TIDE: Token Identity Delivered Everywhere in LLMs

Ajay Jaiswal **Lauren Hannah** **Han-Byul Kim**
Duc Hoang **Mehrdad Farajtabar** **Minsik Cho**
Apple

Abstract

We revisit a universally accepted but under-examined design choice in every modern LLM: a token index is looked up once at the input embedding layer and then permanently discarded. This *single-injection assumption* induces two structural failures: (i) the *Rare Token Problem*, where a Zipf-type distribution of vocabulary causes rare-token embeddings are chronically under-trained due to receiving a fraction of the cumulative gradient signal compared to common tokens; and (ii) the *Contextual Collapse Problem*, where limited parameters models map distributionally similar tokens to indistinguishable hidden states. As an attempt to address both, we propose TIDE, which augments the standard transformer with EMBEDDINGMEMORY: an ensemble of K independent MEMORYBLOCKS that map token indices to context-free semantic vectors, computed once and injected into every layer through a depth-conditioned softmax router with a learnable null bank. We *theoretically* and *empirically* establish the benefits of TIDE in addressing the issues associated with single-token identity injection as well as *improve performance* across multiple language modeling and downstream tasks.

1 Introduction

Scaling modern large language models (LLMs) involves devoting substantial representational capacity towards *contextualizing* tokens through innovating attention mechanisms, enlarging feed-forward modules, and stacking deep transformer layers. In contrast, a critical LLM component that has been widely overlooked in recent advancements is the **token index** - *the only piece of information that unambiguously identifies what a token is*. The token index is looked up once at the input embedding layer and then permanently discarded. Every subsequent computation across all L transformer layers operates on a contextualized hidden state that never again directly consults which vocabulary entries are being processed. This single-injection assumption creates two distinct failure modes:

❶ **The Rare Token Problem:** Natural language vocabularies obey power law scaling, specifically Zipf's law [Zipf, 1949, Pilgrim and Hills, 2021]: the most frequent 1% of tokens account for $\sim 80\%$ of corpus occurrences. Under SGD, cumulative gradient signal for each token embedding is proportional to its frequency (Section 2.1), leaving rare-token embeddings (e.g. rare named entities, technical terms, low-frequency morphological forms) persistently under-trained (Figure 1).

❷ **Contextual Hidden State Collapse:** During training, FFNs are forced into *representational overloading* where they simultaneously implement structural transformations of the residual stream and serve as the primary store of token-specific factual knowledge [Meng et al., 2022, Dai et al., 2022]. The token index is never re-consulted at intermediate layers, and the only mechanism the FFNs have to differentiate two tokens at depth relies on contextual mixture of residual and attention output. However, in case when two semantically distinct tokens appear in nearly identical syntactic environments, the context provides limited differentiating signal and their hidden states become nearly indistinguishable across the network (Figure 2).

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

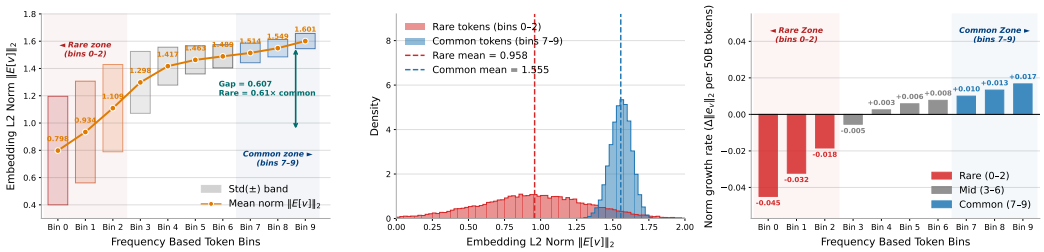

Figure 1: **Empirical Evidence that Rare Token Embeddings Remain Under-trained:** (a) Mean embedding $l2$-norm of LLaMa-Base-1B pretrained checkpoint showing a monotonic increase in norm from rare to common bins; (b) Embedding norm distributions for rare and common tokens: existence of *wide rare distribution versus the narrow common peak* confirms that rare embeddings remain noise-dominated and under-trained; (c) Bin-wise norm growth rate across intermediate training checkpoints per 50 billion tokens: rare token norms exhibit *monotonic decline* with continued training while common token continue growing.

Motivated by these challenges, we pose a critical question: *How can we provide every transformer layer with persistent, token-identity-conditioned knowledge, independent of and complementary to the contextual residual stream?* Unlike prior approaches that focus on post-hoc analysis of *de facto* FFNs [Geva et al., 2022b, Meng et al., 2022, 2023] or retrofit external retrieval at inference time [Lewis et al., 2020b, Borgeaud et al., 2022b, Izacard et al., 2023], we adopt an alternative approach: designing and training from scratch a novel transformer architecture that maintains a dedicated semantic memory indexed directly by static token identity information.

In this work, we propose TIDE (**T**oken **I**dentity **D**elivered **E**verywhere), an architectural modification to standard transformer that maintains a dedicated semantic memory indexed directly by token identity (Figure 3). TIDE introduces EMBEDDINGMEMORY, an ensemble of K independent MEMORYBLOCKs each mapping token indices to static and context-free learned semantic vectors that can be injected to each transformer layer with a persistent, token-conditioned signal in parallel to the contextual residual stream. Our key contributions can be summarized as:

- **Architectural.** TIDE introduces a *token-level unified embedding memory* that enables K disjoint pathways for token-level gradient accumulation. The tensor from memory embeddings tensor is computed *once per forward pass* and injected into *every* transformer layer via a per-layer softmax routing mechanism conditioned on the post-attention hidden state.
- **Theoretical.** We formalize the two failure mode in standard transformer and prove that TIDE (i) asymptotically generalizes the standard transformer; (ii) amplifies the per-token cumulative gradient signal by a factor of K, and (iii) routes around the FFN's Lipschitz constraint by exposing a discrete, token-indexed input with no obligation to hidden states.
- **Empirical.** We empirically validate that TIDE significantly benefits *rare* tokens and mitigates contextual collapse problem. Across model scales from 350M to 1B parameters, TIDE consistently delivers up to performance improvements over standard transformer across various language modeling datasets (*e.g.,* Wikitext, PubMed, DCLM) as well as downstream tasks (*e.g.,* HellaSwag, ARC, PIQA).

2 When Context is Not Enough: Diagnosing Standard Transformers

2.1 The Rare Token Problem.

Gradient Starvation Bottleneck: Under minibatch SGD with batch size B, sequence length T, and per-token squared gradient norm bounded by G^2, the embedding $e_v \in \mathbb{R}^d$ for token v receives a non-zero gradient only when v appears in the current batch. In this setting, the expected cumulative squared gradient norm after τ training steps satisfies:

$$\mathbb{E}\left[\sum_{s=1}^{\tau} \|\nabla_{e_v} \mathcal{L}_s\|^2\right] \leq \tau \cdot f_v \cdot B \cdot T \cdot G^2, \tag{1}$$

where $f_v := \Pr[\text{uniformly drawn token position equals } v] \in (0, 1)$ is the unigram probability of v, with $\sum_{v \in \mathcal{V}} f_v = 1$. Token v is *rare* if $f_v = \epsilon$ for some $\epsilon \ll 1/(BT)$, and token u is *common* if $f_u \geq c$ for some constant $c > 0$ independent of $|\mathcal{V}|$. The full derivation of (1) is given in Appendix C.

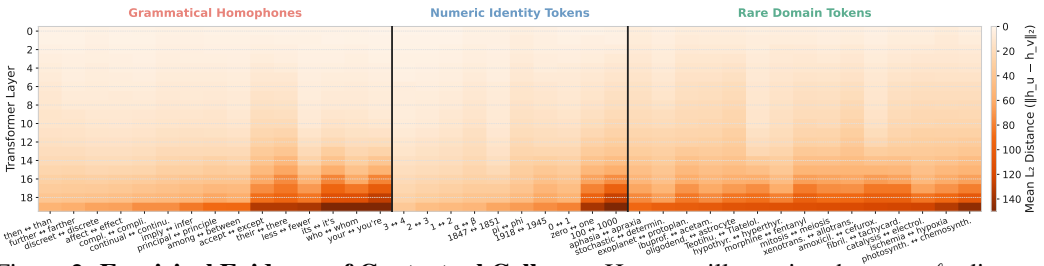

Figure 2: **Empirical Evidence of Contextual Collapse:** Heatmap illustrating the mean ℓ_2-distance $\|h_u^{(\ell)} - h_v^{(\ell)}\|$ between hidden states (LLaMa-`Base`-1B) of token pairs across 250 template sentences from three example categories of contextual collapse. For all sampled pairs, the distance remains *near-zero* for majority of layers (except towards end) confirming the presence of contextual collapse.

In an example corpus of Wikitext-103 [Merity et al., 2016] tokenized using LLaMA-3 tokenizer ($|\mathcal{V}| = 128{,}256$) to generate frequency bins (Appendix B), the gradient disparity between *rare* and *common* tokens becomes severe. Over a training budget of 200B tokens with $B = 8$, $T = 2048$, the expected number of non-zero gradient updates to token v's embedding is given as:

Tier	Bin(s)	Count	f_v	$\mathbb{E}[N_v]$
Hapax (rarest)	0	1	8.3×10^{-9}	$\approx$1,660
Near-hapax	1	$\sim$4	3.3×10^{-8}	$\approx$6,640
Uncommon	2	$\sim$10	8.3×10^{-8}	$\approx$16,600
Mid-freq.	3–6	$\sim10^{2-3}$	$\sim10^{-6}$	$\approx10^{5-6}$
Common (highest)	7–9	$\sim10^{6}$	8.3×10^{-3}	$\approx1.66\times10^{9}$

Table 1: Expected non-zero gradient updates $\mathbb{E}[N_v]$ of token bins with 200B training tokens.

$$\mathbb{E}[N_v] \;=\; \tau\big(1-(1-f_v)^{BT}\big) \;\approx\; \tau\cdot f_v\cdot BT \quad \text{for small } f_v\,. \tag{2}$$

In reference to frequency bins defined in Appendix B, Table 1 instantiates this across the 200B tokens in our training dataset illustrating the existence of high gradient update disparity between rare and common tokens. Additionally, it can empirically inferred from the Figure 1(c) that this disparity doesn't limit itself as a cold-start artifact but grows *monotonically* as the training progresses. The rare tokens' norms decline while common tokens' norms continuously increase. We find similar increasing trend in publically released pre-trained checkpoints ([Grattafiori et al., 2024]).

Ratio of gradient signal for rare and common tokens: For rare v ($f_v = \varepsilon$) and common u ($f_u \geq c > 0$), let $G_{\min}^2 > 0$ be a lower bound on the per-step squared gradient norm conditioned on token u appearing in a batch. The ratio of cumulative gradient signals satisfies:

$$\frac{\mathbb{E}\big[\sum_s \|\nabla_{e_v}\mathcal{L}_s\|^2\big]}{\mathbb{E}\big[\sum_s \|\nabla_{e_u}\mathcal{L}_s\|^2\big]} \;\leq\; \frac{\varepsilon\, BT\, G^2}{\kappa\, G_{\min}^2} \;=\; O(\varepsilon/c) \tag{3}$$

where $\kappa := 1-(1-c)^{BT} > 0$ with BT, G^2, and $G_{\min}^2$ as fixed positive constants. The full derivation is given in Appendix C.1. For the empirical instantiation in Table 1, the ratio between rare tokens (Bin 1) and common tokens (Bin 9) is $\varepsilon/c \approx 10^{-6}$, a disparity of six orders of magnitude of gradient signal between rare and common tokens over the same training budget.

2.2 Contextual Collapse and the FFN's Blind Spot.

As mentioned before, the gradient starvation issue causes the rare-token embeddings to converge to low-norm, noisy representations. More seriously, when two distinct tokens carry poorly trained embeddings of similar magnitude, a deeper structural failure arises: the hidden states produced for those tokens across all transformer layers may become indistinguishable, which can more problematic with similar context shared. We formalize this failure mode and show that it is an inherent consequence of the Lipschitz continuity imposed on any FFN by its continuous domain.

The Contextual Collapse Phenomenon: At each layer ℓ, the hidden state $h_v^{(\ell)} \in \mathbb{R}^d$ of a token v is produced by the attention mechanism operating on the surrounding context. When two tokens $u \neq v$ appear in nearly identical syntactic environments, such as in case for grammatical homophones (*their* or *there*), numeric identity tokens (*1847*, *1851*, or *1849*), or rare domain-specific synonyms (*ibuprofen* or *acetaminophen*), the context provides no distinguishing signal and thereby attention produces similar outputs for both. We define this as:

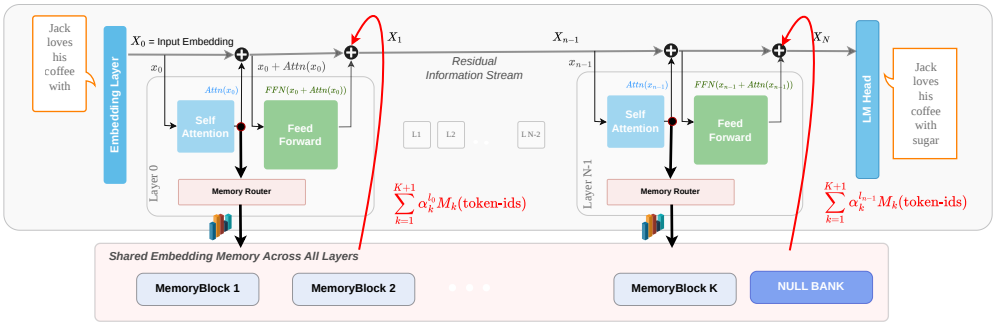

Figure 3: **Main Architecture Diagram:** TIDE augments standard transformers with a parallel and globally shared EMBEDDINGMEMORY module (red region) consisting of K independent MEMORYBLOCK, each mapping raw token indices to *context-free token identity signal*. Each layer uses a linear router to combine memory blocks signals and injects that into the residual stream additively.

Definition 2.1 (Contextual Collapse Set)**.** For a tolerance $\delta > 0$, the *contextual collapse set* at layer ℓ can be formally defined as:

$$\mathcal{C}_\delta^{(\ell)} := \big\{(u, v) \in \mathcal{V}^2 : u \neq v,\ \|h_u^{(\ell)} - h_v^{(\ell)}\| \leq \delta\big\},$$

where the hidden states are averaged over a representative corpus of contexts.

Figure 2 provides direct empirical evidence of contextual collapse in LLaMa-`Base`-1B standard model estimated using 150 template sentences that differ by a single token pair under consideration. For each of the three example canonical categories, the mean ℓ_2 distance $\|h_u^{(\ell)} - h_v^{(\ell)}\|$ remains persistently small across the entire depth axis except the last few final layers, confirming the prevalent existence of collapse. Note that this phenomenon is more severe across numerical tokens category having notable collapse (small δ) even within the final layer's hidden states.

Proposition 2.2 (FFN Approximation Lower Bound on Collapsed Tokens)**.** *Let $(u, v) \in \mathcal{C}_\delta^{(\ell)}$ be a collapsed token pair and let $g : \mathcal{V} \to \mathbb{R}^d$ be any target function satisfying $\|g(u) - g(v)\| = C > 0$. Then for* any *choice of weights W_1, W_2:*

$$\max\big\{\|\mathrm{FFN}(h_u) - g(u)\|,\ \|\mathrm{FFN}(h_v) - g(v)\|\big\} \ \geq\ \frac{C - L_{\mathrm{FFN}}\,\delta}{2}.$$

When $C > L_{\mathrm{FFN}}\,\delta$, the right-hand side is strictly positive: the FFN cannot approximate g to arbitrary precision on the collapsed pair (u, v), regardless of how many parameters it has.

Proof sketch. Since $(u, v) \in \mathcal{C}_\delta^{(\ell)}$, the Lipschitz bound forces $\|\mathrm{FFN}(h_u) - \mathrm{FFN}(h_v)\| \leq L_{\mathrm{FFN}}\delta$. Applying the triangle inequality to the target separation $C = \|g(u) - g(v)\|$ and substituting this bound yields $\|\mathrm{FFN}(h_u) - g(u)\| + \|\mathrm{FFN}(h_v) - g(v)\| \geq C - L_{\mathrm{FFN}}\delta$. Since the maximum of two non-negative terms is at least half their sum, the result follows. See Appendix D for details.

In this bound, δ is determined by the embeddings and attention layers; it is fixed before the FFN acts. The separation target C is determined by the downstream task.The Lipschitz constant L_{FFN} is the only term the FFN controls, but it is bounded in practice because large L_{FFN} amplifies *every* input perturbation, degrading performance on the majority of non-collapsed tokens. The bound exposes a structural limitation: given fixed upstream representations, no FFN, regardless of width, can resolve a collapsed token pair without destabilizing other inputs. The token index is injected once at the embedding layer and never reintroduced; unlike position, which is re-injected via RoPE at every attention layer, token identity has no recovery mechanism. Once intermediate layers erase the distinction, it is permanently lost to all subsequent computation.

3 TIDE: Token Identity Delivered Everywhere

In section 2, we investigated and formalized two failure mode, *i.e.,* rare token and contextual collapse problem, within the standard transformer architecture. In this work, we address these issues with a novel architecture modification: TIDE counters the single-injection assumption in conventional design of modern LLMs. TIDE stops discarding the token identity information after embedding layer and instead make it directly accessible at every depth, so that each layer retains a token-discriminative signal independent of the contextual residual stream.

3.1 Preliminaries and Notations.

Let $\mathcal{V}$ denote a vocabulary of size $|\mathcal{V}|$, d the model hidden dimension, d_b the MEMORYBLOCK embedding dimension, K the number of MEMORYBLOCKs, L the number of transformer layers, T the input sequence length, and B the batch size. We use $x \in \mathbb{Z}^{B\times T}$ for a batch of token index sequences and $h^{(\ell)} \in \mathbb{R}^{B\times T\times d}$ for hidden states at layer ℓ. The standard LLaMA-style transformer block at layer ℓ computes:

$$\tilde{h}^{\ell} = h^{\ell-1} + \mathrm{Attn}\big(\mathrm{RMSNorm}(h^{\ell-1})\big), \tag{4}$$

$$h^{\ell} = \tilde{h}^{\ell} + \mathrm{FFN}\big(\mathrm{RMSNorm}(\tilde{h}^{\ell})\big), \tag{5}$$

where Attn is multi-head self-attention with rotary position embeddings and FFN is a SiLU-gated feed-forward network. The primary embedding table $E \in \mathbb{R}^{|\mathcal{V}|\times d}$ maps each token index to an initial hidden state $h^{(0)} = E[x]$ that will be processed by different transformer blocks.

3.2 TIDE Architecture Design.

TIDE augments the standard transformer with a parallel *token-identity memory pathway* composed of three components:

MEMORYBLOCKs: Each of the K MEMORYBLOCKs maintains a dedicated embedding table $E_k \in \mathbb{R}^{|\mathcal{V}|\times d_b}$ and maps a token index $v \in \mathcal{V}$ to a d_b-dimensional vector via a single embedding lookup followed by RMSNorm [Zhang and Sennrich, 2019]:

$$M_k(v) = \mathrm{RMSNorm}\big(E_k[v]\big) \in \mathbb{R}^{d_b}. \tag{6}$$

Each block maintains its own independent embedding table with no parameter sharing across blocks, encouraging each MEMORYBLOCK to learn a distinct projection of the token identity space.

EMBEDDINGMEMORY ensemble: The K MEMORYBLOCKs are stacked into a single memory tensor computed *once* per forward pass and shared across all L transformer layers:

$$\mathbf{M} = \mathrm{Stack}_k\big(M_k(x)\big) \in \mathbb{R}^{B\times T\times K\times d_b}. \tag{7}$$

Depth-conditioned router and additive fusion: Within each transformer block, the post-attention normalised hidden state $\tilde{n}^{\ell} = \mathrm{RMSNorm}(\tilde{h}^{\ell})$ is fed to a lightweight linear router to generate composition ratio α_k^{ℓ} corresponding to k-th memory block. We additionally introduce a **null bank** at slot $K{+}1$ satisfying $M_{K+1}(v) = \mathbf{0}$ for all v, giving the router a learned "off" switch for with no dedicated parameters. The full TIDE layer update is:

$$\boldsymbol{\alpha}^{\ell} = \mathrm{Softmax}\big(W_r^{\ell}\,\tilde{n}^{\ell}\big) \in \mathbb{R}^{K+1}, \tag{8}$$

$$m^{\ell}(v) = \sum_{k=1}^{K+1} \alpha_k^{\ell}\, M_k(v), \qquad h^{\ell} = \tilde{h}^{\ell} + \mathrm{FFN}\big(\tilde{n}^{\ell}\big) + m^{\ell}(v). \tag{9}$$

where $W_r^{\ell} \in \mathbb{R}^{(K+1)\times d}$ is a per-layer learned weight matrix and $\sum_{k=1}^{K+1} \alpha_k^{\ell} = 1$, $\alpha_k^{\ell} > 0$ for all k. The memory vector $m^{\ell}(v)$ is added *additively* and *independently* of the FFN output: neither pathway interact with the other, preserving the residual stream's role as a shared communication channel [Elhage et al., 2021]. Given that $\mathbf{M}$ is indexed by discrete token identity v, not by hidden state h^{ℓ}, the memory contribution of each token is independent of contextual mixing at any depth.

3.3 TIDE: Theoretical Perspectives and Observations.

3.3.1 Asymptotic Generalization to Standard Transformer.

Proposition 3.1 (Asymptotic Generalization)**.** *Let $\mathcal{F}_{\mathrm{base}}$ denote the function class of standard transformers* (5) *and $\mathcal{F}_{\mathrm{TIDE}}$ the class of our proposed* TIDE *models* (9)*. For any $\epsilon > 0$, there exist finite router parameters W_r^{ℓ} such that*

$$\big\|m^{\ell}(v)\big\| < \epsilon \qquad \forall\, v \in \mathcal{V},\ \ell \in \{1,\dots,L\}.$$

That is, $\mathcal{F}_{\mathrm{TIDE}}$ can approximate the standard transformer $\mathcal{F}_{\mathrm{base}}$ to an arbitrary precision.

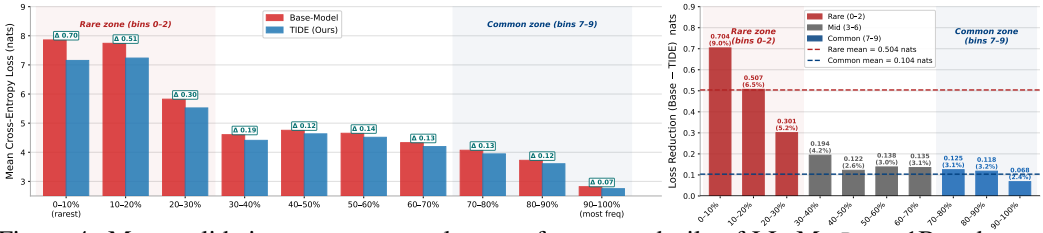

Figure 4: Mean validation cross-entropy loss per frequency decile of LLaMa-Base-1B and TIDE-8E-1B trained with 200B tokens. TIDE strictly improves over the baseline on every decile with gain concentrated for *rare* tokens and follows monotonically decreasing trend as *rare* > *mid* > *common*.

Proof sketch. Since $M_{K+1}(v) = \mathbf{0}$, any weight assigned to the null bank contributes nothing to $m^\ell(v)$. By the softmax constraint, increasing the null logit z^ℓ_{K+1} jointly suppresses all active bank weights: $\sum_{k=1}^{K} \alpha^\ell_k = K/(K + e^{z^\ell_{K+1}}) \to 0$ as $z^\ell_{K+1} \to \infty$.

The triangle inequality then gives $\left\|m^\ell(v)\right\| \le (1-\alpha^\ell_{K+1}) \cdot C \to 0$, where $C = \max_{v,k} \|M_k(v)\| < \infty$. Setting $z^\ell_{K+1} = s^* = \log(K(C-\epsilon)/\epsilon)$ achieves $\left\|m^\ell(v)\right\| < \epsilon$ at a finite parameter configuration. The full proof can be found in Appendix E.

3.3.2 TIDE's K-Pathway Gradient Amplification.

In section 2.1 for the standard transformer, we discussed that the embedding e_v of a rare token v receives a non-zero gradient update only in steps where v appears in the batch, yielding an expected cumulative squared gradient norm bounded by $\tau \cdot f_v \cdot BT \cdot G^2$. Our proposed TIDE's architecture provides a design advantage of K independent MEMORYBLOCKs that enable K distinct, parallel gradient pathways into each token's embedding tables on *every* training step, regardless of how rarely it occurs in the corpus. We formalize the advantage as:

Proposition 3.2 (K-Pathway Gradient Amplification)**.** *Let $\mathcal{L}_s$ denote the loss at step s and let $e^{(k)}_v \in \mathbb{R}^{d_b}$ be the embedding of token v in* MEMORYBLOCK *k. Under minibatch SGD, the total expected cumulative squared gradient norm across all K embedding tables for token v satisfies:*

$$\mathbb{E}\left[\sum_{s=1}^{\tau}\sum_{k=1}^{K}\left\|\nabla_{e^{(k)}_v}\mathcal{L}_s\right\|^2\right] \;\ge\; K \cdot \tau \cdot \kappa_v \cdot G^2_{\min}, \tag{10}$$

where $\kappa_v = 1-(1-f_v)^{BT} \approx f_v \cdot BT$ for small f_v, and $G^2_{\min} > 0$ is a lower bound on the per-step squared gradient norm conditioned on token v appearing in the batch. Consequently, TIDE *provides a K-fold amplification of gradient signal relative to the standard single-embedding baseline.*

Proof sketch. Each MEMORYBLOCK k maintains an independent embedding table $E_k \in \mathbb{R}^{|\mathcal{V}| \times d_b}$ with no parameter sharing across blocks[1]. Within a forward pass during training, MEMORYBLOCK k's output $M_k(v)$ is injected into every transformer layer ℓ via the routing weight α^ℓ_k, contributing to the residual stream and thereby to the loss. Since the K blocks are independent, the event $\{v \in \text{batch}_s\}$ triggers gradient flow through all K embedding tables simultaneously. Because router weights are strictly positive for finite logits, each table receives a non-degenerate gradient on every step that v appears. Summing across blocks and applying the lower bound from Appendix C.1 independently to each yields the K-fold amplification. Please see Appendix F for details.

★ **Empirical Investigation *[Rare Tokens Benefits from* TIDE*]*:** Figure 4(a) illustrate the mean cross-entropy of LLaMa-Base-1B and TIDE-8E-1B at the matched 200B-token training budget across all 10 token frequency deciles. Clearly, we can observe that TIDE *strictly outperforms* LLaMa-Base-1B on every decile, but the absolute performance gap is sharply asymmetric for *rare vs. common* tokens. Per-decile loss reduction in Figure 4(b) decays *monotonically* from 0.704 nats (9.0% relative) on the rarest decile to 0.068 nats (2.4%) on the most frequent decile, yielding a $\sim$4.8$\times$ disparity in absolute gain between *rare* and *common* mean. This rare-skewed improvement profile is precisely the empirical signature provide support for K-fold gradient amplification to assist tokens where base embedding E is gradient starved during training.

[1]For the simplicity, we state the argument for a router over K active banks.

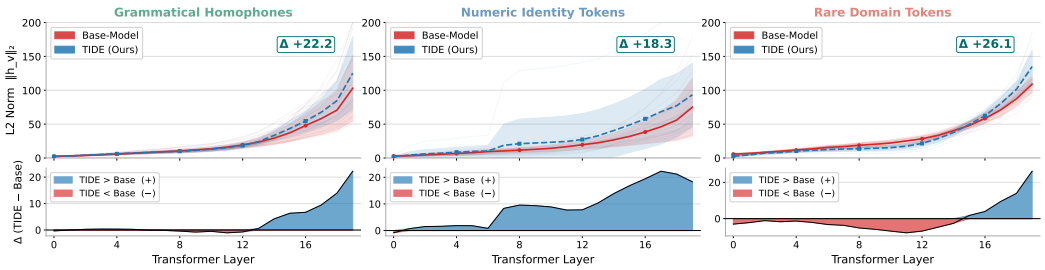

Figure 5: Layer-wise ℓ_2 separation $\|h_u^{(\ell)} - h_v^{(\ell)}\|$ between hidden states of token pairs from the three example contextual collapse categories, averaged across 150 template sentences.

3.3.3 Contextual Collapse and TIDE K-MEMORYBLOCKS.

In a standard transformer, FFN receives $h^{(\ell)}$ as input, and when $\left\|h_u^{(\ell)} - h_v^{(\ell)}\right\| \leq \delta$ is small, Lipschitz continuity forces its outputs to remain close regardless of the weights chosen (see Section 2.2). TIDE architectural design permits to break this constraint since each MEMORYBLOCK is indexed by the discrete token identity v unlike $h^{(\ell)}$, so its output carries no continuity obligation with respect to δ. We formalize this observation as:

Proposition 3.3 (Memory Ensemble Resolves Collapsed Token Separation)**.** *Let $(u, v) \in \mathcal{C}_\delta^{(\ell)}$ be a collapsed token pair satisfying $\left\|h_u^{(\ell)} - h_v^{(\ell)}\right\| \leq \delta$, and let $C > 0$ be any target separation. For any $K \geq 1$, there exist* EMBEDDINGMEMORY *parameters $\{E_k\}_{k=1}^K$ such that:*

$$\|M_k(u) - M_k(v)\| = C \tag{11}$$

regardless of $\delta = \left\|h_u^{(\ell)} - h_v^{(\ell)}\right\|$ and independently of L_{FFN}.

Proof sketch. Each MEMORYBLOCK output $M_k(v) = \text{RMSNorm}(E_k[v])$, where $E_k[v]$ is the row of embedding table E_k indexed by the discrete token identity v. The hidden state $h^{(\ell)}$ does not appear in this computation, so $M_k(u)$ and $M_k(v)$ depend only on their respective rows $E_k[u]$ and $E_k[v]$. Since these rows are *separate, uncoupled parameters*, they can be assigned freely and independently for any token pair (u, v), regardless of how small δ is. In particular, one can choose $E_k[u]$ and $E_k[v]$ such that the resulting RMSNorm outputs can achieve any prescribed separation $C > 0$, which satisfies (11). See Appendix G for the additional details.

We would like to clarify that TIDE does not attempt to fight the Lipschitz constraint of the FFN; it routes around it by exploiting a fundamentally different input signal during the training. Because $m^{(\ell)}(v) = \sum_{k=1}^K \alpha_k^\ell M_k(v)$ is re-injected in additive fashion at every transformer layer via independent per-layer router weights W_r^ℓ, this token-discriminative signal persists throughout the residual stream, it enables effective separation at every layer ℓ.

★ **Empirical Investigation *[Contextual Collapse is Moderated by* TIDE*]*:** To empirically validate the contribution of additive pathways of MEMORYBLOCKS, we revisit the three example contextual collapse categories from Figure 2 (grammatical homophones, numeric identity tokens, rare domain tokens) and compare the layer-wise ℓ_2 separation $\|h_u^{(\ell)} - h_v^{(\ell)}\|$ between LLaMa=Base-1B and TIDE on the same template sentences. Figure 5 (top row) reports the mean ℓ_2 norm averaged over all sampled token pairs in each category and bottom row reports the per-layer difference $\Delta = \|\cdot\|_{\text{TIDE}} - \|\cdot\|_{\text{BASE}}$. Across all three categories, we can clearly observe that TIDE's token-discriminative signal injection significantly increase in ℓ_2 separation prominently from middle to terminal layers which are distant from base embedding E. Note that *numerical tokens* which suffers acute collapse (Figure 2), are the predominant beneficiary of the token identity injection throughout all layers.

To test whether collapse persists under higher syntactic diversity, we sample 250 naturally occurring 2048-token DCLM contexts containing a numerical keyword and pair each with a copy in which the keyword is replaced by its reverse. We found that collapse persist but become less severe than in templates (layer-8 mean $\|h_u - h_v\| = 16.2$ vs. 11.7). Overall, TIDE's separation gain remains consistent (mean $\Delta = +16.9$ vs. $+18.3$ on templates).

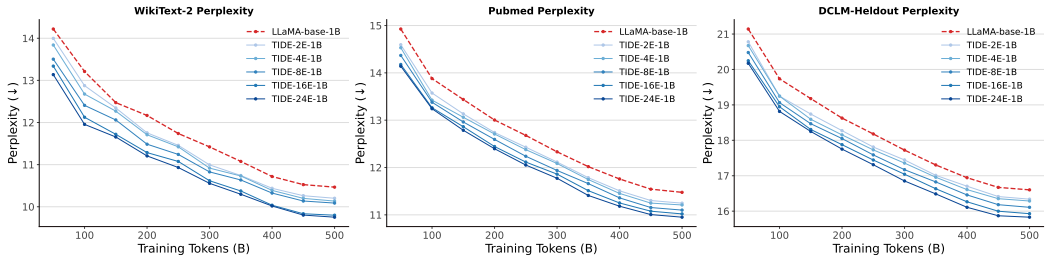

Figure 6: Wikitext-2, PubMed, DCLM validation PPL as a function of training tokens indicating monotonic improvement across increasing K blocks of TIDE variants.

3.4 Computational and Memory Overhead of TIDE

In TIDE, each $M_k(v) = \mathrm{RMSNorm}(E_k[v])$ is a single embedding lookup followed by RMSNorm and *contributes no matrix multiplications*, so the per-layer overhead reduces to one $(K{+}1)$-way softmax router and a weighted sum of d_b-dimensional vectors. This is negligible relative to the baseline FFN. More importantly, every E_k is indexed by discrete token identity v independent of h^ℓ, so once training completes the EMBEDDINGMEMORY tables are static and can be 4-bit quantized (negligible performance impact) and offloaded to SSD for on-demand asynchronous prefetch augmented with appropriate caching mechanism. As Figure 7 shows, this maintains the *effective VRAM footprint* of TIDE similar as LLaMA-Base-1B level (1.03 GB in 8-bit) while the SSD footprint scales from 0 to 3.152 GB from $K{=}0$ to $K{=}24$. Additional details regarding inference overhead and MEMORYBLOCKS compression techniques can be found in Appendix I and J.

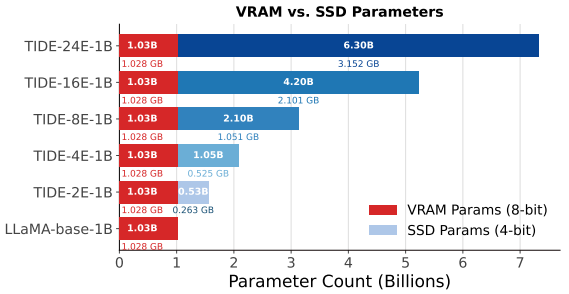

Figure 7: VRAM & SSD parameter breakdown across LLaMA-Base-1B and TIDE-1B model family with $K \in \{2, 4, 8, 16, 24\}$.

Table 2: Benchmark results for LLaMA-Base and TIDE variants at 750M, 1B and 3B parameter scales. PPL is LAMBDA perplexity (lower is better); BoolQ and LAMBADA use accuracy; all other columns use normalized accuracy (%). Best results per scale are **bolded**. *Total* counts all trained parameters; *Active* counts VRAM-resident parameters at inference.

Model	Params (Total / Active)	PPL ↓	ARC-C	ARC-E	BoolQ	HellaSwag	LAMBADA	OBQA	PIQA	SciQ	Average
750M Parameters											
LLaMA-Base	0.74B / 0.74B	5.63	34.6	60.4	**63.5**	60.9	62.8	36.8	73.8	85.1	59.7
TIDE-8E-750M	2.31B / 0.74B	**5.18**	**36.0**	**61.4**	63.0	**62.6**	**64.9**	**37.2**	**74.8**	**85.8**	**60.7**
1B Parameters											
LLaMA-Base	1.03B / 1.03B	5.19	37.5	64.4	61.7	63.9	64.6	37.6	74.9	86.9	61.4
TIDE-2E-1B	1.56B / 1.03B	4.97	37.6	65.7	68.7	64.9	65.1	36.4	75.5	87.1	62.6
TIDE-8E-1B	3.13B / 1.03B	4.89	37.5	64.5	69.3	65.3	64.7	**40.8**	75.5	86.6	63.0
TIDE-16E-1B	5.23B / 1.03B	4.78	38.7	65.5	**69.7**	65.3	65.7	37.8	75.9	**87.9**	63.3
TIDE-24E-1B	7.33B / 1.03B	**4.60**	**38.9**	**66.3**	69.5	**66.3**	**66.4**	37.2	**77.3**	87.2	**63.7**
3B Parameters											
LLaMA-Base	2.94B / 2.94B	4.00	41.2	74.8	69.0	71.9	69.4	40.2	78.1	**93.3**	67.2
TIDE-8E-3B	6.10B / 2.94B	**3.86**	**44.3**	**75.5**	**72.3**	**72.2**	**70.2**	**40.6**	**78.3**	93.2	**68.3**

4 Experiments and Ablation Studies

4.1 Performance Benchmarking of TIDE and Standard Transformer.

➢ **Perplexity and Training Dynamics:** TIDE introduces a parallel additive EMBEDDINGMEMORY pathways within conventional transformers to address the challenges associated with rare tokens and contextual collapse (Section 3.3.3, 3.3.2). Here, we first investigate the influence of token-indexed memory's ability to improve the language modeling quality of standard transformers. Figure 6 presents the validation perplexity on three datasets - Wikitext [Merity et al., 2016], PubMed [Jin et al., 2019] and DCLM [Li et al., 2024] held-out corpora as a function of total training tokens for

LLaMa-Base-1B and TIDE-1B with $K \in \{2, 4, 8, 16, 24\}$. Clearly, each TIDE variant *strictly outperforms* LLaMa-Base-1B monotonically from $K = 2$ to $K = 24$ without saturation. Performance gap opens early during training where with 100B tokens TIDE with merely $2 - 4$ MEMORYBLOCKs already matches the perplexity baseline reaches with 200B tokens, indicating that the additional gradient pathways translate to faster effective convergence.

➢ **Influence of K across across *Rare, Mid, and Common* tokens:** While perplexity based evaluation provide an overall performance benefit of TIDE, a natural question arises from Proposition 3.2 as: *Do these K* MEMORYBLOCK *pathways empirically benefits rare and common tokens equally, and does the marginal benefit of additional K scale with token frequency?* Figure 8 decomposes held-out cross-entropy loss across *rare, mid, and common* bins as a function of increasing K. It can be observed that the absolute loss reduction over LLaMA-Base-1B is largest on rare tokens, moving from $K = 0$ to $K = 24$ reduces rare-bin loss from 6.671 to 6.250 nats (-0.421) compared to only -0.075 nats on common bin which is a $5.6\times$ difference in absolute gain. In addition, the per-block marginal benefit (the slope of each curve) is $3.7\times$ steeper on rare tokens than on common tokens, illustrating the alignment with Section 3.3.2. Note that even the smallest configuration ($K = 2$) can deliver $\sim$55% of the total rare-token improvement obtained at $K = 24$ (also reflected in PPL in Figure 6), suggesting that the bulk of the benefit can be achieved with modest 2-4 memory blocks.

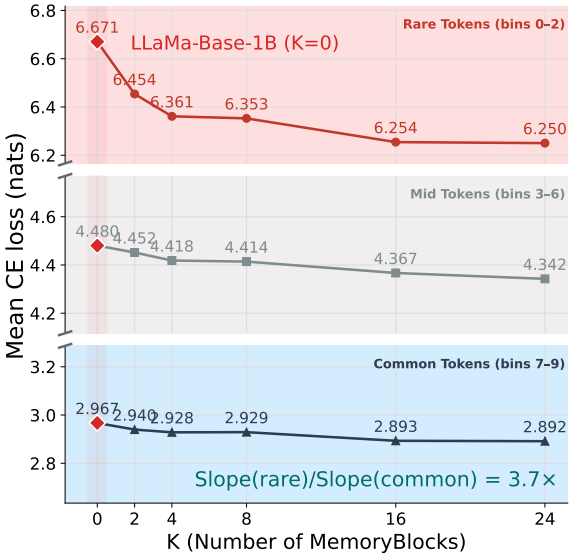

Figure 8: Mean cross entropy loss across *rare, mid,* and *common* tokens with increasing K MEMORYBLOCKs.

Table 3: Parameter-matched comparison at the 1B scale (200B training tokens). *Total* counts all trained parameters; *Active* counts parameters that must reside in VRAM at inference. Rare/Common CE are mean cross-entropy (nats) over rare (Bins 0–2) and common (Bins 7–9) tokens.

Model	Total Params	Active Params	WikiText PPL ↓	PubMed PPL ↓	Rare CE ↓	Common CE ↓
LLaMa-Base-1B	1.03B	1.03B	12.167	13.003	7.858	2.818
LLaMa-WideEMB	3.17B	3.17B	12.113	12.933	7.942	2.669
LLaMa-WideFFN	3.01B	3.01B	**10.892**	**12.106**	7.794	**2.287**
TIDE-8E-1B	3.13B	1.03B	11.485	12.593	**7.154**	2.750

➢ **TIDE and Downstream Task Performance:** Table 2 reports zero-shot accuracy on a suite of eight benchmarks (ARC-C, ARC-E, BoolQ, HellaSwag, LAMBADA, OBQA, PIQA, SciQ) across 750M, 1B, and 3B parameter scales of LLaMa-Base and TIDE variants. Across all settings, TIDE variants consistently outperform standard transformer baselines, confirming the robustness of our proposed architecture. More specifically, at the 1B scale, TIDE improves the average score from 61.4 (Base) to 63.7 ($K = 24$), a $+2.3\%$ absolute gain, with monotonic improvement in K on the perplexity column and on six of eight downstream tasks. On the rare-term-dense PubMedQA [Jin et al., 2019], TIDE-8E-1B improves accuracy from 27.44 (Base) to 29.69, an 8.2% relative gain.

➢ **Parameter-Matched Comparison:** TIDE-8E-1B adds 2.10B MEMORYBLOCK parameters to a 1.03B backbone, raising the question: *how much of* TIDE*'s gain comes from the architecture rather than the extra parameters?* We train two baselines matched in *total* parameters to TIDE-8E-1B with the same 200B-token budget: (a) LLaMa-WideEMB, which enlarges the primary embedding to $(K+1)|\mathcal{V}|/(|\mathcal{V}|+d)$ times its width with a down-projection to d; and (b) LLaMa-WideFFN, which increases the FFN expansion ratio in every layer. Table 3 reports the evaluation across these baselines. First, simply enlarging the embedding table does not reproduce TIDE's gains: LLaMa-WideEMB barely moves perplexity and its rare-decile loss slightly *worsens* (7.942 vs. 7.858), whereas TIDE reduces it to 7.154. Second, LLaMa-WideFFN achieves lower overall perplexity than TIDE-8E-1B, but it does so with $\sim 3\times$ more *active* (VRAM-resident) parameters and correspondingly higher per-token

compute, and its improvement is concentrated on common tokens (common-decile CE 2.287) while barely changing rare tokens (7.794). Consistently, its relative gain over `Base` is smaller on rare-token-heavy PubMed (6.9%) than on WikiText (10.5%). TIDE instead delivers the largest rare-token improvement of all variants at the same active-parameter cost as `Base`, since MEMORYBLOCKs are static, token-indexed, and can be offloaded from VRAM.

4.2 A Deeper Investigation of MEMORYBLOCKs and the NULL Bank.

The performance results in Section 4.1 establish that TIDE K-pathways provide informative signal beyond the contextual residual stream. We now turn the investigation inward to understand the information stored across MEMORYBLOCKs and per-layer router dynamics after training.

➢ **Distance between Primary Embedding and MEMORYBLOCKs:** We first aim to investigate if TIDE's K-MEMORYBLOCKs converge to substantially distinct subspaces or collapse to replicate the base embedding E. Figure 9 (a) reports mean cosine distance $(1 - \frac{1}{|\mathcal{V}|}\sum_v \cos(A[v], B[v])$ between every pair of embedding tables in TIDE-8E-1B. We make two key observations: (a) every M_k without any explicit diversity loss, is highly distant from E (mean cosine distance to E ranges from 0.65 to 0.99), confirming MEMORYBLOCKs do not replicate the input-embedding subspace but encode complementary token-identity signal; (b) inter-M_k distance is relatively smaller, indicating convergence of K blocks to overlapping but non-collapsed subspaces.

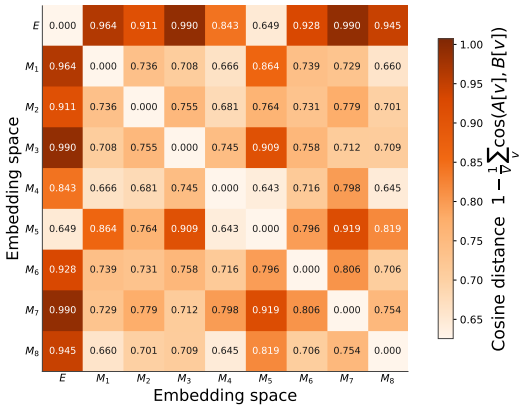

Figure 9: Mean cosine distance between primary embedding E and 8 MEMORYBLOCKs.

➢ **Bin-wise Router Statistics for MEMORYBLOCKs and the NULL Bank:** In reference to our Proposition 3.1 which state that TIDE asymptotically generalizes the standard transformer through the NULL bank, an empirical question persists: *How does the router actually utilize this NULL bank, and does it do so in a token-aware manner?* Figure 10 reports the mean routing weight $\bar{\alpha}_k$ allocated to each memory block M_k and to the NULL bank from last layer stratified by frequency bin. We highlight two key observations: (a) the NULL-bank weight is *monotonically non-decreasing in token frequency*: it rises from $\bar{\alpha}_{\text{NULL}} = 0.530$ on the rarest decile to 0.889 on the most common decile. The router has therefore learned to open the gate and admit substantial memory-bank mass ($1 - \bar{\alpha}_{\text{NULL}} \approx 0.47$) for rarest tokens in comparison to common tokens; (b) the router weight is non-uniform across blocks where M_5 carries an outsized share on rare tokens ($\bar{\alpha}_5 \approx 0.31$) before collapsing to near-zero on common tokens wile M_4 specializes for mid-decile tokens illustrating that distinct banks specialize to distinct frequency regimes *rather than redundantly co-firing*.

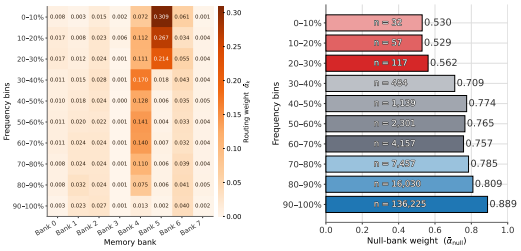

Figure 10: Bin-wise mean router weights $\bar{\alpha}_k$ across MEMORYBLOCKs (left) and the NULL bank (right), stratified by token frequency decile.

5 Conclusion

In this work, we propose TIDE, a transformer architecture that addresses two empirically established failure modes of standard LLMs: gradient starvation of rare tokens and contextual collapse of semantically distinct tokens. We introduce EMBEDDINGMEMORY, an ensemble of K independent MEMORYBLOCKs that map token indices directly to semantic vectors, injected at every layer via a depth-conditioned router with a NULL bank. TIDE provides each transformer layer with a persistent, token-specific signal that is immune to contextual collapse by construction. We *theoretically* and *empirically* establish the benefits of TIDE in addressing the issues associated with single-token identity injection. With extensive experiments across different model scale, we found TIDE consistently *improves performance* across multiple language modeling and downstream tasks.

References

S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022a.

S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, et al. Improving language models by retrieving from trillions of tokens. In *Proceedings of ICML*, pages 2206–2240, 2022b.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

D. Chen and W.-t. Yih. Open-domain question answering. In A. Savary and Y. Zhang, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 34–37, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-tutorials.8. URL `https://aclanthology.org/2020.acl-tutorials.8/`.

D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang, and F. Wei. Knowledge neurons in pretrained transformers. In *Proceedings of ACL*, pages 8493–8502, 2022.

G. Dar, M. Geva, A. Gupta, and J. Berant. Analyzing transformers in embedding space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16124–16170, 2023.

N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. `https://transformer-circuits.pub/2021/framework/index.html`.

M. Geva, R. Schuster, J. Berant, and O. Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, 2021.

M. Geva, A. Caciularu, K. Wang, and Y. Goldberg. 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, 2022a.

M. Geva, A. Caciularu, K. R. Wang, and Y. Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of EMNLP*, pages 30–45, 2022b.

A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwinska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. P. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476, 2016. URL `https://api.semanticscholar.org/CorpusID:205251479`.

K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. Retrieval augmented language model pretraining. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

X. O. He. Mixture of a million experts. *arXiv preprint arXiv:2407.04153*, 2024.

H. Inan, K. Khosravi, and R. Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proceedings of ICLR*, 2017.

G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, et al. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251): 1–43, 2023.

A. Jaiswal, L. Yin, Z. Zhang, S. Liu, J. Zhao, Y. Tian, and Z. Wang. From galore to welore: How low-rank weights non-uniformly emerge from low-rank gradients. *arXiv preprint arXiv:2407.11239*, 2024.

A. Jaiswal, L. Hannah, H.-B. Kim, D. Hoang, A. Kundu, M. Farajtabar, and M. Cho. Memoryllm: Plug-n-play interpretable feed-forward memory for transformers. In *arXiv preprint arXiv:2602.00398*, 2026.

S. Jie, Y. Tang, K. Han, Y. Li, D. Tang, Z.-H. Deng, and Y. Wang. Mixture of lookup experts. *arXiv preprint arXiv:2503.15798*, 2025.

Q. Jin, B. Dhingra, Z. Liu, W. Cohen, and X. Lu. Pubmedqa: A dataset for biomedical research question answering. 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 2567–2577, 2019.

V. Karpukhin, B. Oguz, S. Min, P. S. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020.

A. Kaushal, T. Vaidhya, and I. Rish. Lord: Low rank decomposition of monolingual code llms for one-shot compression. *arXiv preprint arXiv:2309.14021*, 2023.

U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

G. Lample, A. Sablayrolles, M. Ranzato, L. Denoyer, and H. Jégou. Large memory layers with product keys. *Advances in Neural Information Processing Systems*, 32, 2019.

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020a.

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020b.

J. Li, A. Fang, G. Smyrnis, M. Ivgi, M. Jordan, S. Gadre, H. Bansal, E. Guha, S. Keh, K. Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.

Y. Li, Y. Yu, Q. Zhang, C. Liang, P. He, W. Chen, and T. Zhao. Losparse: Structured compression of large language models based on low-rank and sparse approximation. In *International Conference on Machine Learning*, pages 20336–20350. PMLR, 2023.

K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, pages 17359–17372, 2022.

K. Meng, A. S. Sharma, A. Andonian, Y. Belinkov, and D. Bau. MEMIT: Mass-editing memory in a transformer. In *Proceedings of ICLR*, 2023.

S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

E. Nichani, J. D. Lee, and A. Bietti. Understanding factual recall in transformers via associative memories. *arXiv preprint arXiv:2412.06538*, 2024.

F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard, et al. Kilt: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, 2021.

C. Pilgrim and T. T. Hills. Bias in zipf's law estimators. *Scientific reports*, 11(1):17309, 2021.

O. Press and L. Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, 2017.

A. Roberts, C. Raffel, and N. Shazeer. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

R. Sadhukhan, S. Cao, H. Dong, C. Zhao, A. Purpura-Pontoniere, Y. Tian, Z. Liu, and B. Chen. Stem: Scaling transformers with embedding modules. In *arXiv preprint arXiv:2601.10639*, 2026.

T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

R. Sennrich, B. Haddow, and A. Birch. 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, 2016.

S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks. *Advances in neural information processing systems*, 28, 2015.

S. Sukhbaatar, E. Grave, G. Lample, H. Jegou, and A. Joulin. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019.

A. Virmaux and K. Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in neural information processing systems*, 31, 2018.

B. Wang, W. Ping, P. Xu, L. McAfee, Z. Liu, M. Shoeybi, Y. Dong, O. Kuchaiev, B. Li, C. Xiao, et al. Shall we pretrain autoregressive language models with retrieval? a comprehensive study. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 7763–7786, 2023a.

H. Wang, S. Agarwal, Y. Tanaka, E. Xing, D. Papailiopoulos, et al. Cuttlefish: Low-rank model training without all the tuning. *Proceedings of Machine Learning and Systems*, 5:578–605, 2023b.

J. Weston, S. Chopra, and A. Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

B. Zhang and R. Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.

A Background Work

A.1 Memory Augmented Architectures.

Memory-augmented models are designed to expand a model's effective parameter space without incurring large computational overhead. Early work on memory networks was introduced by Weston et al. [2014], and later extended with fully end-to-end trainable variants with Sukhbaatar et al. [2015]. Neural Turing Machines [Graves et al., 2014, 2016] incorporate an external, trainable memory that works alongside other neural components to simulate a differentiable, trainable computing system. Product-key networks [Lample et al., 2019] improve the efficiency and scalability of memory retrieval and propose a key-value memory layer that can scale to very large sizes while keeping exact search on the key space. More recently, PEER [He, 2024] has advanced these ideas by replacing traditional vector-based memory values with rank-one matrices, linking memory-augmented architectures with mixture-of-experts models.

Accurate factual generation remains a critical objective for generative models, often evaluated using open-domain question answering benchmarks [Chen et al., 2017, Chen and Yih, 2020] and other tasks requiring substantial knowledge [Petroni et al., 2021]. Models that can effectively encode factual knowledge from training data are better equipped to provide correct responses to knowledge-intensive queries. While larger models generally demonstrate improved factual accuracy [Roberts et al., 2020, Brown et al., 2020], hallucination remains a persistent challenge. One effective approach for mitigating this issue is retrieval-augmented generation, which leverages external knowledge sources to improve factual consistency [Lewis et al., 2020a, Karpukhin et al., 2020, Khandelwal et al., 2019]. Several language models have incorporated text retrieval from the pretraining stage. REALM [Guu et al., 2020] augments a BERT model with one retrieval step to solve QA tasks. Retro Borgeaud et al. [2022a] enhances auto-regressive decoding with multiple rounds of retrieval, once per 64 tokens. The retrieved texts are injected through a two-layer encoder and then several cross-attention layers in the decoder. Retro++ [Wang et al., 2023a] explores the scalability of Retro by reproducing Retro up to 9.5B parameters. Meanwhile, several models are adapted to retrieval in the finetuning stage. WebGPT [Nakano et al., 2021] learns to use search engine through imitation learning in a text-based web-browsing environment. Toolformer [Schick et al., 2023] performs decoding with multiple tools including search engine, and the finetuning data is labeled by the language model itself.

A.2 Understanding Feed-Forward Networks in Transformers.

Several studies have investigated the role of feed-forward networks (FFNs) in transformers, particularly their contribution to storing and retrieving knowledge learned during pretraining. Geva et al. [2021] demonstrated that FFNs can be interpreted as key–value memories that activate on specific lexical or semantic patterns, while follow-up work showed that FFNs promote vocabulary-level concepts during prediction [Geva et al., 2022a]. Additional related analyses in embedding space further explored how FFN activations correspond to linguistic features and factual recall [Dar et al., 2023, Nichani et al., 2024]. Within their framework, the first layer acts as a pattern detector (“keys") while the second layer projects specific information into the residual stream (“values"). This modularity is evidenced by the identification of specific “knowledge neurons" responsible for storing distinct facts. More broadly, the interpretation of neural networks as associative or persistent memory systems connects this line of work to earlier memory-augmented architectures [Sukhbaatar et al., 2019]. However, these analyses rely on contextualized residual activations and require extensive post-hoc mining of calibration data, making the inferred query space indirect and difficult to interpret. Furthermore, since FFNs operate exclusively on its contextualized residual stream, their ability to distinguish tokens is mathematically bottlenecked when distinct token appears in identical syntactic context which leads to the contextual collapse problem. Recently, MoLE [Jie et al., 2025], illustrates that in mixture-of-experts (MoEs), majority of experts can be trained directly with token-level input embeddings. Following the static routing concept, MemoryLLM [Jaiswal et al., 2026] completely decouples FFNs from the contextual residual stream by directly training a layer-local and token-indexed embedding table to enhance interpretability and reduce compute. Concurrently, in the STEM [Sadhukhan et al., 2026] architecture, FFN is partially replaced to embedding table, with the substitution occurring at the up-projection layer. TIDE builds upon this token-level intuition but fundamentally diverges from standard FFNs. Instead of relying on contextual mixtures vulner-

able to collapse, TIDE bypasses this entirely by injecting a context-free token identity directly into residual stream at every depth.

A.3 Advancements with Embedding and Modern LLMs.

Standard transformer models rely on a single-injection assumption where token embeddings are looked up once at the input layer and subsequently faded out. Since language vocabularies strictly obey Zipf's law [Zipf, 1949, Pilgrim and Hills, 2021], majority of tokens infrequently appear in the training corpus. Sub-word tokenization [Sennrich et al., 2016] is introduced to mitigate out-of-vocabulary issue, yet they do not resolve the fundamental long-tail distribution of tokens, which continues to degrade the performance of contextualized embeddings on rare words. Under standard stochastic gradient descent, this skewed distribution leads to gradient starvation for rare tokens. Embedding sharing [Inan et al., 2017, Press and Wolf, 2017] attempt to stabilize training of embedding by tying input and output embedding weights, allowing input representations to benefit directly from the richer gradient signal of the pre-softmax layer. However, simply sharing parameters between the input and output layers does not structurally resolve the gradient starvation on low-frequency tokens. TIDE directly solve this gradient starvation bottleneck by utilizing independent memory blocks which lead to amplification of gradient signals to token representations, disproportionately benefiting rare tokens.

B Details of Frequency Bins Generated from Vocabulary

The WikiText-103 training split is tokenized with the LLaMA-3 tokenizer ($|\mathcal{V}| = 128{,}256$, sequence length $T = 2{,}048$), producing a token stream of $\sim$120M tokens over $\sim$58k sequences, of which 65,569 vocabulary entries appear at least once. Raw occurrence counts are then passed through a structural filter that removes BOS/EOS special tokens, pure-whitespace tokens, and non-alphanumeric punctuation. In total, 28 tokens (0.04% of observed types) are removed, leaving 65,541 token types for the binning procedure.

Table 4: Frequency decile bin reference for WikiText-103 with the LLaMA-3 BPE tokenizer (128K vocabulary) after structural filtering. Each bin contains $\approx$6,554 token types ranked by corpus frequency. Representative example tokens are drawn from each tier for semantic illustration.

Bin	Freq. range	Types	Description	Role
0	1–2	6,555	Hapax & near-hapax tokens	**Rare**
1	2–6	6,554	Domain-specific, rare names	**Rare**
2	6–20	6,554	Uncommon words, rare entities	**Rare**
3	20–61	6,554	Infrequent content words	**Mid**
4	61–133	6,554	Occasional content words	**Mid**
5	133–240	6,554	Moderate-frequency words	**Mid**
6	240–416	6,554	Fairly common content words	**Mid**
7	416–769	6,554	Common function + content words	**Common**
8	769–1,856	6,554	High-frequency content words	**Common**
9	1,856–999,999	6,554	Highest-frequency (below cap)	**Common**
Rare examples (Bins 0–2)	`cefuroxime`, `morgan`, `Produto`, `Teotihuacan`, `toujours`			
Mid examples (Bins 3–6)	`volcano`, `diocese`, `battalion`, `peninsula`, `sculptor`, `harbour`			
Common examples (Bins 7–9)	`there`, `their`, `also`, `however`, `first`, `time`			

Decile assignment of Token Types: The 65,541 cleaned types are sorted by ascending corpus frequency and partitioned into $B = 10$ equal-cardinality bins, each containing $\approx$6,554 types, with bin index assigned as:

$$b(v) \;=\; \min\!\left(\left\lfloor \frac{\mathrm{rank}(v)}{|\mathcal{V}_{\mathrm{clean}}|} \cdot B \right\rfloor,\; B-1\right). \tag{12}$$

Bin assignment is determined by rank alone while the absolute frequencies establish the ordering. Crucially, while every bin contains the same number of token *types*, the bins account for vastly different shares of the training token *stream* under Zipf's law. Throughout this paper, Bins $\{0-2\}$ are referred to as **rare**, Bins $\{3-6\}$ as **mid-frequency**, and Bins $\{7-9\}$ as **common** tokens.

C Gradient Starvation Bound: Derivation of Equation (1)

The loss $\mathcal{L}_s$ depends on e_v only through positions in batch_s that equal v. If $v \notin \mathrm{batch}_s$, then $\partial \mathcal{L}_s / \partial e_v = \mathbf{0}$ exactly. Formally, we can write as:

$$\nabla_{e_v} \mathcal{L}_s = \mathbf{0} \quad \text{whenever } v \notin \mathrm{batch}_s. \tag{13}$$

Let's define $X_s := \mathbf{1}[v \in \mathrm{batch}_s]$. Since each of the BT positions is drawn i.i.d. from $\{f_v\}$, the event $\{v \notin \mathrm{batch}_s\}$ requires all BT draws to avoid v, each with probability $(1 - f_v)$. Therefore:

$$\Pr[v \in \mathrm{batch}_s] \;=\; \mathbb{E}[X_s] \;=\; 1 - (1 - f_v)^{BT} \;\le\; f_v \cdot B \cdot T, \tag{14}$$

where the inequality applies the Bernoulli bound $(1 - f_v)^{BT} \ge 1 - f_v BT$, valid for $f_v \in [0, 1]$.

Next, we bound the cumulative squared gradient using equation (13), as $\|\nabla_{e_v} \mathcal{L}_s\|^2 \le G^2 \cdot X_s$. Taking expectations and summing over τ steps:

$$\mathbb{E}\left[\sum_{s=1}^{\tau} \|\nabla_{e_v} \mathcal{L}_s\|^2\right] \le \sum_{s=1}^{\tau} G^2 \cdot \mathbb{E}[X_s] \;\le\; \tau \cdot f_v \cdot B \cdot T \cdot G^2, \tag{15}$$

which completes the derivation of (1).

C.1 Understanding the Ratio of Gradient for Rare and Common Tokens

Let v be a rare token with $f_v = \varepsilon \ll 1/(BT)$ and u a common token with $f_u \ge c > 0$. Using the aforementioned derivation, we have

$$\mathbb{E}\left[\sum_{s=1}^{\tau} \|\nabla_{e_v} \mathcal{L}_s\|^2\right] \;\le\; \tau \cdot \varepsilon \cdot BT \cdot G^2. \tag{16}$$

To determine the lower bound for common frequency tokens, we assume a standard non-degeneracy condition: whenever token u appears in batch s, the per-step squared gradient norm satisfies $\|\nabla_{e_u} \mathcal{L}_s\|^2 \ge G_{\min}^2 > 0$ on the event $\{u \in \mathrm{batch}_s\}$. This holds throughout training whenever the cross-entropy loss has not been minimized on token u.

Defining $X_s^{(u)} := \mathbf{1}[u \in \mathrm{batch}_s]$ and using $\|\nabla_{e_u} \mathcal{L}_s\|^2 \ge G_{\min}^2 \cdot X_s^{(u)}$, we take expectations and sum over τ steps:

$$\mathbb{E}\left[\sum_{s=1}^{\tau} \|\nabla_{e_u} \mathcal{L}_s\|^2\right] \;\ge\; \tau \cdot \Pr[u \in \mathrm{batch}] \cdot G_{\min}^2 \;=\; \tau\big(1 - (1 - f_u)^{BT}\big) G_{\min}^2. \tag{17}$$

Since $f_u \ge c > 0$ and $1 - (1 - x)^n$ is non-decreasing in x:

$$1 - (1 - f_u)^{BT} \;\ge\; 1 - (1 - c)^{BT} \;=:\; \kappa \;>\; 0, \tag{18}$$

where κ is a strictly positive constant depending only on c, B, T. Substituting it into (17) gives:

$$\mathbb{E}\left[\sum_{s=1}^{\tau} \|\nabla_{e_u} \mathcal{L}_s\|^2\right] \;\ge\; \tau\, \kappa\, G_{\min}^2. \tag{19}$$

To determine the ratio between rare and common tokens, we divide (16) by (19):

$$\frac{\mathbb{E}\left[\sum_s \|\nabla_{e_v} \mathcal{L}_s\|^2\right]}{\mathbb{E}\left[\sum_s \|\nabla_{e_u} \mathcal{L}_s\|^2\right]} \;\le\; \frac{\tau\, \varepsilon\, BT\, G^2}{\tau\, \kappa\, G_{\min}^2} \;=\; \frac{BT \cdot G^2}{\kappa \cdot G_{\min}^2} \cdot \varepsilon \tag{20}$$

Since BT, G^2, and $G_{\min}^2$ are fixed positive constants. By the first-order Taylor expansion for small c, we have:

$$\kappa \;=\; 1-(1-c)^{BT} \;\approx\; c\cdot BT\;, \tag{21}$$

so the prefactor satisfies:

$$\frac{BT\cdot G^2}{\kappa\cdot G_{\min}^2} \;=\; \frac{BT\cdot G^2}{(c\cdot BT)\cdot G_{\min}^2} \;=\; \frac{G^2}{c\cdot G_{\min}^2} \;=\; O\!\left(\frac{1}{c}\right), \tag{22}$$

and the full bound on the ratio is $O(\varepsilon/c)$, which completes the proof of (3).

Concrete evaluation on WikiText-103. With $\varepsilon = 8.3\times10^{-9}$ (Bin-0 rare tokens, $n_v = 1$), $c = 8.3\times10^{-3}$ (Bin-9 common token, $n_u \approx 10^6$), $B = 8$, $T = 2048$:

$$\kappa \;=\; 1-(1-c)^{BT} \;\approx\; 1-e^{-c\,BT} \;=\; 1-e^{-136} \;\approx\; 1, \tag{23}$$

$$\frac{\varepsilon}{c} \;=\; \frac{8.3\times10^{-9}}{8.3\times10^{-3}} \;=\; 10^{-6}. \tag{24}$$

Under the conservative assumption $G^2/G_{\min}^2 = 10$, the gradient signal accumulated by a Bin-0 hapax embedding is bounded above by 10^{-5} times that of a Bin-9 common token over the same training run, there exists a gradient disparity of five orders of magnitude.

D Full Proof of Proposition 2.2

We prove that for any collapsed pair $(u,v) \in \mathcal{C}_\delta^{(\ell)}$ and any target function $g : \mathcal{V} \to \mathbb{R}^d$ with $\|g(u)-g(v)\| = C > L_{\mathrm{FFN}}\delta$, no setting of the FFN weights can approximate g to error less than $(C - L_{\mathrm{FFN}}\delta)/2$ on both tokens simultaneously.

By definition of the contextual collapse set, $(u,v) \in \mathcal{C}_\delta^{(\ell)}$ implies $\|h_u - h_v\| \le \delta$. Since FFNs are Lipschitz in its hidden-state input [Virmaux and Scaman, 2018], we have

$$\|\mathrm{FFN}(h_u) - \mathrm{FFN}(h_v)\| \;\le\; L_{\mathrm{FFN}}\,\delta. \tag{25}$$

The FFN outputs for u and v must therefore lie within a ball of radius $L_{\mathrm{FFN}}\delta$ of each other — they cannot be far apart, regardless of how the weights W_1, W_2 are chosen.

Using triangle inequality on the target separation, we can write the target separation as:

$$\begin{aligned} C &= \|g(u)-g(v)\| \\ &= \|g(u)-\mathrm{FFN}(h_u)+\mathrm{FFN}(h_u)-\mathrm{FFN}(h_v)+\mathrm{FFN}(h_v)-g(v)\| \\ &\le \|g(u)-\mathrm{FFN}(h_u)\| + \|\mathrm{FFN}(h_u)-\mathrm{FFN}(h_v)\| + \|\mathrm{FFN}(h_v)-g(v)\|. \end{aligned} \tag{26}$$

By substituting Lipschitz bound (25) into (26):

$$C \;\le\; \|g(u)-\mathrm{FFN}(h_u)\| + L_{\mathrm{FFN}}\,\delta + \|g(v)-\mathrm{FFN}(h_v)\|. \tag{27}$$

Rearranging (27) to isolate error terms:

$$\|g(u)-\mathrm{FFN}(h_u)\| + \|g(v)-\mathrm{FFN}(h_v)\| \;\ge\; C - L_{\mathrm{FFN}}\,\delta. \tag{28}$$

The left-hand side is the sum of two non-negative approximation errors. When $C > L_{\mathrm{FFN}}\delta$, the right-hand side is strictly positive, so at least one of the two errors is positive. Specifically, since the maximum of two non-negative numbers is at least half their sum:

$$\max\!\big\{\|g(u)-\mathrm{FFN}(h_u)\|,\; \|g(v)-\mathrm{FFN}(h_v)\|\big\} \;\ge\; \frac{C - L_{\mathrm{FFN}}\,\delta}{2}. \tag{29}$$

when $C > L_{\mathrm{FFN}}\,\delta$, the right side is strictly positive, which completes the proof.

Three quantities govern the bound among which none of which are under the FFN's control:

- **Target separation** (C): The required difference $\|g(u)-g(v)\|$ between the optimal representations of tokens u and v. This is determined by what the downstream task needs — for example, the grammatical-class distance between *"their"* (possessive determiner) and *"there"* (locative adverb) is fixed by the language, not by the model's architecture.

- **Input proximity** (δ): The distance $\|h_u - h_v\|$ between the hidden states that the attention layer produces for u and v. When two tokens appear in nearly identical contexts with same surrounding words and same syntactic position - attention cannot distinguish them, and δ is small. The FFN receives h_u and h_v as inputs; it cannot *choose* to receive different inputs.
- **FFN change limit** (L_{FFN}): The Lipschitz constant controls how rapidly the FFN output can change per unit change in input. While L_{FFN} depends on the weights and could in principle be made large, doing so causes exploding gradients and training instability. A large L_{FFN} amplifies the FFN's response to *every* input perturbation which is not only limited to the gap between h_u and h_v. This sharply degrades performance on the majority of tokens whose hidden states are not collapsed.

E Full Proof of Proposition 3.1

We prove that for any $\epsilon > 0$ there exist finite router parameters W_r^ℓ such that $\left\|m^\ell(v)\right\| < \epsilon$ for all $v \in \mathcal{V}$ and $\ell \in \{1, \ldots, L\}$.

Given $M_{K+1}(v) = \mathbf{0}$ by definition, for any router weight $\alpha_{K+1}^\ell \in (0, 1)$:

$$\alpha_{K+1}^\ell \cdot M_{K+1}(v) = \alpha_{K+1}^\ell \cdot \mathbf{0} = \mathbf{0}.$$

This ensures that null bank contributes nothing to TIDE. The memory term therefore simplifies to:

$$m^\ell(v) = \sum_{k=1}^{K+1} \alpha_k^\ell \, M_k(v) = \sum_{k=1}^{K} \alpha_k^\ell \, M_k(v).$$

By the softmax constraint (8), $\sum_{k=1}^{K} \alpha_k^\ell = 1 - \alpha_{K+1}^\ell$. Applying the triangle inequality, we can bound the memory norm as:

$$\left\|m^\ell(v)\right\| \;\le\; \sum_{k=1}^{K} \alpha_k^\ell \, \|M_k(v)\| \;\le\; \left(1 - \alpha_{K+1}^\ell\right) \cdot C, \tag{30}$$

where $C = \max_{v \in \mathcal{V},\, k \le K} \|M_k(v)\| < \infty$.

Next, we express the active weight sum in terms of the null logit. Upon setting $z_{K+1}^\ell = s > 0$ and $z_k^\ell = 0$ for all $k \le K$. By softmax:

$$1 - \alpha_{K+1}^\ell = \frac{K}{K + e^s}. \tag{31}$$

As $s \to \infty$, $K/(K + e^s) \to 0$, so the total active bank weight vanishes.

Substituting (31) into (30): $\left\|m^\ell(v)\right\| \le KC/(K+e^s)$. For any $\epsilon \in (0, C)$, solving $KC/(K+e^s) = \epsilon$ gives:

$$s^* = \log\left(\frac{K(C - \epsilon)}{\epsilon}\right). \tag{32}$$

Now, we have: $KC/(K + e^{s^*}) = KC\epsilon/KC = \epsilon$. Therefore $\left\|m^\ell(v)\right\| \le \epsilon$ uniformly over all v and ℓ for any $s \ge s^*$.

Note that, the threshold s^* is finite for any $\epsilon \in (0, C)$. Setting W_r^ℓ so that $W_r^\ell \tilde{n}^\ell \approx s^* \, \mathbf{e}_{K+1}$ is a finite parameter assignment under which $\left\|m^\ell(v)\right\| < \epsilon$ for all v and ℓ.

Additional Remark: From (31), $\sum_{k=1}^{K} \alpha_k^\ell = K/(K + e^{z_{K+1}^\ell})$ depends *only* on the null logit z_{K+1}^ℓ. A single large null logit jointly suppresses all K active banks through softmax competition facilitating the suppression degree of freedom to one scalar, regardless of K.

F Full Proof of Proposition 3.2: K-Pathway Gradient Amplification

For simplicity, Proposition 3.2 is stated for a simplified router over K active banks that excludes the null bank at slot $K+1$.

For a fixed token $v \in \mathcal{V}$ and let $e_v^{(k)}$ denote the row of embedding table E_k corresponding to v, for $k = 1, \ldots, K$. Define $X_s := \mathbf{1}[v \in \text{batch}_s]$. As established in Appendix C with Bernoulli Bound, we have:

$$\mathbb{E}[X_s] = 1 - (1 - f_v)^{BT} =: \kappa_v \leq f_v \cdot BT. \tag{33}$$

Given that the K MEMORYBLOCKs have no shared parameters, the gradient with respect to $e_v^{(k)}$ is identically zero whenever $v \notin \text{batch}_s$, and otherwise:

$$\nabla_{e_v^{(k)}} \mathcal{L}_s = \sum_{\ell=1}^{L} \alpha_k^{\ell} \cdot \frac{\partial \mathcal{L}_s}{\partial m^{\ell}(v)} \cdot \frac{\partial M_k(v)}{\partial e_v^{(k)}}, \tag{34}$$

where $\partial \mathcal{L}_s / \partial m^{\ell}(v)$ is the upstream gradient from layer ℓ's residual stream. Since $M_k(v)$ enters every layer, each block accumulates gradient contributions across all L layers.

During training, whenever $v \in \text{batch}_s$ and the loss has not been minimized on token v, we assume the standard non-degeneracy condition: for each k, there exists at least one layer ℓ^* such that

$$\left\| \nabla_{e_v^{(k)}} \mathcal{L}_s \right\|^2 \geq G_{\min}^2 > 0 \quad \text{on the event } \{v \in \text{batch}_s\}. \tag{35}$$

Since the K blocks are independent and each satisfies Equation 35, and $\nabla_{e_v^{(k)}} \mathcal{L}_s = \mathbf{0}$ exactly when $X_s = 0$ (token v absent from batch), we have $\left\| \nabla_{e_v^{(k)}} \mathcal{L}_s \right\|^2 \geq G_{\min}^2 \cdot X_s$. Summing across K blocks:

$$\sum_{k=1}^{K} \left\| \nabla_{e_v^{(k)}} \mathcal{L}_s \right\|^2 \;\geq\; K \cdot G_{\min}^2 \cdot X_s. \tag{36}$$

Taking expectations and summing over τ steps completes the proof of (10):

$$\mathbb{E}\left[\sum_{s=1}^{\tau} \sum_{k=1}^{K} \left\| \nabla_{e_v^{(k)}} \mathcal{L}_s \right\|^2 \right] \;\geq\; K \cdot \tau \cdot \kappa_v \cdot G_{\min}^2. \tag{37}$$

Additional Remark: The standard transformer upper bound gives $\mathbb{E}\left[\sum_s \|\nabla_{e_v} \mathcal{L}_s\|^2\right] \leq \tau \cdot f_v \cdot BT \cdot G^2$ from Appendix C. The TIDE lower bound is K times the analogous single-block lower bound, confirming a K-fold amplification under the assumption $G^2 / G_{\min}^2 = O(1)$.

G Additional Details for Proposition 3.3: Contextual Collapse and TIDE K-MEMORYBLOCKS

Proposition 3.3 state that for any collapsed pair $(u, v) \in \mathcal{C}_{\delta}^{(\ell)}$ *with* $\left\| h_u^{(\ell)} - h_v^{(\ell)} \right\| \leq \delta$*, and any target separation* $C > 0$*, there exist* EMBEDDINGMEMORY *parameters* $\{E_k\}_{k=1}^{K}$ *such that* $\|M_k(u) - M_k(v)\| = C$ *for any* $K \geq 1$.

From Equation 6, we have $M_k(v) = \text{RMSNorm}(E_k[v])$, where $E_k[v]$ is the row of $E_k \in \mathbb{R}^{|\mathcal{V}| \times d_b}$ selected by discrete index v.

The hidden state $h_v^{(\ell)}$ does not appear in (6), so $M_k(u)$ and $M_k(v)$ are independent of $\delta = \left\| h_u^{(\ell)} - h_v^{(\ell)} \right\|$. This stands in direct contrast to the FFN, for which Lipschitz continuity forces $\|\text{FFN}(h_u) - \text{FFN}(h_v)\| \leq L_{\text{FFN}} \delta$ regardless of the weights chosen, bounding the output separation from above by $L_{\text{FFN}} \delta$.

Since rows $E_k[u]$ and $E_k[v]$ are uncoupled parameters, the table E_k assigns one dedicated row per vocabulary entry with no joint constraint, so assigning one row places no restriction on the other, regardless of δ. $M_k(u)$ and $M_k(v)$ can be set to any two vectors in the output range of RMSNorm independently. In particular, for any target $C > 0$ there trivially exist row assignments such that $\|M_k(u) - M_k(v)\| = C$, regardless of $\delta = \left\| h_u^{(\ell)} - h_v^{(\ell)} \right\|$ and independently of L_{FFN}.

H Understanding Layer-wise Contribution of EMBEDDINGMEMORY

In TIDE, we propose to add EMBEDDINGMEMORY to the residual stream at every transformer layer, a natural question arises: *Is each layer's memory injection equally important, or does it contribute disproportionately at certain depths of the network?* To probe this, we sweep the layer index $\ell \in \{0, 1, \ldots, L-1\}$ of our TIDE-1B model and, at each sweep point, replace the routed memory contribution at layer ℓ alone with the zero vector while leaving the router and MEMORYBLOCKS pathway at every other layer untouched. Concretely, the standard TIDE forward pass at layer ℓ is given as per Equation 9 as:

$$h^\ell = \tilde{h}^\ell + \mathrm{FFN}\big(\tilde{n}^\ell\big) + m^\ell(v).$$

becomes $h^\ell = \tilde{h}^\ell + \mathrm{FFN}\big(\tilde{n}^\ell\big)$ for the single ablated layer ℓ, where $\tilde{h}_\ell = \mathrm{RMSNorm}(h_\ell + \mathrm{Attn}_\ell(h_\ell))$ and Mem_ℓ denotes the routed sum over the $K{=}24$ MEMORYBLOCKS. We then evaluate perplexity on WikiText-2, DCLM, and PubMed, repeating for every ℓ to obtain a per-layer marginal-contribution profile.

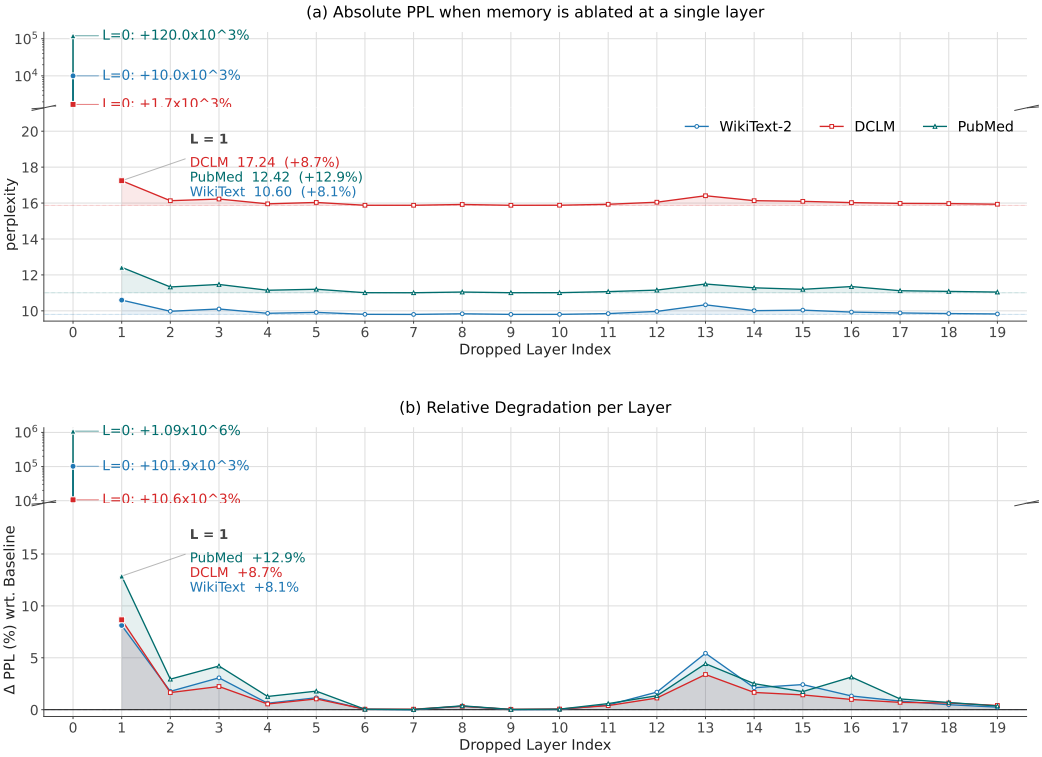

Figure 11: **Marginal contribution of each layer's memory injection in TIDE.** The routed EMBEDDINGMEMORY output at one layer ℓ is zeroed while all other layers retain their full pathway; we report perplexity on WikiText-2, DCLM, and PubMed. Relatively higher degradation in PubMed performance also aligns with our rare-token problem.

Memory contribution is intermittent, not monotone: Figure 11 reveals that the contribution of EMBEDDINGMEMORY is *not uniform across depth* of the trained TIDE checkpoint. Dropping layer 0 collapses the model entirely and the perplexity rises by more than $10^3\%$ on every dataset and reaches $1.09 \times 10^6\%$ on PubMed, indicating that the first memory injection carries an irreplaceable importance for the model. Layer 1 remains substantially load-bearing ($+8.1\%$ to $+12.9\%$ across datasets), suggesting a brief consolidation period during which token-identity information is propagated into the residual stream. After this, degradation falls sharply: every layer in the contiguous range $\ell \in [4, 12]$ lead to less than $\sim$2% PPL cost.

We additionally observe a clear secondary peak at layer 13, where layer dropping costs new spike in performance degradation. We interpret this pattern as evidence that token-identity information injected by early EMBEDDINGMEMORY layers *persists* in the residual stream for several intermediate

layers, during which any single memory contribution becomes redundant. Once this token-identity signal is consumed by the ongoing contextual computation, memory information is again required to refresh it at intermittent intervals.

I Understanding Decoding Cost with MEMORYBLOCKs

In this section, we aim to investigate how does our proposed architecture TIDE-1B empirically perform wrt. LLaMa-Base-1B in terms of decoding speed (ms/token). All experiments are reported using 1×B200 GPU averaged across 5,120 generated tokens.

Table 5: Token Decoding estimated for TIDE-1B variants in comparison to LLaMa-Base-1B transformer model.

	LLaMa-Base-1B	TIDE-2E-1B	TIDE-4E-1B	TIDE-8E-1B	TIDE-16E-1B	TIDE-24E-1B
Decoding Speed (ms/token)	11.085	11.236	11.854	12.688	12.901	13.422

J Investigating Compression Strategies for EMBEDDINGMEMORY

Our proposed TIDE architectures provide an opportunity to offload each static MEMORYBLOCKs within the EMBEDDINGMEMORY to storage devices in resource-constrained settings with asynchronous pre-fetching. This leads to the question: *How does the trade-off between VRAM and storage devices look like and what can be done to minimize* MEMORYBLOCK*s storage cost?*

We first estimate the total storage cost of MEMORYBLOCK as follows:

$$\text{Storage Size} = \text{vocab_size} \times \text{num_blocks} \times \text{hidden_dim} \times \text{bits_per_param} \tag{38}$$

For our TIDE-8E-1B model with 24 layers and 2048 hidden dimension which are trained with LLaMa-3.1 tokenizer with 128,256 vocabulary size, ∼4.2 GB of storage space is required for EMBEDDINGMEMORY with F16 precision. To address our question, we performed a preliminary investigation[2] of storage challenges of EMBEDDINGMEMORY from **two** different perspectives:

D1. Quantization of EMBEDDINGMEMORY, and

D2. Low Rank compression of individual MEMORYBLOCK,

J.1 Quantization of EMBEDDINGMEMORY.

Table 6: Performance comparison of TIDE-8E-1B with various low-precision of MEMORYBLOCKs.

Precision	Size (GB)	Wikitext-2 (↓)	PubMed (↓)	DCLM (↓)
16-bit	4.20 GB	10.088	11.100	16.108
8-bit	2.10 GB	10.089	11.100	16.113
4-bit	1.05 GB	10.263	11.277	16.343

J.2 Low Rank Compression of Token-wise MEMORYBLOCKs.

Several recent works [Li et al., 2023, Wang et al., 2023b, Kaushal et al., 2023] have explored the low-rank characteristics associated with weights and gradients to address storage demands and computational complexity linked to the large matrices of LLMs. For our TIDE-8E-1B model checkpoint with 8 MEMORYBLOCKs, each block holds an embedding table $M_k \in \mathbb{R}^{|\mathcal{V}| \times d_b}$ that maps a token index $v \in \mathcal{V}$ to a d_b-dimensional vector. A rank-r SVD decomposition of M_k yields two matrices

[2] An effective novel compression technique for EMBEDDINGMEMORY compression is out of the scope of this work. Our preliminary investigation reveals a high redundancy within the MEMORYBLOCKs and leaves sophisticated studies to capitalize these redundancies as future work.

$U \in \mathbb{R}^{|\mathcal{V}| \times r}$ and $V \in \mathbb{R}^{r \times d_b}$, and rather than storing M_k directly we can store the factored representation (U, V) provided r is small enough that the factored form has fewer parameters. We estimate the rank r, below which storage of (U, V) will save space as follows:

$$(|\mathcal{V}| \cdot r) + (r \cdot d_b) \;\leq\; |\mathcal{V}| \cdot d_b \quad \Longrightarrow \quad r \;\leq\; \frac{|\mathcal{V}| \cdot d_b}{|\mathcal{V}| + d_b}. \tag{39}$$

For TIDE-8E-1B with hidden dimension $d = 2048$, bottleneck dimension $d_b = 2048$, and the LLaMA-3.1 tokenizer of vocabulary size $|\mathcal{V}| = 128{,}256$, Equation 39 gives $r \leq 2015$, so any uniform rank reduction of at least $\sim$2% across the eight MEMORYBLOCKs is sufficient to reduce storage relative to the dense parameterization.

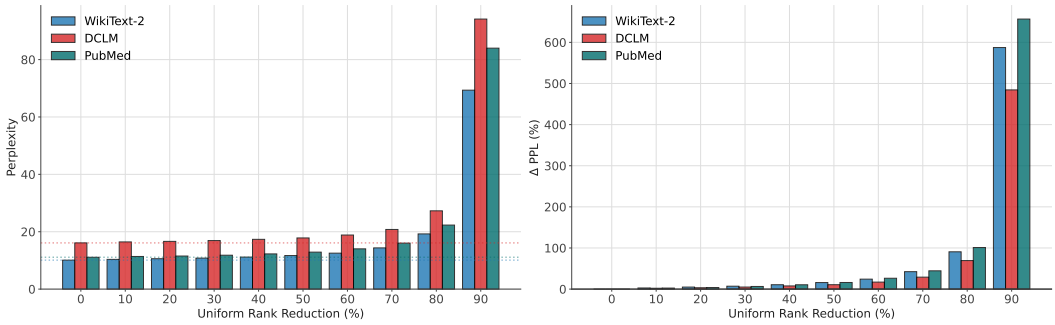

Figure 12: **Uniform rank reduction across all 8 MEMORYBLOCKs of TIDE-8E-1B.** Each $M_k \in \mathbb{R}^{|\mathcal{V}| \times d_b}$ is replaced by its rank-r SVD approximation with $r = \lceil (1-p) \cdot d_b \rceil$ applied identically to every block. **(a)** Absolute perplexity on WikiText-2, DCLM, and PubMed; dotted horizontal lines mark each dataset's uncompressed baseline. **(b)** Relative degradation of perplexity which is largely flat through $\sim$30% reduction, degrades gradually through $\sim$60%, and rises sharply beyond 70%.

Figure 12 sweeps the uniform reduction percentage from 0% to 90% in 10% increments and reports perplexity on WikiText-2, DCLM, and PubMed. At modest reductions (10–30%, $r \in [1434, 1844]$), perplexity degradation remains almost flat while parameter count per MEMORYBLOCK drops to as little as 71% of the dense form. At moderate reductions (40–60%, $r \in [820, 1229]$), degradation grows around $\sim$10% to $\sim$25% but remains gradual. However, beyond 70% reduction the curves bend sharply upward and the relative ΔPPL reaches 587% on WikiText-2, 484% on DCLM, and 657% on PubMed. Our findings indicate that a TIDE MEMORYBLOCKs can be significantly compressed up to a $\sim$50% rank reduction ($r = 1024$, halving the per-block parameter count) at uniform with marginal drop in performance for practical purpose with limited resource availability. Provided the existence of non-uniform low-rank properties across different layers [Jaiswal et al., 2024], we strongly believe that EMBEDDINGMEMORY can be further compressed using non-uniform rank reduction techniques for relatively superior performance compared to uniform SVD.

K K-Nearest Neighbor Study of Base Embedding and MEMORYBLOCKs:

To understand Base and MEMORYBLOCK embeddings from semantic perspective, we ask an interesting question: *At individual token-level, do the* MEMORYBLOCK*s recover semantic neighbors that the primary embedding E has failed to learn during training?* To probe this, we compute the per-token Jaccard overlap J_k between the top-10 cosine neighbors across 200 randomly sampled rare and common token under E and each M_k.

In Figure 13, we present the distribution of J_k over tokens and find that rare-token boxes lie consistently below the common-token boxes for every K. It indicates that for common tokens, the neighbor sets agree more closely with E but for rare tokens neighbor sets are *substantially disjoint* adding complementary and non-overlapping information about them. For example, in Table 7 with concrete examples: for a rare token `asynchronously`, E's top-10 are dominated by adverbs ending in {-ly}; but each MEMORYBLOCKs contribute additional closely related technical information (*e.g.* M_2 surfaces *Asynchronous JavaScript and XML, callbacks*; M_3 surfaces *defensively, securely*) in pursuit to enhance the semantic structure.

Table 7: Top-10 Cosine-Nearest neighbors of two rare query tokens in the primary embedding table E and across 8 MEMORYBLOCK in TIDE-8E-1B ($M_1, M_2, \ldots, M_8$) model checkpoints. Row backgrounds encode per-block Jaccard rank against the Base top-10 within each query, with *darker* shades indicating *higher* J_k (more neighbor-set agreement with Base).

Query	Pathway	Top-10 nearest neighbors (cosine)	J_k
asynchronously	Base E	asynchronous, ynchronously, sequentially, recursively, dynamically, ynchronous, concurrently, independently, horizontally, seamlessly	–
	M_1	asynchronous, ynchronously, synchronous, ynchronous, silently, globally, callback, concurrently, digitally, unsafe	0.25
	M_2	asynchronous, ynchronously, Asynchronous JavaScript and XML, callbacks, synchronous, ynchronous, LSD, hashtags, conditionally, breathable	0.18
	M_3	ynchronously, asynchronous, recursively, ynchronous, sequentially, defensively, resonate, dynamically, callbacks, securely	0.43
	M_4	asynchronous, ynchronously, concurrently, synchronous, ynchronous, dynamically, anonymously, externally, parallel, simultaneously	0.33
	M_5	asynchronous, ynchronously, simultaneously, recursively, spontaneously, sequentially, efficiently, tirelessly, separately, independently	0.33
	M_6	asynchronous, synchronous, Asynchronous JavaScript and XML, instantiated, serialize, scalable, serialized, ynchronously, caching	0.11
	M_7	asynchronous, ynchronously, manually, Premiere, optionally, reordered, indefinitely, RMS, factorial, charcoal	0.11
	M_8	asynchronous, synchronous, ynchronously, ynchronous, dynamically, synchronized, concurrently, electronically, simultaneously, automatically	0.33
fred	Base E	Fred, Fred, Larry, Roger, Doug, Charlie, Sean, jim, Mike, Jake	–
	M_1	Fred, Fred, joe, john, Ginny, Frederick, Doug, Lena, Woody, zar	0.18
	M_2	Fred, Fred, alf, Freder, Maggie, Carlo, Viktor, alan, Noel, Amit	0.11
	M_3	Fred, Fred, fred, mary, bob, Bob, Frederick, Bob, Freder, Herbert	0.11
	M_4	Fred, Fred, martin, Martin, Zack, Bernie, Frederick, alex, Charlie, Albert	0.18
	M_5	Fred, Fred, christ, Nora, brahim, Dani, tek, Enterprise, Yo, Practice	0.11
	M_6	Fred, Fred, Katy, Hans, Ogre, Nel, Sing, Ian, Berk, Toby	0.11
	M_7	Fred, Fred, fred, Ron, Doug, COST, Todd, Evelyn, Lindsey, Apache	0.11
	M_8	Fred, Fred, fred, Frederick, Freddy, Ned, Freddie, Geoff, Fried, red	0.11

The rare-name query `fred` shows the same pattern: E returns only first-name neighbors, while individual MEMORYBLOCKs additionally recover orthographic variants (`Fred`, `Frederick`, `Freddy`), tokenizer fragments (`Freder`), and cross-lingual variants (`Hans`, `Viktor`). The complementary-information picture therefore not a global statistical artifact but reflects genuine per-token specialization of MEMORYBLOCKs.

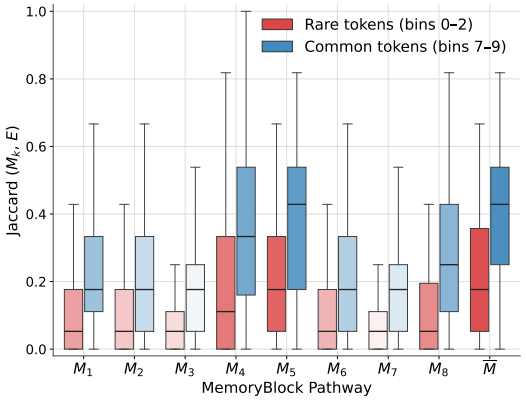

Figure 13: Cosine-Nearest Agreement between primary embedding E and memory blocks M_k across for *rare* and *common* tokens. Rare-token boxes lie consistently below common-token boxes indicating memory blocks pathways encode neighbor sets that are *substantially disjoint* from E for the rare tokens and add complementary new information to model.

L Limitations and Future Work

While TIDE delivers consistent gains across model scales and downstream tasks, we would like to acknowledge some limitations. ❶ *Storage overhead:* Despite the EmbeddingMemory tables are static and quantization friendly, the SSD footprint still scales linearly with K. Deployments with strict storage budgets need to rely on the compression strategies discussed in Appendix J to reduce SSD overhead. ❷ Our experiments cover model scales from 750M to 3B parameters trained on 200–500B tokens of DCLM, with evaluations on WikiText, PubMed, DCLM, and eight zero-shot benchmarks. The benefits of TIDE remain unexplored for longer training horizons, after instruction tuning or RLHF and is left to future work. ❸ Per-block router statistics and nearest-neighbor analyses (Appendix 7) suggest that distinct memoryBlocks specialize to distinct frequency regimes, we do not provide a principled account of*what* each block learns. A more fine-grained interpretability study of memoryblocks specialization is an important direction for future work.

M Model Training Implementation Details

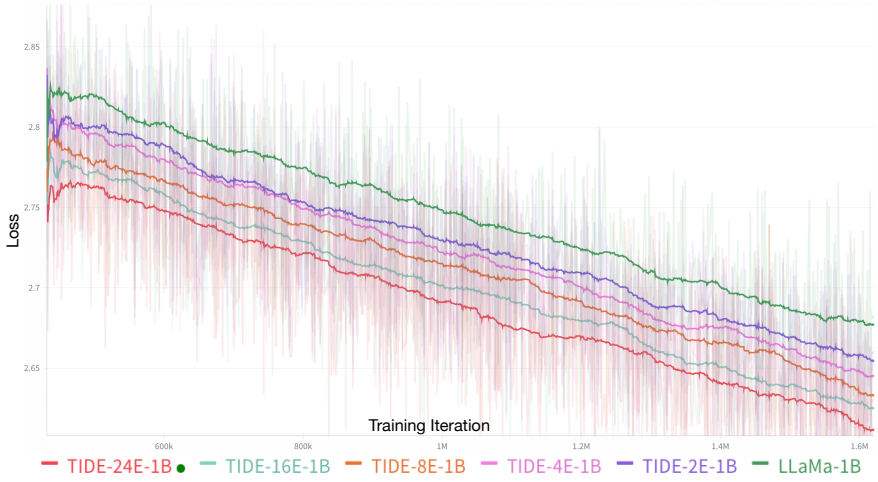

Figure 14: Training Loss Comparison of LLaMa-1B and TIDE-1B with 2, 4, 8, 16, and 24 MEMORYBLOCKS.

Table 8: Model training configurations for our Base, TIDE Models. All model checkpoints are trained within the paper adopt exactly same configuration for fair comparison.

Category	**Key**	**Value**
Common	Tokens Count	400-500 Billion
	Vocabulary size	128,256
	Tokenizer	meta-llama/Llama-3.1-8B
	Dataset	mlfoundations/dclm-baseline-1.0
	Sequence Length	2048
	Hidden Activation	SiLU
Loss	Name	Cross Entropy
	Z-loss	1.0e-6
Optimizer	Name	Adam
	Weight Decay	0.1
	Beta1	0.9
	Beta2	0.95
Schedular	Warmup Initial LR	1e-06
	Warmup Iterations	10000
	Type	Cosine
	Max LR	1.0e-04
	Min LR	1.0e-05

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: Section 3.3.2 and 3.3.2 provided details for rare token and contextual collapse problem. Section 4.1 provide performance results across multiple model scale and datasets.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

Justification: Appendix L.

Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: Appendix G, D, C, and F.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Appendix M

   Guidelines:

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

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

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

   Answer: [No]

   Justification: All datasets used are publicly available and cited.

   Guidelines:

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

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Training details regarding datasets and hyperparameters are provided throughout paper and in Appendix M.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Major plots have statistical significance reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: All experiments are conducted using 64$\times$B200 GPUs for 20-25 days.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Etices code is followed.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This work proposes a new architecture design for modern transformer and we do not identify any positive societal impacts and negative societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: This work proposes a new architecture design for modern transformer and we do not identify any positive societal impacts and negative societal impacts.

Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Licenses are credited and cited.

    Guidelines:

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

13. **New assets**

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

    Answer: [N/A]

    Justification: No new asset is introduced.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

Answer: [N/A]

Justification: No crowdsourcing is involved.

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [N/A]

    Justification: No IRB involved.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLM is used for grammer-correction, editing, or formatting purposes.

    Guidelines:

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