GEM: Towards Interpretable Language Models via Geometric Embedding Alignment

Dimitris Tsaras[1,2] **Yankun Hong**[2] **Lei Chen**[2] **Zhiyao Xie**[1] **Mingxuan Yuan**[2]
[1]Hong Kong University of Science and Technology, Hong Kong
[2]Noah's Ark Lab, Huawei, Hong Kong
dimitris.tsaras@gmail.com

Abstract

Large language models encode rich representations in their hidden states, yet these representations remain largely opaque and disconnected from the model's vocabulary. We introduce Geometric Embedding Mixture (GEM), a pre-training regularization method that aligns transformer hidden states with the geometry of the vocabulary embedding space. GEM utilizes a self-supervised objective to encourage the final hidden state to approximate a probability-weighted mixture of token embeddings. This effectively minimizes the free energy of the models and pushes the representation into the convex hull of the vocabulary. We pre-train models at four scales (135M, 360M, 760M, and 1.7B) and demonstrate that GEM yields superior predictive uncertainty compared to standard training, reducing perplexity by up to 53% on benchmarks. Furthermore, we show that this geometric structural constraint mitigates representation anisotropy and takes a step toward *native interpretability*. Compared to baseline models, GEM improves direct layer-wise decoding of intermediate hidden states with the untuned language head and achieves higher average causal faithfulness, revealing coherent semantic trajectories with less reliance on auxiliary probes. Our work demonstrates that imposing geometric structure during pre-training improves both performance and transparency, offering a path toward language models that are interpretable by construction.

1 Introduction

Large language models (LLMs) operate as black boxes. While their capabilities in reasoning and generation are well-established, the internal representations that drive these behaviors remain largely opaque. This lack of transparency presents a fundamental challenge: without a clear understanding of the model's internal decision-making process, it is difficult to guarantee safety, steer behavior, or predict failures such as hallucinations.

Current efforts to interpret these models are primarily post-hoc. Techniques such as the Logit Lens [1] or Tuned Lens [2] attempt to map intermediate hidden states onto the vocabulary to reveal the model's evolving "beliefs." However, these methods rely on the assumption that the hidden state geometry is meaningful when projected onto the vocabulary space. In standard transformers, this is rarely the case. Hidden states often suffer from severe anisotropy [3, 4] or drift into manifolds topologically distinct from the token embeddings. Consequently, interpreting a model often requires training auxiliary probes, leaving us with an ambiguity: are we analyzing the model's knowledge, or the probe's capacity to extract it? We propose a different paradigm: rather than building better tools to inspect opaque models, we should constrain the models to be more transparent by design.

We introduce Geometric Embedding Mixture (GEM), a pre-training objective that enforces geometric alignment between the transformer's hidden states and its vocabulary. The intuition behind GEM is

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

that a representation predicting a set of concepts should physically resemble those concepts in vector space. GEM regularizes the final hidden state to approximate a probability-weighted mixture, or barycenter, of the token embeddings it predicts.

Geometrically, this constrains the hidden representations to lie within the convex hull of the relevant vocabulary items. This "what you predict is what you are" constraint serves two purposes. First, it promotes native interpretability. By sharing a common geometric substrate, the model's internal states become more directly decomposable into vocabulary contributions, allowing intermediate layers to be read at the vocabulary level without learned projections. Second, it acts as a structural regularizer. By pulling the representation toward a mixture of likely tokens, GEM implicitly draws conceptually similar tokens closer together in the embedding space. We hypothesize that this geometric structure is a prerequisite for reliable control. An LLM that organizes its internal states coherently is not only easier to interpret but potentially easier to steer and monitor for deviations that signal hallucination.

We validate GEM by pre-training models at the 135M, 360M, 760M, and 1.7B scales. Our experiments demonstrate that imposing this geometric structure enhances rather than hinders learning. GEM-trained models achieve significantly better predictive uncertainty, reducing perplexity by up to 53% on LAMBADA and 16% on Wikitext, while consistently improving zero-shot accuracy across standard benchmarks by an average of 2.1%. These gains suggest that GEM does not merely organize the embedding space for interpretability, but actively promotes a more robust and calibrated representation of language.

Contributions.

- We introduce GEM, a regularization method that aligns the final predictive representation with the vocabulary embedding geometry via a distribution-weighted mixture target.
- We provide a theoretical framework linking GEM to the minimization of free energy in Energy-Based Models (EBMs), suggesting that GEM encourages representations to settle in low-energy regions of the vocabulary landscape.
- We demonstrate that GEM mitigates the representation degeneration (anisotropy) common in standard transformers, yielding a more isotropic, structural, and semantically clustered embedding space.
- We show that GEM moves toward native interpretability: the model's raw hidden states can be decoded into vocabulary distributions at intermediate layers using the LM head, with higher average causal faithfulness than the Baseline.

2 Methodology

2.1 Problem Setup and Notation

Let $\mathcal{V}$ denote a vocabulary of size $|\mathcal{V}|$ and let d be the model width. We write the input embedding matrix as $E \in \mathbb{R}^{|\mathcal{V}|\times d}$, where the i-th row $e_i^\top$ is the embedding of token i. Given a token sequence $x_{1:T}$, a decoder-only Transformer produces a hidden state sequence $h_{1:T} \in \mathbb{R}^{T\times d}$, which is then normalized by a final normalization layer (LayerNorm or RMSNorm) to yield $\tilde{h}_{1:T} \in \mathbb{R}^{T\times d}$. A linear language-modeling head ("LM head") maps $\tilde{h}_t$ to logits $z_t \in \mathbb{R}^{|\mathcal{V}|}$, and the next-token distribution is given by the softmax function

$$p_t(i) \;=\; \mathrm{softmax}(z_t)_i \;=\; \frac{\exp(z_t(i))}{\sum_{j\in\mathcal{V}} \exp(z_t(j))}, \tag{1}$$

which defines a categorical distribution over the vocabulary.

In the common *weight-tied* setting, the LM head weight is tied to E (up to transpose), so that logits can be written (ignoring bias for simplicity) as

$$z_t(i) \;=\; \tilde{h}_t^\top e_i. \tag{2}$$

This parameterization makes the token embedding geometry directly define the scoring geometry used by the softmax classifier.

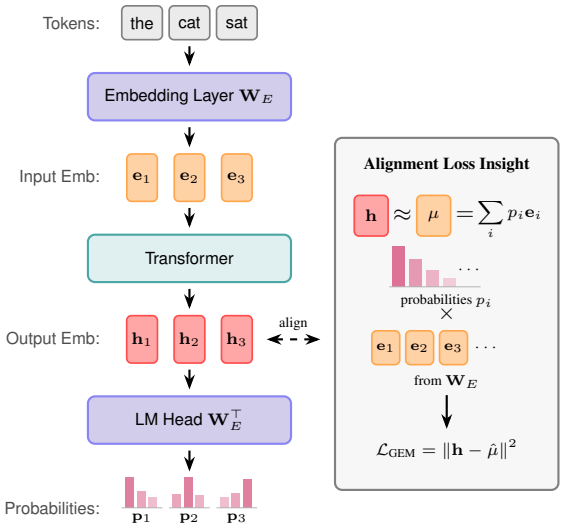


Figure 1: GEM applies an additional alignment loss on the produced hidden embeddings post layer normalization to bring the hidden embeddings in the convex hull of the vocabulary

2.2 GEM Regularization

Motivation. GEM is designed to encourage the model's final representation $\tilde{h}_t$ (i.e., the representation directly consumed by the LM head) to lie in the same geometric space as the vocabulary embeddings E. The core idea is to view $\tilde{h}_t$ as being close to an *embedding mixture* of vocabulary vectors, weighted by the model's own predictive distribution. This aligns with the broader view that sequence models can store and utilize information through structured embedding geometry rather than purely associative lookup, and that strengthening geometric structure can provide headroom for improved generalization.

Expected embedding (mixture) target. For each position t, define the expected embedding under the model distribution as

$$\mu_t \;=\; \sum_{i \in \mathcal{V}} p_t(i)\, e_i \in \mathbb{R}^d, \tag{3}$$

which is a convex combination of the token embeddings and therefore lies in the convex hull of $\{e_i\}_{i \in \mathcal{V}}$.

Top-k approximation. Computing Eq. (3) over the full vocabulary can be expensive. We therefore approximate μ_t using only the top-k predicted tokens: let $\mathcal{K}_t \subset \mathcal{V}$ be the set of indices of the k largest logits at step t, and define a renormalized distribution

$$\tilde{p}_t(i) \;=\; \frac{\exp(z_t(i))}{\sum_{j \in \mathcal{K}_t} \exp(z_t(j))} \quad \text{for } i \in \mathcal{K}_t. \tag{4}$$

We then set

$$\mu_t^{(k)} \;=\; \sum_{i \in \mathcal{K}_t} \tilde{p}_t(i)\, e_i. \tag{5}$$

Since $\mu_t^{(k)}$ is a convex combination over k embeddings, it lies in the convex hull of the selected top-k embeddings, i.e., on a (typically) lower-dimensional face of the full vocabulary convex hull.

GEM alignment loss. We regularize the post-normalization hidden state $\tilde{h}_t$ to be close to $\mu_t^{(k)}$. We consider the following $L2$ loss function:

$$\mathcal{L}_{\text{GEM}}^{\text{L2}} = \frac{1}{|\mathcal{M}|\, d} \sum_{t \in \mathcal{M}} \left\| \tilde{h}_t - \mu_t^{(k)} \right\|_2^2, \tag{6}$$

where $\mathcal{M}$ is the set of token positions included in the loss (e.g., positions where the next-token label is not masked).

Total training objective. We train using standard next-token cross-entropy plus GEM:

$$\mathcal{L} \;=\; \mathcal{L}_{\mathrm{CE}} \;+\; \lambda\,\mathcal{L}_{\mathrm{GEM}}, \tag{7}$$

where $\lambda \geq 0$ controls the strength of the regularization.

Practical schedules and stability. Early in training, the predictive distribution p_t can be high-entropy, making $\mu_t^{(k)}$ closer to a generic average of several embeddings. To avoid overly constraining the representation before the model learns meaningful token discrimination, λ can be scheduled (e.g., warmup then ramp). Additionally, we stop gradients through $\tilde{p}_t$ when constructing $\mu_t^{(k)}$, turning GEM into a pure alignment constraint on $\tilde{h}_t$ rather than allowing the model to reduce the regularizer by changing p_t. GEM adds minimal computational overhead as it reuses the logits already computed for the CE loss. The additional per-position cost is: top-k selection $O(|V|)$, softmax over k elements $O(k)$, weighted embedding sum $O(k \cdot d)$, and L2 distance $O(d)$ which amounts to a total of per position: $O(|V| + k \cdot d)$.

Mechanistic explanation. Eq. (1) can be understood as turning the similarity between $\tilde{h}_t$ and e_i (i.e., $\tilde{h}_t^T e_i$) into probabilities. This implicitly assumes $\tilde{h}_t$ and E share the same representation space, yet standard training provides no constraint to ensure this. GEM introduces the missing alignment by approximating $\tilde{h}_t$ as a probability-weighted mixture of the very embeddings it is scored against. In addition, a transformer ranking next tokens is implicitly performing "next-concept prediction": the top-k candidates at any position typically share semantic or syntactic properties. Minimizing the distance between $\tilde{h}$ and a weighted mixture of these co-probable tokens has two effects: 1. It anchors $\tilde{h}$ to the vocabulary manifold, producing more consistent token scoring. 2. It pulls conceptually related tokens closer in embedding space, creating the semantic clustering.

2.3 Theory: Geometric Interpretation of GEM

2.3.1 Energy-based model interpretation

Beside the heuristic mechanistic explanation, we reformulate GEM into an Energy-Based Model (EBM) framework that serves as a fundamental theoretical justification for this heuristic idea. Recent works [5, 6] have proposed a unified perspective in which LLMs can be interpreted through the lens of EBM or Hopfield network. This viewpoint treats the token selection mechanism not merely as a softmax classifier, but as a structured Gibbs distribution defined by an underlying energy function.

We re-define the logit for a token i as $z_t(i) = \tilde{h}_t^\top e_i - \frac{1}{2}\|\tilde{h}_t\|_2^2$ rather than $\tilde{h}_t^\top e_j$. This modification preserves the softmax distribution due to the translation invariance of the softmax function, but aligns with the requirement of a quadratic energy with respect to $\tilde{h}_t$. Following the energy-based interpretation, the energy of token i is defined as the negative logit:

$$E_i(\tilde{h}_t) := \frac{1}{2}\|\tilde{h}_t\|_2^2 - \tilde{h}_t^\top e_i. \tag{8}$$

The standard softmax distribution is then precisely the Gibbs measure

$$p_t(i \mid \tilde{h}_t) = \frac{\exp(-E_i(\tilde{h}_t))}{Z(\tilde{h}_t)}, \tag{9}$$

$$Z(\tilde{h}_t) = \sum_{j \in \mathcal{V}} \exp(-E_j(\tilde{h}_t)), \tag{10}$$

where Z represents a partition function. This formulation is mathematically equivalent to the usual logit-based softmax, but it exposes a deeper structure: the model is implicitly sampling from an energy landscape indexed by $\tilde{h}_t$.

Given the Gibbs distribution above, the *free energy* associated with representation $\tilde{h}_t$ is defined as $F(\tilde{h}_t) := -\log Z(\tilde{h}_t)$. A key identity from statistical mechanics and modern EBM theory is that the gradient of the free energy equals the expected gradient of the energy:

$$\nabla_{\tilde{h}_t} F(\tilde{h}_t) = \sum_{i \in \mathcal{V}} p_t(i)\, \nabla_{\tilde{h}_t} E_i(\tilde{h}_t). \tag{11}$$

For the energy $E_j(\tilde{h}_t) := \frac{1}{2}\|\tilde{h}_t\|_2^2 - \tilde{h}_t^\top e_j$, we have $\nabla_{\tilde{h}_t} E_i(\tilde{h}_t) = \tilde{h}_t - e_i$, and therefore

$$\nabla_{\tilde{h}_t} F(\tilde{h}_t) = \tilde{h}_t - \sum_{i \in \mathcal{V}} p_t(i)\, e_i = \tilde{h}_t - \mu_t. \tag{12}$$

Hence, in an EBM representation, $\mathcal{L}_{\mathrm{GEM}}$ aims to minimize the free energy associated to LLMs's output token distribution, which aligns the training target of an EBM [7] and a Hopfield networks [6].

For the top-k approximation, the GEM loss corresponds to minimizing the truncated free energy restricted to $\mathcal{K}_t$ when using the renormalized distribution $\tilde{p}_t$ in Eq. (4). For more discussion of EBM, please refer Appendix G.3.

2.3.2 Convex-hull and "face" interpretation

Because p_t is a probability distribution over $\mathcal{V}$ (Eq. (1)), the expected embedding μ_t in Eq. (3), as well as $\mu_t^{(k)}$ in Eq. (5), are convex combinations of token embeddings and therefore lies in

$$\mathrm{conv}(\{e_1, \ldots, e_{|\mathcal{V}|}\}) = \left\{ \sum_{i=1}^{|\mathcal{V}|} \alpha_i e_i : \alpha_i \geq 0, \sum_i \alpha_i = 1 \right\}, \tag{13}$$

which is a $(V-1)$-dimensional subspace of the token embedding space $\mathrm{span}(E)$. Specifically, the top-k approximation $\mu_t^{(k)}$ lies in $\mathrm{conv}\{e_i : i \in \mathcal{K}_t\}$, which is a convex polytope spanned by at most k embedding vectors and is a lower-dimensional face (or subset) of the global vocabulary convex hull.

Thus, GEM can be viewed as a geometric constraint that encourages $\tilde{h}_t$ to remain near the convex region supported by the model's own plausible next-token set, and therefore, reinforces the model to learn a shared representation space for both the hidden state $\tilde{h}_t$ and the token embedded space.

2.3.3 The Importance of Post-Normalization

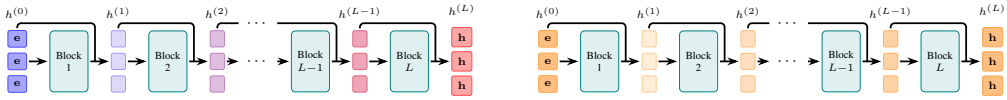
In our architecture, the logits are produced from the *post*-normalization states $\tilde{h}_t$ (final LayerNorm/RMSNorm output), not from the raw residual stream. RMSNorm normalizes activations using their root mean square and provides re-scaling invariance properties, meaning that uniform scaling of inputs is largely removed by the normalization [8, 9]. As a consequence, the direction (angular information) of $\tilde{h}_t$ becomes particularly important for determining which token embeddings e_i yield large inner products $\tilde{h}_t^\top e_i$ and therefore high probability under softmax [8].

The convex-hull interpretation of GEM in Section 2.3.2 complements this perspective. Through GEM regularization, the hidden state $\tilde{h}_t$ and the token embedding vectors $\{e_i\}$ are encouraged to share the same representation space. The inner products $\tilde{h}_t^\top e_i$ effectively compute the cosine similarity between the hidden state $\tilde{h}_t$ and the vocabulary embeddings e_i, which is then transformed into a probability distribution in the exponential family via the softmax. This interpretation holds only when $\tilde{h}_t$ and e_i reside in the same representational space.

2.3.4 Connection to geometric memory

The work of Noroozizadeh et al. argues that deep sequence models can store knowledge not only as associative tables but also as structured embedding geometry ("geometric memory"), where dot products reflect multi-hop relationships and enable generalization beyond memorized co-occurrences [10]. GEM can be interpreted as explicitly encouraging such geometry to be operational at the model's output interface by aligning the final predictive representation with a mixture of vocabulary embeddings, thereby reinforcing the view that prediction and representation share a common geometric substrate.

In addition, within the GEM loss (Eq. (6)), we aim to minimize the discrepancy between $\tilde{h}_t$ and $\mu_t^{(k)} = \sum_{i \in \mathcal{K}_t} \tilde{p}_t(i)\, e_i$. When training with respect to the embedding matrix E, this loss encourages each embedding vector e_i to move closer to $\tilde{h}_t$, producing a cluster of top-k candidate tokens around the predicted state $\tilde{h}_t$. In general, tokens with top-k probability should share some underlying relation, as they represent alternatives for the same position in the same token sequence. Following this intuition, the clustering effect induced by the GEM loss promotes a more coherent geometric structure in the vocabulary embedding space.



(a) Transformer with vanilla cross entropy loss

(b) Transformer with GEM

Figure 2: Interpretational Spaces: In contrast to the vanilla loss, GEM stays closer to vocabulary space throughout the layer

2.3.5 Alignment of spaces of transformer blocks

In standard Transformers, the residual connection [11] sums the initial embedding e_t with subsequent block outputs. Without specific constraints, the vocabulary embedding space (Fig. 2a $h^{(0)}$ space) and the hidden state spaces of the final block (Fig. 2a $h^{(L)}$ space) are typically unaligned. Due to the residual mixture, intermediate representations become "mixtures" of disparate spaces, rendering them undecodable by the final LM head (Fig. 2a spaces of $h^{(1)}$ to $h^{(L-1)}$ are mixtures of two different spaces of $h^{(0)}$ and $h^{(L)}$).

The GEM loss resolves this by aligning the vocabulary embedding and all representation spaces into a shared, consistent manifold. This prevents space mixing (Fig. 2b) and ensures that intermediate layers are inherently interpretable. Crucially, unlike Tuned Lens [2] which trains a "shortcut" to predict the final output, a probe whose explicit goal is "reproduce the final distribution" cannot present what the model actually represents at intermediate layer. However, GEM enforces global space consistency by design, thus, it allows for direct decoding of intermediate states to track their internal evolution, which is a task inherently infeasible for shortcut-based methods. Thus, GEM provides a principled tool for understanding the model's progressive representational refinement.

3 Experiments

In this section, we evaluate the performance of GEM across three primary dimensions: language modeling capabilities, the geometric structure of the embedding space, and the interpretability of internal representations. Section 3.1 details our training configuration. Section 3.2 presents downstream performance benchmarks. Section 3.3 analyzes the geometric properties of the learned embeddings, and Section 3.4 investigates the model's native interpretability.

3.1 Experimental Setup

We train models at four scales, 135M, 360M, 760M, and 1.7B parameters, based on the SmolLM2 architecture [12]. For each scale, we train two variants. The first is a Baseline model trained with the standard cross-entropy objective. The second is the GEM model, which includes the GEM alignment loss with $\lambda = 0.05$ and $k = 8$. An ablation study of k and λ is available in Appendix B.

We train on 5B, 10B, 20B, and 20B tokens respectively, sampled from the FineWeb-Edu dataset [13]. For the three smaller models, these token counts meet or exceed the compute-optimal scaling ratio of $20\times$ training tokens per parameter suggested by Chinchilla [14]; the 1.7B model is trained on approximately 12 tokens per parameter.

Training is performed using the `litgpt` library with a maximum sequence length of 2048 tokens and a global batch size of 0.5M tokens. We use the AdamW optimizer with a weight decay of 0.1 and a cosine annealing schedule (peak learning rate 10^{-3}, 100 warmup steps). A complete list of hyperparameters is provided in Appendix A.

3.2 Language Modeling

We evaluate GEM on a suite of standard benchmarks including LAMBADA [15], WikiText [16], PIQA [17], HellaSwag [18], Winogrande [19], ARC [20], and BoolQ [21].

Our primary focus is predictive uncertainty. As shown in Table 1, GEM dramatically improves perplexity across all model scales. On the LAMBADA and WikiText benchmarks, we observe

perplexity reductions of up to 53.12% and 15.91% respectively for the 135M model, with consistent gains at larger scales.

Crucially, this regularization does not hamper downstream reasoning. We observe an average accuracy increase of 1.88%, 2.26%, 1.46%, and 2.64% across benchmarks for the 135M, 360M, 760M, and 1.7B models respectively. This indicates that forcing the model to align with the vocabulary geometry improves its internal probability estimation without compromising its ability to solve reasoning tasks.

Table 1: Zero-shot performance on standard benchmarks. GEM consistently improves perplexity (ppl) and achieves competitive or superior accuracy across reasoning tasks.

Model	**LMB.** ppl↓	**Wiki.** ppl↓	**LMB.** acc↑	**PIQA** acc_n↑	**Hella.** acc_n↑	**Wino.** acc↑	**ARC-e** acc_n↑	**ARC-c** acc_n↑	**BoolQ** acc↑	**Avg.** ↑
135M/5B										
Baseline	285.7	51.10	19.19	60.12	29.41	49.01	43.94	19.37	50.83	39.63
GEM	134.0	42.97	25.13	61.70	30.97	50.75	44.49	25.09	52.42	41.51
360M/10B										
Baseline	102.3	35.86	24.65	63.38	33.90	52.17	47.60	27.22	55.47	43.48
GEM	59.64	31.70	29.17	64.64	36.62	50.20	51.43	26.71	61.44	45.74
760M/20B										
Baseline	66.94	31.37	28.37	66.76	39.55	50.91	51.26	28.33	60.12	46.47
GEM	47.74	29.37	32.51	67.08	43.83	50.43	53.87	30.20	57.61	47.93
1.7B/20B										
Baseline	46.88	28.53	30.37	66.05	38.10	51.38	51.94	29.52	59.91	46.75
GEM	26.23	24.05	37.28	68.50	43.67	54.38	55.56	29.86	56.48	49.39

3.3 Geometry

A key mechanism of GEM is the creation of the mixture embedding as a barycenter. This effectively pulls highly probable tokens closer together in the embedding space. Assuming the model produces a reasonable distribution of probable words, the top-k tokens usually share active conceptual features. For example, the next token for the sentence "My phone number is 6" is likely a digit. Consequently, the GEM loss pulls the digit tokens closer to the hidden state, and by extension, closer to each other.

We visualize this effect in Fig. 4, which shows UMAP projections of 10 conceptual categories in the 135M and 760M model. While the Baseline model (Fig. 4a) exhibits some structural organization, there is significant mixing between categories. In contrast, GEM (Fig. 4b) creates clearly separated and compact classes. Notably, the unambiguous category of digits (marked in red) is extremely compact in the GEM model, whereas the Baseline fails to capture this conceptual tightness. Additionally, as the number of dimensions scales with the model size, conceptual clusters appear to be more well-defined. A UMAP projection of all models is shown in Fig. 5 in Appendix C.

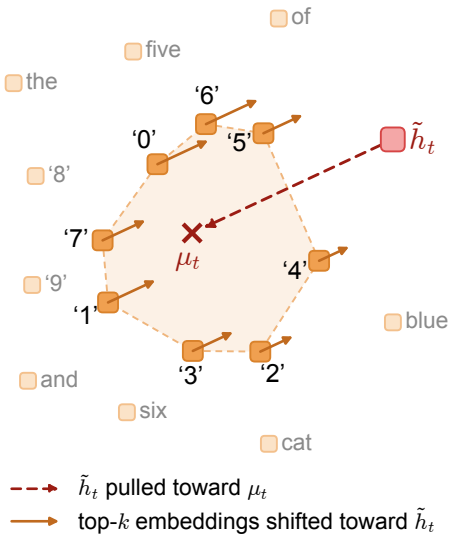
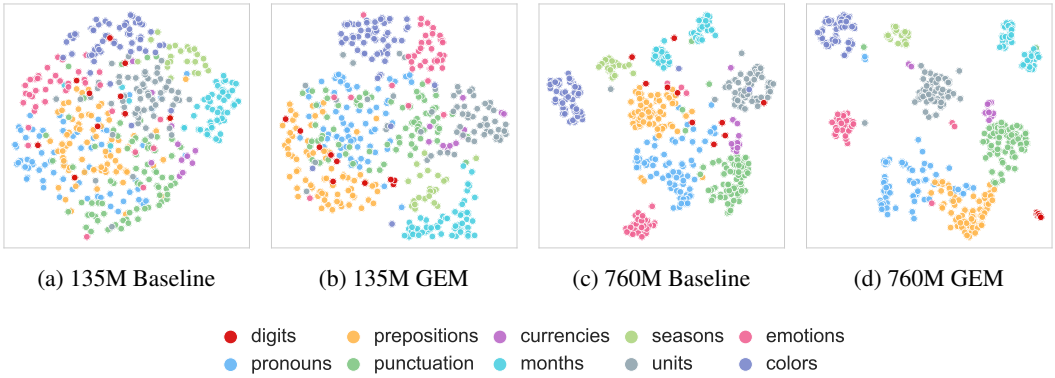


Figure 3: GEM's alignment step for "My phone number is 6" (GEM 360M). The top-8 next tokens are all digits and μ_t is their probability-weighted mixture. GEM pulls $\tilde{h}_t$ toward μ_t and shifts each top-k embedding toward $\tilde{h}_t$ in proportion to its probability (arrow lengths). Digit positions are a linear projection of the real embeddings; the placement of $\tilde{h}_t$ and of the surrounding tokens is schematic.

To quantify the quality of these clusters, we evaluate geometric cohesion across 10 distinct semantic categories (e.g., *Digits*, *Colors*). We compute Intra-class Distance (compactness), Inter-class Distance (distinctness), Separation Score (Inter − Intra), and Purity@k (fraction of same-category neighbors among k nearest neighbors).



(a) 135M Baseline (b) 135M GEM (c) 760M Baseline (d) 760M GEM

Figure 4: UMAP projection of vocabulary embeddings for selected semantic categories. GEM induces clearer separation and tighter clustering compared to the Baseline.

As summarized in Table 2, GEM consistently improves cluster quality (see Appendix Table 11 for full breakdown). Across all scales, GEM achieves significantly higher separation and purity. For instance, in the 760M model, the average separation score improves $\times 4$ (0.062 to 0.245), and Purity@10 increases by over 50%. This confirms that the distinct clusters observed in the UMAP projections represent a robust improvement in semantic organization.

Table 2: Aggregated Embedding Geometry Metrics (Average across 10 semantic categories). GEM consistently improves cluster separation and purity across all scales.

Model	**Avg. Separation** ↑		**Avg. Purity@10** ↑	
	Baseline	GEM	Baseline	GEM
135M	0.028	0.046	0.061	0.102
360M	0.058	0.251	0.152	0.300
760M	0.062	0.245	0.195	0.295
1.7B	0.038	0.239	0.019	0.221

Relational structure. Beyond category clusters, we test whether the vocabulary embeddings encode relations through vector arithmetic. An analogy question such as *man* : *woman* :: *king* : ? is answered with the vocabulary token whose embedding is closest (cosine similarity) to $e_{woman} - e_{man} + e_{king}$, excluding the three query words, and counts as correct if that token is *queen* (3CosAdd [22]). We use the Google analogy set and BATS 3.0 [23], which cover semantic (e.g., country–capital, gender) and morphological (e.g., plurals, verb tenses) relations, keeping questions whose four words are single tokens (10,231 and 48,137 questions). As shown in Table 3, GEM's embeddings solve more analogies than the Baseline's at every scale (e.g., 71.2% vs. 18.0% on Google at 360M and 68.9% vs. 36.9% at 760M). For reference, we also report the official SmolLM2 base models, which were trained on far more tokens (2T–11T). GEM scores higher than them at 360M and 1.7B on both benchmarks, while at 135M the official model remains well ahead.

Table 3: Word-analogy accuracy (%) of the vocabulary embeddings. SmolLM2: official base models, for reference (no 760M model was released).

Model	**Google**			**BATS 3.0**		
	Baseline	GEM	SmolLM2	Baseline	GEM	SmolLM2
135M	3.20	7.86	58.23	3.21	6.83	22.87
360M	17.99	71.23	56.89	11.01	32.65	20.58
760M	36.90	68.93	–	15.51	34.73	–
1.7B	0.04	49.10	47.67	0.04	24.63	16.08

Finally, we examine global geometry using Isotropy Score (see Appendix G.2) [4], Cosine regularization [24], and effective rank. Table 4 shows that GEM improves all metrics across all scales. This

indicates that GEM's "soft" clustering, rather than collapsing into degenerate manifolds, promotes a richer, more isotropic global structure while maintaining local coherence.

Table 4: Embedding space quality metrics for all model sizes.

Metric	135M		360M		760M		1.7B	
	Base	GEM	Base	GEM	Base	GEM	Base	GEM
IsoScore↑	0.2573	0.4040	0.1946	0.3371	0.1735	0.3102	0.1215	0.2210
IsoScore*↑	0.3680	0.5446	0.2870	0.5348	0.2625	0.5249	0.2489	0.4960
CosineReg↓	0.1547	0.0622	0.1353	0.0478	0.1709	0.0417	0.0895	0.0540
EffRank↑	511.5	529.2	849.0	868.8	1127.4	1152.9	1708.2	1737.9

3.4 Interpretability

Intermediate interpretability seeks to decode internal states into vocabulary distributions to reveal the model's representational evolution. We evaluate two zero-shot methods: (1) **Direct Decoding**: $\text{softmax}(Eh_l)$, and (2) **Logit Lens (Norm)**: $\text{softmax}(E \cdot \text{RMSNorm}(h_l))$, alongside (3) **Tuned Lens** [2] as a trained baseline. Table 6 shows the decoding accuracy relative to the final prediction. GEM significantly outperforms the Baseline in early and middle layers across all scales. This gap arises because standard models suffer from a "representation-head mismatch" (Sec. 2.3.5); by aligning intermediate spaces with the final vocabulary space, GEM enables the LM head to accurately reflect state evolution. While Tuned Lens achieves higher accuracy, it functions as a predictive "shortcut" that bypasses the actual intermediate representation. A probe with this objective cannot be a faithful window into intermediate representations, highlighted by its similar performance on both methods.

To further assess intermediate representation quality, we measure causal faithfulness: whether the distribution decoded from a hidden embedding is sufficient to drive the rest of the network. We convert the distribution to an embedding mixture μ_l, substitute it for h_l in the forward pass, and check whether the model's prediction is preserved. Since the standard Tuned Lens is trained to reproduce the final output rather than to support substitution, we additionally introduce Loupe (Causal Tuned Lens), a per-layer affine probe with the same architecture as the Tuned Lens but trained end-to-end through the substitution. As shown in Table 7, GEM achieves higher faithfulness across every probe and scale (see Appendix E for details).

Table 5: Layers in the E, M, L columns of Tables 6 and 7. F (Final) is included only in Avg, which covers all layers.

	135M	360M	760M	1.7B
E	0–6	0–7	0–7	0–5
M	7–14	8–15	8–15	6–11
L	15–21	16–23	16–23	12–17
F	22–29	24–31	24–31	18–23

Qualitative Analysis. Our experiments show that the improved layer-wise decodability allows us to trace the model's "thought process" with greater clarity. We also examine the semantic structure of the embeddings via vector arithmetic analogies. These results are shown in Appendix F.

4 Related Work

Geometry of Representations and Memory The geometry of contextualized word representations has been widely examined. Prior work shows that embeddings from models such as ELMo, BERT, and GPT-2 are highly *anisotropic*, with upper layers exhibiting stronger context dependence and variance that cannot be captured by static embeddings [3, 24, 25]. Subsequent studies demonstrate that reducing anisotropy while preserving contextual information can improve downstream performance, motivating explicit objectives that shape embedding geometry [26]. Complementary evidence from Noroozizadeh et al. suggests that deep sequence models often develop *geometric memory*, arranging embeddings to encode global relational structure—such as multi-hop graph distances—beyond what is explicitly supervised [10]. Our method aligns with this perspective by constraining the final predictive representation to remain consistent with a convex combination of vocabulary embeddings, thereby strengthening the connection between model predictions and the underlying embedding geometry. See Appendix G.1 for more discussions.

GEM is related to these lines of work but differs because, GEM regularizer operates directly in the hidden-state and embedding representational spaces, shaping the *manifold* of Transformer hidden

Table 6: Layerwise Decoding Accuracy (%). E: Early, M: Mid, L: Late (layer ranges in Table 5).

		135M				360M				760M				1.7B			
Model	**Method**	**E**	**M**	**L**	**Avg**	**E**	**M**	**L**	**Avg**	**E**	**M**	**L**	**Avg**	**E**	**M**	**L**	**Avg**
Baseline	Direct	2.49	3.74	6.93	11.8	5.69	5.24	7.68	13.2	3.73	5.01	6.48	10.8	7.72	9.60	13.7	17.0
	Norm	2.49	3.74	6.93	11.8	5.70	5.24	7.68	13.2	3.73	5.01	6.48	10.8	6.81	8.41	11.3	15.0
	Tuned*	15.7	18.0	21.7	21.4	15.6	18.3	**25.5**	23.4	**17.7**	21.2	25.7	24.6	22.0	24.9	28.6	30.6
GEM	Direct	9.87	16.7	23.0	25.2	14.2	12.7	17.5	23.5	11.0	8.46	10.7	17.9	13.0	13.7	16.9	21.6
	Norm	9.12	14.5	20.1	23.2	8.01	7.03	12.3	18.1	6.14	4.67	7.63	13.7	11.7	10.1	10.2	17.0
	Tuned*	**16.1**	**19.5**	**23.4**	**22.7**	**15.8**	**19.9**	25.0	**23.7**	17.1	**22.4**	**27.0**	**25.4**	**23.8**	**27.4**	**33.1**	**34.1**

* Tuned Lens is trained to reproduce the final output distribution, so its accuracy does not reflect the intermediate representation. Best results in bold.

Table 7: Causal Faithfulness (%). At each layer, we form $\mu = \sum_i p_i e_i$ from the decoded distribution, replace the hidden state h_l with μ, and measure top-1 agreement with the unintervened model output. *Direct*: lm_head(h). *Norm*: lm_head(RMSNorm(h)). *Tuned*: standard Tuned Lens. *Loupe*: Causal Tuned Lens (trained end-to-end through the substitution). E: Early, M: Mid, L: Late (layer ranges in Table 5).

		135M				360M				760M				1.7B			
Model	**Method**	**E**	**M**	**L**	**Avg**	**E**	**M**	**L**	**Avg**	**E**	**M**	**L**	**Avg**	**E**	**M**	**L**	**Avg**
Baseline	Direct	5.01	3.18	2.24	2.75	3.59	1.84	0.75	1.72	3.19	1.96	0.90	1.65	2.44	2.76	3.96	2.54
	Norm	5.47	2.71	2.46	2.92	3.11	1.96	0.51	1.45	3.00	2.02	0.91	1.52	3.66	2.69	4.56	3.27
	Tuned	3.07	1.93	0.87	1.63	3.61	2.02	1.01	1.76	3.21	1.86	1.17	1.73	2.95	2.29	3.32	2.59
	Loupe	24.0	23.5	27.8	28.4	15.7	19.4	23.8	23.0	8.92	14.8	19.8	18.0	5.98	8.28	6.12	11.9
GEM	Direct	7.72	5.60	9.70	13.3	6.96	7.46	15.3	16.3	4.16	5.03	9.93	11.4	13.3	8.87	11.4	16.3
	Norm	8.22	6.37	8.17	13.2	4.82	3.38	9.91	13.1	2.56	3.43	5.28	10.1	11.3	6.44	8.78	14.6
	Tuned	8.89	8.04	10.8	14.9	13.7	15.7	24.1	23.5	7.54	18.0	25.6	22.0	12.3	9.41	12.1	17.5
	Loupe	28.3	28.0	31.8	31.9	18.4	25.3	31.0	29.1	8.40	20.2	30.0	25.5	9.67	6.86	17.4	18.2

states by encouraging them to lie near convex combinations of embedding vectors. This induces a more structured geometry for contextual representations, aligning with the goals of prior work that investigates the geometric properties of representation spaces [26, 10].

5 Conclusion

In this work, we introduced Geometric Embedding Mixture (GEM), a regularization method that aligns LLM hidden states with the convex hull of their output vocabulary. Our results demonstrate that this geometric constraint improves model performance by reducing perplexity and increasing isotropy, while simultaneously unlocking *native interpretability*. Unlike standard transformers, GEM-trained models allow for layer-wise decoding using the un-tuned language head. This makes intermediate states readable at the vocabulary level, although trained probes are still needed to fully interpret the model's computation. Our findings suggest that geometric constraints during pre-training can meaningfully improve transparency. We hope this work encourages a shift toward models that are interpretable by construction, ensuring that transparency scales alongside capability.

Acknowledgments and Disclosure of Funding

This work is partially supported by Hong Kong Research Grants Council (RGC) CRF-YCRG C6003-24Y.

References

[1] Nostalgebraist, "Interpreting gpt: the logit lens." https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020. Ac-

cessed: 2026-01-29.

[2] N. Belrose, I. Ostrovsky, L. McKinney, Z. Furman, L. Smith, D. Halawi, S. Biderman, and J. Steinhardt, "Eliciting latent predictions from transformers with the tuned lens," 2025.

[3] K. Ethayarajh, "How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings," 2019.

[4] W. Rudman and C. Eickhoff, "Stable anisotropic regularization," 2024.

[5] M. Blondel, M. E. Sander, G. Vivier-Ardisson, T. Liu, and V. Roulet, "Autoregressive language models are secretly energy-based models: Insights into the lookahead capabilities of next-token prediction," 2025.

[6] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, "Hopfield networks is all you need," 2021.

[7] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang, *et al.*, "A tutorial on energy-based learning," *Predicting structured data*, vol. 1, no. 0, 2006.

[8] B. Zhang and R. Sennrich, "Root mean square layer normalization," 2019.

[9] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[10] S. Noroozizadeh, V. Nagarajan, E. Rosenfeld, and S. Kumar, "Deep sequence models tend to memorize geometrically; it is unclear why," 2025.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[12] L. B. Allal, A. Lozhkov, E. Bakouch, G. M. Blázquez, G. Penedo, L. Tunstall, A. Marafioti, H. Kydlíček, A. P. Lajarín, V. Srivastav, J. Lochner, C. Fahlgren, X.-S. Nguyen, C. Fourrier, B. Burtenshaw, H. Larcher, H. Zhao, C. Zakka, M. Morlon, C. Raffel, L. von Werra, and T. Wolf, "Smollm2: When smol goes big – data-centric training of a small language model," 2025.

[13] G. Penedo, H. Kydlíček, L. B. allal, A. Lozhkov, M. Mitchell, C. Raffel, L. V. Werra, and T. Wolf, "The fineweb datasets: Decanting the web for the finest text data at scale," 2024.

[14] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, *et al.*, "Training compute-optimal large language models," *arXiv preprint arXiv:2203.15556*, 2022.

[15] D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández, "The lambada dataset: Word prediction requiring a broad discourse context," 2016.

[16] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," 2016.

[17] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi, "Piqa: Reasoning about physical commonsense in natural language," in *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[18] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?," 2019.

[19] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "Winogrande: An adversarial winograd schema challenge at scale," *arXiv preprint arXiv:1907.10641*, 2019.

[20] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," *ArXiv*, vol. abs/1803.05457, 2018.

[21] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, "Boolq: Exploring the surprising difficulty of natural yes/no questions," 2019.

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[23] A. Gladkova, A. Drozd, and S. Matsuoka, "Analogy-based detection of morphological and semantic relations with word embeddings: What works and what doesn't," in *Proceedings of the NAACL Student Research Workshop*, 2016.

[24] J. Gao, D. He, X. Tan, T. Qin, L. Wang, and T.-Y. Liu, "Representation degeneration problem in training natural language generation models," 2019.

[25] J. Mu, S. Bhat, and P. Viswanath, "All-but-the-top: Simple and effective postprocessing for word representations," 2018.

[26] C. Xiao, Y. Long, and N. A. Moubayed, "On isotropy, contextualization and learning dynamics of contrastive-based sentence representation learning," 2023.

[27] O. Press and L. Wolf, "Using the output embedding to improve language models," 2017.

[28] H. Inan, K. Khosravi, and R. Socher, "Tying word vectors and word classifiers: A loss framework for language modeling," 2017.

[29] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013.

[30] F. Bertolotti and W. Cazzola, "By tying embeddings you are assuming the distributional hypothesis," in *Forty-first international conference on machine learning*, 2024.

[31] N. Pappas, L. M. Werlen, and J. Henderson, "Beyond weight tying: Learning joint input-output embeddings for neural machine translation," 2018.

[32] N. Pappas and J. Henderson, "Gile: A generalized input-label embedding for text classification," 2019.

[33] M. Majurski, S. Menon, P. Farvardin, and D. Chapman, "A method of moments embedding constraint and its application to semi-supervised learning," 2024.

[34] S. Rajaee and M. T. Pilehvar, "A cluster-based approach for improving isotropy in contextual embedding space," 2021.

[35] T. Jiang, J. Jiao, S. Huang, Z. Zhang, D. Wang, F. Zhuang, F. Wei, H. Huang, D. Deng, and Q. Zhang, "Promptbert: Improving bert sentence embeddings with prompts," 2022.

[36] F. Stollenwerk and T. Stollenwerk, "Better embeddings with coupled adam," 2025.

[37] B. Hoover, Y. Liang, B. Pham, R. Panda, H. Strobelt, D. H. Chau, M. J. Zaki, and D. Krotov, "Energy transformer," 2023.

[38] A. Gladstone, G. Nanduru, M. M. Islam, P. Han, H. Ha, A. Chadha, Y. Du, H. Ji, J. Li, and T. Iqbal, "Energy-based transformers are scalable learners and thinkers," 2025.

[39] A. Langedijk, H. Mohebbi, G. Sarti, W. Zuidema, and J. Jumelet, "Decoderlens: Layerwise interpretation of encoder-decoder transformers," 2024.

[40] D. Halawi, J.-S. Denain, and J. Steinhardt, "Overthinking the truth: Understanding how language models process false demonstrations," 2024.

[41] A. Hunklinger and N. Ferruz, "Toward the explainability of protein language models," 2025.

[42] K. Pal, J. Sun, A. Yuan, B. Wallace, and D. Bau, "Future lens: Anticipating subsequent tokens from a single hidden state," in *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, p. 548–560, Association for Computational Linguistics, 2023.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract mentions some of the results and the intro lists our contributions.

   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: The Conclusion (Section 5) notes that trained probes are still needed to fully interpret the model's computation.

   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: Section 2.3 clearly labels formulas and all related assumptions

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: We provide all the details in the experimental setup3.1 and detailed hyperparameters at 8

   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: The company does not allow for publishing artifacts

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: It's mentioned at the experimental setup 3.1

   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: [No]

   Justification: The results are mostly deterministic

   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: [No]

   Justification: We cannot disclose due to company policy

   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: To the best of our ability we believe it conforms

   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 belongs does not have any direct societal impact.

    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: No such risks

    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: We credit the original models that we borrow the architecture from

    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: [N/A]

    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: [N/A]

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: [N/A]

    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: [N/A]

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.

A Hyperparameters

We provide the complete set of hyperparameters used for pre-training the 135M, 360M, 760M, and 1.7B models in Table 8. Our architecture follows the SmolLM2 specifications [12]. The 135M, 360M, and 760M models use Grouped Query Attention (GQA), and the 1.7B model uses standard multi-head attention, as in the corresponding SmolLM2 configurations. All models use Rotary Positional Embeddings (RoPE). Training was performed on packed token sequences using the AdamW optimizer with a cosine decay schedule.

Attribute	135M	360M	760M	1.7B
Vocab Params	28.3M	47.1M	62.9M	100.7M
Total Params	135M	361M	760M	1.71B
Layers	30	32	32	24
Emb. Dimension	576	960	1280	2048
FFN Dimension	1536	2560	4608	8192
Attention Heads	9	15	20	32
Attention Variant	GQA			MHA
Key-Value Heads	3	5	5	32
Position Emb.	RoPE			
RoPE θ	100000			130000
Layer Norm Type	RMSNorm			
Layer Norm ϵ	1e-5			
Sequence Length	2048			
Vocab Size	49152			
Batch Size	524288			
Training Steps	9536	19073	38146	38146
Training Tokens	5B	10B	20B	20B
Optimizer	AdamW			
Weight Decay	0.1			
Warmup Steps	100			
Peak Learning Rate	1e-3			
Min Learning Rate	4e-5			
Lr Scheduler	Cosine			

Table 8: Hyperparameters

B Ablation

We test various settings of k and λ with the same experimental setup as in the main section for the 135M model.

Table 9: Sensitivity Ablation of k and λ for the 135M Model.

k	λ	**LAMBADA PPL (↓)**	**Wiki PPL (↓)**
- (Base)	-	285.7	51.10
4	0.01	152.8	42.40
4	0.05	138.2	42.17
4	0.10	139.3	41.97
4	0.20	159.6	43.33
8	0.01	126.8	42.02
8 (Main)	0.05	134.0	42.97
8	0.10	136.5	41.66
8	0.20	155.4	43.18
16	0.01	141.8	42.33
16	0.05	146.5	42.49
16	0.10	159.5	42.70
16	0.20	151.2	42.65

Table 10: Comparison with CosReg [24] and GEM with $k \in \{1, 8, |\mathcal{V}|\}$ ($\lambda = 0.05$) for the 135M model trained on 5B tokens.

Method	**LAMBADA PPL (↓)**	**Wiki PPL (↓)**
CE baseline	285.99	51.09
CosReg	213.90	48.17
GEM ($k = 1$)	227.32	48.27
GEM ($k = 8$)	134.23	43.01
GEM ($k = |\mathcal{V}|$)	124.26	41.57

C Extended Geometric Analysis

In this section, we provide a detailed breakdown of the geometric properties of the embedding space. We first define the metrics used to quantify clustering quality and then present the full per-category results.

C.1 Metric Definitions

To rigorously evaluate the semantic organization of the embedding space, we curate 10 distinct semantic categories (e.g., *Digits*, *Months*, *Colors*) containing N_c tokens each. Let $V_c = \{v_1, \ldots, v_{N_c}\}$ be the set of normalized embedding vectors for category c, and $V_{\neg c}$ be the set of all embeddings not in category c. We define the distance metric $d(x, y) = 1 - \cos(x, y)$.

Intra-class Distance. This measures the compactness of a cluster. It is defined as the average pairwise cosine distance between all unique pairs of vectors within the same category:

$$\text{Intra}(c) = \frac{2}{N_c(N_c - 1)} \sum_{i=1}^{N_c} \sum_{j=i+1}^{N_c} d(v_i, v_j) \tag{14}$$

Inter-class Distance. This measures the distinctness of a cluster from the rest of the vocabulary. It is defined as the average cosine distance between vectors in category c and all vectors in other categories:

$$\text{Inter}(c) = \frac{1}{N_c|V_{\neg c}|} \sum_{v \in V_c} \sum_{u \in V_{\neg c}} d(v, u) \tag{15}$$

Separation Score. To quantify how well-separated a category is, we compute the difference between its Inter-class and Intra-class distances. A positive score indicates that items are, on average, closer to members of their own category than to external items:

$$\text{Sep}(c) = \text{Inter}(c) - \text{Intra}(c) \tag{16}$$

Purity@k. This metric evaluates the local neighborhood structure. For each token in a category, we retrieve its k nearest neighbors in the embedding space. Purity is the fraction of those neighbors that share the same semantic category label. We report the average purity across all tokens in the category for $k = 10$:

$$\text{Purity@}k(c) = \frac{1}{N_c} \sum_{v \in V_c} \frac{|\text{KNN}_k(v) \cap V_c|}{k} \tag{17}$$

C.2 Visual and Quantitative Results

Figure 5 visualizes the embedding space for all model scales using UMAP. The Baseline models (top row) exhibit significant mixing of categories, particularly in the 760M scale. In contrast, the GEM models (bottom row) show clear, compact clusters for categories such as Digits (red) and Months (blue).

Table 11 provides the quantitative breakdown for every category. GEM consistently improves the Separation Score and Purity across nearly all categories and scales. For example, the *Digits* category in the 760M model improves from a negative separation score (-0.04) in the Baseline to a strongly positive one (0.55) in GEM, with Purity@10 increasing from 0.0 to 0.45.

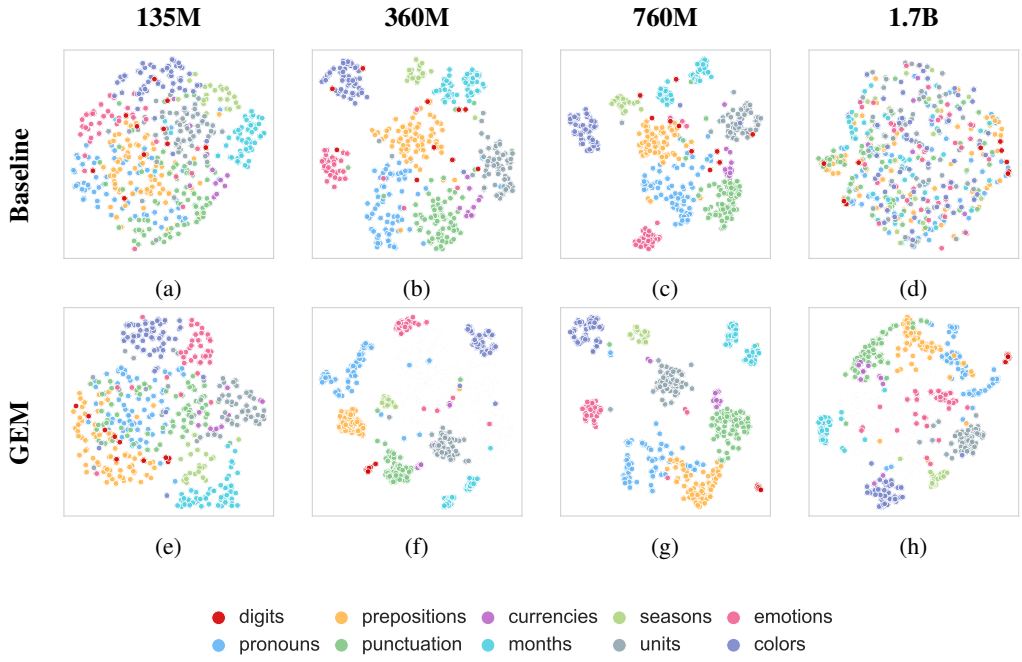


Figure 5: UMAP projections of 10 semantic categories across all model scales. GEM (bottom row) consistently produces tighter, more distinct clusters than the Baseline (top row).

Table 11: Complete Category Analysis: Intra-class Distance, Inter-class Distance, Separation Score (Inter−Intra), and kNN Purity for all model sizes.

		Baseline				GEM			
Category	**Size**	Intra ↓	Inter ↑	Sep ↑	Purity@10 ↑	Intra ↓	Inter ↑	Sep ↑	Purity@10 ↑
Digits	135M	1.0426	1.0011	−0.0415	0.0000	0.9577	1.0111	0.0535	0.0200
	360M	1.0377	0.9961	−0.0416	0.0000	0.7135	1.0091	0.2956	0.2550
	760M	1.0457	1.0015	−0.0442	0.0000	0.4400	0.9957	0.5557	0.4500
	1.7B	0.9649	1.0161	0.0512	0.0500	0.4984	0.9987	0.5003	0.4200
Months	135M	0.9358	0.9938	0.0580	0.2245	0.9141	0.9939	0.0797	0.3255
	360M	0.8835	0.9828	0.0993	0.3755	0.6291	0.9636	0.3345	0.7094
	760M	0.8899	0.9870	0.0971	0.5755	0.7206	1.0038	0.2832	0.5783
	1.7B	0.9239	0.9940	0.0701	0.0311	0.7372	0.9478	0.2107	0.3038
Seasons	135M	0.9342	1.0042	0.0701	0.0636	0.9379	1.0008	0.0629	0.0955
	360M	0.8837	0.9887	0.1049	0.2091	0.5781	0.9941	0.4160	0.4341
	760M	0.8820	0.9858	0.1038	0.2409	0.6759	0.9787	0.3027	0.3273
	1.7B	0.9828	1.0068	0.0240	0.0023	0.7449	1.0010	0.2561	0.1659
Colors	135M	0.9606	0.9981	0.0375	0.0965	0.9481	1.0000	0.0519	0.1561
	360M	0.8961	0.9901	0.0940	0.3018	0.6533	0.9705	0.3173	0.4974
	760M	0.8911	0.9871	0.0959	0.3272	0.7163	0.9784	0.2621	0.3605
	1.7B	0.9634	1.0032	0.0398	0.0061	0.7163	1.0127	0.2965	0.2447
Emotions	135M	0.9702	0.9963	0.0261	0.0342	0.9616	0.9986	0.0370	0.0526
	360M	0.9224	0.9957	0.0734	0.0961	0.7519	0.9473	0.1954	0.1250
	760M	0.9261	0.9842	0.0581	0.0961	0.7945	1.0053	0.2108	0.1250
	1.7B	0.9982	1.0013	0.0031	0.0026	0.8171	1.0078	0.1907	0.0868
Currencies	135M	0.9391	0.9976	0.0586	0.0455	0.9353	0.9948	0.0595	0.0682
	360M	0.8787	0.9873	0.1086	0.1182	0.6904	1.0132	0.3228	0.1409
	760M	0.8678	0.9987	0.1309	0.1455	0.7190	0.9970	0.2780	0.1636
	1.7B	0.9734	0.9918	0.0185	0.0000	0.8103	1.0052	0.1950	0.0818
Units	135M	0.9759	0.9992	0.0233	0.0306	0.9714	0.9992	0.0278	0.0573
	360M	0.9260	0.9952	0.0692	0.1484	0.7426	0.9675	0.2249	0.2290
	760M	0.9194	0.9929	0.0735	0.1855	0.7936	1.0027	0.2091	0.1911
	1.7B	0.9853	0.9988	0.0135	0.0073	0.8164	1.0370	0.2206	0.1298
Prepositions	135M	0.9812	1.0016	0.0204	0.0425	0.9661	1.0058	0.0397	0.1103
	360M	0.9552	0.9933	0.0380	0.1199	0.7938	0.9974	0.2036	0.2986
	760M	0.9524	0.9965	0.0441	0.1534	0.8361	0.9795	0.1434	0.3363
	1.7B	0.9648	1.0082	0.0434	0.0356	0.8003	1.0057	0.2053	0.3192
Pronouns	135M	0.9833	1.0059	0.0226	0.0581	0.9719	1.0052	0.0332	0.1162
	360M	0.9717	0.9965	0.0249	0.1191	0.8873	1.0141	0.1268	0.2610
	760M	0.9723	1.0009	0.0286	0.1456	0.8785	1.0071	0.1286	0.2772
	1.7B	0.9654	1.0043	0.0390	0.0235	0.8375	1.0118	0.1743	0.2493
Punctuation	135M	0.9866	1.0015	0.0149	0.0340	0.9755	1.0089	0.0334	0.0604
	360M	0.9655	0.9999	0.0344	0.0875	0.8358	1.0054	0.1696	0.1694
	760M	0.9403	0.9995	0.0592	0.1611	0.8536	1.0225	0.1689	0.2174
	1.7B	0.9526	1.0176	0.0650	0.0354	0.8152	1.0227	0.2076	0.2056

D Extended Interpretability Results

In the main text, we summarized the ability to decode intermediate hidden states using the pre-trained language head. Table 12 provides the complete layer-wise accuracy for all model scales. We report accuracy using two methods:

- **Direct:** Decoding the raw hidden state h.
- **Norm:** Decoding the normalized state $\text{RMSNorm}(h)$.

The data reveals a consistent trend across all scales: GEM models exhibit significantly higher decodability in the early and middle layers (e.g., layers 5–20) compared to Baseline models. Furthermore, GEM consistently prefers **Direct** decoding, suggesting that the magnitude of the hidden state vector carries semantic information aligned with the embedding space. In contrast, the Baseline model's performance is largely invariant to normalization, indicating a reliance on angular information alone.

Table 12: Detailed Per-Layer Direct vs. Normalized (Logit Lens) Accuracy for all models.

Layer	135M				360M				760M				1.7B			
	Baseline		GEM		Baseline		GEM		Baseline		GEM		Baseline		GEM	
	Direct	Norm	Direct	Norm	Direct	Norm	Direct	Norm	Direct	Norm	Direct	Norm	Direct	Norm	Direct	Norm
0	0.0170	0.0170	0.0364	0.0384	0.0165	0.0165	0.1498	0.0862	0.0230	0.0230	0.1163	0.0718	0.0648	0.0494	0.1311	0.1210
1	0.0230	0.0230	0.0823	0.0786	0.0441	0.0441	0.1487	0.0862	0.0273	0.0273	0.1136	0.0686	0.0729	0.0582	0.1311	0.1203
2	0.0251	0.0252	0.1040	0.0979	0.0541	0.0540	0.1501	0.0856	0.0349	0.0349	0.1137	0.0653	0.0775	0.0689	0.1302	0.1204
3	0.0263	0.0263	0.1080	0.0984	0.0626	0.0626	0.1501	0.0834	0.0379	0.0380	0.1125	0.0652	0.0791	0.0740	0.1310	0.1121
4	0.0262	0.0262	0.1143	0.1032	0.0621	0.0622	0.1462	0.0807	0.0387	0.0387	0.1109	0.0629	0.0828	0.0784	0.1311	0.1159
5	0.0277	0.0277	0.1176	0.1065	0.0663	0.0664	0.1396	0.0756	0.0425	0.0426	0.1062	0.0577	0.0859	0.0797	0.1276	0.1137
6	0.0288	0.0288	0.1281	0.1154	0.0733	0.0733	0.1301	0.0745	0.0470	0.0470	0.1068	0.0520	0.0849	0.0803	0.1271	0.1107
7	0.0299	0.0299	0.1419	0.1255	0.0766	0.0766	0.1249	0.0690	0.0468	0.0468	0.0997	0.0480	0.0896	0.0841	0.1263	0.1119
8	0.0316	0.0316	0.1533	0.1326	0.0592	0.0592	0.1265	0.0707	0.0472	0.0472	0.0977	0.0474	0.0940	0.0806	0.1363	0.1056
9	0.0344	0.0345	0.1512	0.1295	0.0519	0.0519	0.1304	0.0754	0.0484	0.0484	0.0916	0.0448	0.0959	0.0811	0.1372	0.0991
10	0.0359	0.0359	0.1558	0.1319	0.0528	0.0528	0.1309	0.0737	0.0494	0.0493	0.0828	0.0449	0.1048	0.0865	0.1473	0.0916
11	0.0379	0.0380	0.1695	0.1454	0.0486	0.0486	0.1247	0.0675	0.0485	0.0485	0.0831	0.0461	0.1070	0.0918	0.1455	0.0880
12	0.0400	0.0400	0.1735	0.1473	0.0507	0.0507	0.1196	0.0630	0.0504	0.0504	0.0794	0.0456	0.1165	0.0924	0.1471	0.0881
13	0.0435	0.0436	0.1866	0.1617	0.0492	0.0492	0.1280	0.0672	0.0505	0.0506	0.0759	0.0449	0.1238	0.1011	0.1554	0.0941
14	0.0462	0.0462	0.2044	0.1836	0.0518	0.0518	0.1272	0.0715	0.0522	0.0522	0.0798	0.0476	0.1292	0.1107	0.1626	0.1007
15	0.0527	0.0527	0.1992	0.1717	0.0552	0.0553	0.1315	0.0734	0.0541	0.0542	0.0867	0.0524	0.1441	0.1180	0.1708	0.0999
16	0.0595	0.0595	0.2072	0.1808	0.0591	0.0591	0.1319	0.0761	0.0544	0.0544	0.0864	0.0543	0.1468	0.1218	0.1824	0.1105
17	0.0618	0.0618	0.2185	0.1921	0.0608	0.0608	0.1354	0.0819	0.0557	0.0557	0.0909	0.0595	0.1622	0.1345	0.1946	0.1205
18	0.0669	0.0669	0.2335	0.2083	0.0675	0.0675	0.1484	0.0914	0.0584	0.0584	0.0964	0.0647	0.1764	0.1497	0.2149	0.1452
19	0.0795	0.0795	0.2490	0.2170	0.0735	0.0735	0.1628	0.1096	0.0627	0.0628	0.1022	0.0702	0.1880	0.1641	0.2389	0.1678
20	0.0778	0.0778	0.2434	0.2143	0.0777	0.0776	0.1876	0.1326	0.0630	0.0631	0.1123	0.0786	0.2208	0.1900	0.2845	0.2132
21	0.0870	0.0869	0.2563	0.2252	0.0829	0.0829	0.1919	0.1421	0.0681	0.0680	0.1123	0.0839	0.2703	0.2494	0.3522	0.2899
22	0.1037	0.1037	0.2884	0.2648	0.0912	0.0912	0.2128	0.1652	0.0730	0.0730	0.1190	0.0921	0.3626	0.3728	0.4742	0.4521
23	0.1171	0.1171	0.3124	0.2870	0.1019	0.1019	0.2288	0.1857	0.0831	0.0830	0.1338	0.1073	-	-	-	-
24	0.1397	0.1397	0.3370	0.3233	0.1144	0.1144	0.2626	0.2138	0.0890	0.0890	0.1483	0.1203	-	-	-	-
25	0.1641	0.1641	0.3754	0.3545	0.1290	0.1289	0.2918	0.2486	0.0971	0.0972	0.1828	0.1485	-	-	-	-
26	0.2257	0.2257	0.4481	0.4227	0.1554	0.1554	0.3338	0.2844	0.1149	0.1149	0.2324	0.1884	-	-	-	-
27	0.3243	0.3244	0.5213	0.5060	0.2010	0.2011	0.3877	0.3370	0.1405	0.1405	0.2931	0.2402	-	-	-	-
28	0.4975	0.4970	0.6447	0.6416	0.2574	0.2576	0.4653	0.4068	0.1706	0.1706	0.3659	0.3033	-	-	-	-
29	-	-	-	-	0.3507	0.3507	0.5540	0.4925	0.2339	0.2339	0.4685	0.4038	-	-	-	-
30	-	-	-	-	0.5180	0.5180	0.6589	0.6835	0.3840	0.3840	0.6288	0.6035	-	-	-	-

E Interpretability

Causal faithfulness. Probing accuracy measures whether a layer's hidden state contains enough information for a decoder to predict a target token. It does not check whether the decoded distribution corresponds to what the rest of the network uses, since the decoder may simply learn to reproduce the final output. We use a stricter test. Given a decoder that maps h_l to a distribution p_l over the vocabulary, we form the embedding mixture $\mu_l = \sum_i p_{l,i}\, e_i$, rescale it to match the norm of h_l, and substitute μ_l for h_l before running the remaining blocks. We then compare the resulting top-1 prediction to the model's original prediction. The fraction of tokens on which they agree is the causal faithfulness of the decoder at layer l. A decoder with high probing accuracy but low causal faithfulness reproduces the output without describing what later layers actually depend on.

Causal Tuned Lens (Loupe). The standard Tuned Lens [2] fits a per-layer affine map (A_l, b_l) by minimizing $\mathrm{KL}(p_{\text{model}} \,\|\, \mathrm{softmax}(\mathrm{lm_head}(\mathrm{RMSNorm}(A_l h_l + b_l))))$. The training signal asks the affine map to reproduce the final distribution from the layer-l hidden state, so the probe is rewarded for skipping the remaining blocks. We train a second probe, Loupe, with the same architecture but a causal objective. For each layer l we compute $p_l = \mathrm{softmax}(\mathrm{lm_head}(\mathrm{RMSNorm}(A_l h_l + b_l)))$, build $\mu_l = \sum_i p_{l,i}\, e_i$ rescaled to the norm of h_l, replace h_l with μ_l in the forward pass, and minimize $\mathrm{KL}(p_{\text{model}} \,\|\, p_{\text{intervened}})$ over non-padded tokens. Gradients pass through the substitution, the remaining

blocks, and the frozen unembedding back into (A_l, b_l), so the affine map is updated to produce a mixture the rest of the network can use. We train each layer independently for 250 Adam steps with $\text{lr} = 10^{-3}$ and batch size 8 on a held-out FineWeb-Edu split, optionally warm-started from the standard Tuned Lens. The base model and unembedding are frozen. Loupe has the same parameter count as the Tuned Lens (d^2+d per layer), so any difference between the two reflects the training objective rather than capacity.

F Qualitative Analysis

The improved layer-wise decodability allows us to trace the model's "thought process" with greater clarity. For the prompt "The capital of France is", the 760M Baseline model wanders through unrelated concepts before converging at the very last layers:

```
Base:        L26:[situated] -> L27:[Belfast] -> L28:[Belfast]
-> L29:[Luxembourg] -> L30:[Brussels] -> L31:[Paris] ->
L32:[Paris].
```

In contrast, GEM displays a coherent semantic drift towards the target concept much earlier in the forward pass:

```
GEM:        L26:[located] -> L27:[located] -> L28:[France] ->
L29:[France] -> L30:[Paris] -> L31:[Paris] -> L32:[Paris].
```

Next, we show the semantic structure of the embeddings via vector arithmetic analogies. As shown below, GEM retrieves semantically valid answers for standard analogies where the Baseline often fails, retrieving morphological matches or unrelated tokens. This confirms that GEM not only aligns hidden states but also structures the embedding space to capture relational semantics.

```
Baseline:
Analogy: 'Paris' - 'France' + 'Italy'
  - Result: ['Leipzig', 'Macedonia', '3', 'LM', 'ijing']

Analogy: 'mice' - 'mouse' + 'dollar'
  - Result: ['Zimbabwe', 'lb', 'dL', 'bullets', 'currency']

Analogy: 'darker' - 'dark' + 'cool'
  - Result: ['cooling', 'cooled', 'colder', 'sweeter', 'heavier']

GEM:
Analogy: 'Paris' - 'France' + 'Italy'
  - Result: ['Rome', 'Rome', 'Italians', 'Italian', 'Baltimore']

Analogy: 'mice' - 'mouse' + 'dollar'
  - Result: ['dollars', 'euros', 'inflation', 'pounds', 'dividend']

Analogy: 'darker' - 'dark' + 'cool'
  - Result: ['cooler', 'cools', 'milder', 'cooled', 'warmer']
```

G Background and prior works

G.1 Output Layers, Weight Tying, and Embedding Geometry

Early work on the output layer of neural language models highlighted that the pre-softmax weight matrix itself constitutes a useful word embedding and that tying this matrix to the input embedding improves both perplexity and parameter efficiency. Press and Wolf recommend *weight tying*, showing that a single shared embedding matrix used for both input and output yields better word representations and reduces model size without harming performance in language modeling and machine translation [27–30]. They also prototype explicit regularization of the output embedding, indicating that constraining the geometry of the output layer can be beneficial.

Subsequent work generalized weight tying by learning *joint input–output embeddings*. Pappas et al. propose a structure-aware output layer that embeds both translation contexts and output classifiers into

a shared space, allowing more flexible sharing than strict equality of input and output embeddings while controlling the effective capacity of the output layer [31, 32]. They show that such joint embeddings can outperform both untied and strictly tied output layers in neural machine translation, suggesting that explicitly modeling structure in the output embedding space is useful beyond simple parameter sharing.

More generally, recent work on embedding constraints has explored method-of-moments style regularization in latent spaces. For example, a Method of Moments embedding constraint has been proposed to encourage latent clusters with Gaussian structure and to support explicit modeling of joint distributions through Gaussian mixture heads, improving robustness and outlier handling in semi-supervised learning [33]. Conceptually, GEM can be viewed as an improvement of the joint input–output embeddings method, a first-order (mean) moment constraint on the final hidden state with respect to the vocabulary embedding distribution, suggesting possible extensions to higher-order moment matching or mixture-based heads.

G.2 Anisotropy, Isotropy, and IsoScore

Anisotropy in contextual embeddings. Several works have shown that contextualized word representations in models like ELMo, BERT, and GPT-2 are highly *anisotropic*: instead of filling the embedding space uniformly, token vectors occupy a narrow cone and exhibit high average pairwise cosine similarity [3, 24, 25]. Ethayarajh finds that in all layers of these models, contextual embeddings are far from isotropic, and that upper layers are particularly anisotropic while also being more context-specific [3]. This anisotropy has been linked to the *representation degeneration problem*, where embeddings fail to spread out sufficiently, degrading their usefulness for similarity-based tasks [24].

Some works attempt to alleviate this by subtracting dominant principal components or mean vectors, reporting improvements in similarity tasks, though later analyses have questioned whether these operations genuinely restore isotropy [25]. Rajaee and Pilehvar study how fine-tuning affects geometry, showing that fine-tuning often increases the number of elongated directions that carry essential linguistic information, while not necessarily improving global isotropy [34]. They also propose cluster-based approaches for improving isotropy in contextual embedding space, suggesting that simple global centering is insufficient to capture the complex local geometric structure that emerges during fine-tuning.

IsoScore: measuring isotropy properly. Many early measures of isotropy, such as average random cosine similarity or partition scores, were shown to be brittle and not aligned with the mathematical definition of isotropy [4, 3, 25]. Rudman and Gillman introduce *IsoScore*, a metric that quantifies how uniformly a point cloud utilizes the ambient vector space by analyzing the distribution of variance across principal components. IsoScore is designed to be [4]: (i) mean-agnostic (insensitive to translation), (ii) scale- and rotation-invariant with respect to the covariance structure, and (iii) a *global* estimator of isotropy, unlike many previous local or direction-specific metrics. Using IsoScore, they show that contextualized embedding models utilize even fewer effective dimensions than previously thought and challenge claims that deeper layers necessarily become less isotropic. They also emphasize that relying on average cosine similarity can lead to misleading conclusions about isotropy, and advocate IsoScore (and related intrinsic dimensionality measures) as more faithful tools.

Follow-up work further disentangles isotropy from semantic quality. Jiang et al. argue that anisotropy is not, by itself, the root cause of poor semantic isometry in BERT embeddings; instead, latent biases and non-semantic clusters can harm both isotropy and semantic alignment [35]. This suggests that simply increasing isotropy is not guaranteed to improve semantics; geometry-shaping methods must consider both global variance distribution and the semantic structure encoded in elongated directions.

IsoScore-based regularization and optimization. Recently, IsoScore has been used not only as a diagnostic tool but also as a component in training objectives. I-STAR (IsoScore*-based Stable Anisotropic Regularization) proposes an IsoScore-derived regularizer that can either increase or decrease isotropy in embedding space in a stable way during training [4]. This method allows practitioners to tune anisotropy as a hyperparameter, balancing the benefits of more uniform variance distribution against the role of anisotropic directions that encode important features. Other work,

such as Coupled Adam, explores modifying the optimization dynamics for embedding parameters specifically to alleviate anisotropy and improve embedding quality [36].

Implications for GEM and experimental guidance. GEM does not directly optimize IsoScore or any particular anisotropy measure; instead, it enforces that the final predictive hidden state lies near a convex combination of vocabulary embeddings. This could affect anisotropy in the following way: If the vocabulary embeddings themselves are relatively anisotropic, GEM may pull hidden states toward the same anisotropic subspace, potentially increasing alignment and improving IsoScore as shown in our experiments.

Given the nuanced relationship between anisotropy and semantics, we avoid assuming *a priori* that GEM "fixes" anisotropy. Instead, we (i) systematically measure IsoScore (and intrinsic dimensionality) of both vocabulary embeddings and final hidden states in baseline and GEM models; and (ii) correlate these metrics with semantic and generalization benchmark performance.

G.3 Energy Based Models

Energy-based models (EBMs) constitute a class of machine learning models that assign a scalar energy to each configuration of variables and learn to shape the energy landscape such that observed data lie in low-energy regions [7]. Recent research has explored connections between EBMs and Transformer architectures. Some works take a more direct approach by reformulating Transformers themselves as energy-based models, either by designing explicit energy functions within Transformer blocks [37] or by learning a parameterized energy function for inference via optimization [38]. Specifically, Ramsauer et al. [6] show that the Transformer attention mechanism can be interpreted as a modern Hopfield network, thereby providing an energy-based view of attention and establishing theoretical foundations for designing energy functions. In parallel, Blondel et al. [5] provide a unified theoretical perspective showing that autoregressive language models, including Transformers, can be viewed as energy-based models under suitable formulations.

In EBMs interpretation, in contrast to prior work that aims to directly cast Transformers themselves as EBMs, our GEM method—inspired by the unifying perspectives of [6, 5]—does not modify the Transformer architecture into an energy-based model. Instead, we introduce an auxiliary minimum-energy regularization term, motivated by classical EBM principles [7], into standard Transformer-based language models. This GEM regularization encourages the model to learn representations that better conform to energy-based interpretations, thereby enhancing the interpretability of the learned language representations.

G.4 Interpretability via Intermediate-Layer Decoding

A common approach to interpreting hidden representations is to *decode* them into vocabulary space. The *logit lens* applies the model's final LM head to intermediate residual-stream activations, producing a sequence of layer-wise predictive distributions [1, 39]. For a hidden state $\tilde{h}_\ell^{(t)}$ at layer ℓ, embedding matrix E and the RMSNorm $\mathrm{norm}(\cdot)$ of the final layer, the induced distribution is

$$p_\ell(x_{t+1} \mid x_{\le t}) = \mathrm{softmax}\big(E \cdot \mathrm{norm}(\tilde{h}_\ell^{(t)})\big), \tag{18}$$

revealing how the model's beliefs about the next token evolve across layers. This technique has been used to study the emergence of syntactic and semantic structure, reasoning dynamics, and safety-relevant behaviors in both language and protein models [40, 41].

In vanilla transformer models, the original logit lens can be brittle because intermediate representations may lie in a basis misaligned with the final unembedding, producing noisy or misleading predictions. The *tuned lens* mitigates this by learning a small affine translator per layer that maps hidden states into the basis expected by the LM head [2, 42]. Moreover, these translators often transfer across nearby layers, and causal basis analyses show that the influential directions for tuned-lens outputs align with those driving the model's own predictions, supporting their interpretability.

Our GEM regularizer is naturally compatible with these tools. By encouraging the post-normalization hidden state $\tilde{h}_t$ to align with the expectation of token embeddings, GEM biases the final representations toward the same embedding geometry used by the LM head. Consequently, applying directly the raw logit lens (LM head) to intermediate layers produce more semantically meaningful intermediate

predictions, reducing the basis mismatch that tuned lenses are designed to correct and eliminating the need for additional translator training.