal{D}}(w)$ is a distribution-dependent term such that $\lim_{w\to\infty} g_{\mathcal{D}}(w) = 0$. The full dependence of the shape factor $g_{\mathcal{D}}(w)$ is unknown for arbitrary distributions. Moreover, whether a width w is large enough is unknown *a priori*. We show that controlled approximations of the distribution $\mathcal{D}$ can yield learning rates which transfer better to the real data. To this effect, we tune peak learning rates on 2-layer and 4-layer μP decoder-only proxy transformers with widths $d \in \{64, 96, 128, 160, 192, 256\}$ and then analyze the transfer to targets with various widths. We observe that HP transfer is more reliable if HP tuning for the proxy model is done with data transferred from a controlled approximation of the real dataset. This is shown in Figure 3, where we plot the target excess loss obtained as a function of the data complexity. The excess loss is always greater when $\nu^\star$ is chosen by tuning the data on the real data, as compared to any of it's synthetic approximation. This is because $\nu^\star$ monotonically increases with the complexity of the controlled dataset (with a weak width-dependence not shown here). The $\nu^\star$ from tuning on the real dataset however shows a non-monotonic effect with width, as shown by the black dashed line which sits lower than the optimal learning rate obtained from tuning on the real data. While a formal characterization of which proxy distribution to tune on, remains an open question - choosing the median learning rate over different proxy distributions shows reliable and robust transfer.

A complete mechanistic understanding of this observation is beyond the scope of the present work. However, we present a simple toy model of data from tree-grams, which we show can give rise to a mechanism for the drift in the optimal learning rate $\nu^\star$. The model is a Gaussian surrogate of hierarchical features,in the spirit of the Random Hierarchy Model (Cagnetta et al., 2024) and of multiscale Gaussian signal models (Wornell & Oppenheim, 1992a). It consists of k levels, with a constant branching factor of 2 and constant branching dilution of α. This leads to geometrically decaying power factor $\pi_k = 2^{-\alpha k}$. A finite-context cutoff is introduced by imposing a strict upper bound on the lookback distance $K(n) \equiv \log_2[n-1]$ and the appropriate normalization. Each level is carried by d channels $a = 1, \ldots, d$, with clean features $S_{k,a} \sim \mathcal{N}(0, \pi_k/d)$ and fixed centered logit directions $u_{k,a} \in \mathbb{R}^V$ that are orthonormal in the Fisher metric at the uniform distribution. Distinct directions keep the d features of a level distinguishable, so d sets the resolution of each level. The label depends on the clean features only:

$$g(S) = \sum_{k=0}^{K(n)} \sum_{a=1}^{d} S_{k,a}\, u_{k,a} \in \mathbb{R}^V, \qquad Y \mid S \sim \text{Categorical}\big(\text{softmax}(g(S))\big). \tag{7}$$

Finite capacity enters through the observed features

$$\Phi_{k,a} = \sqrt{q}\, S_{k,a} + \sqrt{1-q}\, \Xi_{k,a}$$

where the noise term $\Xi_{k,a} \sim \mathcal{N}(0, \pi_k/d)$ is independent of (S, Y). The quality factor $q(d) = d/(d+d_0)$ sets the signal-to-noise ratio d/d_0. The posterior mean $\widehat{S}_{k,a} = \sqrt{q}\, \Phi_{k,a}$ preserves power $\pi_k/(d+d_0)$ which montonically decreases to 0 as $d \to \infty$.

Solving a minimal quadratic model like that of (LeCun et al., 1991) with aligned features, the single-step optimal learning rate has the following dependence:

$$\nu^\star_{w,\Lambda} = \underbrace{\nu^\star_{\infty,\infty}}_{\text{asymptote}} + \underbrace{\Delta(\Lambda)}_{\text{data complexity}} + \underbrace{g_\Lambda(w)}_{\text{finite-width effect}} \tag{8}$$

Here $\Lambda \equiv \Lambda(n, d)$ is a quantity that characterizes the complexity of the synthetic data. $\Delta(\Lambda)$ is a term that depends solely on the approximation parameters n and d, while $g_\Lambda(w)$ is a shape factor

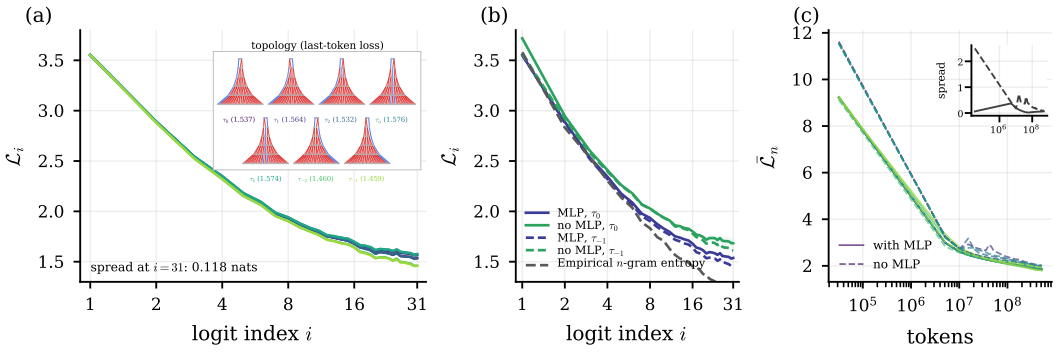

Figure 4: **(a)** Role of topologies. At convergence, the topologies show slight difference at long n. (Inset) The seven different topologies we experimented on. The anti-aligned topologies seem to significantly out-perform other topologies for last-token prediction (losses in nats mentioned). **(b)** Role of MLPs. MLPs have a significant effect in reducing the loss even at matched parameter count. **(c)** MLPs help control the drift in the losses for different topologies, giving more stable training.

which governs the effect of finite width. More importantly, $\Delta(\Lambda)$ increases monotonically with d and n. The full derivation is presented in Appendix E. We emphasize that the toy model presented is not intended to provide a causal mechanism for our empirical observations.

4.2 Pre-training and finetuning

The value of a pre-training distribution depends on how well it matches the fine-tuning distribution. (Isik et al., 2025) find that downstream quality improves monotonically with more pre-training data only when the two distributions are well aligned. (Sun & Dredze, 2025) show that the gains from further pre-training can stay latent until fine-tuning, and (Springer et al., 2025) that pre-training for longer can make a model worse after fine-tuning. Synthetic corpora have been used to ask which structure transfers to natural language: music and code (Papadimitriou & Jurafsky, 2020), artificial languages with controlled dependency structure (Ri & Tsuruoka, 2022; Chiang & Lee, 2022), and formal languages with hierarchical dependencies (Hu et al., 2025) can all transfer, depending on the structure they carry. These corpora differ from the target in kind. A tree-gram corpus is instead a fitted approximation of the target itself, with an exact horizon, a capacity knob and an exactly known entropy rate. Tree-grams therefore provide a structured manner to introduce, and to measure, misalignment between the pre-training and finetuning data, through either the context horizon n or the model capacity d.

By fine-tuning we mean continued next-token training of all parameters on the target corpus, as in studies of transfer between text distributions Hernandez et al. (2021); Hu et al. (2025). Unlike a narrower downstream task, a target here can be more complex than its source, so we avoid low-rank adapters Biderman et al. (2024). We re-warm and cosine re-decay the learning rate from a fresh optimizer state Ibrahim et al. (2024); Gupta et al. (2023), but without the replay that Ibrahim et al. (2024) recommend. This is because replaying source data would mix the source into every target run and remove the same-distribution control on the diagonal. For a generator q, we estimate its Bayes floor on the sampled path as

$$H_q = \frac{1}{N}\sum_{t=1}^{N}\left[-\sum_{v\in\mathcal{V}} q(v \mid x_{t-n+1:t-1}) \log q(v \mid x_{t-n+1:t-1})\right]. \tag{9}$$

The conditional entropy at each sampled context is computed from the generator's full softmax during generation — it is not inferred from the downstream model's validation loss. Appendix H specifies the model ladder, data split and optimization schedule.

Figure 3 reports the result of this experiment. Pre-training on distributions from the capacity-limited regime shows a nearly uniform performance gap when fine-tuned on different distributions. On the other hand, pre-training on distributions from the horizon-limited regime shows a non-monotonic performance gap, unless the finetuning distribution is too simple. Note that a higher performance

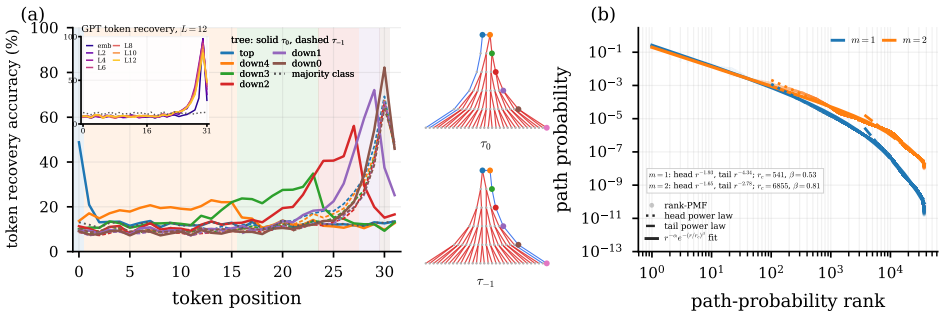

Figure 5: **(a)** Effective receptive fields. We train linear probes to predict input tokens from latent activations from the terminal nodes (marked with corresponding colors in the central panel). Accuracy for the aligned topology τ_0 in tree-gram peaks at different intermediate context windows for different tree-gram layer, while the accuracy peaks at the last context for all layers for the anti-aligned topology τ_{-1} and transformers. This suggests that different layers in an aligned topology preserves information about different context windows, while the anti-aligned topology and transformers mix features for last-token prediction. **(b)** Emergent sparsity in tree-grams. Tree-grams that are allowed to choose sequences through routing, demonstrate a preference for few topologies. The paths follow a power-law with a generalized exponential cutoff, with the tail showing a steeper decay.

gap occurs as the complexity of the target distribution increases. Interestingly, the entropy of the distribution does *not* provide a demarcation, as we show in Appendix I. This suggests that alignment depends on certain directional components.

5 Ablations

Role of topologies We focus on three kinds of topologies: aligned, balanced, and anti-aligned. The aligned topologies contain all single connections on the left-hand side (topologies labelled τ_0, τ_1, τ_2 in Figure 4), while the anti-aligned topologies contain all single connections on the right-hand side (labelled τ_{-1}, τ_{-2}). Balanced topologies have the single connections in the middle (labelled τ_a, τ_b). With proper hyper-parameter tuning, all seven converge to nearly the same loss at the end of training. The difference in their per-token losses occurs only in the last few positions. We notice that the balanced and aligned topologies typically have a higher loss, while the anti-aligned topologies converge to a lower last-token loss. However, we perform experiments with τ_0 since it is an interpretable model 6. There is some difference in the training *dynamics*, which is reported in Fig. 4. The anti-aligned topologies start at a higher loss but converge to a lower loss and vice versa for the algined topologies.

Role of MLPs MLPs have a significant effect on loss. We compare a width-1024 tree-gram with MLPs to a roughly parameter-matched width-1848 tree-gram without MLPs. For either topology, removing the MLP worsens performance. This effect is more pronounced at higher values of n. However, the comparison changes both the MLP/nonlinearity and the contraction rank to maintain parameter parity, so it does not isolate which factor causes the gap. We also observe that MLPs control the variation between different tree-gram topologies during training. We leave an in-depth investigation of the role of MLPs to future work.

6 Interpretability

Effective receptive fields The aligned topology labelled τ_0 (Figure 4) has the unique property that terminal nodes in different layers "focus" on different parts of the context. On the other hand, the anti-aligned topology τ_{-1} mixes information from the whole context. We use a linear probe to uncover input tokens from latent activations of the terminal nodes in tree-grams. We observe that for τ_0, different terminal nodes in different layers recover each input token in its scope. The accuracy rises as we move down the layers, which indicates the difficulty of reconstruction increases as we pass through multiple non-linear passes. In contrast, for τ_{-1} and transformers, recovery peaks

at the last token, given the full context. This highlights that the aligned topology has *effective receptive fields* for each node in the downward pass, in contrast with anti-aligned topologies and transformers. Activation patching also confirms this causally (Fig. 13). Restoring a single node recovers the prediction exactly when the swapped token lies in its scope and not at all otherwise. The corresponding plots are Appendix I. We emphasize that we do not claim that transformers are implementing a τ_{-1} topology. Recovery of trees from transformers need careful investigation due to known biasing effects from the parsing algorithm, feature definition as well as inherent architectural biases (Li et al., 2020). We also emphasize that these experiments do not necessarily demonstrate that receptive-field boundaries emerge through learning.

Emergent sparsity through topology routing To test whether the network develops a preference among topologies, we implement a dynamic tree-gram with one linear router at each merge layer. The routing decision is sequence-level rather than token-level, unlike token-choice Mixture-of-Experts routing. At layer ℓ, the router applies the same linear map to every active node state and then averages across active positions, in the style of (Zaheer et al., 2017):

$$r_g^{(\ell)}(x_{1:n-1}) = \frac{1}{|A_\ell|} \sum_{t \in A_\ell} w_{\ell,g}^\top \, \mathrm{LN}(z_t^{(\ell)}), \qquad p^{(\ell)} = \mathrm{softmax}(\beta r^{(\ell)} + a^{(\ell)}). \tag{10}$$

Here A_ℓ is the active-node set, g indexes the valid grouping patterns at that layer, and $a^{(\ell)}$ is a non-gradient bias trick adapted from (DeepSeek-AI, 2025). During training, $a^{(\ell)}$ is updated toward a slightly perturbed uniform selection target to reduce route collapse, which we switch off after 5% of training time for better performance. We select the hard top-m indices under $p^{(\ell)}$ and mix their outputs using the corresponding differentiable softmax weights. Gradients flow through the selected weights; the index selection itself uses neither a straight-through estimator nor a Gumbel relaxation. We report the path probabilities $p_{\text{path}} = \mathbb{E}_x \left[\prod_{g_\ell \in \text{path}} p_{g_\ell}^{(\ell)} \right]$ where g_ℓ denotes the topology chosen at a layer ℓ.

We observe that the network has a sharp cutoff in the probabilities, indicating that it has a strong preference for a few topologies. For the specific case of $n = 32$, the top-100 (out of 36,720 topologies) carry a substantial $O(1)$ fraction of the probability mass. This can also be seen through a comparison of the slope of the head and the tail, where the exponent for the tail has a higher value than that of the head. Raising m flattens both exponents together, and spreads the mass. These results are shown in Fig. 5. Although the concentration persists despite the balancing bias, current experiments are insufficient to rule out initialization or router-optimization effects as compared to learnt-language preference. The exact preferred topologies chosen is an effect of the random seed. We emphasize that we treat topology as an architectural prior. We do *not* infer linguistic meaning from the studied topologies, as cautioned by (Williams et al., 2018).

7 Conclusion, Limitations, Future Work

In this work, we present tree-grams, trainable and controllable n-gram approximators of language. They match optimal n-gram estimates at small n. A noticeable gap from deep models remains at longer contexts. We find that controlled data approximations can aid hyper-parameter transfer under μP. We also use them to study performance gaps after finetuning by distinguishing pre-training and fine-tuning distributions.

Computational constraints prevent scaling to longer context windows, particularly to $n \approx 256$, which is the typical length of paragraphs in TinyStories. The $O(Bn^2d)$ activation cost of prefix unfolding is a primary bottleneck, which could be reduced by sharing prefix states. TinyStories is a synthetic corpus generated by language models and has constrained vocabulary and syntax. Testing a less constrained language corpus is necessary before generalizing these empirical findings. In addition, the scope and scale of the models and data is limited. Understanding of different individual components of the tree-gram architecture also remains limited.

Developing faster, stabler and robust optimization routines for tree-grams remains an open engineering problem. It would also be interesting to train tree-grams on distributions very different from language.We leave these and other directions mentioned in the text for future work.

DECLARATION OF LLM USAGE

In this work, we used generative AI tools to design or provide feedback on research methodology or experiments, propose or refine hypotheses, help develop theoretical models or conceptual frameworks, fomrulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, and to implement several baseline methods. We have not used generative AI tools to generate synthetic data sets. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, draft parts of a research paper, summarize or analyse existing literature, discover research topics or identify gaps, suggest a structure for a research paper, and to propose a keywords for a research paper. We have reviewed all AI-assisted work. In particular, we checked LLM-generated research ideas for potential plagiarism through a manual literature survey. LM-generated code was verified and tested for correctness by one author. LM-generated proofs were checked by one author. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The authors pledge to adhere to the Code of Ethics. No human subjects were used in this project. Experiments were performed on TinyStories, which is a GPT-generated text dataset.

REPRODUCIBILITY STATEMENT

Code to train tree-grams and reproduce the plots in the main text can be found in the Supplementary Material, and will be released publicly. Any unlawful or unethical use or distribution of the code is strictly prohibited.

REFERENCES

Mahmoud Assran and Mike Rabbat. On the convergence of Nesterov's accelerated gradient method in stochastic settings. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 410–420, 2020.

Borja Balle, Ariadna Quattoni, and Xavier Carreras. Spectral learning techniques for weighted automata, transducers, and grammars. Tutorial at the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014. URL https://borjaballe.github.io/emnlp14-tutorial/. Accessed: 2026-08-29.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003a.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003b. URL https://www.jmlr.org/papers/v3/bengio03a.html.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. LoRA learns less and forgets less. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2405.09673.

Fernando Blat, María José Castro, Salvador Tortajada, and Joan Andreu Sánchez. A hybrid approach to statistical language modeling with multilayer perceptrons and unigrams. In *Proceedings of the 8th International Conference on Text, Speech and Dialogue (TSD 2005)*, pp. 195–202. Springer, 2005. doi: 10.1007/11551874_25.

Taylor L. Booth. Probabilistic representation of formal languages. In *10th Annual Symposium on Switching and Automata Theory*, pp. 74–81. IEEE, 1969. doi: 10.1109/SWAT.1969.7.

T. Brants, Ashok Popat, Peng Xu, Franz Josef Och, and Jeffrey Dean. Large language models in machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2007. URL https://api.semanticscholar.org/CorpusID:633992.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

Francesco Cagnetta, Leonardo Petrini, Umberto M. Tomasini, Alessandro Favero, and Matthieu Wyart. How deep neural networks learn compositional data: The random hierarchy model. *Phys. Rev. X*, 14:031001, Jul 2024. doi: 10.1103/PhysRevX.14.031001. URL https://link.aps.org/doi/10.1103/PhysRevX.14.031001.

Francesco Cagnetta, Alessandro Favero, Antonio Sclocchi, and Matthieu Wyart. Scaling laws and representation learning in simple hierarchical languages: Transformers vs. convolutional architectures, 2025. URL https://arxiv.org/abs/2505.07070.

Francesco Cagnetta, Allan Raventós, Surya Ganguli, and Matthieu Wyart. Deriving neural scaling laws from the statistics of natural language. https://github.com/fracagnetta/small-language-modelling, 2026. arXiv:2602.07488.

M.J. Castro, F. Prat, and F. Casacuberta. MLP emulation of n-gram models as a first step to connectionist language modeling. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pp. 910–915 vol.2, 1999. doi: 10.1049/cp:19991228.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 310–318. Association for Computational Linguistics, 1996. doi: 10.3115/981863.981904. URL https://aclanthology.org/P96-1041/.

Song Cheng, Lei Wang, Tao Xiang, and Pan Zhang. Tree tensor networks for generative modeling. *Physical Review B*, 99(15):155131, 2019. doi: 10.1103/PhysRevB.99.155131.

Cheng-Han Chiang and Hung-yi Lee. On the transferability of pre-trained language models: A study from artificial datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 2022.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. Learning to compose task-specific tree structures. In *AAAI*, 2018.

Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956. doi: 10.1109/TIT.1956.1056813.

Noam Chomsky. *Syntactic Structures*. Mouton, The Hague, 1957.

Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959. doi: 10.1016/S0019-9958(59)90362-6.

Noam Chomsky and Marcel-Paul Schützenberger. The algebraic theory of context-free languages. In *Computer Programming and Formal Systems*, volume 35 of *Studies in Logic and the Foundations of Mathematics*, pp. 118–161. North-Holland, 1963. doi: 10.1016/S0049-237X(08)72023-8.

Stephen Crain and Mineharu Nakayama. Structure dependence in grammar formation. *Language*, 63(3):522–543, 1987. doi: 10.2307/415004.

Marina Danilova and Grigory Malinovsky. Averaged heavy-ball method. *Computer Research and Modeling*, 14(2):277–308, 2022. doi: 10.20537/2076-7633-2022-14-2-277-308.

DeepSeek-AI. DeepSeek-V3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

E. DeGiuli. Random language model. *Phys. Rev. Lett.*, 122:128301, Mar 2019. doi: 10.1103/PhysRevLett.122.128301. URL https://link.aps.org/doi/10.1103/PhysRevLett.122.128301.

François Denis, Mattias Gybels, and Amaury Habrard. Dimension-free concentration bounds on Hankel matrices for spectral learning. *Journal of Machine Learning Research*, 17(31):1–32, 2016. URL https://www.jmlr.org/papers/v17/14-501.html.

Nai Ding, Lucia Melloni, Hang Zhang, Xing Tian, and David Poeppel. Cortical tracking of hierarchical linguistic structures in connected speech. *Nature Neuroscience*, 19(1):158–164, 2016. doi: 10.1038/nn.4186.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proceedings of NAACL-HLT*, pp. 199–209, 2016. doi: 10.18653/v1/N16-1024.

Ronen Eldan and Yuanzhi Li. TinyStories: How small can language models be and still speak coherent English?, 2023. URL https://arxiv.org/abs/2305.07759.

Jeffrey L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–225, 1991. doi: 10.1007/BF00114844.

Martin B. H. Everaert, Marinus A. C. Huybregts, Noam Chomsky, Robert C. Berwick, and Johan J. Bolhuis. Structures, not strings: Linguistics as part of the cognitive sciences. *Trends in Cognitive Sciences*, 19(12):729–743, 2015. doi: 10.1016/j.tics.2015.09.008.

Richard Futrell and Michael Hahn. Linguistic structure from a bottleneck on sequential information processing. *Nature Human Behaviour*, 10:589–600, 2026. doi: 10.1038/s41562-025-02336-w.

Nikhil Ghosh, Denny Wu, and Alberto Bietti. Understanding the mechanisms of fast hyperparameter transfer, 2025. URL https://arxiv.org/abs/2512.22768.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Joshua Goodman. A bit of progress in language modeling, 2001. URL https://arxiv.org/abs/cs/0108005.

Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L. Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. Continual pre-training of large language models: How to (re)warm your model? In *ES-FoMo Workshop, International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2308.04014.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan Brennan. Finding syntax in human encephalography with beam search. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2727–2736, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1254. URL https://aclanthology.org/P18-1254/.

Marc D. Hauser, Noam Chomsky, and W. Tecumseh Fitch. The faculty of language: What is it, who has it, and how did it evolve? *Science*, 298(5598):1569–1579, 2002. doi: 10.1126/science.298.5598.1569.

Soufiane Hayou. A proof of learning rate transfer under μP. In Emtiyaz Khan, Yingzhen Li, Arno Solin, and Aaditya Ramdas (eds.), *Proceedings of The 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, pp. 3322–3330. PMLR, 02–05 May 2026. URL https://proceedings.mlr.press/v300/hayou26a.html.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021. URL https://arxiv.org/abs/2102.01293.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *Proceedings of NAACL-HLT*, pp. 4129–4138, 2019. doi: 10.18653/v1/N19-1419.

John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

Michael Y. Hu, Jackson Petty, Chuan Shi, William Merrill, and Tal Linzen. Between circuits and Chomsky: Pre-pretraining on formal languages imparts linguistic biases. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and scalable strategies to continually pre-train large language models. *Transactions on Machine Learning Research (TMLR)*, 2024. URL https://openreview.net/forum?id=DimPeeCxKO.

Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. Scaling laws for downstream task performance in machine translation. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2402.04177.

Dayal Singh Kalra. nanoGPT-mup. https://github.com/dayal-kalra/nanoGPT-mup, 2025. (Slight modifications made by authors).

Dayal Singh Kalra and Maissam Barkeshli. Quantifying hyperparameter transfer and the importance of embedding layer learning rate, 2026. URL https://arxiv.org/abs/2605.21486.

Yoon Kim, Chris Dyer, and Alexander M. Rush. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of ACL*, pp. 2369–2385, 2019. doi: 10.18653/v1/P19-1228.

Dan Klein and Christopher D. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004. doi: 10.3115/1218955.1219019.

Aravind Krishnan, Jesujoba Alabi, and Dietrich Klakow. On the n-gram approximation of pre-trained language models, 2023. URL https://arxiv.org/abs/2306.06892.

Adhiguna Kuncoro, Chris Dyer, Laura Rimell, Stephen Clark, and Phil Blunsom. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of ACL*, pp. 1426–1436, 2018. doi: 10.18653/v1/P18-1132.

Karim Lari and Steve J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language*, 4(1):35–56, 1990. doi: 10.1016/0885-2308(90)90022-X.

Yann LeCun, Ido Kanter, and Sara A. Solla. Eigenvalues of covariance matrices: Application to neural-network learning. *Phys. Rev. Lett.*, 66:2396–2399, May 1991. doi: 10.1103/PhysRevLett.66.2396. URL https://link.aps.org/doi/10.1103/PhysRevLett.66.2396.

Huayang Li, Lemao Liu, Guoping Huang, and Shuming Shi. On the branching bias of syntax extracted from pre-trained language models, 2020. URL https://arxiv.org/abs/2010.02448.

Henry W. Lin and Max Tegmark. Critical behavior in physics and probabilistic formal languages. *Entropy*, 19(7):299, 2017. doi: 10.3390/e19070299.

Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens, 2025. URL https://arxiv.org/abs/2401.17377.

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)?, 2025. URL https://arxiv.org/abs/2409.07953.

Lovish Madaan, Srinadh Bhojanapalli, Himanshu Jain, and Prateek Jain. Treeformer: Dense gradient trees for efficient attention computation. In *ICLR*, 2023.

Stéphane G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989. doi: 10.1109/34.192463.

Jacob Miller, Guillaume Rabusseau, and John Terilla. Tensor networks for probabilistic sequence modeling. In *AISTATS*, 2021.

Masaaki Nakamura and Kiyoshi Shikano. A study of English word category prediction based on neural networks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 731–734, Glasgow, Scotland, 1989.

Gerardo G. Naumis and Germinal Cocho. The tails of rank-size distributions due to multiplicative processes: from power laws to stretched exponentials and beta-like functions. *New Journal of Physics*, 9(8):286, 2007. doi: 10.1088/1367-2630/9/8/286. URL https://doi.org/10.1088/1367-2630/9/8/286.

Timothy Nguyen. Understanding transformers via n-gram statistics. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 98049–98082. Curran Associates, Inc., 2024. doi: 10.52202/079017-3112. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b1c446eebd9a317dd0e96b16908c821a-Paper-Conference.pdf.

Christophe Pallier, Anne-Dominique Devauchelle, and Stanislas Dehaene. Cortical representation of the constituent structure of sentences. *Proceedings of the National Academy of Sciences*, 108(6): 2522–2527, 2011. doi: 10.1073/pnas.1018711108. URL https://www.pnas.org/doi/abs/10.1073/pnas.1018711108.

Isabel Papadimitriou and Dan Jurafsky. Learning music helps you read: Using transfer to study linguistic structure in language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Ankur P. Parikh, Avneesh Saluja, Chris Dyer, and Eric Xing. Language modeling with power low rank ensembles. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1487–1498, Doha, Qatar, 2014a. Association for Computational Linguistics. doi: 10.3115/v1/D14-1158. URL https://aclanthology.org/D14-1158/.

Ankur P. Parikh, Avneesh Saluja, Chris Dyer, and Eric Xing. Language modeling with power low rank ensembles. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1487–1498. Association for Computational Linguistics, 2014b. doi: 10.3115/v1/D14-1158. URL https://aclanthology.org/D14-1158/.

Vasily Pestun and Yiannis Vlassopoulos. Tensor network language model. *arXiv preprint arXiv:1710.10248*, 2017.

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. doi: 10.1016/0041-5553(64)90137-5.

Ryokan Ri and Yoshimasa Tsuruoka. Pretraining with artificial language: Studying transferable knowledge in language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022.

Claude E. Shannon. Prediction and entropy of printed English. *Bell System Technical Journal*, 30 (1):50–64, 1951. doi: 10.1002/j.1538-7305.1951.tb01366.x.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *ICLR*, 2019.

Jacob Mitchell Springer, Sachin Goyal, Kaiyue Wen, Tanishq Kumar, Xiang Yue, Sadhika Malladi, Graham Neubig, and Aditi Raghunathan. Overtrained language models are harder to fine-tune, 2025. URL https://arxiv.org/abs/2503.19206.

Zhan Su, Yuqin Zhou, Fengran Mo, and Jakob Grue Simonsen. Language modeling using tensor trains. *arXiv preprint arXiv:2405.04590*, 2024.

Kaiser Sun and Mark Dredze. Amuro and Char: Analyzing the relationship between pre-training and fine-tuning of large language models, 2025. URL https://arxiv.org/abs/2408.06663.

Anej Svete, Nadav Borenstein, Mike Zhou, Isabelle Augenstein, and Ryan Cotterell. Can transformers learn n-gram language models? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9851–9867, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.550. URL https://aclanthology.org/2024.emnlp-main.550/.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, 2015.

Stefan Thurner, Rudolf Hanel, Bo Liu, and Bernat Corominas-Murtra. Understanding Zipf's law of word frequencies through sample-space collapse in sentence formation. *Journal of the Royal Society Interface*, 12(108):20150330, 2015. doi: 10.1098/rsif.2015.0330.

Aditya Varre, Gizem Yüce, and Nicolas Flammarion. Learning in-context n-grams with transformers: Sub-n-grams are near-stationary points, 2025. URL https://arxiv.org/abs/2508.12837.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267, 2018.

Gregory W. Wornell and Alan V. Oppenheim. Estimation of fractal signals from noisy measurements using wavelets. *IEEE Transactions on Signal Processing*, 40(3):611–623, 1992a. doi: 10.1109/78.120804.

G.W. Wornell and A.V. Oppenheim. Estimation of fractal signals from noisy measurements using wavelets. *IEEE Transactions on Signal Processing*, 40(3):611–623, 1992b. doi: 10.1109/78.120804.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer, 2022. URL https://arxiv.org/abs/2203.03466.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. PCFGs can do better: Inducing probabilistic context-free grammars with many symbols. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1487–1498, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.117. URL https://aclanthology.org/2021.naacl-main.117/.

Xiulin Yang, Heidi Getz, and Ethan Gotlieb Wilcox. Function words as statistical cues for language learning, 2026. URL https://arxiv.org/abs/2601.21191.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3770–3785, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.292. URL https://aclanthology.org/2021.acl-long.292/.

Oğuz Kaan Yüksel, Rodrigo Alvarez Lucendo, and Nicolas Flammarion. Incremental learning of sparse attention patterns in transformers, 2026. URL https://arxiv.org/abs/2602.19143.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. In *Advances in Neural Information Processing Systems*, 2017.

Lipeng Zhang, Peng Zhang, Xindian Ma, Shuqin Gu, Zhan Su, and Dawei Song. A generalized language model in tensor space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7450–7458, 2019. doi: 10.1609/aaai.v33i01.33017450. URL https://doi.org/10.1609/aaai.v33i01.33017450.

A Model, Code and Data release

The tree-gram architecture and pre-trained models will be released under the **Creative Commons Attribution 4.0 International (CC BY 4.0)** license, allowing use, modification and redistribution with proper attribution.

The models will be released through a public Github or HuggingFace repository. As presented in the Supplementary Material, scripts describing the architecture and use will be uploaded, with clear instructions for reproducibility.

The Github repo will be additionally tracked to allow users to report and fix bugs. We will further invite the community to contribute to existing models through bug reports, suggestions for improvement and new dataset and feature suggestions. Each update will be accompanied by well-documented release notes, detailing changes and new requirements.

B Detailed Background and Related Works

We provide a detailed description of previous and related works, to motivate tree-grams presented in this work.

Synthetic models of data Modelling language has been an area of study for a long time. The classic work of (Chomsky, 1956) used finite-state Markov models, phrase-structure grammars and transformational grammars. The theory of formal languages classifies grammars by their generative capacity (Chomsky, 1959). In particular, context-free grammars use recursive production rules to generate hierarchical and nested structures, such as Dyck languages, that finite-state Markov models cannot represent (Chomsky & Schützenberger, 1963). Later work extended context-free grammars into stochastic models by assigning probabilities to production rules (Booth, 1969; Lari & Young, 1990), while synthetic grammatical data became a controlled setting for studying whether neural networks can learn hierarchical structure (Elman, 1991). More recently, there has been a growing interest in understanding models of data with randomly chosen underlying rules, eg. the Random Language Model (DeGiuli, 2019) and the Random Hierarchy Model (Cagnetta et al., 2024). The justification for such hierarchical structures comes from theoretical studies (Chomsky, 1957), statistical analysis (Thurner et al., 2015; Goodfellow et al., 2016), information theory (Futrell & Hahn, 2026; Yang et al., 2026) and cognitive representations (Crain & Nakayama, 1987; Pallier et al., 2011; Ding et al., 2016; Hale et al., 2018). Tree-grams are a natural extension of such hierarchical and exponentially expressive models of language-like data. By making these models trainable, we allow tree-grams to capture necessary features from the data, without explicit interpretation of hidden nodes.

n-gram estimators Language modeling is essentially an n-gram estimation task. Recall that the n-gram estimation amounts to estimating the conditional probability tensor $p(x_n \mid x_{1:n-1})$. The frequentist method quickly becomes infeasible due to an exponential dependence on sample complexity and memory requirements. However, numerous empirical studies indicate that this tensor is extremely sparse, which gives rise to different algorithms. Smoothing re-distributes the probability mass through a linear interpolation (Chen & Goodman, 1996); PLRE takes advantages of conditional factorization into lower-rank tensors (Parikh et al., 2014a); Some algorithms use back-off (Goodman, 2001; Nakamura & Shikano, 1989) or scales by brute force (Brants et al., 2007) up to the unbounded suffix-array construction of infini-gram (Liu et al., 2025)

A different method is to approximate this with factorized tensor sequence models, MPS and tensor-train models. (Pestun & Vlassopoulos, 2017; Miller et al., 2021; Su et al., 2024) We use (Loconte et al., 2025) as a primary motivation for our rank-3 tensor factorization.

Neural networks can also be used for a parametric estimation. This has been tried by (Castro et al., 1999) with MLPs and one-hot encodings, (Bengio et al., 2003b) with MLPs and learnable encodings, (Blat et al., 2005) with interpolation with unigrams.

Modern recurrent networks or sequence-to-sequence models take advantage of implicit biases or statistical or nested properties of language to directly model the n-gram probabilities. (Nguyen, 2024; Krishnan et al., 2023; Cagnetta et al., 2026; 2025)

Controlled approximation of languages n-gram estimators provide a truncated-horizon approximation of language data. In addition, several model families introduce monotonic measures of capacity. Weighted automata uses finite Hankel rank (Denis et al., 2016). MPS uses bond dimension (Pestun & Vlassopoulos, 2017; Miller et al., 2021; Su et al., 2024) (Yang et al., 2021) uses latent state size, while (Parikh et al., 2014a) use matrix and tensor ranks to represent intermediate levels of dependence between successive n-gram orders.

Studying transformers on such controlled languages / n-grams There have been attempts to study transformers on synthetic formal grammars. In addition, n-gram models of data has also been formulated as a learning problem by (Svete et al., 2024) and (Varre et al., 2025) in an in-context setting. (Yüksel et al., 2026) show that transformer incremental learn higher order Markov structure. However, to the best of our knowledge, such controlled approximations have never been applied for understanding practical situations like HP transfer, due to the limitation posed by scalability and fidelity to a realistic text corpora.

Trainable tree-structured models Differentiable tree structures exist. In particular, Tree-LSTMs consume a given parse (Tai et al., 2015) while (Choi et al., 2018) learns the composition order (tree) with a Gümbel relaxation. (Shen et al., 2019) forces a soft hierarchy into a recurrent net's gates. A key tension arises with (Williams et al., 2018), who caution that the structures such models learn need not be the linguistically meaningful ones. Transformer architectures also take advantage of trees, eg. (Madaan et al., 2023) uses a decision-tree index over keys.

C Details of the architecture

C.1 Analytic expressions for various properties of the network

Write n_k for the number of nodes at level k, so that $n_0 = n$ and $n_{k+1} = \lfloor n_k/2 \rfloor + 1$. Unrolling the recursion gives the closed form

$$n_k = 2 + \left\lfloor \frac{n-2}{2^k} \right\rfloor . \tag{11}$$

The expressions for the depth d_n, number of nodes s_n and the number of different topologies t_n are as follows

$$d_n = \lceil \log_2(n-1) \rceil \sim \log_2 n \tag{12}$$

$$s_n = \sum_{k=0}^{d_n} n_k = n + 2d_n + \sum_{k=1}^{d_n} \left\lfloor \frac{n-2}{2^k} \right\rfloor \sim 2n + O(\log n) \tag{13}$$

$$t_n = \prod_{k=0}^{d_n - 1} \binom{n_{k+1}}{n_k - n_{k+1}} \sim n^{\log \sqrt{n} + O(1)} \tag{14}$$

Only the d_n levels above the leaves carry weight matrices, so the total number of parameters is

$$P_{\text{tree-gram}}(n, r) \approx Vd + d^2 \left(1 + \sum_{k=1}^{d_n} n_k + 2d_f(2d_n + 1) \right) \tag{15}$$

If MLPs are not included then $d_f = 0$.

D Description of n-gram baselines used for comparison

Raw count A ***raw count*** n-gram model estimates $p(x_n \mid x_{1:n-1})$ by normalizing counts for each observed n-tuple of tokens. Smoothed raw count models redistribute probability mass to unseen

events and shorter histories through interpolation and absolute discounting, while low-rank ensembles control the correlation of n-grams through low-rank matrices (Chen & Goodman, 1996; Parikh et al., 2014b). We report classical raw-count, smoothed-count, and low-rank count baselines only for $n = 2$ to $n = 8$. This is because at $n = 8$, we already observe that 60% of the n-grams appear only once. Tree-grams match these tested count baselines at small n and out-perform them for larger n.

Infini-gram Infini-gram uses a suffix array to retrieve continuation counts for the longest matching suffix of the available history and backs off when longer matches are unavailable (Liu et al., 2025). In our finite-context comparison, we cap this search at $n - 1$ tokens and interpolate with a smoothed unigram (***infinigram + unigram***) or a neural LM (***infinigram + neural LM***). On TinyStories, infinigram + unigram improves through $n = 4$, but its loss worsens and then plateaus at larger n as long exact suffix matches become rare. Infinigram + neural LM improves but plateaus by $n \approx 8$. In contrast, tree-grams improve throughout the tested range and outperform infinigram at every shared value of n.

Spectral automaton models A spectral weighted-finite-automaton (***spectral WFA***) estimator constructs empirical Hankel blocks of substring probabilities and uses a truncated singular-value decomposition to recover a finite-dimensional state and token-transition operators (Balle et al., 2014). After tuning the rank and Hankel window $W \in \{3, \ldots, 6\}$, the resulting curve remains nearly flat as n increases and performs worse even compared to the raw trigram. Tree-grams outperform the spectral estimator at every shared order, with the gap increasing toward larger n.

Tensor train approximations We compare to three tensor-network language models. The uniform matrix product state (***u-MPS***) of (Miller et al., 2021) is a Born machine, $p(x_n \mid x_{1:n-1}) \propto |\alpha^\top A^{x_1} \cdots A^{x_n} \omega|^2$, with a single core $A \in \mathbb{R}^{|V| \times D \times D}$ shared across positions. The tree tensor network (***tree TN***) of (Cheng et al., 2019) is a Born machine on a balanced binary tree over the context, with a learned $|V| \to v$ feature map at the leaves and one order-3 tensor at each internal node; we train it end to end by gradient descent rather than by DMRG sweeps. TTLM-Large (***TTLM***) (Su et al., 2024) is a tensor-train recurrence in which a token-dependent factor updates the state multiplicatively, followed by a softmax readout tied to the input embedding tensor. At every n, the bond dimension D, the tree width v and the rank R are set to match the parameter count of the $d = 1024$ tree-gram ($D = 47$–131, $v = 164$–316, $R = 46$–128). The tree TN is best at $n = 3$ (3.36 nats) and falls to near the unigram level from $n = 8$ on. Tree-grams outperform all three at every shared order, by 0.19 nats at $n = 3$ and 0.95 nats at $n = 32$ against the strongest of them, TTLM.

Neural network approximation of n-grams A neural n-gram model embeds a fixed window of $n - 1$ preceding tokens and uses an MLP with a softmax output to estimate the next-token distribution. This may be done with one-hot encodings (Castro et al., 1999) (***MLP***) or through learnable embedding structure (Bengio et al., 2003b) (***neural LM***). Later works combined MLP with a unigram model (Blat et al., 2005) which we adapt here as the ***MLP + unigram*** model with the learnable embedding MLP. The parameter-matched neural baseline improves through $n = 16$ but worsens at $n = 32$. However, it has consistently higher cross-entropy than tree-grams. Thus, tree-grams outperform all tested finite-window neural n-gram estimators at every shared order.

Deep neural sequence models Well-trained deep sequence models provide a stronger estimate of n-gram statistics at large n (Nguyen, 2024; Krishnan et al., 2023; Cagnetta et al., 2026). In this case, we compare tree-grams to GPT-2 with absolute positional embeddings (***GPT-2 (APE)***) and rotary positional embeddings (***GPT-2 (RoPE)***), Llama-3.2 (***Llama***), and Mamba (***Mamba***). These are converged multi-epoch runs, which use approximately six times the token budget. In Fig. 1b, tree-grams track these estimators through approximately $n = 8$, while the deep references remain lower at larger n. However, in Fig. 1(c), cross-entropy decreases monotonically as the rank d increases over the tested range.

E A Toy Model for Tree-gram data

We provide details of the toy model that we defined in the main text, which suggests a plausible mechanism for the HP transfer experiment and PT-FT experiment results we observe empirically.

This model is based primarily on the random hierarchy model (Cagnetta et al., 2024) and the wavelet analysis of (Wornell & Oppenheim, 1992b; Mallat, 1989). The toy model for the data has two layers. A finite dyadic hierarchy allocates predictive power across look-back scales. A Gaussian channel limits how well each latent feature can be recovered. Each of these layers are used to implement a different mode of control that tree-grams offer. We discuss this in more detail below. We make an attempt to highlight every assumption that a conclusion depends on through the labels (A1) – (A10). Figure 6 gives an intuitive picture.

E.1 Scales and signal strength

Let $X = (x_{t-n+1}, \dots, x_{t-1})$ be a context of $n-1$ tokens, and let $g(X) = \log p(\cdot \mid X) - V^{-1} \sum_y \log p(y \mid X)\, \mathbf{1}$ be its centered Bayes logit vector. We assume $p(y \mid X) > 0$ and $\mathbb{E}\|g\|^2 < \infty$. The dyadic suffix window W_k holds the 2^k most recent tokens, so $W_0 \subset W_1 \subset \cdots$. Passing from W_{k-1} to W_k reveals the tokens at distances $2^{k-1}+1, \dots, 2^k$; for $k = 0$, it reveals x_{t-1}. This is the coarse-to-fine organization of multiresolution analysis (Mallat, 1989). Let $\mathcal{W}_k$ be the σ-algebra generated by W_k, and let $\mathcal{W}_{-1}$ be the trivial one. The level-k increment and its signal strength are

$$\Delta_k = \mathbb{E}[g \mid \mathcal{W}_k] - \mathbb{E}[g \mid \mathcal{W}_{k-1}], \qquad s_k = \mathbb{E}\big[\Delta_k^\top F_0\, \Delta_k\big], \tag{16}$$

where $F_0 = \mathrm{diag}(p_0) - p_0 p_0^\top$ is the categorical Fisher matrix at the uniform distribution p_0. It is positive definite on centered vectors, and we write $\|v\|_{F_0}^2 = v^\top F_0 v$. The increments are martingale differences. For a constant metric they are therefore orthogonal, $\mathbb{E}[\Delta_j^\top F_0 \Delta_k] = 0$ for $j \neq k$, and their powers add: $\sum_{k \le K} s_k = \mathbb{E}\|\mathbb{E}[g \mid \mathcal{W}_K] - \mathbb{E}[g]\|_{F_0}^2$. The strength s_k is signal power, not a Hessian eigenvalue.

(A1) Modelling the effect of horizon and capacity equivalents The horizon n and the channel count d are hyperparameters of the data distribution. In our experiments, they stand for the context window and the width of the tree-gram that generated a corpus. Here, we attempt to model the dependence of the distribution on these hyper-parameters through the toy model that we describe above. n sets a finite context window by an abrupt cutoff on the lookback distance $K(n) = \log_2[n-1]$. d controls the fidelity of the different modes through a quality factor $q(d)$ which we define later.

(A2) Suffix windows, one summary per level. The model keeps one d-channel summary per level and ignores the internal organization of each new block.

E.2 Scale powers

A hierarchy alone does not determine s_k. Suppose that the level-k increment has amplitude $C_k = C_0 \prod_{\ell=1}^k A_\ell$, with $s_k = \mathbb{E}[C_k^2]$.

(A3) Self-similar branch law. The factors $C_0, A_1, A_2, \dots$ are mutually independent, and $\mathbb{E}[A_\ell^2] = 2^{-\alpha}$ for every ℓ.

Then $s_k = \mathbb{E}[C_0^2]\, 2^{-\alpha k}$. Such a power law was also obtained in (Lin & Tegmark, 2017). The empirically relevant case is $\alpha > 0$. Only the levels whose window fits in the context are active:

$$K(n) = \lfloor \log_2(n-1) \rfloor. \tag{17}$$

This sets the context approximation in **(A1)**. Hence K is constant for $n \in \{2^K+1, \dots, 2^{K+1}\}$, and the tokens at distances $2^{K(n)}+1, \dots, n-1$ are dropped.

(A4) Fixed total budget. For every n, the total clean power is one: $\sum_k s_k = 1$.

With $r = 2^{-\alpha}$, the level powers are

$$\pi_k(n) = \frac{r^k}{Z_{K(n)}}\, \mathbf{1}[k \le K(n)], \qquad Z_K = \sum_{\ell=0}^{K} r^\ell = \begin{cases} (1 - r^{K+1})/(1-r), & r \neq 1, \\ K+1, & r = 1. \end{cases} \tag{18}$$

The largest level power is $\pi_{\max}(K) = 1/Z_K$ for $\alpha \geq 0$ and $r^K/Z_K = 1/\sum_{j=0}^{K} r^{-j}$ for $\alpha < 0$. It decreases strictly with K and tends to $1-r$, 0, and $1-1/r$ for $\alpha > 0$, $\alpha = 0$, and $\alpha < 0$. Assumption (A4) deserves care. For one fixed source, the increments equation 16 do not depend on n, so a longer context adds levels without weakening the old ones. The normalization therefore describes a family of teachers, one for each n, each scaled to unit predictable power. It is also the only source of n-dependence in the spectral edge below. Without it, $\max_k r^k$ is independent of n for $\alpha \geq 0$ and grows with n for $\alpha < 0$.

E.3 The feature–label generator

For each sample, each active level k, and each channel $a = 1, \ldots, d$, draw

$$S_{k,a},\ \Xi_{k,a} \overset{\text{iid}}{\sim} \mathcal{N}\big(0, \pi_k(n)/d\big), \qquad \Phi_{k,a} = \sqrt{q}\, S_{k,a} + \sqrt{1-q}\, \Xi_{k,a}, \qquad q(d) = \frac{d}{d+d_0}. \tag{19}$$

Choose centered logit directions $u_{k,a} \in \mathbb{R}^V$ that are orthonormal in the F_0 metric, which requires $d\,(K(n)+1) \leq V-1$. Generate

$$g(S) = \sum_{k=0}^{K(n)} \sum_{a=1}^{d} S_{k,a}\, u_{k,a}, \qquad Y \mid S \sim \text{Categorical}\left(\text{softmax}(g(S))\right). \tag{20}$$

The dataset contains (Φ, Y), not S or Ξ. The index k labels the level and a labels its channels. Each channel has its own direction $u_{k,a}$, so the d features of a level stay distinguishable in their effect on the prediction, and d is the number of resolved features per level. The latent summaries play the role of the context in Section E.1: $\Delta_k = \sum_a S_{k,a} u_{k,a}$ and $s_k = \pi_k(n)$. The base logit is zero, so p_0 is uniform. A non-uniform unigram distribution would enter as a fixed logit offset.

(A5) Independent, balanced Gaussian channels. All $S_{k,a}$ and $\Xi_{k,a}$ are jointly Gaussian and independent, and the d channels of a level share its power equally.

(A6) Scale-uniform recovery. The recovery fraction $q(d) = d/(d+d_0)$ is the same at every level and does not depend on n.

The per-channel signal-to-noise ratio is $q/(1-q) = d/d_0$. Because the channel is Gaussian, $S_{k,a} \mid \Phi_{k,a} \sim \mathcal{N}\big(\sqrt{q}\,\Phi_{k,a}, (1-q)\pi_k/d\big)$, and by independence $\mathbb{E}[S_{k,a} \mid \Phi] = \mathbb{E}[S_{k,a} \mid \Phi_{k,a}]$. The posterior mean and its power are

$$\widehat{S}_{k,a} = \sqrt{q}\,\Phi_{k,a}, \qquad \omega_k(n,d) \equiv \text{Var}(\widehat{S}_{k,a}) = q(d)\,\frac{\pi_k(n)}{d} = \frac{\pi_k(n)}{d+d_0}. \tag{21}$$

The recoverable covariance is $\Sigma_{\text{rec}} = \text{diag}\{\omega_k\} = q(d)\,\Sigma_\Phi$, with trace $q(d)$. More capacity recovers more total signal, but each channel carries less of it: ω_k falls with d while $\text{tr}\,\Sigma_{\text{rec}}$ rises. The recoverable power stays finite as $d \to 0$. The normalized spectrum $\omega_k/\,\text{tr}\,\Sigma_{\text{rec}} = \pi_k/d$ is known in closed form, and no corpus statistic enters it.

E.4 A matched quadratic learner

We now provide a minimal quadratic model to set up the learner for the data distribution above.

The learner has one coordinate $\theta_{k,a}$ for every channel it can represent, including channels that are inactive at the current (n,d). It predicts $f_\theta(\Phi) = \sqrt{\gamma}\sum_{k,a} \theta_{k,a}\widehat{S}_{k,a}u_{k,a}$. Take a fixed regression target $g^\dagger(\Phi)$, such as $\mathbb{E}[g(S) \mid \Phi] = \sum_{k,a} \widehat{S}_{k,a}u_{k,a}$. The population objective and its Hessian are

$$\mathcal{Q}(\theta) = \tfrac{1}{2}\,\mathbb{E}\,\|f_\theta(\Phi) - g^\dagger(\Phi)\|^2_{F_0} + \tfrac{\lambda}{2}\|\theta\|_2^2, \qquad H = \lambda I + \gamma\Sigma_{\text{rec}}, \qquad h_{k,a} = \lambda + \gamma\,\omega_k(n,d). \tag{22}$$

The off-diagonal entries $\gamma\,\mathbb{E}[\widehat{S}_{k,a}\widehat{S}_{k',a'}]\,u_{k,a}^\top F_0 u_{k',a'}$ vanish because the output directions are F_0-orthonormal, even if the features are correlated. Inactive coordinates have $\widehat{S} \equiv 0$ and curvature λ. Data features and optimization modes therefore align by construction. The target moves the optimum but not the Hessian.

(A7) Matched features. The learner reads $\widehat{S} = \sqrt{q}\,\Phi$, not Φ. This is a strong assumption. We justify it here by assuming that the warmup phase of learning aligns the model's features with the data features, so that the optimal learning rate is driven in this pseudo-constant phase where the learner model has aligned features.

(A8) Weak signal. Because f_θ is linear in θ, the regularized cross-entropy objective has the exact Hessian $H_{\mathrm{CE}}(\theta) = \mathbb{E}[J^\top F(p_\theta(\cdot \mid \Phi))J] + \lambda I$, with $J = \partial f_\theta/\partial\theta$ and $p_\theta = \mathrm{softmax}(f_\theta)$. Equation equation 22 replaces $F(p_\theta)$ by F_0. This assumes that predictions stay near the reference distribution, that is, that the total clean power σ^2 is small in F_0 units. Assumption (A4) sets $\sigma^2 = 1$, and a smaller σ^2 is absorbed into γ.

The replacement is exact at $\theta = 0$. At any θ, the correction that is first order in the logits vanishes in expectation, because it is odd under $(S, \Xi) \to (-S, -\Xi)$. The error is therefore of relative order σ^2. Numerically, take $V = 16$, $K = 2$, $d = 3$, $d_0 = 2$, $r = 1/2$, and 24 random draws of the $u_{k,a}$. At the cross-entropy optimum, the median of $\|H_{\mathrm{CE}} - H\|/\|H\|$ is 1%, 12%, and 28% at $\sigma^2 = 0.1$, 1, and 4. The unit budget corresponds to about half a nat of divergence from the uniform distribution, since $\mathrm{KL}(\mathrm{softmax}(g) \,\|\, p_0) \approx \frac{1}{2}\|g\|^2_{F_0}$. The quadratic model is exact for a Gaussian regression target $\widetilde{Y} = g(S) + \varepsilon$ with independent centered noise, since then $\frac{1}{2}\mathbb{E}\|f_\theta - \widetilde{Y}\|^2_{F_0} = \frac{1}{2}\mathbb{E}\|f_\theta - \mathbb{E}[g \mid \Phi]\|^2_{F_0} + \text{const}$.

E.5 Spectral edges and the optimal learning rate

(A9) Ridge-only directions. At least one learner coordinate has no data curvature, for example an inactive channel, so $L = \lambda$. Otherwise, $L = \lambda + \gamma\pi_{\min}/(d + d_0)$ also moves with (n, d).

(A10) Deterministic, non-adaptive optimizer. The optimizer takes full-batch steps on $\mathcal{Q}$ with constant parameters. Minibatch noise, schedules, and Adam-type preconditioning are outside the model.

The upper edge is

$$U(n, d) = \lambda + \gamma\,\frac{\pi_{\max}(K(n))}{d + d_0}. \tag{23}$$

For a spectrum in $[L, U]$, gradient descent $\theta_{t+1} = \theta_t - \eta\nabla\mathcal{Q}(\theta_t)$ has the minimax step $\eta^\star_{\mathrm{GD}} = 2/(L + U)$. This step equalizes $1 - \eta L = \eta U - 1$ and contracts at rate $(U - L)/(U + L)$. The heavy-ball update $\theta_{t+1} = \theta_t - \eta\nabla\mathcal{Q}(\theta_t) + \beta(\theta_t - \theta_{t-1})$ has $\eta^\star_{\mathrm{HB}} = 4/(\sqrt{L} + \sqrt{U})^2$ and $\beta^\star_{\mathrm{HB}} = \big((\sqrt{U} - \sqrt{L})/(\sqrt{U} + \sqrt{L})\big)^2$ (Polyak, 1964; Danilova & Malinovsky, 2022). Nesterov's method, $y_t = \theta_t + \beta(\theta_t - \theta_{t-1})$ and $\theta_{t+1} = y_t - \eta\nabla\mathcal{Q}(y_t)$, commonly uses $\eta = 1/U$ and $\beta = (\sqrt{U} - \sqrt{L})/(\sqrt{U} + \sqrt{L})$ (Assran & Rabbat, 2020). With $L = \lambda$, the gradient-descent rate decomposes exactly as

$$\nu^\star_{\mathrm{GD}}(n, d) \equiv \log_2 \eta^\star_{\mathrm{GD}} = \underbrace{\log_2 \frac{1}{\lambda}}_{\nu^\star_\infty} - \underbrace{\log_2\Big(1 + \frac{\gamma\,\pi_{\max}(K(n))}{2\lambda\,(d + d_0)}\Big)}_{\text{capacity term} \,\geq\, 0}. \tag{24}$$

The capacity term decreases strictly in d and stepwise in n. It vanishes as $d \to \infty$, stays finite as $d \to 0$ because $d_0 > 0$, and saturates in n for $\alpha \neq 0$. When $U \gg L$ and $\alpha \geq 0$, Eq. equation 24 becomes $\nu^\star_{\mathrm{GD}} \simeq 1 - \log_2\gamma + \log_2(d + d_0) + \log_2 Z_{K(n)}$. More generally, $\nu^\star_A = C_A - \log_2 U + o(1)$, with $C_{\mathrm{GD}} = 1$, $C_{\mathrm{HB}} = 2$, and $C_{\mathrm{NAG}} = 0$. The heavy-ball correction is only $O(\sqrt{L/U})$. The stability threshold $2/U$ has the same scaling, so the prediction concerns the dependence on U rather than the optimality criterion.

The monotone drift uses (A1), (A4), (A5), and (A9), together with two facts: new levels receive positive power, and recovery grows sublinearly, so that $q(d)/d = 1/(d + d_0)$ decreases and dilution beats recovery. The drift can fail if the total power grows with depth, if recovery improves faster than linearly over some range of d, or if an unrelated direction sets U.

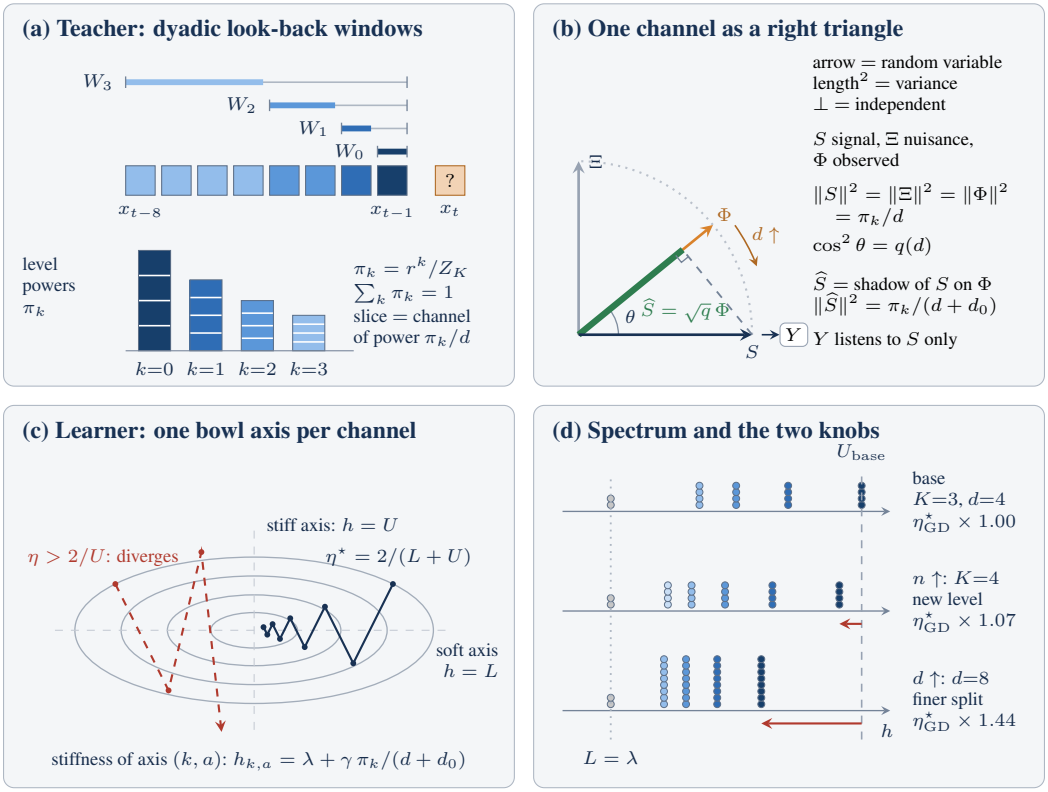

Figure 6: The dyadic toy model (top) and its local quadratic learner (bottom). **(a)** Each doubling of the look-back window reveals one new block of context (colored), whose contribution to the next-token logit has power π_k. The powers form a geometric profile with a fixed total, and each level is carried by d channels; drawn for $\alpha = 1/2$, $K = 3$, $d = 4$. **(b)** Random variables drawn as vectors. The observed feature Φ mixes the signal S and the nuisance Ξ at angle θ, with $\cos^2\theta = q(d)$. The label depends only on S, and the best linear use of Φ is the shadow of S on Φ, the posterior mean $\widehat{S}$. **(c)** The matched learner's loss is a bowl whose principal axes are the channels, drawn for $U/L = 6$. The step $\eta^\star = 2/(L+U)$ shrinks the stiffest and softest axes at the same rate $(U-L)/(U+L)$; a step above $2/U$ diverges along the stiff axis. **(d)** Eigenvalues $h_{k,a}$ stacked by multiplicity, for $\alpha = 1/2$, $d_0 = 2$, $\lambda = 0.01$, $\gamma = 1$; gray dots are ridge-only directions. Adding a level re-divides the fixed budget and lowers U slightly. Doubling d spreads every level over twice as many channels and lowers U substantially. The right column gives the change in $\eta^\star_{\text{GD}}$.

E.6 From the solvable learner to a transformer

Equation equation 22 makes the data features and the Hessian eigenvectors coincide by design. Applying Eq. equation 24 to a transformer requires a hypothesis in three parts. **(H1)** The dominant Hessian eigendirections align approximately with the multiscale predictive features. **(H2)** The top eigenvalue belongs to such a direction and equals an (n, d)-independent gain times its recoverable power. **(H3)** The lower spectral edge does not depend on (n, d).

For a well-specified softmax model at a population optimum, the cross-entropy Hessian reduces to its Gauss–Newton part,

$$H_{\text{CE}} = \mathbb{E}_X\left[J_\theta^\top F(p_\theta(\cdot \mid X))\, J_\theta\right], \tag{25}$$

because the residual term $\mathbb{E}_X\left[\sum_y (p_\theta(y \mid X) - p(y \mid X))\nabla^2_\theta f_{\theta,y}(X)\right]$ vanishes there. The strengths s_k determine neither the singular vectors nor the singular values of the learned logit Jacobian J_θ. A transformer can rotate the features, mix scales, and add unrelated high-curvature directions. The Fisher factor adds a corpus dependence of its own: $F(p_\theta)$ shrinks as predictions sharpen, so the gain in (H2) can itself depend on (n, d). The learned Jacobian is the unresolved first-principles bottleneck.

F TREE-GRAM EXPERIMENTAL DETAILS AND COMPARISON TO TRANSFORMERS

We performed experiments with 4 NVIDIA A100 GPUs. The 529.2M-token run for the $n = 32$, $d = 1024$ model (with MLP) takes 5 hours of wall-clock time. A total of 11,000 A100 GPU hours were used for all pre-training experiments on tree-grams reported in this paper.

We train on the 529.2M-token training split of TinyStories, tokenized with a custom BPE tokenizer (Cagnetta et al., 2026), and validate on a 1.05M-token subset of its 5.34M-token validation split.

Every run trains for a single epoch over the 529.2M training tokens, cut into non-overlapping windows of n tokens with 1024 windows per batch, using AdamW at its default moments. The learning rate warms up linearly from the initial to the peak value over the first 1% of steps and then follows a cosine decay to 1% of the peak. Unless stated otherwise the architecture uses LayerNorm, a residual stream of unit strength, GELU, $d_{\text{ff}} = 4d$ and tied embeddings. We used a 1.05M validation set.

We tuned the peak learning rate and the weight decay for every cell. The n-sweep used $\eta_{\max} \in \{3\times10^{-5}, 10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}\}$ and $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 1\}$, and the d-sweep added $\eta_{\max} = 10^{-2}$; the $n = 32$, $d = 1024$ cell pools this grid with an earlier one that reaches $\eta_{\max} = 3\times10^{-2}$ and $\lambda = 10^{-4}$. The initial learning rate was 10^{-5} except for τ_{-1} and τ_{-2}. The topology ablations used smaller grids to test for learning rate transfer: $\eta_{\max} \in \{3\times10^{-4}, 10^{-3}, 3\times10^{-3}\}$, with $\lambda \in \{10^{-3}, \dots, 1\}$ for τ_1, τ_2 and $\lambda \in \{10^{-2}, 10^{-1}, 1\}$ for τ_a, τ_b. The anti-aligned τ_{-1} and τ_{-2} need smaller rates and were swept over initial rates $\{10^{-6}, 10^{-5}\}$, $\eta_{\max} \in \{10^{-5}, \dots, 10^{-3}\}$ and $\lambda \in \{10^{-4}, \dots, 10^{-1}\}$ (τ_{-2}: $\eta_{\max} \in \{10^{-4}, 3\times10^{-4}, 10^{-3}\}$, $\lambda \in \{10^{-2}, 10^{-1}\}$). Without MLPs only τ_0 and τ_{-1} were swept; the other five topologies were run at a single setting ($\eta_{\max} = 10^{-3}$, $\lambda = 0.1$). Cells were scored by the minimum validation loss over the run, averaged over two seeds. At the selected settings of the cells with MLPs, the seed-to-seed standard deviation is at most 0.0025 nats ($d = 64$), and at most 0.0004 nats for the other n- and d-sweep cells; it reaches 0.06 nats for the untuned topologies without MLPs. We ran 2 random seeds for each HP configuration of the n- and d-sweeps and then ran a total of 4 random seeds for the model with the best HP configuration. Data generation took 200 GPU hours.

For μP transformers comparison, we scanned 10 peak learning rates while holding the weight decay fixed at 0.01, and kept the rate with the lowest validation loss on a single seed. We ran a total of 3 seeds for the transformers with the best value of the hyper-parameter combination. These were also trained with AdamW at batch 1024 for one epoch on the same windows as the tree-grams, with the same 1% warmup from 10^{-5}, decaying to 10% of the peak, and with dropout 0.0. The transformers were based on (Kalra, 2025).

F.1 PARAMETER-MATCHED TRANSFORMER CONFIGURATIONS

Table 1 gives the transformer width D used for each tree-gram and transformer depth in Fig. 2. All transformers use eight heads, MLP width $4D$, GELU, RoPE, tied embeddings, and the same context and token budget as the corresponding tree-gram. Parameter counts are matched to the nearest available width. Thus, width varies with depth and the sweep does not isolate depth.

Table 1: Exact parameter-matched transformer widths. Tree parameters include embeddings.

tree-gram		Tree params	Transformer width D by depth L							
n	d		1	2	3	4	5	6	10	12
3	1024	36,709,376	1440	1072	896	800	720	656	512	480
4	1024	56,637,440	1856	1376	1152	1008	912	832	656	592
8	1024	78,665,728	2240	1648	1376	1200	1072	992	784	704
16	1024	104,892,416	2640	1920	1600	1392	1248	1152	896	832
32	64	1,039,744	112	96	96	80	80	80	64	64
32	128	3,103,488	272	224	208	176	176	160	128	128
32	256	10,302,976	640	512	432	384	352	320	256	240
32	512	36,989,952	1440	1088	912	800	720	656	528	480
32	1024	139,515,904	3088	2240	1856	1616	1456	1344	1040	960

G HP TRANSFER EXPERIMENT DETAILS

G.1 LEARNING RATE SWEEPS

For the hyper-parameter-transfer study in Fig. 3(a,b), all transformers use eight heads, $d_{\text{ff}} = 4d$, GELU, RoPE, tied embeddings, no dropout, sequence length 128 and batch 256 ($32{,}768$ tokens per step). Every proxy and every target trains for one epoch over the full 529.2M-token training split of its corpus ($16{,}024$ steps), so a proxy trained on real data differs from its target only in width. We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$ and weight decay 0.01, a linear warmup from 10^{-5} over the first 1% of steps, and a cosine decay to 10% of the peak.

Proxy widths are $d \in \{64, 96, 128, 160, 192, 256\}$ at $L \in \{2, 4\}$, each trained on all eleven corpora of Fig. 3(c). Targets are trained on real TinyStories only, at $d \in \{384, 512, 640, 832, 1024, 1280, 1600, 2048\}$ for $L = 2$ and $d \in \{288, 384, 480, 608, 768, 928, 1152, 1408\}$ for $L = 4$; the main text reports $d = 1024$ ($L = 2$) and $d = 1408$ ($L = 4$). Peak rates lie on the lattice $\eta_k = 0.0039 \cdot 10^{k/8}$. For the proxies, seed 0 spans $[0.0039, 3.9]$ at $d = 64$ and $[0.00925, 1.64]$ at the wider widths, and seeds 1 and 2 span $[0.052, 1.64]$. This was done to ensure that the optimal learning rate is obtained at the basin of the loss vs learning rate curve. Targets use three seeds on $[0.052, 6.94]$, or on $[0.0123, 2.19]$ for the two widest targets at each depth, plus two to four off-lattice rates per target inserted near its divergence onset.

G.2 CONTINUOUS $\eta^\star$ FROM A SMOOTHING SPLINE

In order to get a more precise estimate of $\eta^\star$ we use a cubic smooothing spline as discussed in (Kalra & Barkeshli, 2026). Writing $\nu = \log \eta$, we discard points with $L > 1.35\, L^\star$, fit a cubic smoothing spline to $L(\nu)$ with smoothing parameter $s = 0.1\, N \operatorname{Var}(L)$, evaluate it at 400 uniformly spaced ν in the retained range, and take its minimizer.

Two adaptations are needed here. First, the $1.35\, L^\star$ window was calibrated on sweeps with a different loss scale; with $L^\star \approx 1.48$ nats the bar sits near 2.0 and admits points already far up the divergence cliff. A single such point inflates $\operatorname{Var}(L)$, hence s, and the cubic cannot turn fast enough to follow the cliff, so it smears it back across the basin. We therefore truncate the window at the divergence onset, cutting at the first increment above $\max(3h,\, 0.05)$ nats, where h is the median $|\Delta L|$ along the descending branch; across all 412 sweeps no descending increment exceeds 0.045 nats while the smallest increment the rule flags is 0.062, so it separates basin from cliff without ever truncating a basin, and the median residual falls to 0.0009 nats. Second, we use the spline only to *locate* optima. A proxy may recommend an η far up the target's cliff — tuning on the real corpus does exactly that — and there the measured loss is precisely the quantity of interest, so we score the target with a shape-preserving (PCHIP) interpolant of its full curve rather than with the spline. That interpolant cannot itself locate an optimum, since it sets a zero derivative at every data extremum and its minimiser is the grid argmin again; only a smoothing spline moves $\eta^\star$ off the lattice.

The re-estimate leaves our conclusions unchanged and sharpens two of them. The median-over-synthetics rule stays within one grid point of each target's own optimum with no grid involved: $\eta^\star_{\text{median}}/\eta^\star_{\text{target}}$ is 0.87–1.05 at $L{=}2$ and 1.11–1.22 at $L{=}4$, against a rung of $4/3$.

H FINETUNING EXPERIMENT DETAILS

H.1 METHODOLOGY AND OPTIMIZATION

For the finetuning experiment the downstream models are $(L, d) = (2, 1024)$ and $(4, 768)$. We evaluate every ordered pair among the bigram corpus, the five horizon-sweep corpora $\{n3, n4, n8, n16, n32\}_{d=1024}$, the five capacity-sweep corpora $n32_{d\in\{64,128,256,512,1024\}}$, and real TinyStories: an 11×11 matrix for each depth and three seeds. Fine-tuned models are validated on a 1.05M-token subset of the target's validation split.

Pre-training. Each source is pre-trained for one epoch over its 80% segment ($12{,}819$ steps). Its peak rate $\eta^\star_{\text{PT}}$ is the argmin, on one seed, over seven lattice rates centred on that corpus's optimum at

$d = 256$ in the HP-transfer sweep, a $5.6\times$ window; no selected rate lies on the edge of its window. This gives $11 \times 2 \times 3 = 66$ pre-trained checkpoints.

Fine-tuning. All parameters are updated from the pre-trained weights with a fresh AdamW state; the learning rate is re-warmed from 10^{-5} over the first 1% of steps and cosine-decayed to 10% of its peak Gupta et al. (2023); Ibrahim et al. (2024), with no replay of source data. The fine-tuning peak rate is set per (source, target) pair: $\eta_{FT}(s \to t) = \eta^{\star}_{PT(t)} \cdot \eta^{\star}_{FT(s\to\text{real})}/\eta^{\star}_{PT(\text{real})}$, snapped to the lattice, where $\eta^{\star}_{FT(s\to\text{real})}$ comes from a one-seed sweep over a shared 22-point window. Tuning the rate per model follows Isik et al. (2025) ; re-warming to the pre-training peak instead trades adaptation against forgetting Gupta et al. (2023), and we score only the target.

Budgets and references. The matrix is read after the full 105.8M fine-tuning tokens (3,205 steps). For a synthetic target, the floor is its generator's entropy rate H_q (Eq. 9). For real TinyStories we use 1.242 nats, the converged loss of the 12-layer, $d = 1024$ GPT-2 of (Cagnetta et al., 2026) at $T = 128$, which is an empirical reference rather than a Bayes floor. Each cell reports the median over the three seeds of the final validation loss. As a no-pre-training reference, the same models trained from scratch on the same 105.8M real tokens, at 0.52 (within one lattice step of their own optimum on this budget), reach 1.688 nats at $L = 2$ and 1.600 nats at $L = 4$ (medians of three seeds), that is 0.36 and 0.29 on the scale of Fig. 3(c,d).

Unstable cells. Of the 242 cells of the transfer matrix, 10 have a standard deviation across the three chained seeds above 0.05 nats, against a median of 0.0006. They lie in two rows: `n8_d1024` at $L = 2$ ($\eta_{\text{FT}} = 0.922$ and 1.23; targets `n16_d1024`, `n32_d1024`, `n32_d64`, `n32_d128`, `n32_d256`, `n32_d512` and `real`; largest standard deviation 0.36 nats) and `n32_d512` at $L = 4$ ($\eta_{\text{FT}} = 1.23$ and 1.64; targets `n32_d256`, `n32_d512` and `real`; largest 0.40 nats). The disagreement is between seeds. We report the median as a majority-voting rule.

I ADDITIONAL PLOTS

We collect here the plots alluded to in the main text, at full size. In order:

- A per-position cross-entropy comparison of every tree-gram in the sweep against itsparameter-matched transformers, at every depth (Fig. 7).
- The transfer phase diagram over every (proxy width, target width) pair, contrasting the median-over-synthetics rule with tuning on the real corpus (Fig. 8).
- The raw loss-versus-learning-rate curves for different widths on different datasets, with the spline-function interpolated values. (Fig. 9).
- Plots of the optimal learning rate for proxy widths as a function of the entropy of the distribution is presented in Figure 10. The non-monotonic behavior of the optimal learning rate as a function of the width for the real dataset is presented in Figure 11.
- The pre-training transfer matrix with both axes re-sorted by entropy rates, which is where the two synthetic sweeps are seen to interleave (Fig. 12).
- Activation patching for the fixed tree-gram at terminal nodes recover the effective receptive fields for the aligned τ_0 and anti-aligned τ_{-1} topology. (Fig. 13).

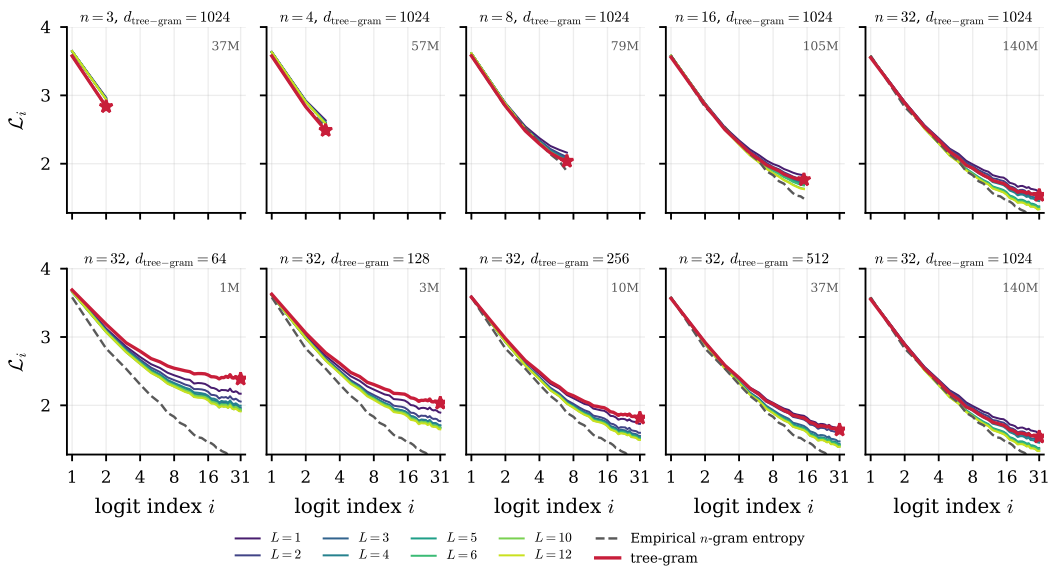

Figure 7: Per-position cross-entropy for every tree-gram in the sweep against its parameter-matched transformers, for various depths L. **(top)** n varies at $d = 1024$; **(bottom)** d varies at $n = 32$. Grey annotations give the parameter count of the tree-gram in that panel (1M to 140M). The tree-gram tracks the transformer family exactly for the first few positions in every panel and peels away only at long context.

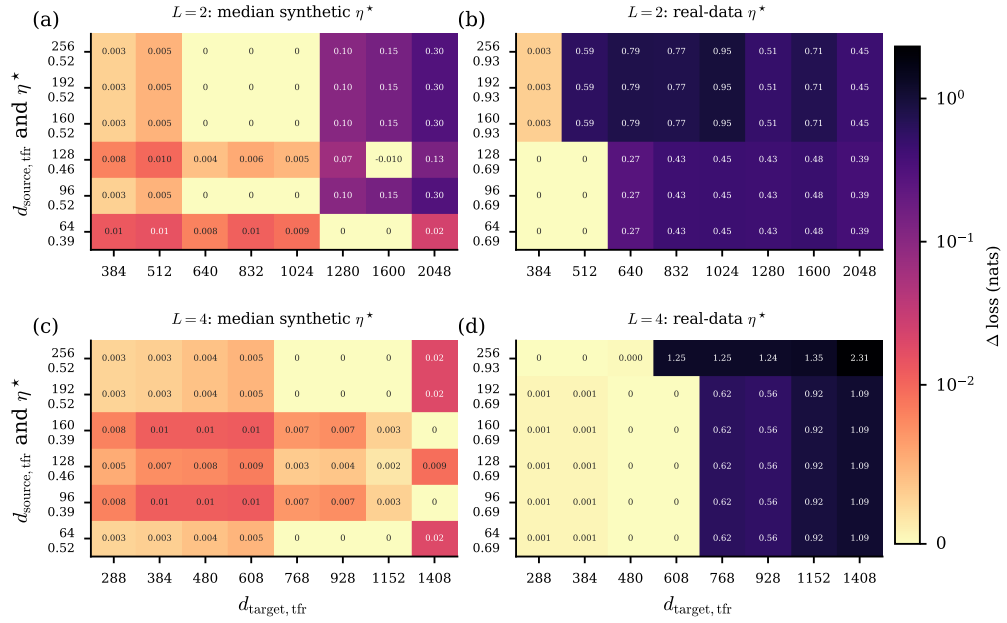

Figure 8: The HP transfer phase diagram. (**Left**) We plot the excess loss obtained when transferring the median optimal learning rate $\nu^\star$ from a source width $d_{\text{source,tfr}}$ to a target width $d_{\text{target,tfr}}$. (**Right**) The same plot when we tune a proxy on the real data. Rows are depth $L = 2$ (top) and $L = 4$ (bottom). The synthetic rule is more robust as compared to tuning on the real data (the standard recipe).

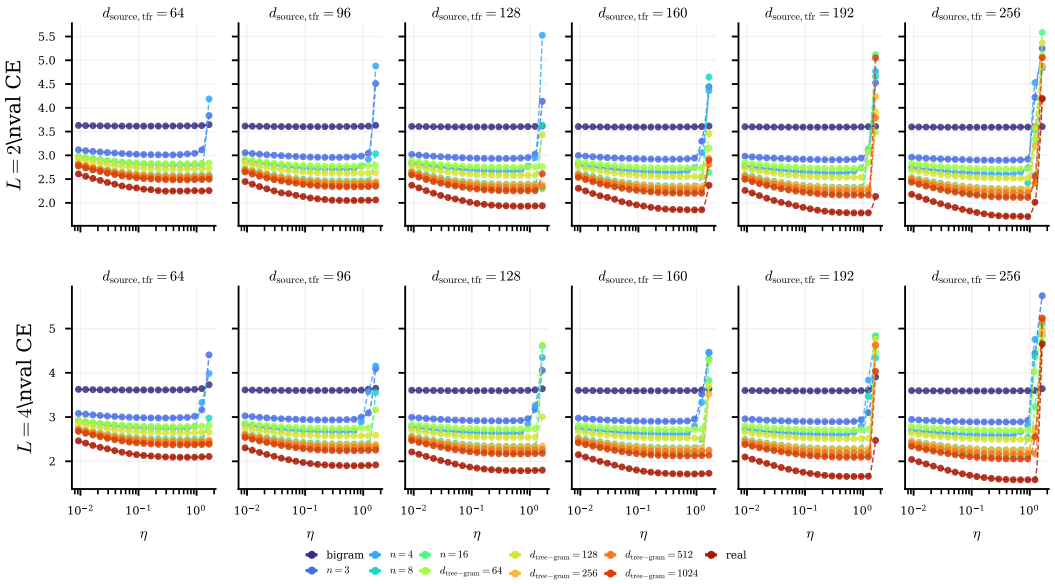

Figure 9: Converged losses as a function of the peak learning rates. The solid dots represent the actual runs while the dashed lines represent the interpolated losses using a spline function.

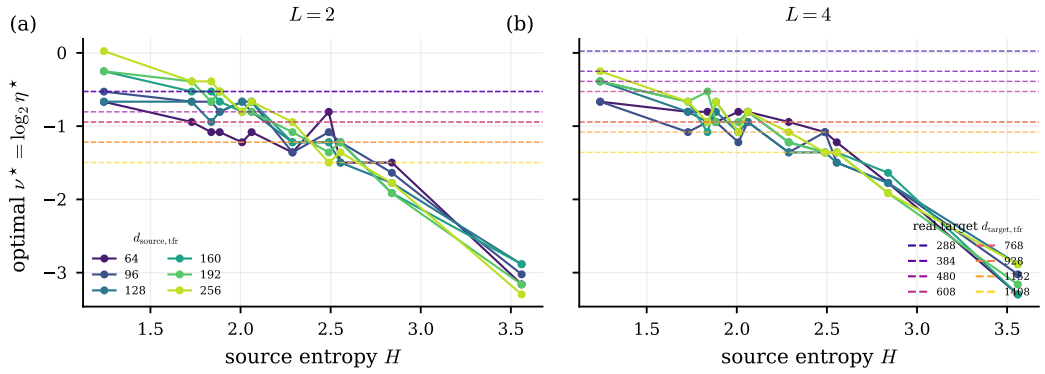

Figure 10: Optimal learning rate $\nu^\star \equiv \log_2 \eta^\star$ as a function of the entropy of the generated distributions. We see a monotonic decrease with increase in entropy. However the width-dependence is extremely weak.

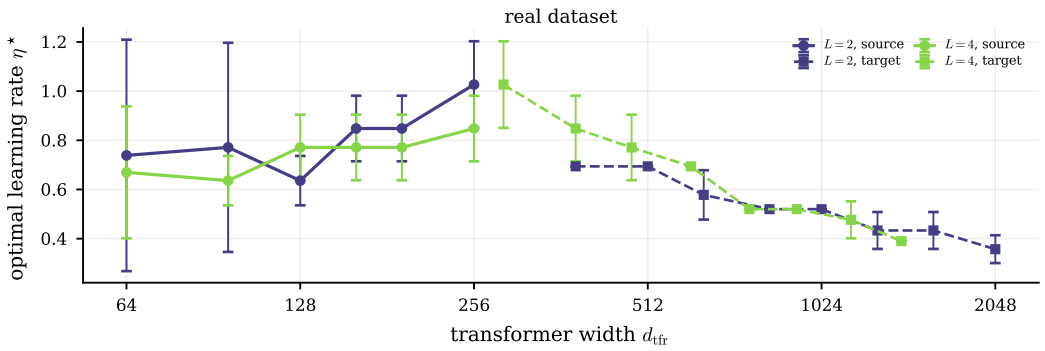

Figure 11: Optimal learning rate $\nu = \log \eta$ for the real dataset as a function of width. We observe a non-monotonic trend.

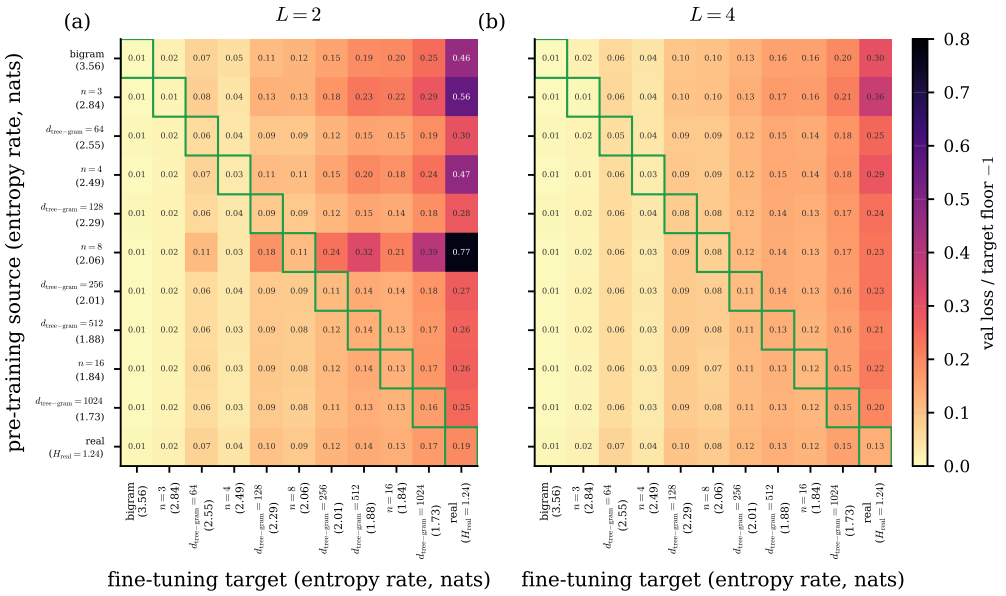

Figure 12: The pre-training transfer matrix of the main text, panels (c) and (d), with both axes re-sorted by *entropy rate* of the synthetic corpus, instead of grouped by mode of approximation. The green boxes are the same-distribution diagonal. We observe that the monotonic trend we observed in the main text, is not longer present. In addition, horizon-limited modes perform worse than capacity-limited modes, even when they have similar entropy rate.

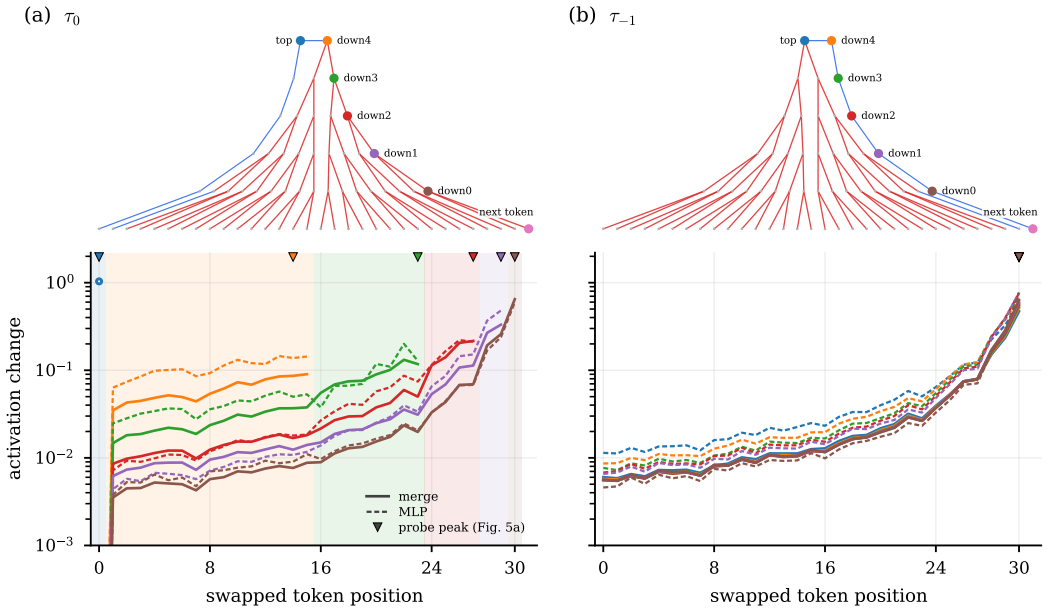

Figure 13: Receptive fields of individual nodes, recovered by activation patching. The token at each position is swapped, one position at a time, for the token another validation sequence holds there (1024 sequences), and every activation of the tree-gram is compared with the unswapped run. Top: the topology with its leaves aligned to the token axis below. Bottom: mean squared change of each node's Merge (solid) and MLP (dashed) activation, relative to its change when the whole context is replaced (log scale). Outside a node's scope its activations are exactly unchanged (no line). Triangles mark where the linear probe of Fig. 5a peaks. **(a)** Each τ_0 node responds only to tokens up to the end of its window and most to the window it adds, peaking in the same windows as the probe; x_0 reaches the down nodes only below 10^{-8}. **(b)** In τ_{-1}, no individual context windows are observed.

J GOODNESS OF FIT FOR RANKED PATH PROBABILITIES

We sort the exact path probabilities and normalize each candidate on the finite rank support, $r = 1, \ldots, 36720$. The probability of a path is the token average of its per-token joint probability, $p(\text{path}) = \mathbb{E}_x\big[\prod_\ell p^{(\ell)}(g_\ell \mid x)\big]$, over 192,448 validation tokens.We fit each router and seed separately, by least squares on $\log p(r)$. This gives equal weight to each rank and resolves the low-probability tail. The primary measures are log-RMSE and ΔAICc. AICc assumes Gaussian log residuals. KS and $D_{\mathrm{KL}}(p\|q)$ are secondary descriptive checks.

Table 2: Goodness of fit for the ranked path-probability curves, fitted separately to the two seeds (s0, s1) of each router. Lower is better in every column, and the best value in each column of a block is in bold. PL denotes power law. GEC denotes generalized exponential cutoff. DGBD denotes the discrete generalized Beta distribution discussed in (Naumis & Cocho, 2007).

	log-RMSE		ΔAICc		KS		KL	
Model	s0	s1	s0	s1	s0	s1	s0	s1
$m = 1$								
Exponential	2.965	2.972	191424.8	175489.6	0.839	0.809	3.597	3.513
Generalized exponential	0.433	0.393	50186.3	26910.6	0.373	0.340	0.580	0.590
Power law / Pareto	1.622	1.940	147131.1	144180.5	0.431	0.459	0.682	0.880
PL with exponential cutoff	0.351	0.371	34644.7	22679.1	0.266	0.279	0.200	0.259
PL with GEC	**0.219**	**0.272**	**0.0**	**0.0**	**0.087**	**0.120**	**0.068**	**0.128**
Broken PL / double Pareto	0.281	0.467	18371.9	39542.0	0.271	0.286	0.211	0.272
DGBD / beta-rank	0.939	1.007	106993.4	96049.9	0.375	0.398	0.443	0.559
$m = 2$								
Exponential	1.736	2.288	179757.3	192998.3	0.743	0.879	2.797	4.086
Generalized exponential	0.369	0.735	66086.3	109589.8	0.341	0.565	0.417	0.956
Power law / Pareto	0.852	0.554	127501.0	88814.8	0.317	0.244	0.333	0.161
PL with exponential cutoff	0.208	0.255	24092.6	31735.5	0.127	0.155	0.066	0.090
PL with GEC	0.201	0.242	21497.1	27930.4	**0.095**	**0.120**	**0.044**	**0.074**
Broken PL / double Pareto	**0.150**	**0.165**	**0.0**	**0.0**	0.178	0.154	0.114	0.090
DGBD / beta-rank	0.512	0.392	90046.1	63404.0	0.241	0.209	0.193	0.125

For $m = 1$ the power law with a generalized exponential cutoff is best on every measure for both seeds, with log-scale $R^2 = 0.9959$ and 0.9948. For $m = 2$ the broken power law has the lowest log-RMSE and AICc for both seeds ($R^2 = 0.9953$ and 0.9929), while the generalized cutoff has the lowest KS and KL. Both place the bend at $r \approx 6{,}000$–$9{,}000$, one as a sharp break and one smoothly. Under the generalized cutoff, $p(r) \propto r^{-\alpha} e^{-(r/r_c)^\beta}$, the cutoff scale separates the two routers, $r_c = 541$ and 777 for $m = 1$ against 6,860 and 8,170 for $m = 2$, while α (1.16–1.33) and β (0.53–0.81) vary about as much between the two seeds of one router as between routers. The 100 most probable paths carry 88% and 85% of the mass for $m = 1$ and 70% and 91% for $m = 2$, but their identity does not reproduce: the two seeds of a router share none of their 100 most probable paths, and only 3 ($m = 1$) and 0 ($m = 2$) of their top 500, against about 7 expected by chance. Pareto and power law are the same finite rank model here. Double Pareto and sharp broken power law are also the same rank curve. The smooth cutoff agrees with the transition from power-law-like ranks to faster tails discussed by (Naumis & Cocho, 2007).

K EXAMPLES OF GENERATED TEXT

Samples from fixed tree-grams on TinyStories (BPE-8192) at temperature 1, with a fixed context and generation seed. Within each block every model receives the same validation prompt, truncated to its own context length $n - 1$; the token stream is lightly de-tokenised (punctuation re-attached) for readability, and [EOS] marks the end-of-story token.

I. WIDTH SWEEP — d VARIES AT $n = 32$

$n = 32$**,** $d = 64$ **(31 context tokens,** $h_{\text{pred}} = 2.537$ **nats/token)**

Prompt. camp. [EOS] One day, a little girl named Lucy found a shoe in her room. It was a big shoe, much too big for her foot.

Continuation. Mia tried to climb up, but she couldn't. She felt so cold and strong. She did not shiver anymore. Her music made her. It made her bow back on the ground. The lemonade tasted like Mr. H ina's face turned into a small sc run. The fit and introduced all her was out to dry and decided to show it to her, making a pile castle. She had some money from

$n = 32$**,** $d = 128$ **(31 context tokens,** $h_{\text{pred}} = 2.247$ **nats/token)**

Prompt. camp. [EOS] One day, a little girl named Lucy found a shoe in her room. It was a big shoe, much too big for her foot.

Continuation. Mia wanted to comfort Fluffy, but she was scared. She did not know what to play with. Kitty's mom saw her feeling sad and did not want to help. So, she hid the medicine in the kitchen. [EOS] One day, a colorful bird named Bob flew in a big tree. March was a happy place to be. He was very happy to find the sun. Bobo flew to a big park where people

$n = 32$**,** $d = 256$ **(31 context tokens,** $h_{\text{pred}} = 1.938$ **nats/token)**

Prompt. camp. [EOS] One day, a little girl named Lucy found a shoe in her room. It was a big shoe, much too big for her foot.

Continuation. Lucy tried to fix the shoe, but it couldn't pinch her dress. Amy asked her mom, "Mom, what is this jacket?" Her mom said, "I don't know, Sue. Maybe they can use a glove to talk to someone." Finally, Tim's mom mentioned the coin to Tim. But when they went back home, Tim forgot about Spot. She gave it back to Tim and said

$n = 32$**,** $d = 512$ **(31 context tokens,** $h_{\text{pred}} = 1.890$ **nats/token)**

Prompt. camp. [EOS] One day, a little girl named Lucy found a shoe in her room. It was a big shoe, much too big for her foot.

Continuation. She tried to pull it, but she couldn't. She felt frustrated. Amy asked her mom, "Mom, what is this round thing?" Her mom smiled and said, "This is a special seat, Amy. It tells us things A test is when we drive around the front to the whole world. We can learn this and learn how to count to 10, to the beauty end. How does that sound?"

$n = 32$**,** $d = 1024$ **(31 context tokens,** $h_{\text{pred}} = 1.756$ **nats/token)**

Prompt. camp. [EOS] One day, a little girl named Lucy found a shoe in her room. It was a big shoe, much too big for her foot.

Continuation. She wanted to repair it, but she did not know how. "Mom, can I reverse our shoe?" Sam requested. His mom smiled and said, "Of course we can! I'll hold it for you." [EOS] One day, a colorful bird named Bob flew to a big tree. He wanted to settle there to rest and rest. While he was sitting, he heard a loud laugh. Rob opened his eyes

II. CONTEXT SWEEP — n VARIES AT $d = 1024$

$n = 3$, $d = 1024$ **(2 context tokens, $h_{\text{pred}} = 2.843$ nats/token)**

Prompt. too big

Continuation. for you. Do you want to buy the blue ball. One sunny day, a boy named Tom came to the war and be friends and her white dress. She ran back home. They played together and had fun. Would you like."Bob wanted to bake with her toys and games. They agreed and they both loved the doll." You can't need to sleep. [EOS] Once upon a time, there was a

$n = 4$, $d = 1024$ **(3 context tokens, $h_{\text{pred}} = 2.537$ nats/token)**

Prompt. much too big

Continuation. for you. Do you want to buy the blue ball. One sunny day, a boy named Tom came to Sam's orange collar and her white boots. She ran back home to play together. [EOS] Once upon a time, there was a rabbit. She had seen the wind's joy and introduced himself. The cat was busy and decided to show it to all her friends. She gave Lily a big hug and said

$n = 8$, $d = 1024$ **(7 context tokens, $h_{\text{pred}} = 2.085$ nats/token)**

Prompt. a big shoe, much too big

Continuation. and old. John felt frustrated. He yelled,'But the parrot wouldn't listen. He came closer and saw what was happening. She did not want to be rude. Tom looked at Sam and said, "Would you like some more?" But Tom could not hear Tim's voice and shake the hole. A big dog came to the castle. The dog did not like the decorations. He wanted some money from

$n = 16$, $d = 1024$ **(15 context tokens, $h_{\text{pred}} = 1.818$ nats/token)**

Prompt. in her room. It was a big shoe, much too big for her

Continuation. mouth. Sue didn't want her spaghetti, but her mom said, "Don't worry, we can make another one later." After that, they played a game where Tom squeezed the bowl and made a loud noise. "Boo!" A sound came out. "Wow, you fit us! You are good at hiding. You can escape through the hole to sleep." Tim followed Max to a big park where there

$n = 32$, $d = 1024$ **(31 context tokens, $h_{\text{pred}} = 1.756$ nats/token)**

Prompt. camp. [EOS] One day, a little girl named Lucy found a shoe in her room. It was a big shoe, much too big for her foot.

Continuation. She wanted to repair it, but she did not know how. "Mom, can I reverse our shoe?" Sam requested. His mom smiled and said, "Of course we can! I'll hold it for you." [EOS] One day, a colorful bird named Bob flew to a big tree. He wanted to settle there to rest and rest. While he was sitting, he heard a loud laugh. Rob opened his eyes