Inside the LLM Word Factory

Benzi Busigin **Yuval Pinter**
Stein Faculty of Computer and Information Science
Ben-Gurion University of the Negev
Beer Sheva, Israel
{busigin@post,uvp@cs}.bgu.ac.il

Abstract

Transformer language models process input provided as subword fragments, but natural language semantics usually rely on word-level concepts. *Detokenization* is the process where models reconcile these two facts, aggregating subwords into word-level representations through their computation. Prior work has found that this takes place mostly in early-to-middle layers, but so far the exact mechanics of the process have not been pinned down. We venture deep into detokenization using activation patching in controlled paired experiments that isolate the contribution of different model components, localizing English detokenization in Llama2-7B to a two-stage process at Layer 1. Attention transmits a token-specific signal from nonfinal subwords, using sequential relays if necessary, while the MLP composes it with the local embedding. This two-stage structure generalizes to twelve models from eight families, but the depth over which it takes place depends on the flavor of positional encoding: RoPE-based models detokenize over 1 to 5 layers, while learned-absolute models take 5 to 10. Finally, we provide a probe for determining the success of the detokenization process based on early-layer activations alone, performing at 0.94–0.97 AUROC depending on the amount of context.

1 Introduction

Subword tokenization is a near-universal preprocessing step in modern transformer language models (Sennrich et al., 2016; Kudo, 2018; Schmidt et al., 2024), yet it creates a systematic mismatch between input representation and internal computation: tokenizers split text into subword units based on statistical criteria, while the model's computations must operate over unified semantic concepts. A given word can be tokenized in multiple ways; the word "table", for instance, may be represented by the single token _table or split into the pair _ta ble, and in the latter case the model must internally reconstruct a single representation of the concept *table* from its fragments. This internal reconstruction, termed *detokenization* (Elhage et al., 2022; Gurnee et al., 2023), is a prerequisite for coherent downstream processing: when it fails, the model can lose access to the concept entirely. Detokenization is therefore a core internal computation that transformer language models must execute correctly on nearly every input. Tokenization artifacts have been shown to produce behavioral failures ranging from anomalous outputs on under-trained tokens (Land and Bartolo, 2024) to degraded performance on arithmetic and numerical reasoning (Singh and Strouse, 2024). These symptoms differ, but each one shows that tokenization can influence model behavior. This motivates us to look beyond behavioral effects and examine the internal computation: how do transformer language models compose subword fragments back into the word-level concepts they encode, and what determines whether this composition succeeds or fails for a given input?

Recent work has made substantial progress on understanding detokenization. Under the *stages-of-inference* framing, early layers of transformer language models are thought to detokenize subword input into an internal vocabulary on which later stages operate (Lad et al., 2024). Kaplan et al. (2025) provide the most comprehensive account to date, showing that word-level representations become decodable from the last subword position in early-to-middle layers, and describing a broad two-stage process: attention aggregates subword tokens, and feed-forward layers (MLP) retrieve a word-level representation. Complementary evidence comes from Feucht et al. (2024), who show that last-token positions of multi-token words "erase" information about preceding tokens in early layers. Kamoda et al. (2025) use a different method: an input-agnostic, weight-based

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 35230–35249
October 24-29, 2026 ©2026 Association for Computational Linguistics

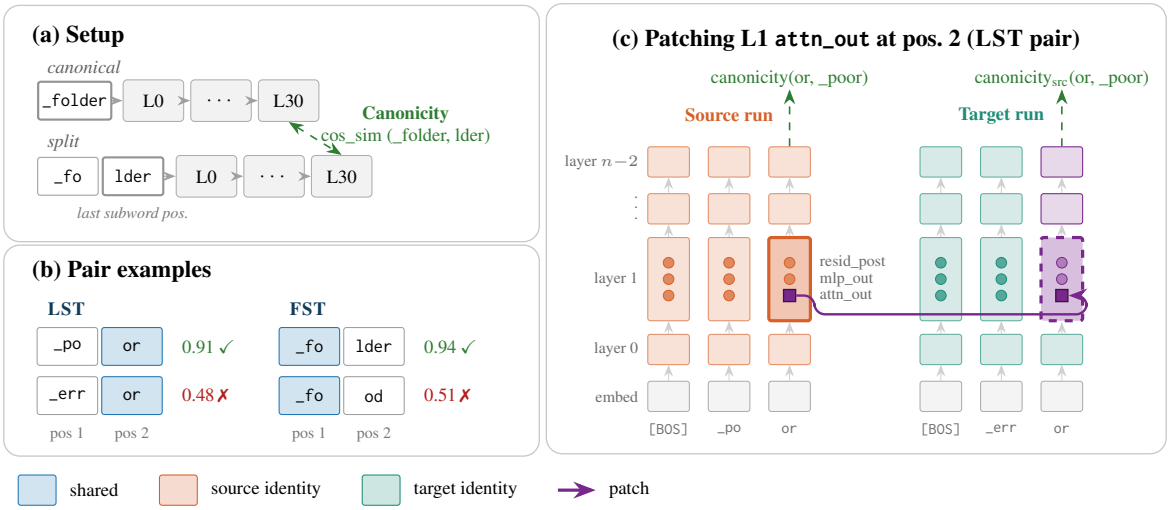


Figure 1: **Experimental design and activation patching.** **(a)** Each word the tokenizer renders as a single token is run twice: as the canonical single token and as an artificial split. *Canonicity* compares the layer-30 residual streams at the canonical position and the split's last subword position. **(b)** LST pairs share the second token; FST pairs share the first. Each pair has one successful and one failed split, with canonicity values shown. **(c)** Activation patching at L1 `attn_out`, position 2, for an LST pair. The source run [BOS, _po or] reconstructs _poor. Replacing the target's L1 `attn_out` at position 2—which normally aggregates the target context _err or—with the source's, then completing the forward pass unchanged, makes the target's L30 residual align with _poor (the *source's* word) rather than _error. A single L1 attention vector thus carries enough cross-position signal to make the target's run reconstruct the source's word instead of its own.

analysis of first-layer attention in GPT-2, which finds structure supporting detokenization. These studies establish that detokenization is a core process in transformer language models to which both attention and MLPs contribute. This understanding is already actionable: Mehta et al. (2026) use detokenization to guide vocabulary expansion for under-served scripts, deciding both which items to add and how to initialize their embeddings.

Existing evidence characterizes detokenization at the level of entire layers or broad component types, relying predominantly on correlational probing or coarse-grained interventions. Three questions remain open. First, within the critical early layers, which **specific** sub-components drive the process, and what signal do they transmit? Second, does the mechanism **generalize** across architectures and word lengths, and what determines how many layers it spans? Third, can the outcome of detokenization, i.e., whether it will succeed or fail for a given input, be **predicted** from the early-layer representation, and would such a prediction hold for natural text as well as for controlled inputs? We address these questions through activation patching combined with a controlled paired-dataset design that, by construction, isolates the causal contribution of attention from that of the MLP.

On English data, across twelve transformer language models, we find that Layer 1 implements a two-stage computation that initiates detokenization. First, a small subset of attention heads transmit a weak, input-dependent signal from the preceding token into the last subword position. Then, Layer 1's MLP composes this signal with the local subword embedding through a continuous transformation. The same division of labor holds across word lengths from two to six subword tokens, with composition still occurring at the last position and intermediate positions serving as relays. Detokenization concentrates in the first 6–30% of layers across all architectures, with the depth governed by the positional encoding strategy. The early-layer signal is linearly readable: a single direction fit at the depth identified by activation patching predicts final-layer detokenization success with AUROC 0.94 on isolated, contextless sequences, and 0.97 on natural text. Together, these results identify a specific, generalizable mechanism for detokenization, whose depth is governed by positional encoding rather than scale, and provide an early-layer linear predictor of detokenization success that transfers to natural text.[1]

[1]We release our code and data at https://github.com/MeLeLBGU/Word-Factory.

2 Setup

Many words that a tokenizer represents as a single token can also be split into multiple subword tokens. For example, the word "table" may be tokenized as a single token _table or split into _ta ble. When a word is split, the model must internally reconstruct the full-word meaning from its parts, a process known as *detokenization*. We study this process by leveraging words that admit both representations: a canonical single-token form and an artificially-split multi-token form. This dual representation provides a ground truth: we can compare what the model computes from the split input against the representation it produces for the intact word. Our analysis is therefore restricted to words that the tokenizer can represent as a single token; inherently multi-token words lack such a reference and are excluded. All experiments before section 4 consider only two-token splits.

Data. For each model, we collect a pool of *single-token words*: English words that exist in the model's tokenizer vocabulary as one token. Restricting to such words is necessary because our analysis compares a split tokenization of a word against its canonical single-token representation; inherently multi-token words allow no such reference. We draw candidate words from three publicly available English word lists: a frequency-ordered list of the 100,000 most common lowercased English words from the `top-english-wordlists` repository; the Google 10,000-word common English list; and the English word frequency dataset (Tatman, 2017), a unigram frequency table of approximately 333,000 entries derived from the Google Web Trillion Word Corpus. More information, as well as details about preprocessing, is available in Appendix A.

Notation. We adopt standard transformer-component notation throughout. We refer to the attention block output as `attn_out` and the MLP block output as `mlp_out`. `resid_pre` denotes the residual stream at the input of a layer, and `resid_post` the residual stream at its output, after both the attention and the MLP blocks have written to it. We index layers from zero, therefore `resid_pre` at Layer 0 is the token embedding alone, while `resid_post` at Layer 0 already contains the output of one complete transformer block. Unless otherwise stated, all activations are taken at the last subword position of the split word.

Model. All primary experiments use Llama 2–7B (Touvron et al., 2023) with 32 layers, $d_{\text{model}} = 4{,}096$, with a word pool of size 8,933. We discuss cross-architecture generalities in section 5.

2.1 Measuring Success of Detokenization

We introduce our primary success metric, *canonicity*, defined as the cosine similarity, at layer $n-2$ (layer 30 of 32 in Llama2-7B), between `resid_post` at the last subword position of the split input and the corresponding canonical single-token representation at the same layer and position (see Figure 1a). High canonicity indicates that the model has successfully reconstructed the full-word representation from fragments; a low value indicates failure. We read at layer $n-2$ because it is late enough for the computation to have largely stabilized, and early enough to precede the final layer, which reshapes the residual stream for next-token prediction (Belrose et al., 2023). A per-layer sweep supports the choice: the correlation between representational and behavioral similarity plateaus across the late layers in every model tested, reaching $\rho_{\text{KL}} = -0.86$ at layer $n-2$ on Llama2-7B, so $n-2$ represents the plateau instead of a tuned optimum. This choice of layer fixes *where* we read. It remains to show that *what* we read there tracks behavior. Canonicity is a strong, monotonic predictor of model behavior rather than a geometric artifact. Across our full word pool, split inputs in the top canonicity quintile match the canonical input's top-1 next-token prediction 78% of the time, versus 24% in the bottom quintile.

We demonstrate the cost of low canonicity using two splits of the same word. Both _ad vanced and _advance d spell "advanced", but only the former matches the canonical next-token distribution (KL 0.05) with predicted continuations such as "technology" and "manufact". The latter predicts "raft" and "iet", suggesting that the model reads the final subword d as the beginning of a new word. We provide both correlational and causal validation of canonicity, together with further qualitative examples, in Appendix B.

2.2 Experimental Design

The success of detokenization depends on both subword tokens. To isolate the contribution of each component, we construct controlled pairs of split words that share one token and differ on the other.

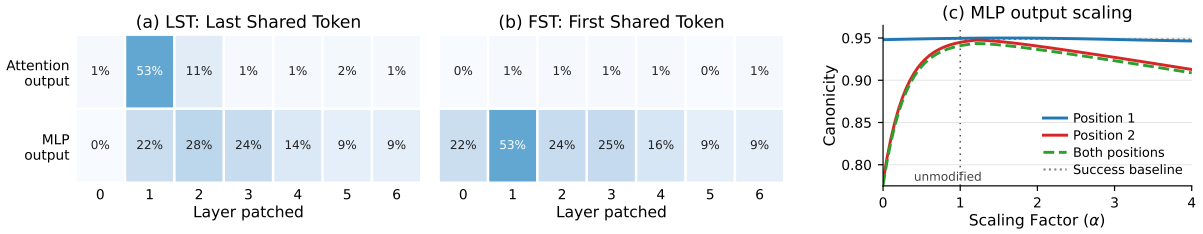


Figure 2: Activation patching at the second token. Values show the percentage of the canonicity$_{\text{src}}$ gap closed. **(a)** In LST, Layer 1 `attn_out` drives the effect. **(b)** In FST, attention is uninformative, and Layer 1 `mlp_out` dominates. **(c)** Scaling the Layer 1 `mlp_out` by a factor α reveals a smooth, continuous effect on detokenization quality. Scaling position 2 raises canonicity from 0.78 ($\alpha = 0$) to the source baseline ($\alpha = 1$); scaling position 1 has little effect.

Pair construction. Each pair consists of two split words that share exactly one token position. Within each pair, one member has high canonicity in our chosen model and one has low canonicity; we refer to these as the *successful* and *failed* members of the pair, respectively, and denote the difference in their canonicity as the *success–failure gap*. We define two complementary settings (Figure 1b):

- **First-Shared-Token (FST):** pairs share the first token and differ in the second. For example, the split _fo lder reconstructs the representation of _folder well (canonicity 0.94, *successful*), while the split _fo od reconstructs the representation of _food poorly (canonicity 0.51, *failed*). The pair members share the first token _fo.
- **Last-Shared-Token (LST):** pairs share the second token and differ in the first. For example, the split _po or reconstructs _poor well (canonicity of 0.91, *successful*), while the split _err or reconstructs _error poorly (canonicity of 0.48, *failed*). The pair members share the second token or.

The experimental value of this design lies in what each setting holds constant. In FST, the first token, and therefore the value vector at position 1, is identical across the pair, so the cross-position contribution that any attention head can read from the first token is approximately matched between runs. Any difference in outcome must therefore arise from how subsequent components, including the MLP, process the differing local embedding at the last position. In LST, the local embedding at the last position is identical across the pair, so the two members are matched in everything an MLP receives from the local position; any difference must arise from cross-position information that attention transmits from the differing first token. This complementary structure separates the contributions of attention and MLP, and is the foundation of the analysis in section 3.

Pairs are constructed by requiring a minimum success–failure gap in canonicity between the two members. Because canonicity spreads vary across architectures (Ethayarajh, 2019), we calibrate this threshold per model to ensure a sufficient number of pairs for each dataset. For Llama2-7B (gap $\geq$ 0.30), this yields $N = 6{,}409$ LST and $N = 6{,}254$ FST pairs; per-model thresholds and counts are listed in Appendix A.

2.3 Activation Patching

Throughout the paper, we use activation patching (Vig et al., 2020; Geiger et al., 2021; Meng et al., 2022) to localize the components responsible for detokenization. Given a pair from the dataset, we patch a chosen component's activation from the *source* (successful member) into the *target* (failed member), complete the target's forward pass, and measure whether the source's canonical representation now emerges in the target (Figure 1c). A high score indicates that the patched component carries the composed concept, or information used in its composition. We measure this as *canonicity*$_{src}$: the layer-$n-2$ cosine similarity between the patched target and the *source*'s canonical representation. We report the fraction of the gap that the patch closes between the target baseline (canonicity$_{\text{src}}$ of the unpatched target run) and the source baseline (the source's standard canonicity). 0% means the patch had no effect; 100% means the patched target reaches the source baseline, indicating that the source concept has been fully reconstructed in the target.

3 The Two-Stage Mechanism

We now turn to the core mechanistic question: how does the model compose two subword tokens into a single semantic representation? All experiments in this section use Llama2-7B on two-token words from the LST and FST pair sets. The unpatched source and target baselines (canonicity$_{\text{src}}$) are approximately 0.95 and 0.43, respectively, in both pair sets. We will refer to the tokens and their representations by their position, 1 and 2.

3.1 Activation Patching Localizes the Mechanism to Layer 1

To localize where information from position 1 first reaches position 2, we perform activation patching in the LST setting. For each layer independently, we patch a single component's activation at position 2 from source into target. Figure 2a,b reports the fraction of the gap closed. Prior to Layer 1, patching any component has no effect: position 2 contains only the local token embedding and has not yet received information from position 1. Patching the Layer 1 attention output, however, closes 53% of the gap, the largest single-component effect at any layer; later layers yield minimal improvement. Layer 1 attention is thus the first and primary mechanism by which information from the first token reaches position 2.

This effect is specific to detokenization, not a generic property of early-layer attention. The same intervention, applied to ordinary two-word sequences that share a last word and require no composition (e.g., the cat vs. a cat), closes 6.2% of the gap, against 50.8% for split-word pairs under a metric adapted to both conditions (Appendix C). Layer 1 attention therefore transmits cross-position information in ordinary sequences too, but its causal contribution is about eight times larger during detokenization.

The MLP's contribution is smaller and more distributed, peaking at Layer 2 (28% gap closed) and declining gradually thereafter. No single MLP layer matches the L1 attention effect. However, since the MLPs read from a residual stream that L1 attention has already written to, LST alone cannot tell us whether the MLP plays a distinct role in the process or merely passes along the attention signal. We turn to FST to find out.

3.2 MLP and Attention Have Distinct Roles

In FST, source and target share the first token, so the value vectors at position 1 are identical by construction. The Layer 1 attention output at position 2 is not: the differing second-token embedding can change the query and the attention pattern there. We therefore measure it directly. Across all 6,254 FST pairs, the within-pair cosine similarity of L1 `attn_out` at position 2 averages 0.95, against 0.74 for words with a different first token (Appendix D). The intervention that drove most of the effect in LST therefore has almost nothing to transfer, and indeed patching L1 `attn_out` in FST closes only 1% of the gap. The reverse intervention is sharper: injecting a target run's `attn_out` into a source run barely degrades performance (canonicity$_{\text{src}}$ 0.92 vs. 0.94). The L1 attention output is genuinely interchangeable across the pair when the first token is the same; it carries no information that distinguishes successful from failed composition. What does distinguish them is the MLP. Patching L1 `mlp_out` closes 53% of the gap, more than it closed in LST, and the only intervention that produces meaningful reconstruction in this setting. Because the pre-MLP residual stream differs between the runs only via the second-token embedding, the MLP's nonlinear transformation is what turns this local input into a successful (or failed) reconstruction. Together, these results establish a two-stage process: attention transmits cross-position information, and the MLP composes it with the local embedding.

3.3 Attention as a Directional Nudge

We established that L1 attention transmits cross-position information from the first token, but not what that information is. We decompose the L1 attention output to find out.

The signal flows through a few heads, via values at position 1. Patching individual per-head, pre-projection output vectors $\mathbf{z}^{(h)}$ localizes the effect to three heads: head 27 alone closes 17% of the gap, and heads 24 and 28 a further 7% when patched together. The three together close 39% of the gap, indicating a super-additive interaction in which the heads act jointly rather than independently. This recovers most of L1 `attn_out`'s 53% effect, with the remainder spread thinly across the other heads. Component-wise, the effect is carried almost entirely by *value* vectors at position 1; patching queries, keys, or attention patterns pro-

duces negligible change.

The signal points toward the canonical embedding, but only weakly. These three dominant heads write vectors weakly but consistently aligned with the canonical embedding direction (mean cosine similarity 0.053, vs. −0.01 for the other heads), and their combined projection moderately predicts detokenization success (Spearman $\rho = 0.52$). Layer 1 attention thus writes a small, input-dependent nudge toward the canonical embedding.

The signal is token-specific, not a generic success flag. A natural concern is that the L1 attention output might encode a generic successful composition flag. The FST results rule this out: there, source and target share the first token, so L1 attention writes nearly the same vector in both runs, yet they diverge in composition success. Whatever attention writes is therefore tied to the first-token identity, not to the success of the eventual composition. Together, these results refine the two-stage picture: L1 attention does not encode the composed concept. Rather, it writes a small, token-specific signal that the MLP uses to assemble it.

3.4 The MLP Composes Continuously

Having characterized the attention signal, we turn to the next stage: how does the Layer 1 MLP use this signal to compose the full-word representation? Unlike the patching experiments, the interventions described below modify a single run instead of transferring activations between runs; we therefore report raw canonicity, not gap-closed percentages.

Necessity. Zero-ablating L1 `mlp_out` at position 2 drops canonicity from 0.95 to 0.78, the single largest MLP ablation effect at any layer; ablating L0 `mlp_out` causes comparable damage. Beyond Layer 3, zero-ablation has negligible impact (canonicity > 0.93); the critical MLP computation is concentrated in the earliest layers.

Continuous operation. To test whether the MLP functions as a discrete gate or a continuous transformation, we scale L1 `mlp_out` by a factor $\alpha \in [0, 4]$ (Figure 2c). While scaling only position 1 has virtually no effect, scaling position 2, or both positions jointly, produces a smooth response: canonicity rises steadily from 0.78 at $\alpha = 0$ to 0.95 at $\alpha = 1$, then declines. The absence of any threshold, plateau, or step, rules out a discrete key-value retrieval account of the per-input computation (Geva et al., 2021, 2022; Kaplan et al., 2025).

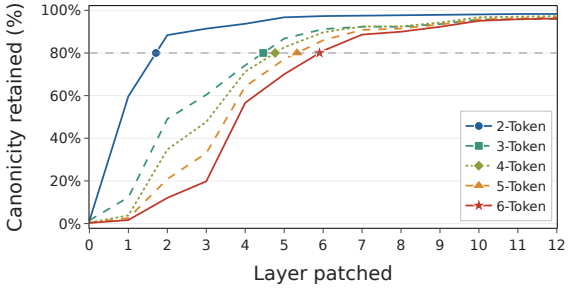


Figure 3: Fraction of the canonicity$_{\text{src}}$ gap closed when patching `resid_post` at the last position from a successful run into a failed one, layer by layer, in the first-token-differs configuration. The 2→3 transition adds about three layers; subsequent token additions cost less than one layer each.

Together with the attentional findings (§3.3), this gives a complete picture of the Layer 1 mechanism: attention writes a small, token-specific directional signal, and the MLP applies a continuous transformation that turns this signal plus the local embedding into the canonical representation.

4 Scaling with Token Count

We have established that for two-token words, the two-stage mechanism of detokenization executes primarily at Layer 1. Words split into more tokens require more layers in practice; we ask why. We extend the controlled-pairs dataset to k-token splits ($k \in [2, 6]$), focusing on the most demanding configuration, where the only differing token is the first subword,[2] and examine how the number of required layers grows with token count, and what the mechanism behind that growth is. These are still in-vocabulary words, split artificially into k tokens, so canonicity is computed against the same canonical reference as before.

4.1 Detokenization Depth Grows Sublinearly with Token Count

To measure how many layers detokenization requires at each word length, we patch `resid_post` at the last position one layer at a time. A high value at layer l means the residual stream after l already carries enough information to recover the canonical representation; a low value means subsequent layers are still required.

[2] An example for $k = 4$ would be _ant | ic | ipa | tion and _part | ic | ipa | tion.

Figure 3 shows the gap-closed curves. Detokenization is shallow at every word length: 80% gap-closed is reached by Layer 2 for two-token words and by Layer 6 for 6-token words—19% of the network in the deepest case. Growth is non-uniform: the transition from two- to three-token words adds about three layers to reach the 80% threshold, while every subsequent token requires less than one.

Word length is not the only factor. In three-token splits where the two members share the first and last tokens and differ only in the middle one, 80% gap-closed is reached by Layer 3, compared to Layer 5 when the first token differs (Appendix E). The distance from the differing token to the last position therefore sets the depth as well.

Both findings suggest that intermediate positions contribute to the computation rather than passively passing the residual stream toward the last position.

4.2 Intermediate Positions Relay First-Token Information

To test whether intermediate positions contribute actively to detokenization, we reverse the patching direction: we corrupt the successful run by replacing intermediate positions' `resid_post` with the failed member's activations at the same position and layer. Because the corrupted positions hold the same tokens in both members, any drop in canonicity at the last position must come from cross-position information those positions read via attention, a direct measurement of the relay. We report canonicity at the last position as a percentage of the successful member's uncorrupted baseline.

Each intermediate position relays in a fixed window. Figure 4 (left) shows per-position corruption curves for 5-token words. Each intermediate position contributes a drop concentrated in a 2–3 layer window, and these windows shift by roughly one layer per position, consistent with sequential propagation along the sequence. The same pattern appears in 3- and 4-token words: a position's relay timing is determined by its distance from position 1, not by total word length (see subsection E.2).

The collective relay contribution grows with word length. Although each individual relay carries less signal in longer words, the collective contribution grows: corrupting all intermediate positions simultaneously produces peak drops at Layer 2: 9% in 3-token words, rising to 21% in 5-token words (Figure 4, right). The collective drop also exceeds the sum of individual drops by an increasing factor (1.4× for 4-token, 1.6× for 5-token): each relay partially compensates when the others are intact, and this redundancy grows with the number of relay positions available.

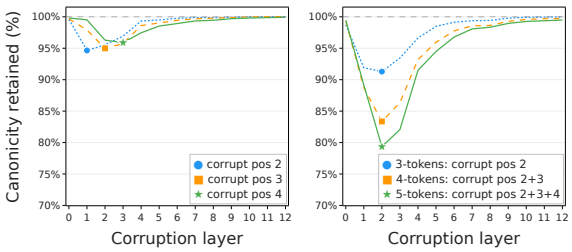


Figure 4: Canonicity at the last position after corrupting intermediate positions, relative to the same word's uncorrupted run; a larger drop indicates that the position relayed more detokenization signal. **Left:** 5-token words, one position corrupted at a time; each position relays in a 2–3 layer window that shifts one layer deeper per position. **Right:** All intermediate positions corrupted jointly. The drop deepens with word length and exceeds the sum of single-position drops (1.4–1.6×), indicating partial redundancy.

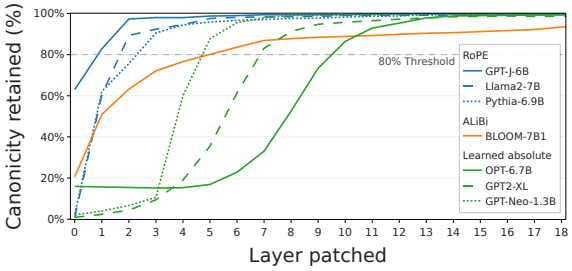


Figure 5: Canonicity retained at the last subword position when patching `resid_post` from a successful run into a failed one, in the LST setting. RoPE models (blue) reach high canonicity within the first 1–5 layers; learned-absolute models (green) require 5–10 layers, accumulating the same effect gradually. ALiBi (BLOOM-7B1, orange) is intermediate.

The pattern of decreasing per-position and increasing collective contribution matches the shrinking marginal depth cost: as word length grows, intermediate positions provide parallel relay paths that limit how much each additional token costs.

5 Cross-Architecture Generalization

Does the two-stage mechanism reflect something about Llama2-7B specifically, or about how transformers compose subword tokens in general? We extend our analysis to twelve models spanning eight families, three positional-encoding schemes

(RoPE (Su et al., 2021), ALiBi (Press et al., 2022), learned absolute (Radford et al., 2019)), and widths from 1,024 to 4,096 (Appendix F). We find that the two-stage process occurs in every model: attention transmits and MLP composes. However, the depth at which it operates follows one of two distinct regimes based on a single architectural variable: *positional encoding*.

The two-stage mechanism is universal. We test each of the two stages separately across the twelve models (Appendix G). For attention, we patch `resid_post` at position 1 from a successful run into a failed one, in the LST setting; the layer at which this patch stops being effective marks when attention at position 2 has finished reading from position 1. In every model, patching position 1 closes most of the gap at the input layer, with effectiveness decaying over some early-layer window. For the MLP, we patch `mlp_out` from a failed run into a successful one and measure the resulting damage to canonicity; damage at a layer indicates the MLP is producing composition-specific output, not just generic local processing. In every model, this damage is localized to a small set of layers adjacent to the attention-read window. Both stages identified in section 3 replicate in all twelve models.

The layer-count splits by positional encoding. Despite this shared structure, models differ sharply in how many layers detokenization occupies (Figure 5). In the *concentrated* regime, 80% of the canonicity$_{\text{src}}$ gap is closed within 1–5 layers; position 1's stream stops being essential by Layer 1–2, and MLP composition damage peaks at Layers 1–4. In the *distributed* regime, 80% gap-closed requires 5–10 layers; position 1 remains essential through Layers 5–8, and MLP damage peaks at mid-network. Positional encoding alone partitions the models: every RoPE model is concentrated and every learned-PE model is distributed, while width, depth, parameter count, tokenizer, MLP activation, and training corpus all cross the regime boundary (Appendix F).

BLOOM-7B1 (ALiBi) sits between these two poles: its processing is concentrated for 2-token words, but gradually transitions to a distributed profile as token count scales up (Appendix H). Since ALiBi and RoPE both provide relative-position access in attention, the property that aligns with the split appears to be relative-position access in general rather than rotary encoding specifically.

We hypothesize that this division exists because a relative-PE head at Layer 1 can attend from position 2 to position 1 with sharp selectivity, executing the cross-position write in one step, whereas learned-PE attention must first recover relative position from absolute coordinates, spreading the same effect over several layers; this account fits all twelve models but is untested, and we leave verification to future work.

6 Early Layers Predict Success

We established, via causal intervention, that detokenization executes primarily in the early layers of every architecture tested. This entails a falsifiable prediction: at a given depth, the residual stream should already linearly encode whether detokenization will succeed.

We test this with a class-mean-difference probe (Rimsky et al., 2024; Marks and Tegmark, 2024). We rank split words by canonicity, and take the top and bottom quintiles as *successful* and *failed* classes (discarding the middle 60% to sharpen the contrast). We then fit

$$\hat{d} = \frac{\mu_{\text{successful}} - \mu_{\text{failed}}}{\|\mu_{\text{successful}} - \mu_{\text{failed}}\|}$$

on `resid_post` at the final subword position in layer l^*, set to the gap-closed-80% depth fixed by the mechanism (Appendix H). Held-out activations are scored by projection onto $\hat{d}$; we report AUROC under 5-fold STRATIFIEDGROUPKFOLD (Widodo et al., 2022) grouped by canonical word.

We run this pipeline in two settings: **Isolated** uses the controlled BOS t1 … tk setup as before. **In-context** embeds each split word into a WikiText-103 (Merity et al., 2017) sentence and recomputes canonicity per ⟨word, sentence⟩ record. Within each setting, the probe is fit and evaluated on that setting's activations.

The signal is linearly readable, in isolation and in context. On Llama2-7B at l^*=L2, Isolated achieves AUROC 0.94 and In-context 0.97 (Figure 6). The probe collapses to chance under label shuffling (AUROC 0.49–0.52), confirming the signal is not an artifact of the fitting procedure. The early-layer geometry thus separates both outcomes.

The isolated direction transfers to natural text. To test if the isolated probe direction, $\hat{d}_{\text{iso}}$, generalizes to natural text, we score it on in-context activations extracted from WikiText sentences. To avoid lexical leakage, we apply an 80/20 train-test split to

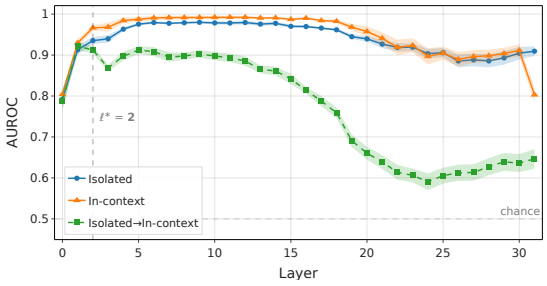


Figure 6: Layerwise probe AUROC on Llama2-7B (k=2). Class-mean-difference probes fit on isolated activations (**Isolated**), in-context activations (**In-context**), or fit on isolated and applied to in-context activations of held-out words (**Isolated**→**In-context**). Vertical line: l^*=L_2, the gap-closed-80% depth (see Table 6).

our vocabulary, stratified by isolated-setting quintiles. We fit $\hat{d}_{\text{iso}}$ strictly on the isolated activations of the training words, and evaluate its AUROC on the in-context activations of the held-out words. On Llama2-7B at l^*=L2, $\hat{d}_{\text{iso}}$ achieves AUROC 0.91 on these in-context activations, within 0.03 of its own isolated test-set AUROC (0.94). At l^*, context from preceding tokens has not yet been fully aggregated into the last subword position, so its residual still resembles the isolated counterpart.

The two directions align at l^* and diverge as context accumulates. At l^*, applying $\hat{d}_{\text{iso}}$ to in-context activations matches its own isolated test AUROC: across all 12 architectures, the gap is at most 0.04 and the median is 0.004 (Appendix I). However, this alignment decays in deeper layers. As context aggregates into the last subword position, the in-context residual encodes outcomes in a way that $\hat{d}_{\text{iso}}$ no longer captures, causing its AUROC to drop well below the In-context probe. This is consistent with the mechanism: $\hat{d}_{\text{iso}}$ is fit where the last position carries composition alone, which is no longer the sole signal once context accumulates.

Cross-architecture replication. The same construction yields Isolated AUROC 0.76–0.94 and In-context 0.86–0.99 at l^* across all 12 models (Appendix I). The two regimes of §5 appear in the layer-by-layer profile: concentrated-regime models reach peak AUROC within the first 10–15% of layers; distributed-regime models build up gradually over the first third. Causal intervention and correlational probing thus identify the same critical depth l^*, supporting a single underlying mechanism.

7 Conclusion

We characterized detokenization in transformer language models as an early-layer two-stage mechanism: attention writes a token-specific directional signal from preceding subwords, and the MLP composes it with the local embedding. The mechanism is consistent across twelve models from eight families and extends to longer words via sequential relays at intermediate positions, with its depth governed by positional encoding.

Beyond mapping the phenomenon, we established that the detokenization outcome is linearly readable from early layers. This opens the door to practical, inference-time interventions, such as flagging tokenization-induced failures before text generation concludes. Furthermore, our findings suggest that early-layer activations can be leveraged as an intrinsic metric for tokenizer evaluation, offering a clean, mechanistic alternative to behavioral criteria that inherently conflate tokenizer and model performance.

Limitations

Canonicity measures alignment to a model-internal reference. Our success metric quantifies how closely a split word's hidden state aligns with the model's own hidden state for the canonical single-token form of the same word (§2.1). This reference is itself a representation the model computes, not an external ground truth for the word's meaning. A split input whose hidden state diverges from the canonical could in principle still support coherent downstream computation through a different geometry, and would be scored as a partial failure under our metric. Consistent with this, the behavioral agreement in §2.1 is strong and monotonic, but not absolute. The mechanism we identify is therefore the one that produces canonical-aligned representations. Whether transformers also implement alternative pathways that produce a word-level representation through different geometry remains outside the scope of our evidence.

Restriction to single-token words. Our paired design requires a canonical single token form to serve as the reference, which forces us to study only words the tokenizer can represent as a single token, artificially split into multi-token sequences. The words for which detokenization arguably matters most behaviorally (rare words, technical terms, and proper names that the tokenizer represents as two or

more tokens) have no single-token counterpart and therefore no comparable ground truth available to our method. Whether the two-stage mechanism we identify governs these naturally-multi-token words is an open question we cannot address with the present methodology, and would require either a different reference signal or an alternative behavioral criterion for detokenization success.

Acknowledgments

We thank Sarah Wiegreffe and Jacob Eisenstein, as well as the anonymous reviewers, for valuable comments. This research was supported by the Israel Science Foundation (grant No. 1166/23) and by the Frenkel Foundation and the Stein Faculty of Computer Science and Information Systems.

References

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from Transformers with the Tuned Lens. *Preprint*, arXiv:2303.08112.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430.

BigScience Workshop. 2022. BLOOM: A 176B-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large scale autoregressive language modeling with Mesh-Tensorflow.

Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Andy Jones, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, and 14 others. 2022. Softmax linear units. *Transformer Circuits Thread*.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Sheridan Feucht, David Atkinson, Byron C Wallace, and David Bau. 2024. Token erasure as a footprint of implicit vocabulary items in LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9727–9739, Miami, Florida, USA. Association for Computational Linguistics.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, volume 34.

Gemma Team. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*.

Go Kamoda, Benjamin Heinzerling, Tatsuro Inaba, Keito Kudo, Keisuke Sakaguchi, and Kentaro Inui. 2025. Weight-based analysis of detokenization in language models: Understanding the first stage of inference without inference. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6339–6358, Albuquerque, New Mexico. Association for Computational Linguistics.

Guy Kaplan, Matanel Oren, Yuval Reif, and Roy Schwartz. 2025. From tokens to words: On the inner lexicon of LLMs. In *The Thirteenth International Conference on Learning Representations*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. 2024. The remarkable robustness of LLMs: Stages of inference? *arXiv preprint arXiv:2406.19384*.

Sander Land and Max Bartolo. 2024. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11631–11646, Miami, Florida, USA. Association for Computational Linguistics.

Samuel Marks and Max Tegmark. 2024. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *Proceedings of the First Conference on Language Modeling (COLM)*.

Maitrey Mehta, Nishant Subramani, Zhichao Xu, Ashim Gupta, and Vivek Srikumar. 2026. Defragmenting language models: An interpretability-based approach for vocabulary expansion. *arXiv preprint arXiv:2604.16656*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of the 5th International Conference on Learning Representations*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

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

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics.

Craig W Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. 2024. Tokenization is more than compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 678–702, Miami, Florida, USA. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Aaditya K. Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier LLMs. *arXiv preprint arXiv:2402.14903*.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2021. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

Rachael Tatman. 2017. English word frequency. `https://www.kaggle.com/datasets/rtatman/english-word-frequency`. Derived from the Google Web Trillion Word Corpus.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 billion parameter autoregressive language model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Slamet Widodo, Herlambang Brawijaya, and Samudi Samudi. 2022. Stratified k-fold cross validation optimization on machine learning for prediction. *Sinkron: jurnal dan penelitian teknik informatika*, 6(4):2407–2414.

Susan Zhang, Stephen Roller, Naman Goyal, and 1 others. 2022. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A Data Sources and Single-Token Word Pools

All experiments operate on a per-model pool of *single-token words*: English words that the model's tokenizer encodes as exactly one token (excluding any special tokens). Restricting to such words is necessary because our analysis compares a split tokenization of a word against its canonical single-token representation (subsection 2.1); inherently multi-token words admit no such reference.

Raw word lists. We draw candidate words from three publicly available English word lists:

- A frequency-ordered list of the 100,000 most common lowercased English words from the `top-english-wordlists` repository.[3]
- The Google 10,000-word common English list.[4]
- The English word frequency dataset of Tatman (2017), a unigram frequency table of approximately 333,000 (word, count) entries derived from the Google Web Trillion Word Corpus.[5]

For both the contrastive-pair datasets used in activation patching (section 3 onward) and the behavioral readout-layer validation (subsection 2.1), we draw on all three lists. We concatenate the three lists and merge them into a single set of candidate words. The merged words are lowercased and deduplicated, yielding a combined candidate pool of 338,850 unique words. Each candidate is then tokenized with the model's own tokenizer, and only words encoded as exactly one content token are retained, giving the per-model pool.

Per-model single-token filtering. Because tokenizer vocabularies differ substantially across architectures, the resulting pool size varies: from roughly 7.5k words for GPT-2–family models to roughly 28k for Gemma-2-2B (SentencePiece Unigram). Pool sizes per model and the resulting number of contrastive pairs are listed in Table 1.

[3] https://github.com/david47k/top-english-wordlists

[4] https://github.com/first20hours/google-10000-english

[5] https://www.kaggle.com/datasets/rtatman/english-word-frequency

Model	Pool	Threshold	2-token pairs
GPT-2 Large	7,521	0.15	4,418
GPT-2 XL	7,521	0.20	4,918
GPT-Neo 1.3B	7,521	0.05	4,407
GPT-J 6B	7,521	0.15	4,791
OPT 1.3B	7,521	0.20	12,376
OPT 6.7B	7,521	0.20	11,207
Pythia 410M	8,651	0.15	8,983
Pythia 1B	8,651	0.15	7,997
Pythia 6.9B	8,651	0.10	7,499
Llama-2-7B	8,933	0.30	12,663
Gemma-2-2B	28,006	0.30	9,954
BLOOM-7B1	8,933	0.30	9,961

Table 1: Per-model dataset sizes. **Pool**: unique words that the model's tokenizer encodes as exactly one token, after merging the three source lists, lowercasing, and deduplication. **Threshold**: per-model minimum metric gap $|\Delta canonicity|$ required to retain a pair, calibrated to cross-architecture variation in cosine-similarity spread (Ethayarajh, 2019). **2-token pairs**: contrastive pairs used in activation patching, summed across the LST and FST settings.

B Choosing the Readout Layer and Validating Canonicity

This appendix supports two claims from §2.1: that layer $n-2$ is the right readout depth, and that canonicity at that depth tracks model behavior.

Setup. For each model we use the flat pool of all valid two-token segmentations of its single-token words, built from the per-model pool described in Appendix A. For each segmentation we run two forward passes, one on the split input and one on the canonical input. At every layer L we record the cosine similarity between their last-position residual streams. At the last token position of each input we also record the next-token distribution and compute three behavioral comparisons between the split and canonical predictions: $\mathrm{KL}(p_{\text{canon}} \,\|\, p_{\text{split}})$, top-1 agreement (whether the two inputs share the same argmax next token), and top-5 overlap. At each layer we then compute the Spearman correlation across all rows between the cosine similarity and each behavioral metric.

Layer selection. Figure 7 shows the three per-layer correlations across all 12 architectures. In every model, all three correlations rise with depth and peak near layer $n-2$, then drop sharply at the final layer due to next-token-prediction reshaping (Belrose et al., 2023). On Llama2-7B, layer 30 yields $\rho_{\text{KL}} = -0.86$, $\rho_{\text{top-1}} = +0.45$, and $\rho_{\text{top-5}} = +0.71$, ranking first among all 32 lay-

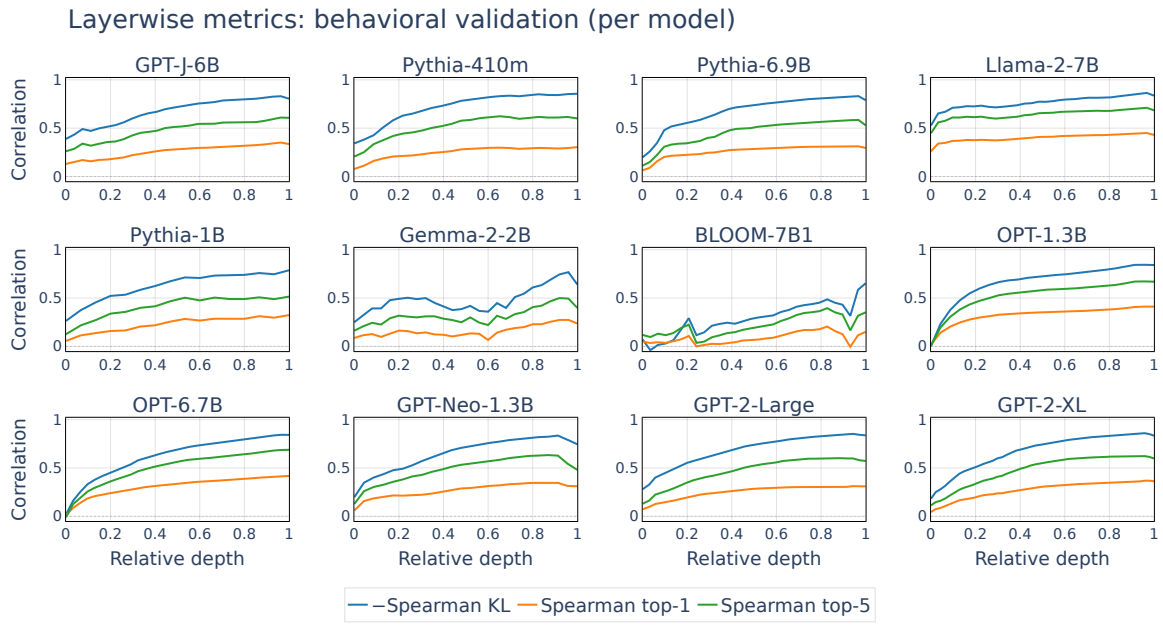


Figure 7: Per-layer Spearman correlation between cosine similarity to the canonical representation and three next-token behavioral metrics—KL divergence (blue), top-1 agreement (orange), and top-5 overlap (green)—across 12 architectures. The x-axis is relative depth; KL correlation is plotted as $-\rho_{\mathrm{KL}}$ so that higher is better on all three curves. Across every model, all three correlations rise with depth and peak near relative depth ~ 0.95 (the $n-2$ layer used as the measurement point in §2.1), then drop at the final layer due to next-token-prediction reshaping. BLOOM-7B1 shows noisier curves but the same overall shape.

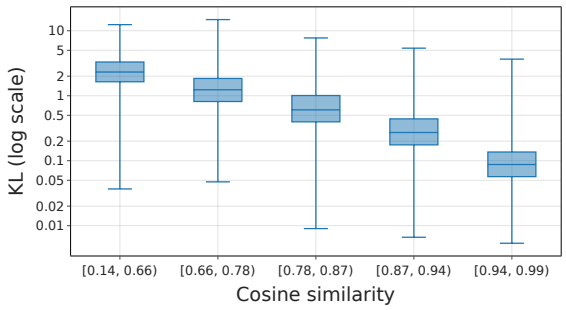


Figure 8: Next-token KL divergence between split-input and canonical-input distributions, binned by canonicity (y-axis log scale). Median KL falls monotonically across bins, dropping roughly $25\times$ from the lowest to the highest canonicity bin.

ers. For the remaining models, $n-2$ ranks first or within a negligible margin of the top-ranked layer, consistently under all three behavioral metrics. We therefore adopt $L = n-2$ as the readout layer throughout the paper.

Behavioral validation. Having fixed the readout layer, we verify that canonicity at that layer predicts behavior rather than reflecting a geometric artifact. We rank words by canonicity and stratify them into quintiles. Canonicity is a strong, monotonic predictor of behavioral agreement: words in the top quintile (mean canonicity 0.97) reach 78% top-1 agreement with the canonical input and a KL divergence of 0.09, while words in the bottom quintile (mean canonicity 0.65) reach only 24% top-1 agreement and a KL divergence of 1.86. Figure 8 shows the full relationship: median KL falls roughly $25\times$ from the bottom to the top of the canonicity range, and the trend is monotonic across all bins.

Qualitative examples. §2.1 gives one example of what a low canonicity value costs. Table 2 gives five more. Each block holds one word, its canonical single-token input, and two splits of that word, one with high canonicity and one with low. Word identity is held constant, so any difference in behavior comes from the segmentation alone. The high-canonicity split reproduces the canonical continuations in every case, while the low-canonicity split shares no token with the canonical top-5 prediction in four of the five. The failed predictions are not noise. In several cases the continuation completes the final subword into a different word: n becomes "nights" for _show n and "nation" for _media n, just as d becomes "draft" and "diet" for _advance d in §2.1. A low canonicity

value therefore means that the model reads the last subword as the start of a new word, rather than as the end of the intended one.

B.1 Causal Behavioral Validation

The validation above is observational: canonicity correlates with next-token behavior across words. Here we confirm the same relationship causally. We take the LST L1 `attn_out` patch and the FST L1 `mlp_out` patch, the interventions that localize the two stages of the mechanism in §3, and ask whether they move model behavior and not only canonicity.

Rather than scoring canonicity, we score the patched run's next-token distribution against the canonical single-token run, using top-1 agreement, top-5 overlap, and $\mathrm{KL}(p_{\text{canon}} \,\|\, \cdot)$. As in §2.3, we report the fraction of the gap closed from the failed split to the successful split: 0% is the unpatched $\text{canonicity}_{\text{src}}$ score of failed split, and 100% is the unpatched canonicity score of the successful split. As a control (RANDOM), we run the same intervention but take the patched activation from an unrelated word at the same component, layer, and position, rather than from the pair's successful member. For example, in LST the failed split [_err][or] is normally patched at L1 `attn_out`, position 2, with the activation of its successful partner [_po][or]; the RANDOM control instead patches the same site with the L1 `attn_out` activation, at its own last position, of an unrelated word such as [_ta][ble].

Table 3 shows that both interventions shift every behavioral metric from the failed split toward the successful one. The L1 `attn_out` patch raises top-1 agreement with the canonical prediction from 3.5% to 27.6%, closing 42% of the gap against only 1% for the RANDOM control, and the L1 `mlp_out` patch reproduces this (1.9% to 26.8%, 40% gap closed against 2%). Across metrics the `attn_out` patch closes 42% of the top-1 gap and 46% of top-5, with KL closing 60%; the $\text{canonicity}_{\text{src}}$ column produces the 53%–the gap stated in §3. The behavioral effect is thus of the same order as the representational one, substantial on every metric, confirming that the intervention moves model behavior, not only canonicity. This is a one-time check on the metric; all other results in the paper are reported in canonicity alone.

C Specificity to Detokenization

Layer 1 attention reads from the preceding position in ordinary text as well as in split words. This appendix asks whether the effect of §3.1 is specific to detokenization, or a generic property of early-layer attention.

We construct 1,000 sequences of the form [BOS][w1][w2], where both words are single-token English words, and pair them so that the two members share [w2] and differ in [w1] (e.g., [BOS][the][cat] vs. [BOS][a][cat]). This is the structural analogue of our LST setting (`_po|or` vs. `_err|or`), with two words in place of two subwords of one word. We then apply the identical intervention: patching Layer 1 `attn_out` at position 2 from the source run into the target run.

A two-word sequence has no canonical single-token reference, so canonicity is undefined for it. We therefore score both conditions by alignment to the source run's own layer-30 representation at the last position. Under this metric the intervention closes 50.8% of the gap for artificially split words, and 6.2% for ordinary two-word sequences.

Layer 1 attention therefore takes part in general cross-token processing, but its causal contribution to the downstream representation is roughly eight times larger during detokenization.

D Attention Output Similarity in FST Pairs

§3.2 states that Layer 1 `attn_out` at position 2 is nearly identical within an FST pair. This does not follow from the pair construction. Sharing the first token fixes the value vector at position 1, but the second-token embedding differs, so the query at position 2, and therefore the attention pattern, may differ as well. We measure the resulting similarity directly.

For each of the 6,254 FST pairs we compute the cosine similarity between the source and target L1 `attn_out` at position 2. As a baseline we repeat the computation against words with a different first token. Table 4 reports both, for the full attention output and for the three heads that carry the detokenization signal (§3.3).

The per-head numbers show what drives the similarity. The baseline falls from 0.739 for the full output to 0.144–0.231 per head. Within a pair, by contrast, the three heads stay at 0.834–0.998. The high within-pair similarity is therefore a consequence of the shared first token, not of a direction

Input	canonicity	KL↓	top-5	Top-3 next tokens (%)
_corpor	—	—	—	ate 82.9 | ation 6.0 | ations 4.1
_cor por	0.95	0.01	1.00	ate 84.9 | ation 6.1 | ations 2.6
_corpo r	0.33	9.54	0.00	ison 9.8 | ub 7.6 | re 4.6
_shown	—	—	—	in 21.1 | on 8.8 | below 8.2
_sh own	0.96	0.06	0.80	in 19.6 | on 7.2 | below 7.2
_show n	0.29	4.51	0.00	ights 4.1 | roll 3.2 | eth 2.7
_median	—	—	—	household 10.6 | income 6.1 | sal 5.7
_med ian	0.92	0.31	0.40	et 8.5 | income 5.2 | of 3.8
_media n	0.26	5.85	0.00	ation 15.9 | in 4.6 | ama 3.3
_gender	—	—	—	, 10.8 | and 6.5 | equality 6.1
_g ender	0.98	0.05	1.00	, 7.0 | - 6.6 | equality 5.1
_ge nder	0.36	3.85	0.20	8 8.3 | k 6.1 | - 4.5
_cried	—	—	—	out 13.4 | , 10.4 | the 10.4
_c ried	0.92	0.27	0.80	out 14.8 | , 6.6 | the 4.0
_cr ied	0.29	4.26	0.00	ie 41.8 | it 3.9 | e 3.0

Table 2: Next-token behavior for two splits of the same word, scored against the canonical single-token input. Each block gives the canonical input, the highest-canonicity split, and the lowest-canonicity split. KL is $\mathrm{KL}(p_{\text{canon}}\|\cdot)$, and top-5 is the overlap with the canonical top-5 prediction. The five words are drawn from those with the largest gap between their best and worst split.

	top-1	top-5	KL↓	$\text{canonicity}_{\text{src}}$
LST: L1 attn_out patch				
Failed split	3.5%	13.1%	3.75	0.44
RANDOM	3.9%	13.8%	3.49	0.48
PATCHED	27.6%	39.6%	1.76	0.69
Successful split	60.4%	70.4%	0.43	0.91
Gap closed (PATCHED)	42%	46%	60%	53%
Gap closed (RANDOM)	1%	1%	8%	8%
FST: L1 mlp_out patch				
Failed split	1.9%	10.7%	4.14	0.41
RANDOM	2.9%	14.4%	3.68	0.47
PATCHED	26.8%	38.6%	1.89	0.68
Successful split	64.0%	74.1%	0.30	0.93
Gap closed (PATCHED)	40%	44%	59%	52%
Gap closed (RANDOM)	2%	6%	12%	12%

Table 3: Behavioral effect of the two interventions of §3, scored against the canonical single-token run. top-1 and top-5 are agreement rates with the canonical next-token prediction; KL is $\mathrm{KL}(p_{\text{canon}}\|\cdot)$, lower is better; $\text{canonicity}_{\text{src}}$ at $n-2$, shown for comparison. RANDOM patches from an unrelated word.

these heads write for every input. Head 27, which alone closes 17% of the gap in §3.3, is the most nearly identical of the three at 0.998.

E The Effect of Token Position on Detokenization Depth

§4 measures detokenization depth as word length grows. This appendix holds length fixed and varies position instead, in two ways: by moving the differing token within the word, and by timing the contribution of each intermediate position.

Component	Within pair	Different first token
attn_out	0.948	0.739
Head 24	0.923	0.214
Head 27	0.998	0.144
Head 28	0.834	0.231

Table 4: Cosine similarity of Layer 1 attn_out at position 2 across all 6,254 FST pairs on Llama2-7B. **Within pair**: source against target. **Different first token**: against words that do not share the first token.

E.1 Depth for Three-Token Splits with a Differing Middle Token

We construct 4,609 three-token pairs on Llama2-7B in which the two members share the first and the last token and differ only in the middle one. Apart from the position of the differing token, the construction follows §2.2. We patch resid_post at the last position, layer by layer, as in Figure 3.

The curve reaches 80% gap-closed at Layer 3. Three-token splits whose first token differs reach the same threshold at Layer 5, and two-token splits reach it at Layer 2. Depth therefore tracks the distance from the differing token to the last position, and not the token count alone: a differing middle token starts one position closer to the readout, and

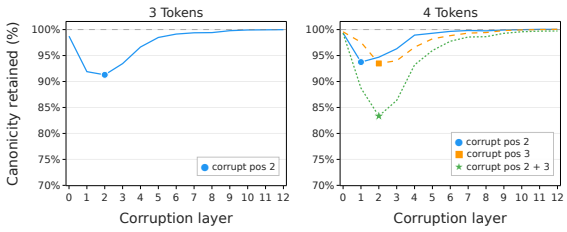


Figure 9: Per-position relay across word lengths. Canonicity at the last position when a single intermediate position's `resid_post` is patched with the failed run's activations at one layer at a time, as a percentage of the uncorrupted baseline. **Left:** 3-token words (one intermediate position: 2). **Right:** 4-token words (intermediate positions: 2, 3). Across both lengths relay timing is set by a position's index, not by total word length.

costs about two layers less. subsection E.2 shows the same dependence from the other direction.

E.2 Relay Timing per Intermediate Position in 3- and 4-Token Splits

§4.2 establishes, on 5-token words, that each intermediate position relays first-token information in a fixed 2–3 layer window whose timing is set by the position's index. This subsection verifies that the same per-position timing holds for shorter words.

We repeat the single-position corruption analysis of Figure 4 on 3- and 4-token words, corrupting each intermediate position's `resid_post` at one layer at a time and measuring the drop in canonicity at the last position. Figure 9 shows the result. The 3-token case has a single intermediate position (position 2), which drops canonicity maximally at Layer 2—the same layer at which position 2 drops canonicity in 5-token words (Figure 4, left). The 4-token case adds position 3, which contributes its own drop one layer later, again matching its 5-token counterpart. A position's relay timing is therefore a function of its index alone; total word length affects the *number* of relays available but not *when* each one activates.

F Model Details

Table 5 lists the 12 models analyzed in this work, grouped by the regime identified in §5. We draw from eight families spanning three positional-encoding schemes: GPT-2 (Radford et al., 2019), GPT-Neo (Black et al., 2021), GPT-J (Wang and Komatsuzaki, 2021), OPT (Zhang et al., 2022), Pythia (Biderman et al., 2023), BLOOM (BigScience Workshop, 2022), Llama 2 (Touvron et al., 2023), and Gemma 2 (Gemma Team, 2024). The table supports the claim that among the architectural variables we vary, positional encoding is the only one that aligns with the regime split. Every other column crosses the regime boundary. *Depth* ranges from 16 to 32 layers within the concentrated regime alone (Pythia-1B to Pythia-6.9B and Llama2-7B), and from 24 to 48 within distributed (OPT-1.3B to GPT2-XL); the ranges overlap substantially. *Width* (d_{model}) similarly spans 1,024–4,096 in concentrated and 1,280–4,096 in distributed. *Parameter count* spans 410M–7.1B in concentrated and 774M–6.7B in distributed—nearly identical ranges. *Training corpus* crosses the boundary in both directions: The Pile trains concentrated-regime models (Pythia family, GPT-J-6B) and a distributed-regime model (GPT-Neo-1.3B); mixed web data trains concentrated-regime models (Llama2-7B, Gemma-2-2B) and distributed-regime models (OPT-1.3B, OPT-6.7B). *Tokenizer* crosses the boundary as well: GPT-2 BPE appears in GPT-J-6B on the concentrated side and in all five distributed models. *MLP activation* likewise crosses: GELU is used by most concentrated models and by three of the five distributed ones.

BLOOM-7B1 (ALiBi) sits between the two regimes, concentrated at 2-token words but drifting toward distributed depth as token count grows (Appendix H). This intermediate behavior is itself informative: ALiBi and RoPE are mechanistically distinct schemes that both provide relative-position access in attention, so the property aligned with the split appears to be relative-position access in general rather than rotary encoding specifically.

G Cross-Architecture Two-Stage Mechanism

Figure 10 reports the two interventions of §5 layer by layer for each of the 12 models. The red curve isolates the attention stage: patching position 1's residual stream from a successful run into a failed one transfers any cross-position information attention might still read from that position, so the layer at which the patch stops closing the gap marks when attention has finished its read. The blue curve isolates the MLP stage: patching `mlp_out` from a failed run into a successful one replaces the MLP's contribution at that layer, so the resulting damage measures how much composition-specific output the MLP is producing there. The two stages are temporally adjacent in every model: the blue peak

Model	Family	Layers	d_{model}	Params	Training data	Tokenizer	MLP	PE	L1 (%)	L@80%
Concentrated										
GPT-J-6B	GPT-J	28	4,096	6.0B	The Pile	GPT-2 BPE	GELU	RoPE	82.8	1
Pythia-410M	Pythia	24	1,024	410M	The Pile	GPT-NeoX BPE	GELU	RoPE	74.0	3
Pythia-6.9B	Pythia	32	4,096	6.9B	The Pile	GPT-NeoX BPE	GELU	RoPE	62.2	3
Llama2-7B	LLaMA	32	4,096	7.0B	Mixed web	SP BPE	SwiGLU	RoPE	60.5	2
Pythia-1B	Pythia	16	2,048	1.0B	The Pile	GPT-NeoX BPE	GELU	RoPE	59.9	3
Gemma-2-2B	Gemma 2	26	2,304	2.6B	Mixed web	SP Unigram	GeGLU	RoPE	51.9	2
Intermediate (ALiBi)										
Bloom-7B1	BLOOM	30	4,096	7.1B	ROOTS	BLOOM BPE	GELU	ALiBi	51.0	5
Distributed										
OPT-1.3B	OPT	24	2,048	1.3B	Mixed web	GPT-2 BPE	ReLU	Learned	25.8	7
OPT-6.7B	OPT	32	4,096	6.7B	Mixed web	GPT-2 BPE	ReLU	Learned	15.7	10
GPT-Neo-1.3B	GPT-Neo	24	2,048	1.3B	The Pile	GPT-2 BPE	GELU	Learned	4.0	5
GPT2-Large	GPT-2	36	1,280	774M	WebText	GPT-2 BPE	GELU	Learned	2.3	7
GPT2-XL	GPT-2	48	1,600	1.5B	WebText	GPT-2 BPE	GELU	Learned	2.3	7

Table 5: Architectural details and patching results for all 12 models, 2-token words, LST configuration. **L1 (%)**: percentage of canonicity$_{\text{src}}$ gap closed by patching Layer-1 `resid_post`. **L@80%**: first layer at which $\geq 80\%$ of the gap is closed. Models are sorted within each group by L1. SP = SentencePiece.

sits within or just after the red decay window, never before it. Concentrated-regime RoPE models complete both stages within the first ∼5 layers, with red decaying by Layer 2–3 and blue peaking at Layer 2–4. Distributed-regime models spread both stages across layers 5–12: the red curve decays gradually from a high plateau, and the blue peak shifts proportionally later. Bloom-7B1 sits between the two regimes. Its red curve decays gradually over 15+ layers rather than sharply within 2–3, and its blue peak (∼13% at Layer 1) is smaller than the peaks of the RoPE models. The two-stage ordering is intact, but the cross-position read is spread over a much wider window, consistent with ALiBi encoding relative position less sharply than RoPE while still providing relative-position access in attention.

H Cross-Architecture Gap-Closed Patching

Section 5 introduced the concentrated/distributed regime distinction using gap-closed-80% depth on 2-token words. This appendix extends the measurement to 3- and 4-token words and reports per-model $\ell^*_{80\%}$ values, which fix the probe depth used in §6.

Figure 11 shows that the RoPE/learned-PE split holds at every word length. The six RoPE models accumulate gap-closed sharply, rising to 80% within the first ∼15% of network depth and then plateauing. Learned-PE models accumulate gradually, rising over roughly the first third of the network before plateauing. Table 6 reports the per-model $\ell^*_{80\%}$ for $k \in \{2, 3, 4\}$. Among the RoPE models, $\ell^*_{80\%}$ stays in single digits across all word

Model	PE	Layers	ℓ^*(80% g-c depth)		
			k=2	k=3	k=4
Concentrated					
GPT-J-6B	RoPE	28	1	2	5
Pythia-410M	RoPE	24	3	5	6
Pythia-6.9B	RoPE	32	3	5	6
Llama2-7B	RoPE	32	2	5	5
Pythia-1B	RoPE	16	3	5	6
Gemma-2-2B	RoPE	26	2	4	5
Intermediate (ALiBi)					
Bloom-7B1	ALiBi	30	5	10	17
Distributed					
OPT-1.3B	Learned	24	7	10	11
OPT-6.7B	Learned	32	10	13	14
GPT-Neo-1.3B	Learned	24	5	8	11
GPT2-Large	Learned	36	7	13	15
GPT2-XL	Learned	48	7	14	17

Table 6: Depth required for detokenization as a function of word length, across all 12 models. $\ell^*_{80\%}$ is the first layer at which patching `resid_post` at the last subword position closes $\geq 80\%$ of the canonicity$_{\text{src}}$ gap.

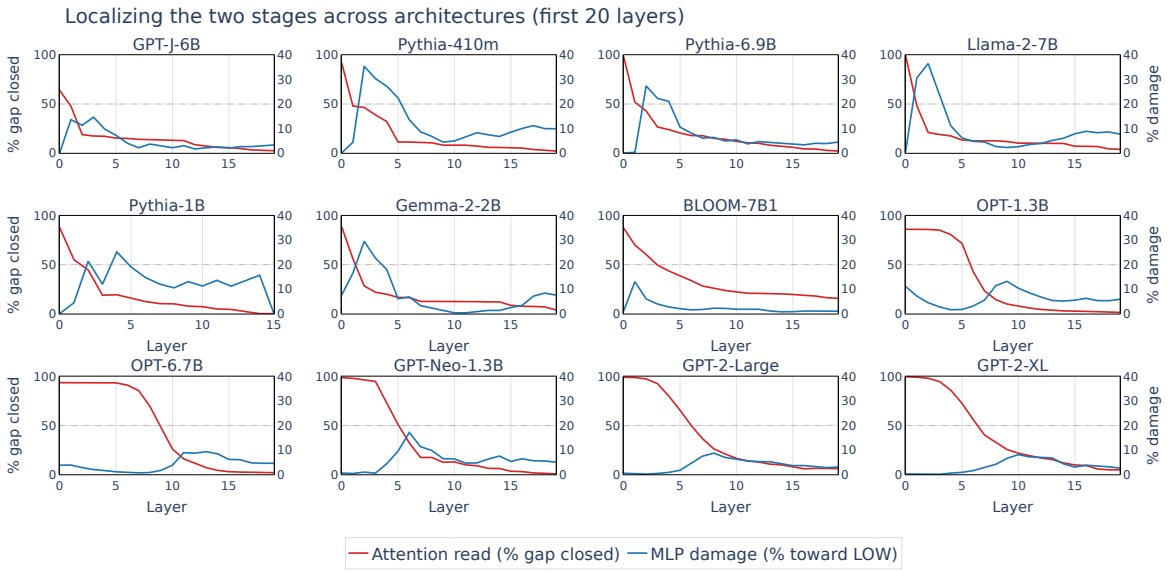


Figure 10: Localizing the two-stage mechanism across twelve architectures. Red curve (left y-axis): percentage of the canonicity$_{\text{src}}$ gap closed by patching `resid_post` at position 1 from a successful run into a failed one, layer by layer. Blue curve (right y-axis): percentage damage to canonicity when patching `mlp_out` from a failed run into a successful one. First 20 layers shown for all models.

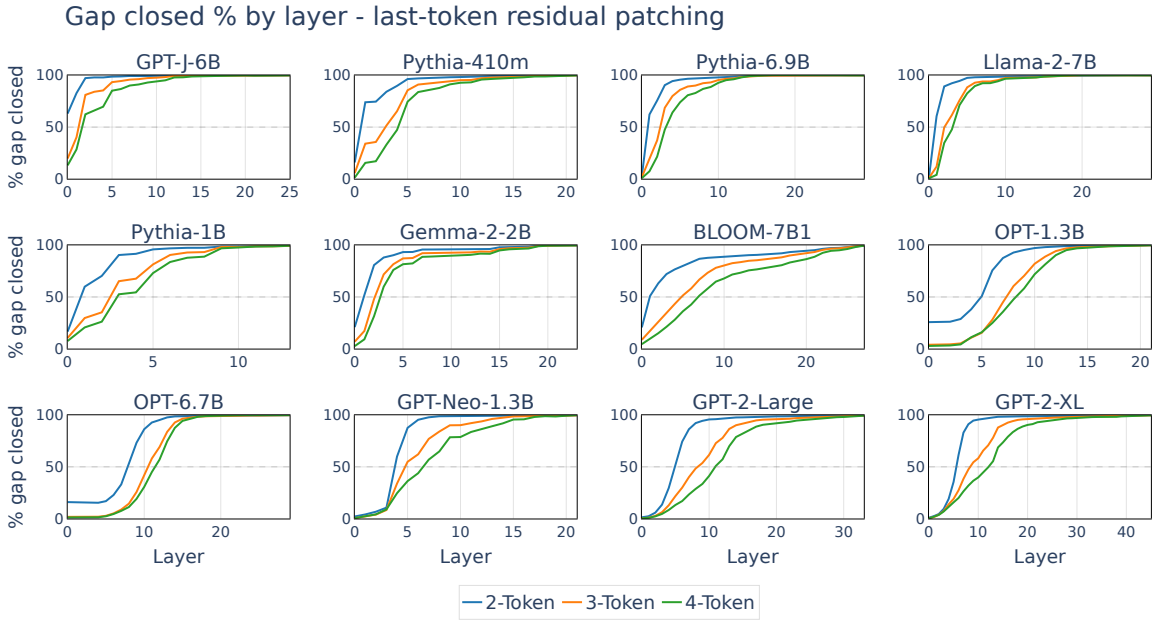


Figure 11: Percentage of canonicity$_{\text{src}}$ gap closed per layer via `resid_post` patching at the final subword position, for words of length $k \in \{2, 3, 4\}$ tokens. One panel per model. First-layer indices at which each curve reaches 80% gap-closed are listed in Table 6.

lengths (5–6 layers at k=4). Among the learned-PE models, $\ell^*_{80\%}$ at k=4 ranges from 11 to 17 layers. Bloom-7B1 (ALiBi) tracks the concentrated regime at k=2 ($\ell^*_{80\%}$=5) but drifts toward distributed behavior at k=4 ($\ell^*_{80\%}$=17), reaching depths comparable to the slowest learned-PE model (GPT2-XL). This is consistent with the BLOOM observation in §5: ALiBi provides relative-position access in attention, but encodes position less sharply than RoPE, and the gap to learned-PE models narrows as the cross-position composition window widens with word length.

I Cross-Architecture Probe Results

Figure 12 reports layerwise AUROC for all 12 architectures at k=2, replicating the three-curve analysis of Figure 6. We elaborate here on choices in the probe pipeline and on per-model observations.

Per-model behavior. Peak Isolated AUROC at l^* ranges from ∼0.76 (Pythia-1B) to ∼0.94 (Llama2-7B, GPT-J-6B, OPT-1.3B). Peak In-context AUROC ranges from ∼0.86 (Gemma-2-2B) to ∼0.99. Pythia models show a persistent Isolated/In-context gap, most pronounced in Pythia-1B and Pythia-6.9B (Isolated∼0.75–0.80, In-context>0.95). Gemma-2-2B is the sole exception where all three curves remain within ∼0.05 AUROC of each other across all layers. While the Isolated→In-context decay is universal, its depth varies considerably by model: it is sharpest in Pythia-6.9B, GPT2-XL, and OPT-6.7B (dropping below 0.7), and shallowest in GPT-J-6B and Gemma-2-2B (remaining above 0.8). This variation suggests that the rate at which context overwrites the early-layer composition signal is architecture-dependent.

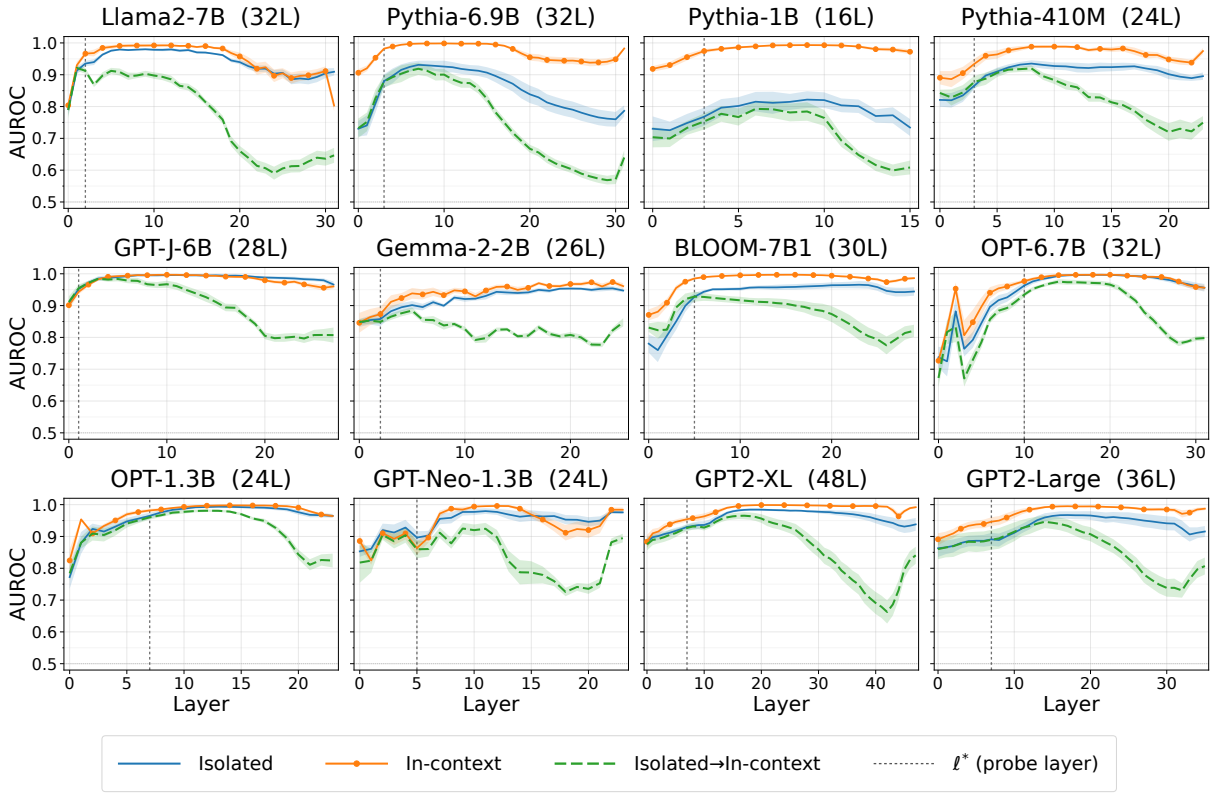


Figure 12: Layerwise probe AUROC for all 12 architectures at k=2. Solid blue: Isolated probe. Solid orange: In-context probe. Dashed green: Isolated→In-context (probe fit on Isolated activations, evaluated on In-context activations of held-out words). Shaded regions show 95% confidence intervals across 5 folds. Vertical dotted line: l^* per model. The Llama2-7B pattern holds across every architecture: Isolated and In-context rise to a plateau by l^* and stay high; Isolated→In-context tracks them through l^* and decays through middle layers as preceding-token context aggregates into the last subword position.