Dispersing Embeddings in Transformer Layers Improves Generalization of Language Models

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Large language models achieve remarkable performance through ever-increasing parameter counts, yet scaling imposes steep computational costs. We observed a geometric phenomenon called *embedding condensation*, where token representations collapse into narrow cones as they propagate through smaller models. Through systematic measurements across multiple transformer families, we show that small models such as `ALBERT-base` and `GPT-2` exhibit severe condensation, whereas large models maintain embedding dispersion. This suggests that superior performance partly arises from sustained representational diversity. We formulate four losses that explicitly encourage embedding dispersion during training. Experiments demonstrate that these losses mitigate condensation, recover dispersion patterns seen in larger models, and yield consistent performance gains across 10 benchmarks, offering a principled path toward improving smaller transformers without additional parameters.

1 Introduction

The remarkable success of large-scale transformer models has fundamentally transformed natural language processing, with performance consistently improving as parameter counts scale from millions to hundreds of billions [1, 2, 3]. However, this scaling paradigm presents significant practical challenges: larger models require substantial computational resources [4, 5, 6], making them inaccessible for many applications. This motivates a critical question: *Can we identify and replicate the key properties that make large models effective, thus improving smaller models without simply adding more parameters?*

Recent theoretical work has shown that transformer embeddings mathematically tend to cluster toward a single point as depth approaches infinity [7], but the empirical manifestation of this phenomenon and its relationship to model performance remain underexplored. In this work, we provide a comprehensive empirical analysis of what we term **embedding condensation**: the tendency for token representations in smaller transformer models to collapse into narrow directional cones as they propagate through deeper layers. Through systematic measurement of pairwise cosine similarities across multiple transformer families, we demonstrate that smaller models (e.g., `GPT2`, `ALBERT-base`) exhibit severe condensation, with token representations becoming increasingly aligned and losing representational diversity (Figure 1). In contrast, larger models (e.g., `GPT2-xl`, `ALBERT-xxlarge`) naturally maintain **embedding dispersion**, which we define as diverse representation directions that preserve expressive capacity.

This geometric perspective reveals a fundamental insight: *condensation may be a key bottleneck limiting the expressiveness of smaller transformers*. We hypothesize that the superior performance of large-scale models is partly a consequence of their ability to maintain representational breadth, suggesting that counteracting condensation could narrow the performance gap between small and large models without increasing parameter count.

To test this hypothesis, we propose four variants of **dispersion losses** that explicitly encourage embedding dispersion during training, serving as auxiliary objectives that promote representational diversity. Our empirical evaluation demonstrates that these losses successfully counteract embedding condensation in smaller models, restoring representational dispersion. More importantly, this geometric improvement leads to overall performance gains on average across 10 language understanding benchmarks when applied to `GPT2` during mid-training.

The key contributions of this work are listed below.

1. We provide an empirical characterization of embedding condensation across transformer scales, revealing a clear size-dependent geometric phenomenon.
2. We formulate four geometrically motivated dispersion loss variants that counteract condensation through different mechanisms. Compared to their existing counterparts in the literature, our implementations include specific design choices to maintain training stability and reduce parameter search space.
3. We demonstrate that explicitly encouraging dispersion improves the performance of smaller models, offering a path toward closing the gap with larger models.

2 The Embedding Condensation Phenomenon

Consider a sequence of N tokens and let $\mathcal{Z}^{(l)} = [z_1^{(l)}, \ldots, z_N^{(l)}]^\top \in \mathbb{R}^{N\times d}$ denote the token embeddings after layer l in a transformer. In the eyes of physicists, $\mathcal{Z}^{(l)}$ can be interpreted as N particles in a d-dimensional space, and transformer layers are external impacts on the particle system. A theory paper [7] has mathematically proven that these embeddings tend to cluster to a single point as $l \to \infty$, but limited empirical evidence has been provided.

In this work, we empirically analyze the spread of $\mathcal{Z}^{(l)}$ across depth l and across model scales, and identify what we term the ***embedding condensation phenomenon***.

2.1 Quantifying layer-by-layer embedding spread in transformers

Let $z_i^{(l)} \in \mathbb{R}^d$ denote the embedding of token i after layer l, we measure the spread of embeddings using pairwise cosine similarities $\texttt{cossim}\left(z_i^{(l)}, z_j^{(l)}\right) = \frac{z_i^{(l)\top} \cdot z_j^{(l)}}{\|z_i^{(l)}\| \cdot \|z_j^{(l)}\|}$.

Cosine similarities lie in $[-1, 1]$, with a value of 1 indicating complete directional alignment, -1 indicating opposite directions, and 0 indicating orthogonality.

For each layer l, we compute pairwise similarities $\{\texttt{cossim}(z_i^{(l)}, z_j^{(l)})\}$ across all token pairs after feeding the input sequence to the transformer. The resulting values form a distribution that we visualize as a histogram for each layer. By stacking these histograms across depth, we create a heatmap that highlights the progression of embedding spread layer by layer.

In this work, every heatmap is created using a population average over $n = 100$ randomly selected input sequences from `wikitext-103` [8].

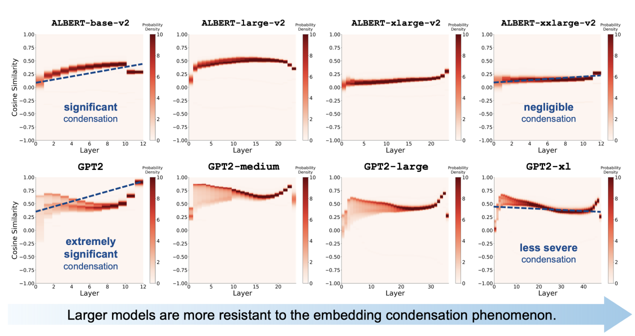

Figure 1: Smaller models (e.g., `ALBERT-base`, `GPT2`) exhibit the embedding condensation phenomenon, as token cosine similarities become increasingly positive as the embeddings proceed to deeper layers. Larger models (e.g., `ALBERT-xxlarge`, `GPT2-xl`) are less vulnerable to embedding condensation, suggesting that broader representation spread is a key property of larger models, and potentially correlated with model performance.

We have experimented with different types of input text (`pubmed_qa` [9], `imdb` [10], and `squad` [11]), and the trends remain highly independent of the dataset.

2.2 Embedding condensation and its implication

Applying the above analysis to multiple transformer families reveals a clear model-size-dependent trend. As shown in Figure 1, smaller models such as `ALBERT-base` and `GPT2` exhibit a **sharp upward drift** of cosine similarity distributions with depth. The embeddings become increasingly aligned, and in `GPT2` the distribution collapses almost entirely near 1, indicating a near-perfect directional alignment. `ALBERT-base` shows the same tendency, though its collapse remains less extreme. We refer to this degeneracy as ***embedding condensation***.

In contrast, larger models such as `ALBERT-xxlarge` and `GPT2-xl` maintain relatively non-extreme cosine similarities across all layers, indicating that they are naturally more resistant to embedding condensation. We refer to this behavior as ***embedding dispersion***.

This geometric view highlights an important implication: condensation reduces the diversity of directions in which tokens can be represented, effectively narrowing the model's expressive capacity (Figure 2). Dispersion, on the other hand, preserves representational breadth, which may underlie the superior performance of large-scale models. These observations motivate the following hypothesis.

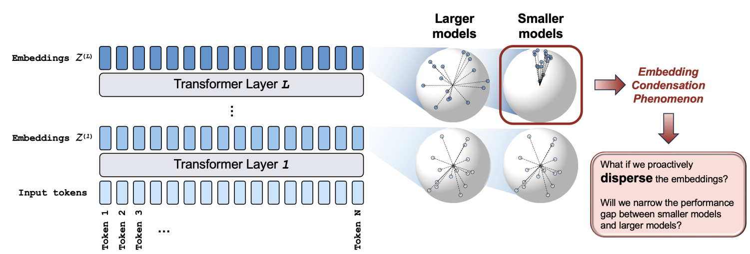

Figure 2: Conceptual illustration of embedding condensation.

HYPOTHESIS: DISPERSION UNDERLIES THE POWER OF LARGER MODELS.

Embedding condensation reduces the expressiveness of small transformers by collapsing token representations into narrow cones. We hypothesize that by **counteracting condensation** and **encouraging dispersion during training**, smaller models can recover properties seen in larger models, thereby *narrowing the performance gap without increasing the number of parameters*.

3 Dispersion Losses for Transformer Embeddings

Our hypothesis motivates the design of auxiliary objectives that explicitly promote embedding dispersion during training. For this purpose, we propose to augment the training loss with a dispersion loss as a regularizer, which gives $\mathcal{L} = \mathcal{L}_{\text{train}} + \lambda_{\text{disp}} \cdot \mathcal{L}_{\text{disp}}$.

We implemented four variants of $\mathcal{L}_{\text{disp}}$, each capturing dispersion through a different geometric lens. The formulations of these variants are summarized in Table 1 and illustrated in Figure 3.

Table 1: **Variants of dispersion losses for transformers.** For the Orthogonalization variant, the distance margin is fixed to $\frac{1}{2}$ since we use angular distance, where $\frac{1}{2}$ corresponds to orthogonality and thus serves as the ideal margin. For ℓ_2-repel and Angular spread, we adopt the `log-sum-exp` trick for numerical stability, which differs from $\log(\text{mean}(\exp(\cdot)))$ only by an additive constant. For ℓ_2-repel, a norm regularization term is added to prevent unbounded expansion of embeddings.

Variant	For generative modeling in diffusion-based models	**For improving generalization performance of smaller language models**	
	formulation [12]	**formulation (ours)**	term definition
Decorrelation	$\sum_{m,n} \mathrm{Cov}^2_{mn}$	$\sum_{m,n}^{m\neq n} \mathrm{Cov}^2_{mn}$	$\mathrm{Cov}^2 = \frac{\mathcal{Z}_c^\top \mathcal{Z}_c}{d-1}$, $\mathcal{Z}_c = \frac{\mathcal{Z}-\mu_d(\mathcal{Z})}{\sigma_d(\mathcal{Z})}$
ℓ_2-repel	$\log \mathbb{E}_{i,j}[\exp(-D(z_i,z_j)/\tau)]$	$\log \sum_{i,j}^{i\neq j}[\exp(-D(z_i,z_j)/\tau)] + \lambda_{\text{norm}}\|\mathcal{Z}\|_2^2$	$D(z_i,z_j) = \|\mathcal{Z}_{i,:} - \mathcal{Z}_{:,j}\|_2^2$
Angular spread	$\log \mathbb{E}_{i,j}[\exp(-D(z_i,z_j)/\tau)]$	$\log \sum_{i,j}^{i\neq j}[\exp(-D(z_i,z_j)/\tau)]$	$D(z_i,z_j) = \frac{\arccos(\texttt{cossim}(z_i,z_j))}{\pi}$
Orthogonalization	$\mathbb{E}_{i,j}[\texttt{max}(0, \epsilon - D(z_i,z_j))^2]$	$\mathbb{E}_{i,j}^{i\neq j}[\texttt{max}(0, \frac{1}{2} - D(z_i,z_j))^2]$	$D(z_i,z_j) = \frac{\arccos(\texttt{cossim}(z_i,z_j))}{\pi}$

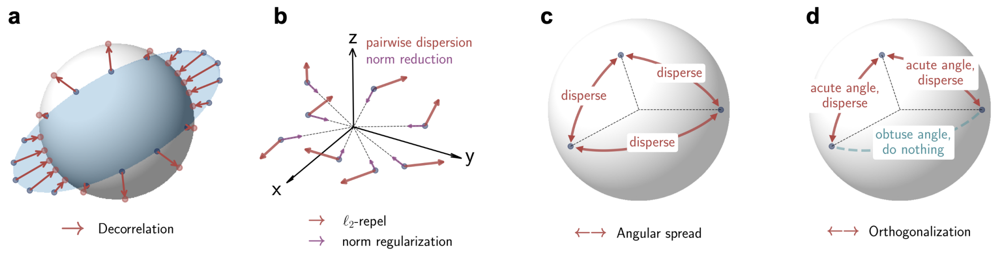

Figure 3: **Illustration of how the four dispersion loss variants respectively promote representation dispersion.** (a) Decorrelation loss suppresses off-diagonal covariance, encouraging different feature dimensions to remain uncorrelated. (b) ℓ_2-repel loss drives pairwise separation in Euclidean space, while the norm regularization prevents unbounded expansion. (c) Angular spread loss enforces uniform angular dispersion by spreading out all pairs along the sphere. (d) Orthogonalization loss selectively spreads out vectors forming acute angles while leaving cobtuse ones unchanged.

4 Empirical Results

We mid-train GPT-2 models for 200M tokens on `wikitext-103` starting from pre-trained weights. Full pre-training from scratch is computationally expensive, which we leave for future investigations.

4.1 Dispersion loss counteracts the embedding condensation phenomenon

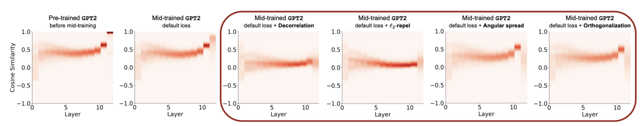

Figure 4: Dispersion losses counteract condensation.

Our dispersion losses effectively counteract the embedding condensation phenomenon (Figure 4). While pre-trained `GPT2` exhibits severe condensation (similarities collapse to 1 in deeper layers) and standard mid-training provides minimal improvement, all four dispersion variants significantly restore the natural non-extreme cosine similarities characteristic of larger models.

4.2 Dispersion loss improves model performance in mid-training

Dispersion losses consistently improve downstream performance (Table 2). Over a diverse set of 10 language tasks, all four dispersion variants outperform the baseline with default cross-entropy loss. Improvements are consistent in most tasks, with particularly strong gains in `LAMBADA`. These results validate our hypothesis: counteracting condensation through geometric regularization improves language understanding, bringing smaller models closer to larger counterparts.

5 Conclusion

We identified embedding condensation as a key limitation of smaller transformers and demonstrated that dispersion losses effectively counteract this phenomenon. In our future endeavors, we will (1) identify better text corpses than `wikitext-103` that may be more beneficial during mid-training, and (2) extend our experiments to pre-training from scratch, as a more direct and definitive investigation.

Table 2: Using dispersion losses during mid-training improve `GPT2` performance on language tasks.

Model	Mid-training		Zero-shot						Few-shot (1)				**Average↑**
	Train	Variant	$\texttt{ANLI}_{\texttt{R2}}$↑	$\texttt{LAMBADA}_{\texttt{openai}}$↑	`OpenbookQA`↑	`PIQA`↑	`TrustfulQA`↑	`WinoGrande`↑	$\texttt{ARC}_{\texttt{easy}}$↑	$\texttt{ARC}_{\texttt{challenge}}$↑	`MedMCQA`↑	`MMLU`↑	
`GPT2`	✗	—	34.4	30.0	16.0	62.0	40.4	53.2	43.2	17.2	25.2	25.2	34.68
	✓	Default loss	$35.0_{(+0.6)}$	$34.4_{(+4.4)}$	$16.2_{(+0.2)}$	$62.2_{(+0.2)}$	$42.7_{(+2.3)}$	$53.2_{(+0.0)}$	$44.4_{(+1.2)}$	$17.2_{(+0.0)}$	$23.6_{(-1.6)}$	$25.5_{(+0.3)}$	$35.44_{(+0.76)}$
		Decorrelation	$35.2_{(+0.8)}$	$35.0_{(+5.0)}$	$15.8_{(-0.2)}$	$61.8_{(-0.2)}$	$42.5_{(+2.1)}$	$52.6_{(-0.6)}$	$43.6_{(+0.4)}$	$17.8_{(+0.6)}$	$25.6_{(+0.4)}$	$25.8_{(+0.6)}$	$35.57_{(+0.89)}$
		ℓ_2-repel	$35.0_{(+0.6)}$	$36.2_{(+6.2)}$	$16.6_{(+0.6)}$	$61.6_{(-0.4)}$	$45.4_{(+5.0)}$	$55.8_{(+2.6)}$	$45.0_{(+1.8)}$	$18.4_{(+1.2)}$	$24.0_{(-1.2)}$	$24.8_{(-0.4)}$	$\mathbf{36.28}_{(+1.60)}$
		Orthogonalization	$35.6_{(+1.2)}$	$34.6_{(+4.6)}$	$16.2_{(+0.2)}$	$62.0_{(+0.0)}$	$43.0_{(+2.6)}$	$53.2_{(+0.0)}$	$43.0_{(-0.2)}$	$18.0_{(+0.8)}$	$25.0_{(-0.2)}$	$25.6_{(+0.4)}$	$35.63_{(+0.95)}$
		Angular spread	$35.4_{(+1.0)}$	$34.8_{(+4.8)}$	$16.4_{(+0.4)}$	$61.0_{(-1.0)}$	$43.2_{(+2.8)}$	$55.0_{(+1.8)}$	$44.2_{(+1.0)}$	$17.8_{(+0.6)}$	$24.8_{(-0.4)}$	$25.4_{(+0.2)}$	$35.80_{(+1.12)}$
`GPT2-m`	✗	—	33.4	40.6	36.4	18.8	66.4	40.6	52.6	49.8	20.4	25.2	38.42
	✓	Default loss	$33.2_{(-0.2)}$	$43.2_{(+2.6)}$	$36.6_{(+0.2)}$	$19.0_{(+0.2)}$	$68.0_{(+1.6)}$	$44.2_{(+3.6)}$	$53.6_{(+1.0)}$	$48.8_{(-1.0)}$	$19.6_{(-0.8)}$	$25.1_{(-0.1)}$	$39.13_{(+0.71)}$
		Decorrelation	$33.4_{(+0.0)}$	$45.4_{(+4.8)}$	$38.6_{(+2.2)}$	$18.8_{(+0.0)}$	$66.4_{(+0.0)}$	$43.8_{(+3.2)}$	$54.4_{(+1.8)}$	$48.0_{(-1.8)}$	$19.6_{(-0.8)}$	$25.4_{(+0.2)}$	$39.39_{(+0.97)}$
		ℓ_2-repel	$33.6_{(+0.2)}$	$44.4_{(+3.8)}$	$38.0_{(+1.6)}$	$18.8_{(+0.0)}$	$67.2_{(+0.8)}$	$44.2_{(+3.6)}$	$52.8_{(+0.2)}$	$48.0_{(-1.8)}$	$19.8_{(-0.6)}$	$25.3_{(+0.1)}$	$39.21_{(+0.79)}$
		Orthogonalization	$33.2_{(-0.2)}$	$45.2_{(+4.6)}$	$37.6_{(+1.2)}$	$18.6_{(-0.2)}$	$67.8_{(+1.4)}$	$43.6_{(+3.0)}$	$53.2_{(+0.6)}$	$48.6_{(-1.2)}$	$20.0_{(-0.4)}$	$25.0_{(-0.2)}$	$39.28_{(+0.86)}$
		Angular spread	$33.4_{(+0.0)}$	$45.0_{(+4.4)}$	$39.4_{(+3.0)}$	$19.2_{(+0.4)}$	$67.4_{(+1.0)}$	$43.8_{(+3.2)}$	$52.2_{(-0.4)}$	$50.0_{(+0.2)}$	$20.2_{(-0.2)}$	$25.4_{(+0.2)}$	$\mathbf{39.60}_{(+1.18)}$
`GPT2-l`	✗	—	33.4	47.6	19.6	71.4	38.9	59.0	53.8	22.4	26.6	25.5	39.83
`GPT2-xl`	✗	—	36.2	49.8	22.8	72.6	38.0	57.8	58.4	24.2	27.2	25.1	41.21

References

[1] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[2] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Jacob Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

[3] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

[4] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[5] Abhimanyu Dubey, Aaron Grattafiori, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Angela Fan, Anirudh Goyal, Aurelien Rodriguez, , et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[6] OpenAI. Introducing gpt-5. OpenAI Blog, 2025.

[7] Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. A mathematical perspective on transformers. *Bulletin of the American Mathematical Society*, 62(3):427–479, 2025.

[8] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *International Conference on Learning Representations*, 2017.

[9] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua 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.

[10] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, 2011.

[11] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[12] Runqian Wang and Kaiming He. Diffuse and disperse: Image generation with representation regularization. *arXiv preprint arXiv:2506.09027*, 2025.