Dispersion Loss Counteracts Embedding Condensation and Improves Generalization in Small Language Models

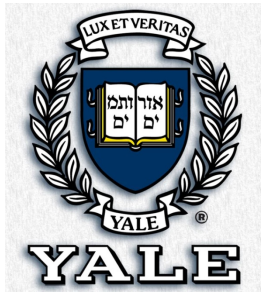
Chen Liu [*1] **Xingzhi Sun** [*1] **Xi Xiao** [*23] **Alexandre Van Tassel** [*1] **Ke Xu** [1] **Kristof Reimann** [1] **Danqi Liao** [1]
Mark Gerstein [1] **Tianyang Wang** [2] **Xiao Wang** [3] **Smita Krishnaswamy** [1]

KrishnaswamyLab/LM-Dispersion Project Page

* Equal contribution [1] Yale University [2] University of Alabama at Birmingham [3] Oak Ridge National Laboratory.
Correspondence to: Smita Krishnaswamy <smita.krishnaswamy@yale.edu>.

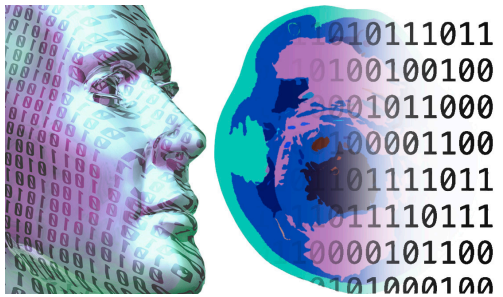
What makes LLMs better than small LMs?
Data? Parameters? Geometry might play a role!

ICML
International Conference
On Machine Learning



UAB
The University of
Alabama at Birmingham

OAK RIDGE
National Laboratory



Motivation

Motivation

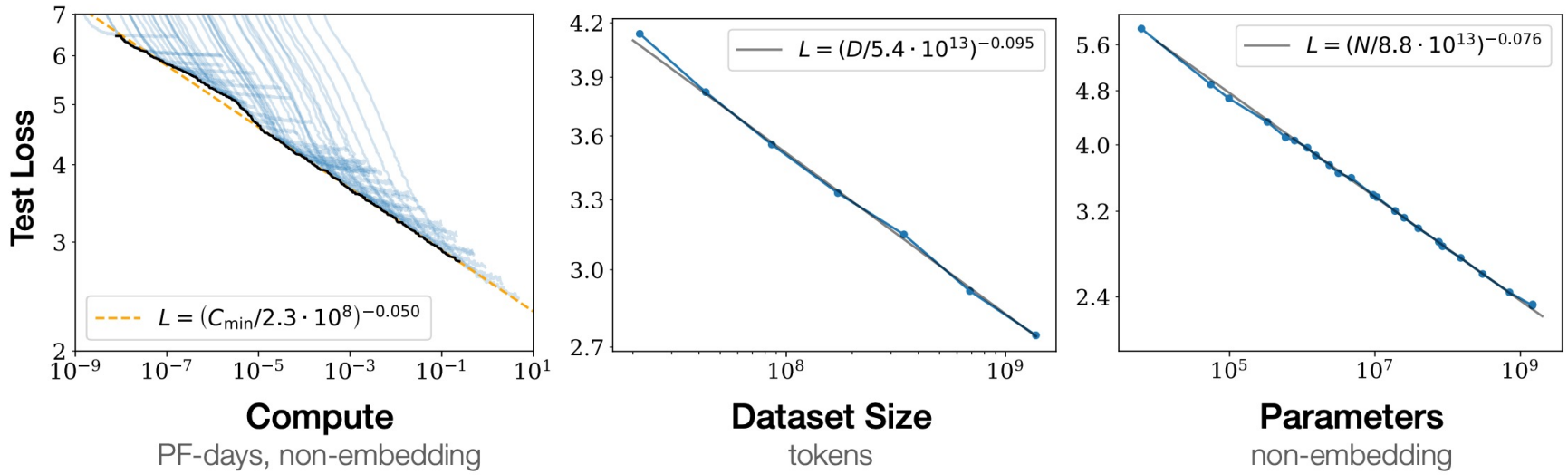
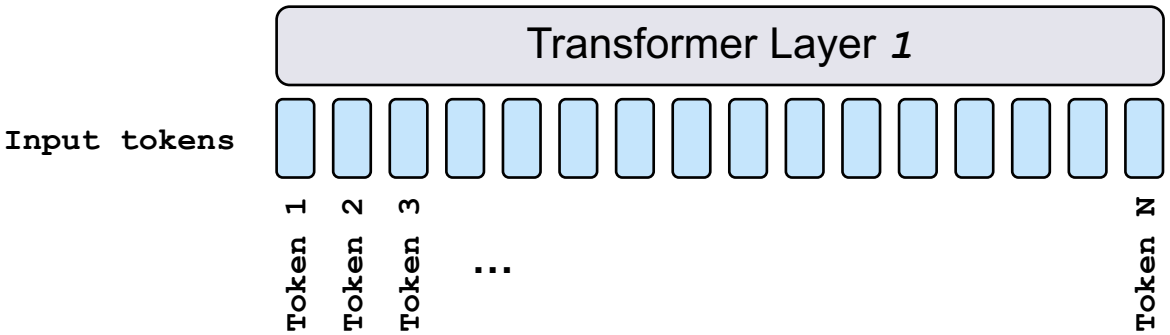


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

[2]Here we display predicted compute when using a sufficiently small batch size. See Figure 13 for comparison to the purely empirical data.

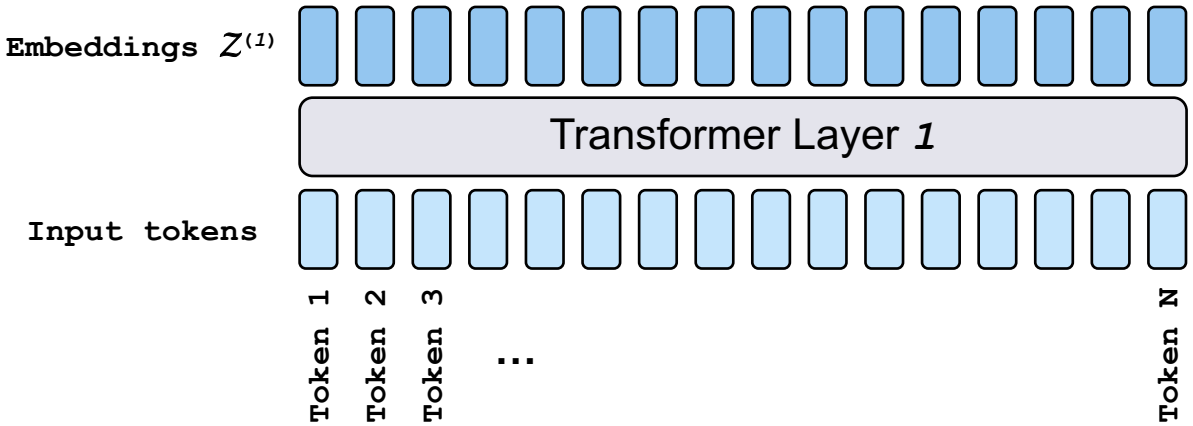
Kaplan, Jared, et al. "Scaling laws for neural language models." *arXiv preprint arXiv:2001.08361* (2020).

Motivation



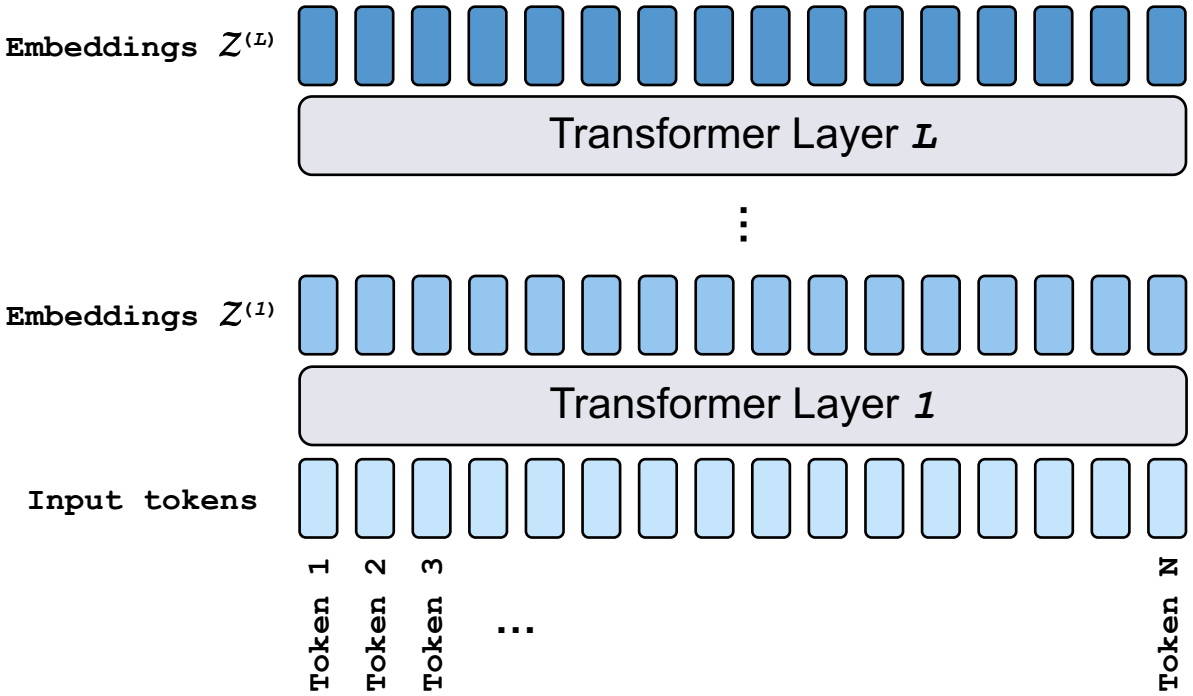
This figure is conceptual. Experimental results will be shown later.

Motivation



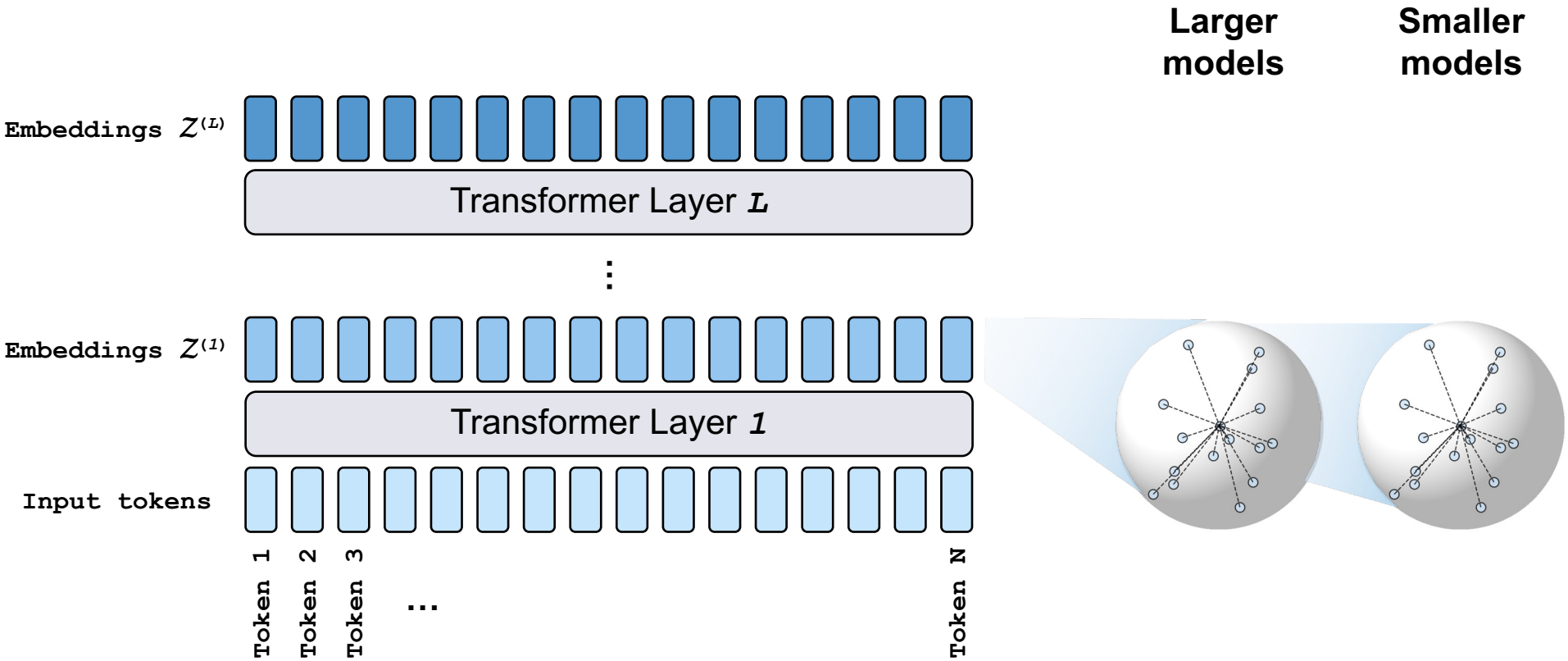
This figure is conceptual. Experimental results will be shown later.

Motivation



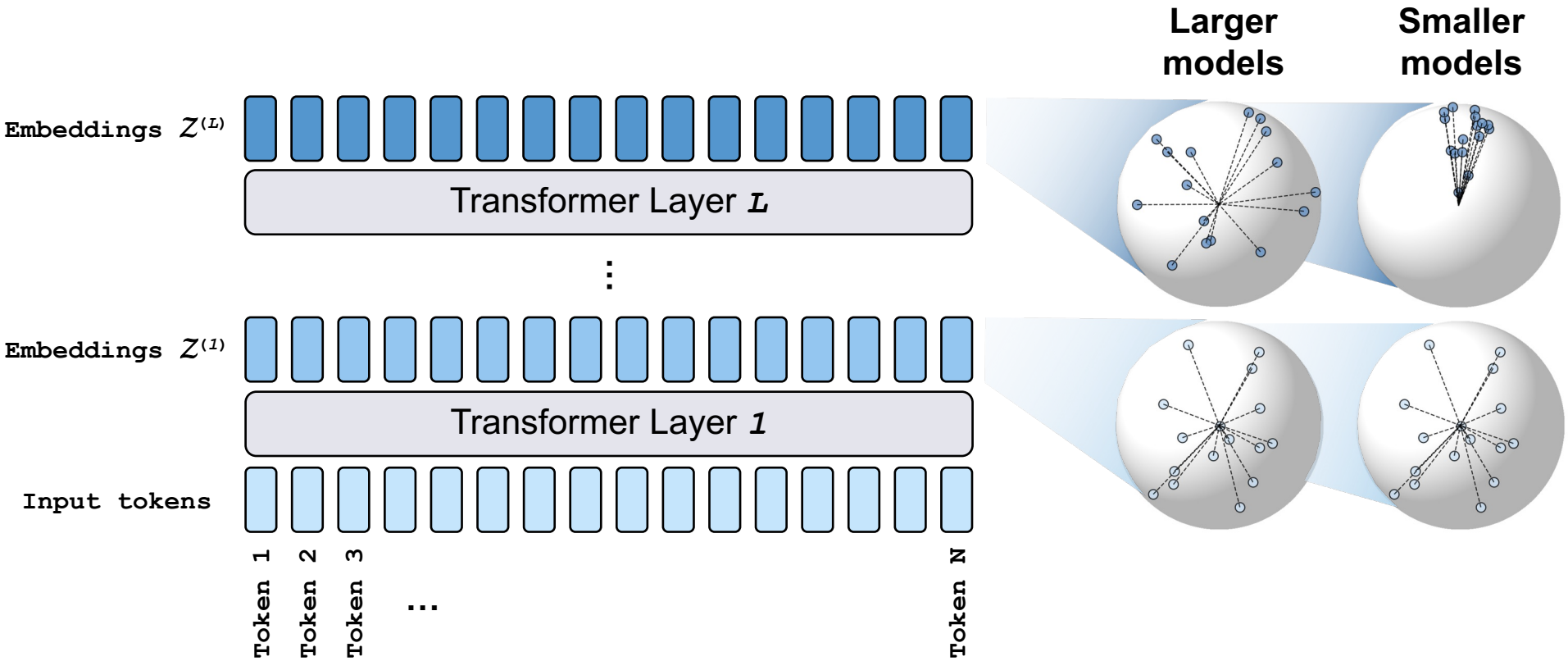
This figure is conceptual. Experimental results will be shown later.

Motivation



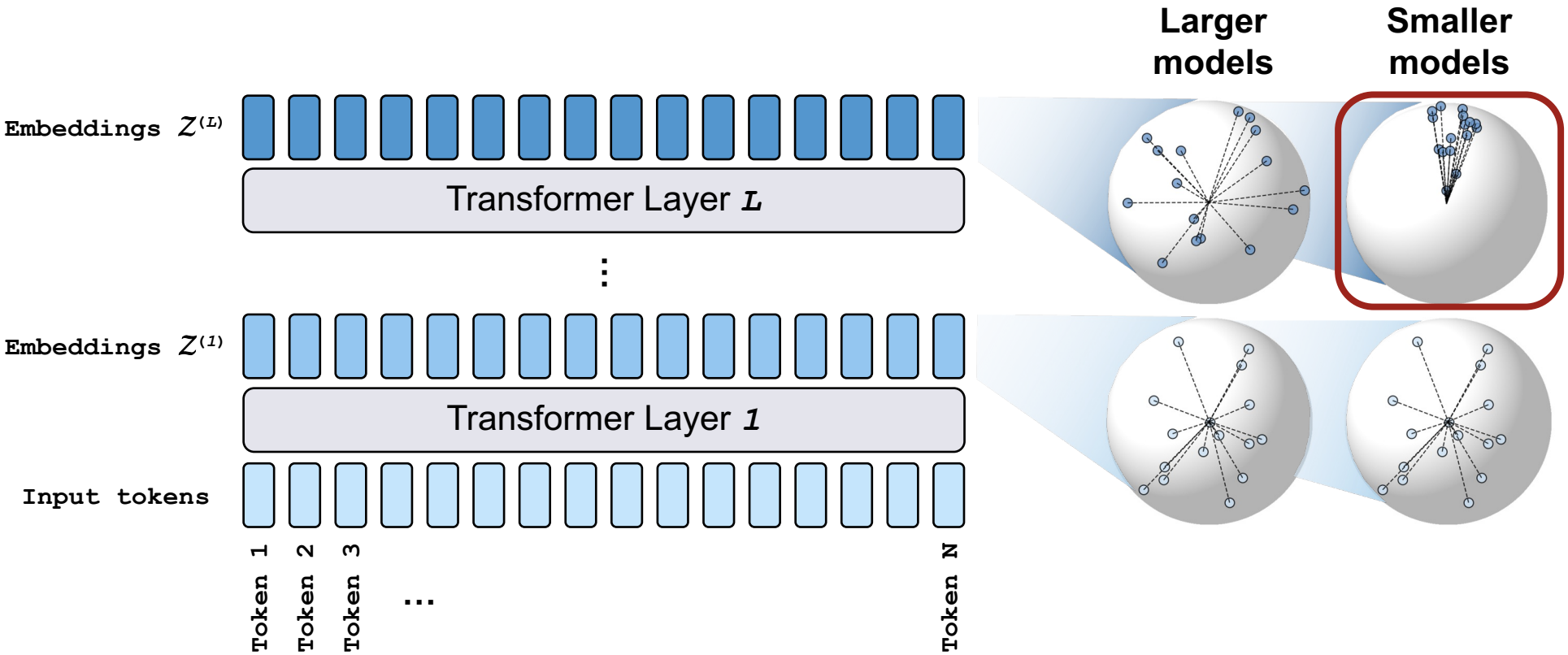
This figure is conceptual. Experimental results will be shown later.

Motivation



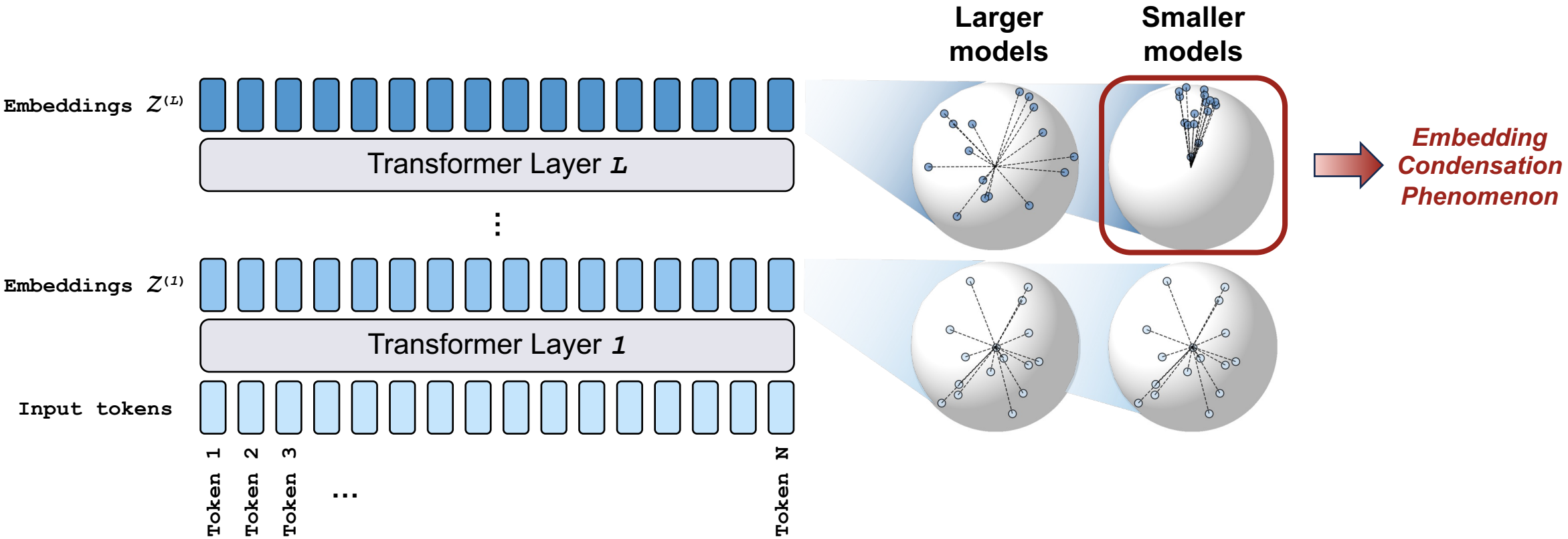
This figure is conceptual. Experimental results will be shown later.

Motivation



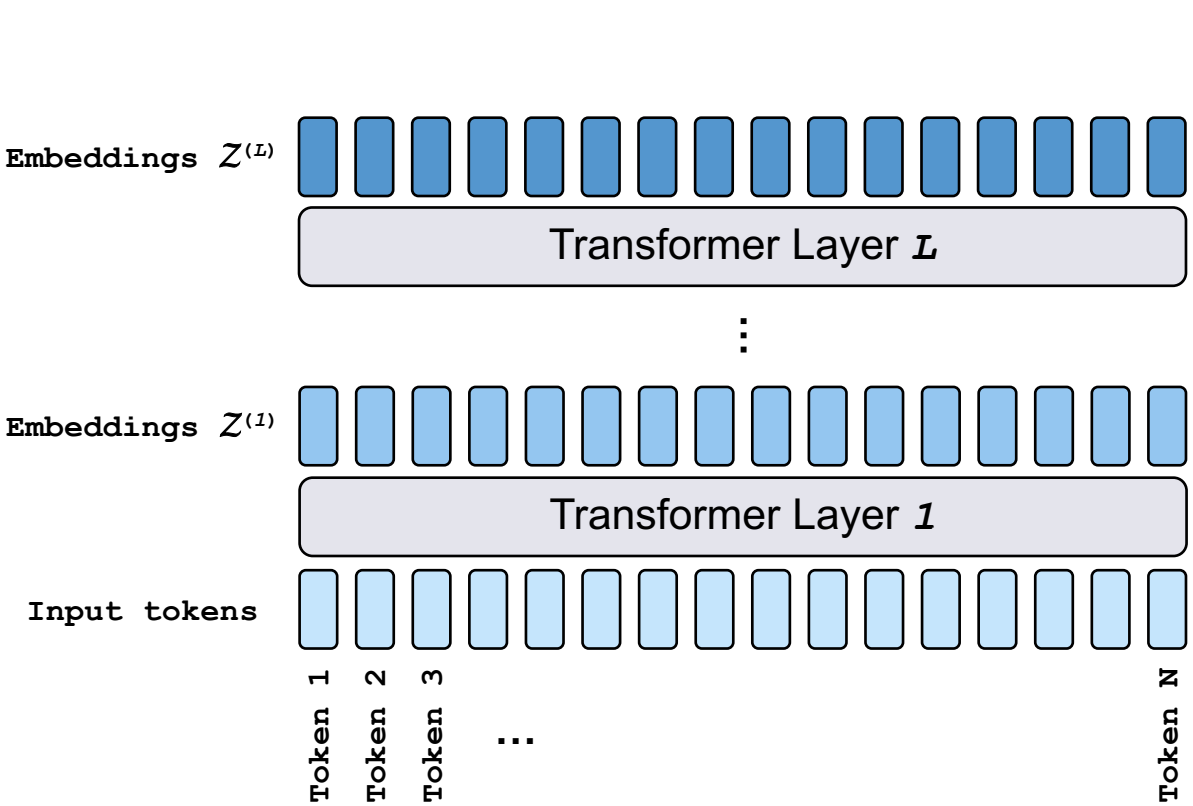
This figure is conceptual. Experimental results will be shown later.

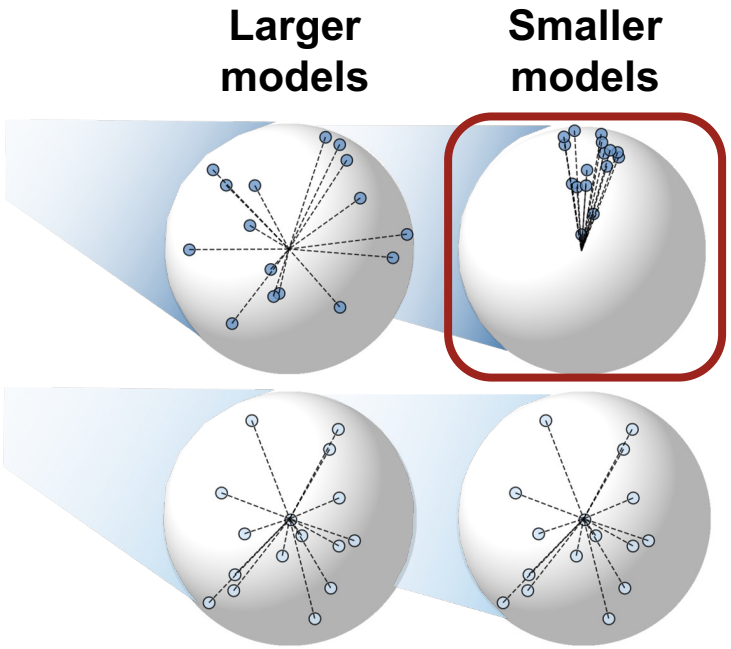
Motivation



This figure is conceptual. Experimental results will be shown later.

Motivation





Embedding Condensation Phenomenon

What if we proactively **disperse** the embeddings?

Will we narrow the performance gap between smaller models and larger models?

This figure is conceptual. Experimental results will be shown later.

Embedding Condensation

Four features of embedding condensation

1. Feature 1
2. Feature 2
3. Feature 3
4. Feature 4

Embedding Condensation

Four features of embedding condensation

1 Feature 1

More severe in smaller models than in larger counterparts (Figure 2).

2 Feature 2

Reproducible under confounder-controlled settings (Figure 3).

3 Feature 3

Emerging at model initialization and gets alleviated by pre-training (Figure 4).

4 Feature 4

Not resolved by knowledge distillation from a larger model (Figure 5).

Embedding Condensation

1 Feature 1

More severe in smaller models than in larger counterparts (Figure 2).

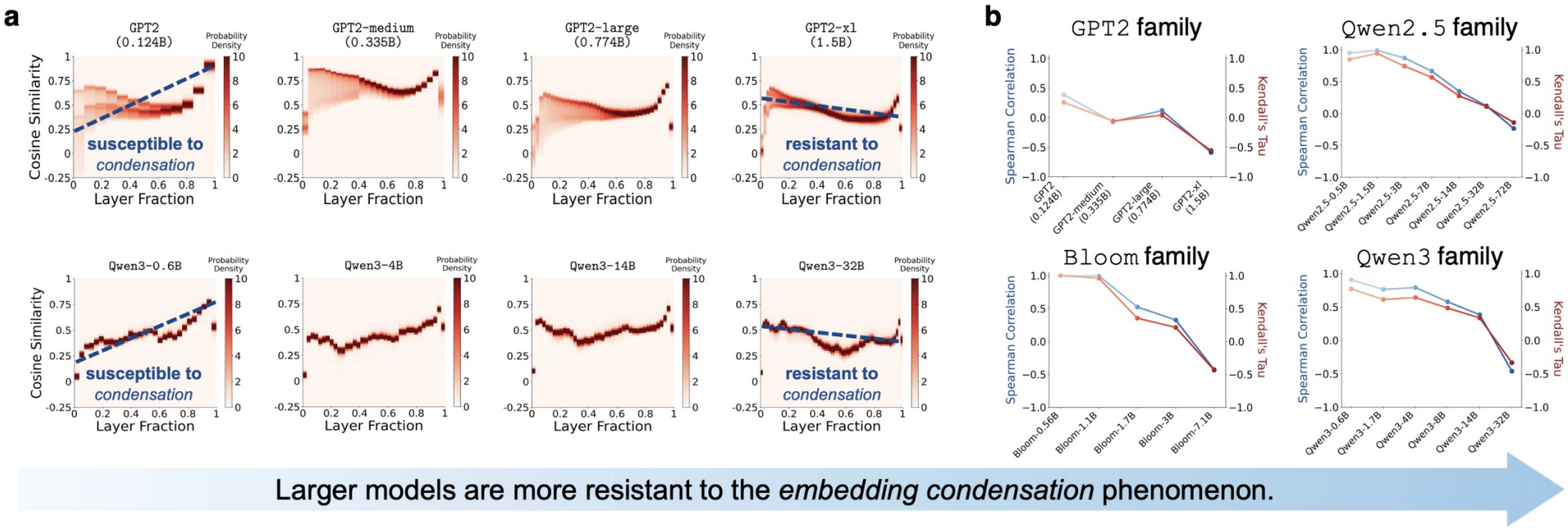


Figure 2. Qualitative and quantitative observations of the embedding condensation phenomenon. **a.** The cosine similarity heatmaps demonstrate that smaller models (e.g., `GPT2`, `Qwen3-0.6B`) are susceptible to condensation, since token cosine similarities become increasingly positive as the embeddings proceed to deeper layers. In contrast, larger models (e.g., `GPT2-xl`, `Qwen3-32B`) are more resistant to embedding condensation. **b.** Quantifications using Spearman correlation and Kendall's Tau demonstrate a consistent trend of "larger model, less condensation" across multiple families of language models. Additional results can be found in Figure S1.

Embedding Condensation

1 Feature 1

More severe in smaller models than in larger counterparts (Figure 2).

Different model families

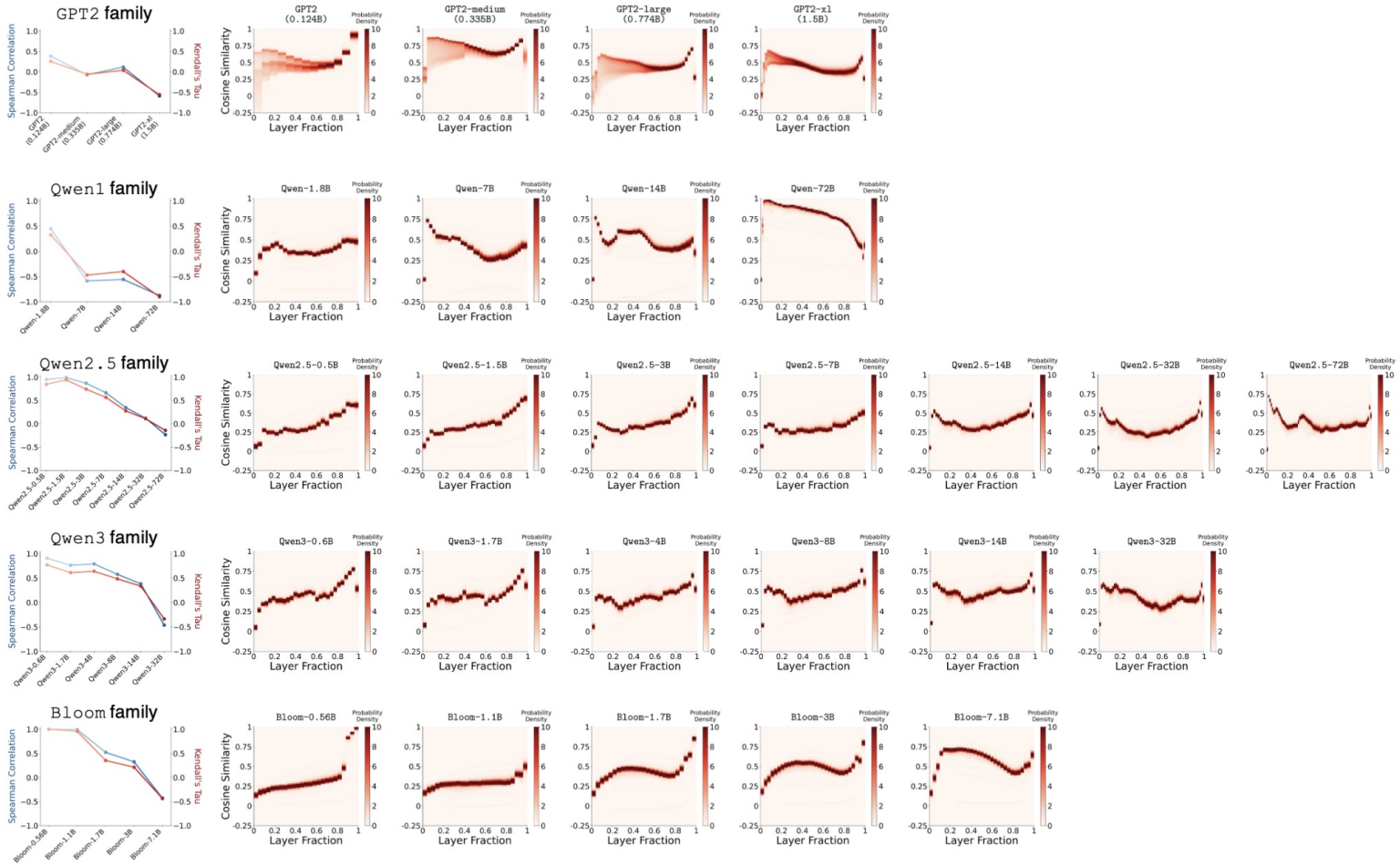


Figure S1. Additional quantitative and qualitative evaluations on `GPT2`, `Qwen1`, `Qwen2.5`, `Qwen3` and `Bloom` families all demonstrate consistent trends that **within each model family, larger models are less susceptible to the embedding condensation phenomenon**.

Embedding Condensation

1 Feature 1

More severe in smaller models than in larger counterparts (Figure 2).

Different input datasets

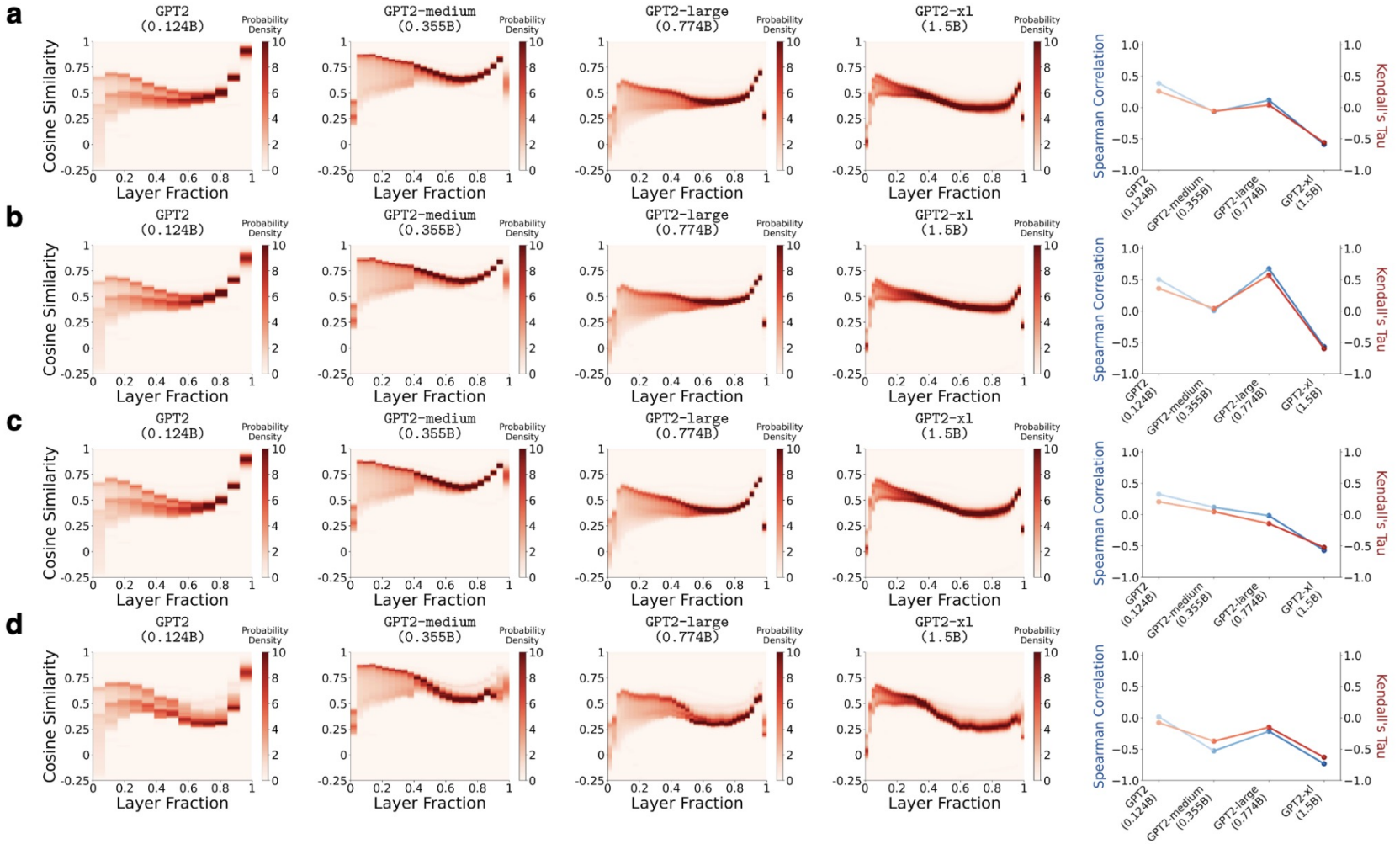


Figure S2. The embedding condensation effect is consistent regardless of the input text dataset. Results are shown for four datasets, namely **(a)** `wikitext`, **(b)** `pubmed_qa`, **(c)** `imdb`, and **(d)** `squad`.

Embedding Condensation

2 **Feature 2**

Reproducible under confounder-controlled settings (Figure 3).

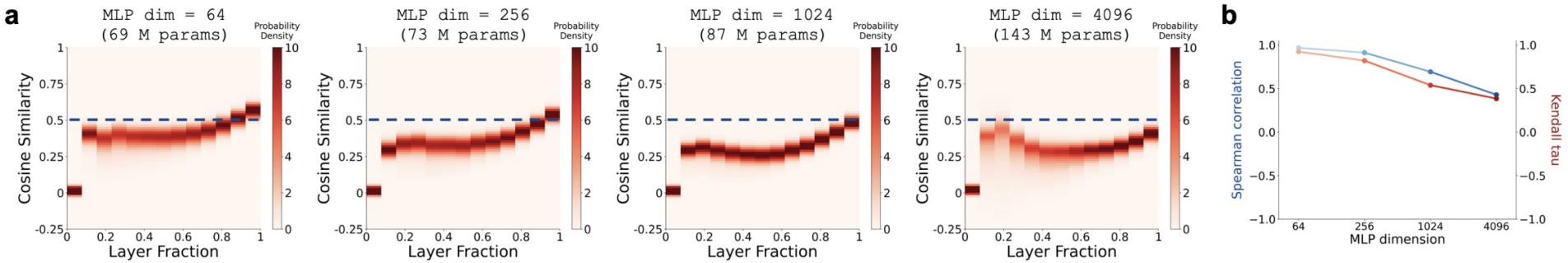


Figure 3. In a highly controlled experiment, we reproduced the observation of "larger model, less condensation". We pre-trained four `GPT2`-like models of varying sizes that differ only in MLP dimension, while keeping all other factors fixed, including the number of layers, embedding dimension, dataset, and training configuration. The resulting models exhibit consistent trends in embedding condensation, shown qualitatively (panel **a**) and quantitatively (panel **b**). Horizontal dashed lines are added to panel **a** for easier visual comparison.

Embedding Condensation

3 Feature 3

Emerging at model initialization and gets alleviated by pre-training (Figure 4).

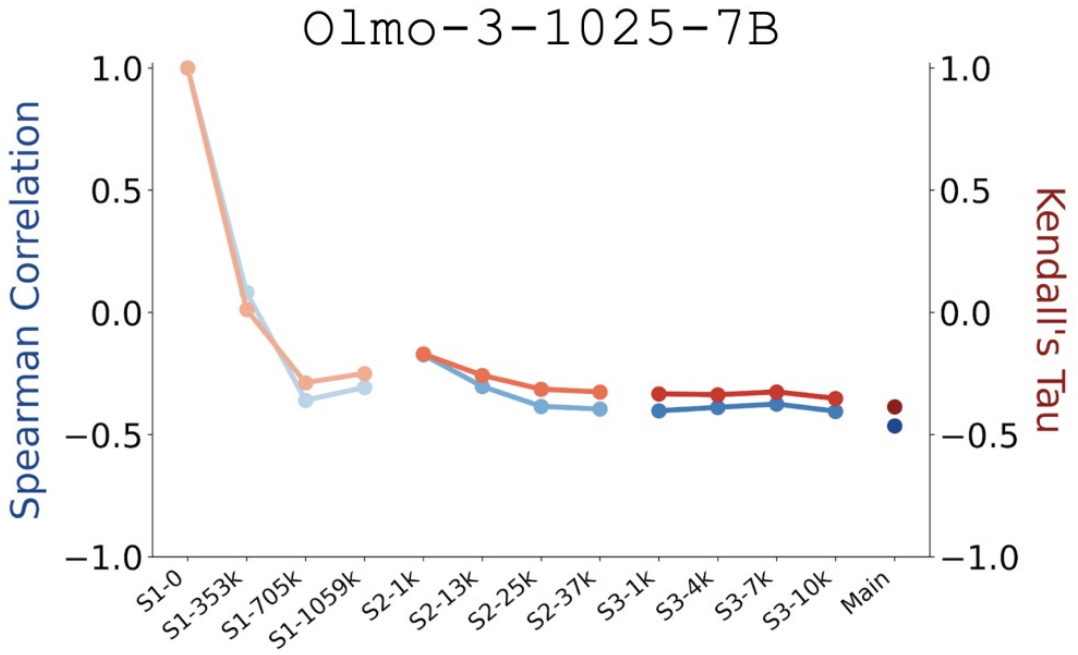


Figure 4. Embedding condensation is observed immediately after model initialization. We analyze checkpoints of `Olmo-3-1025-7B` spanning initialization, intermediate pre-training stages, and the final base model. Each checkpoint is annotated by its training stage and the number of training tokens.

Embedding Condensation

4 Feature 4

Not resolved by knowledge distillation from a larger model (Figure 5).

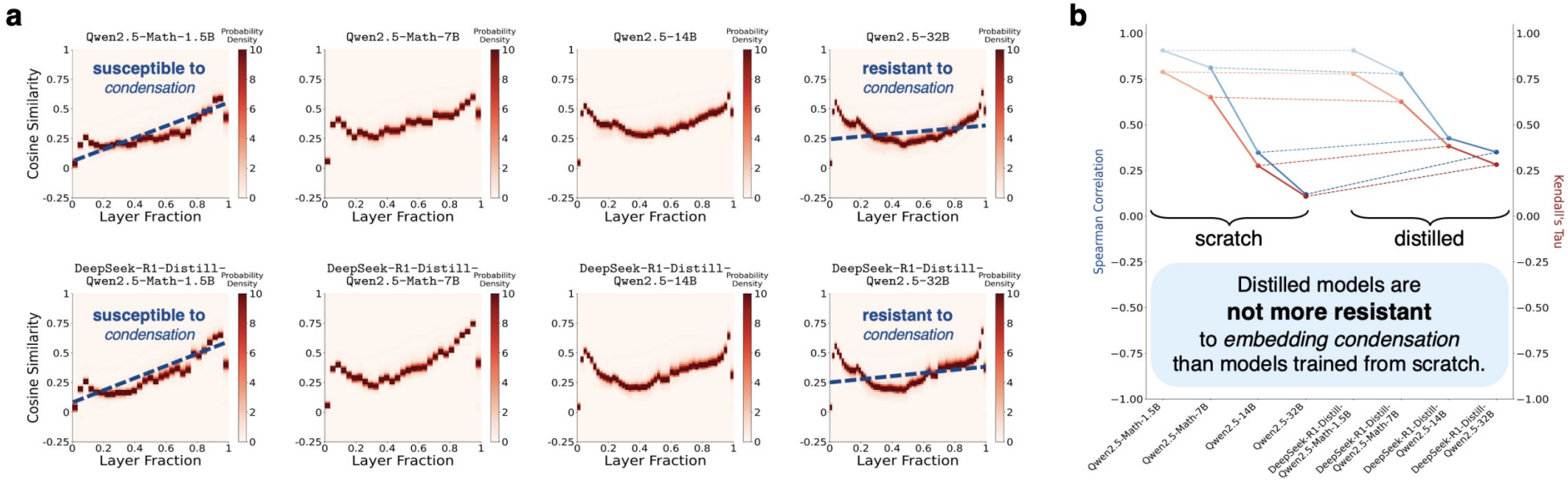


Figure 5. Knowledge distillation is not a remedy to embedding condensation, shown qualitatively (panel **a**) and quantitatively (panel **b**).

Dispersion loss

What if we proactively **disperse** the embeddings?

Will we narrow the performance gap between smaller models and larger models?

Dispersion loss

What if we proactively **disperse** the embeddings?

Will we narrow the performance gap between smaller models and larger models?

$$\mathcal{L}_{\text{disp}} = \log \sum_{i,j}^{i \neq j} e^{-\frac{\arccos\left(\text{cossim}(z_i, z_j)\right)}{\pi \tau}}$$

$$\mathcal{L} = \mathcal{L}_{\text{train}} + \lambda_{\text{disp}} \cdot \mathcal{L}_{\text{disp}}$$

$\mathcal{L}_{\text{train}}$ denotes the standard training loss, which defaults to the cross-entropy loss for next-token prediction in most language models.

Dispersion loss

Table 1. Our dispersion loss and its alternative formulations. Main implementation differences from (Wang & He, 2025) are highlighted in teal and magenta. Including or excluding diagonal terms yields identical gradients and is therefore cosmetic. For dispersion loss and ℓ_2-repel, 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, we include a norm regularization term to prevent unbounded expansion of embeddings. For Orthogonalization, 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 generative modeling in diffusion-based models (Wang & He, 2025)		**For improving generalization of Transformer-based language models (Ours)**	
	formulation	term definition	formulation	term definition
Dispersion loss	$\log \mathbb{E}_{i,j}[\exp(-D(z_i, z_j)/\tau)]$	$D(z_i, z_j) = -\texttt{cossim}(z_i, z_j)$	$\log \sum_{i,j}^{i \neq j}[\exp(-D(z_i, z_j)/\tau)]$	$D(z_i, z_j) = \frac{\arccos(\texttt{cossim}(z_i, z_j))}{\pi}$

Dispersion loss

Table 1. Our dispersion loss and its alternative formulations. Main implementation differences from (Wang & He, 2025) are highlighted in teal and magenta. Including or excluding diagonal terms yields identical gradients and is therefore cosmetic. For dispersion loss and ℓ_2-repel, 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, we include a norm regularization term to prevent unbounded expansion of embeddings. For Orthogonalization, 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 generative modeling in diffusion-based models (Wang & He, 2025)		**For improving generalization of Transformer-based language models (Ours)**	
	formulation	term definition	formulation	term definition
Dispersion loss	$\log \mathbb{E}_{i,j}[\exp(-D(z_i, z_j)/\tau)]$	$D(z_i, z_j) = -\texttt{cossim}(z_i, z_j)$	$\log \sum_{i,j}^{i\neq j}[\exp(-D(z_i, z_j)/\tau)]$	$D(z_i, z_j) = \frac{\arccos(\texttt{cossim}(z_i, z_j))}{\pi}$
Alternative formulations				
Decorrelation	$\sum_{m,n} \text{Cov}^2_{mn}$	$\text{Cov}^2 = \frac{\mathcal{Z}_c^\top \mathcal{Z}_c}{d-1}$, $\mathcal{Z}_c = \frac{\mathcal{Z}-\mu_d(\mathcal{Z})}{\sigma_d(\mathcal{Z})}$	$\sum_{m,n}^{m\neq n} \text{Cov}^2_{mn}$	$\text{Cov}^2 = \frac{\mathcal{Z}_c^\top \mathcal{Z}_c}{d-1}$, $\mathcal{Z}_c = \frac{\mathcal{Z}-\mu_d(\mathcal{Z})}{\sigma_d(\mathcal{Z})}$
ℓ_2-repel	$\log \mathbb{E}_{i,j}[\exp(-D(z_i, z_j)/\tau)]$	$D(z_i, z_j) = \|\mathcal{Z}_{i,:} - \mathcal{Z}_{:,j}\|_2^2$	$\log \sum_{i,j}^{i\neq j}[\exp(-D(z_i, z_j)/\tau)] + \lambda_{\text{norm}}\|\mathcal{Z}\|_2^2$	$D(z_i, z_j) = \|\mathcal{Z}_{i,:} - \mathcal{Z}_{:,j}\|_2^2$
Orthogonalization	$\mathbb{E}_{i,j}[\texttt{max}(0, \epsilon - D(z_i, z_j))^2]$	$D(z_i, z_j) = -\texttt{cossim}(z_i, z_j)$	$\mathbb{E}_{i,j}^{i\neq j}[\texttt{max}(0, \frac{1}{2} - D(z_i, z_j))^2]$	$D(z_i, z_j) = \frac{\arccos(\texttt{cossim}(z_i, z_j))}{\pi}$

Dispersion loss

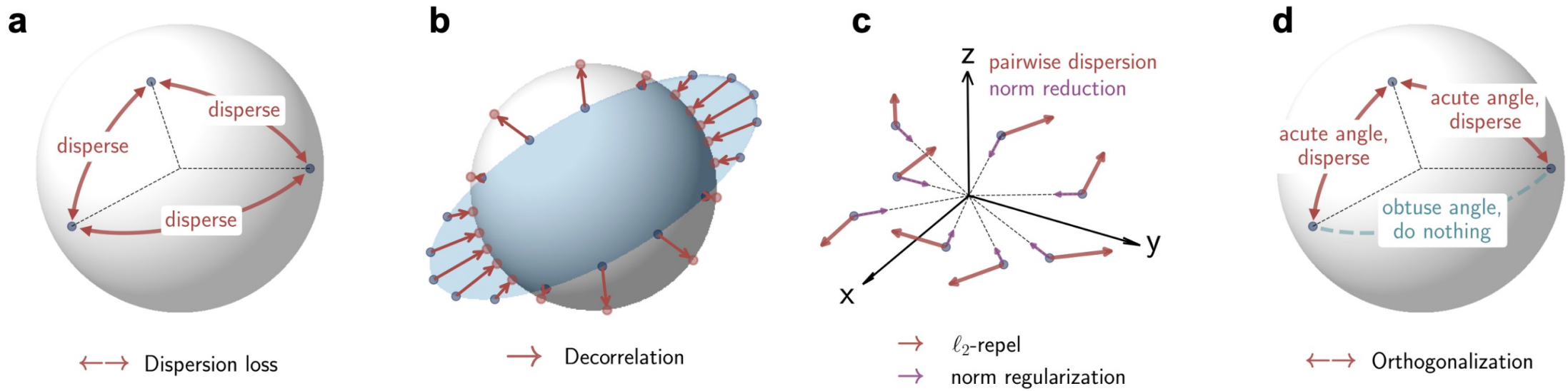


Figure 6. Illustration of how dispersion loss and its alternative formulations promote embedding dispersion. **a.** Dispersion loss enforces uniform angular dispersion by spreading out all pairs along the unit hypersphere. **b.** Decorrelation loss encourages different feature dimensions to remain uncorrelated. **c.** ℓ_2-repel loss increases pairwise Euclidean distance, while the norm regularization prevents unbounded expansion. **d.** Orthogonalization loss spreads out vectors forming acute angles while leaving obtuse ones unchanged.

Empirical results

Empirical results

Dispersion loss **counteracts** the embedding condensation phenomenon

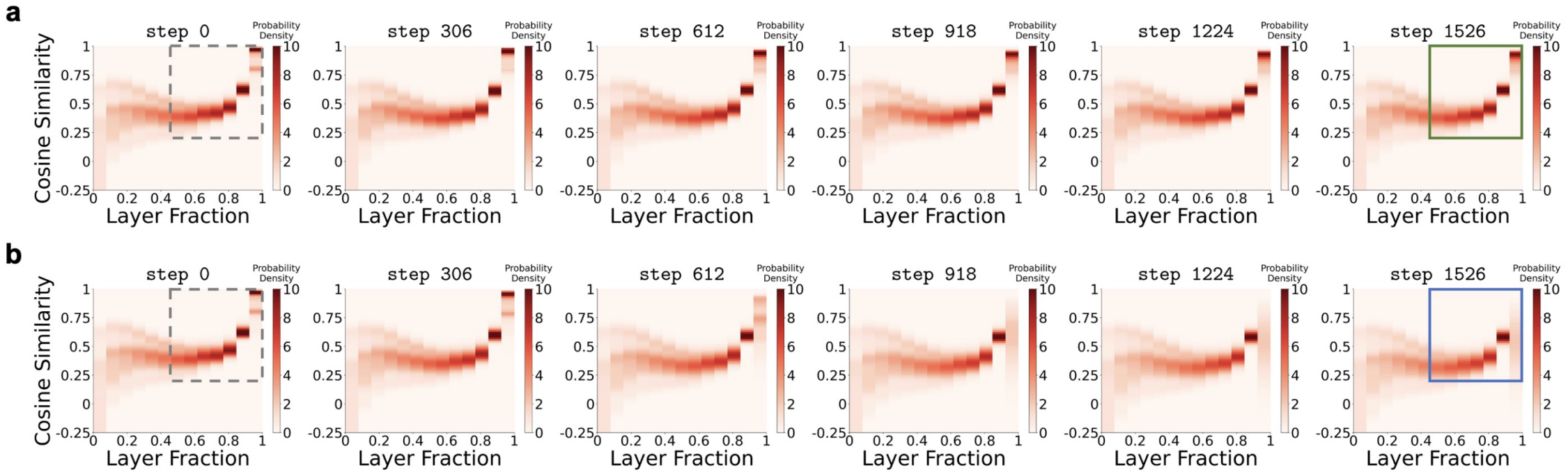


Figure 7. Dispersion loss counteracts the embedding condensation phenomenon. **a.** Starting from condensed embeddings (gray dashed box), mid-training with the default loss has a limited impact (green box). **b.** In contrast, mid-training with our dispersion loss as a regularizer substantially mitigates embedding condensation (blue box).

Empirical results

Dispersion loss is effective in **mid-training**

Table 2. Using dispersion loss during mid-training improves performance on language tasks. For each base model, the best single-benchmark metrics are displayed in bold, whereas the best average ranks and best average performances are boxed and in bold. We also perform the Student's t-tests on the average performances and report the significance level with respect to "$\mathcal{L}_{\text{train}}$ + Dispersion loss".

Model	Mid-training		Training time	Zero-shot						Few-shot				Rank↓	Average↑	t-test
	Train	Loss	A100 hours	ANLI_{R2}↑	$\text{LAMBADA}_{\text{openai}}$↑	OpenbookQA↑	PIQA↑	TruthfulQA↑	WinoGrande↑	ARC_{easy}↑	$\text{ARC}_{\text{challenge}}$↑	MedMCQA↑	MMLU↑			
GPT2	✗	—	—	34.4	30.8	15.6	61.6	40.3	52.4	42.0	16.2	25.4	24.8	6.1	34.35	$p < 0.0001$
	✓	$\mathcal{L}_{\text{train}}$	1.122 (1.00×)	34.0 ± 0.3	32.4 ± 0.6	16.4 ± 0.1	62.2 ± 0.7	43.6 ± 0.2	52.8 ± 3.1	41.6 ± 0.0	17.4 ± 0.6	24.2 ± 3.8	24.8 ± 0.4	6.2	34.95 ± 0.11	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + noisy embedding	1.122 (1.00×)	34.3 ± 0.1	33.0 ± 0.3	16.9 ± 0.4	61.1 ± 0.7	43.9 ± 0.2	53.6 ± 2.0	43.7 ± 0.4	18.0 ± 0.6	21.6 ± 2.5	25.4 ± 0.4	4.3	35.15 ± 0.06	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + active forgetting	1.127 (1.00×)	**35.0** ± 0.3	33.8 ± 1.1	16.6 ± 0.3	61.0 ± 0.0	44.0 ± 0.1	52.2 ± 0.8	**44.3** ± 0.4	**18.6** ± 1.1	22.4 ± 1.7	**25.7** ± 0.2	**3.2**	35.36 ± 0.15	*n.s.*
	✓	$\mathcal{L}_{\text{train}}$ + Decorrelation	1.221 (1.09×)	**35.0** ± 0.0	33.6 ± 0.5	**17.0** ± 0.6	60.8 ± 0.6	43.9 ± 0.3	52.0 ± 1.7	43.8 ± 0.5	18.0 ± 1.3	24.4 ± 2.3	25.6 ± 0.2	3.6	35.41 ± 0.06	*n.s.*
	✓	$\mathcal{L}_{\text{train}}$ + ℓ_2-repel	1.175 (1.05×)	34.6 ± 0.3	32.8 ± 0.2	16.6 ± 0.2	60.8 ± 0.3	43.9 ± 0.2	**54.2** ± 0.8	43.4 ± 0.2	17.6 ± 0.8	24.6 ± 2.0	**25.7** ± 0.1	4.1	35.42 ± 0.11	*n.s.*
	✓	$\mathcal{L}_{\text{train}}$ + Orthogonalization	1.210 (1.08×)	34.4 ± 0.1	31.2 ± 1.1	16.8 ± 0.7	61.6 ± 0.0	**44.2** ± 0.4	53.4 ± 0.8	44.2 ± 1.0	18.0 ± 0.3	24.8 ± 3.3	25.6 ± 0.2	**3.2**	35.42 ± 0.19	*n.s.*
	✓	$\mathcal{L}_{\text{train}}$ + Dispersion loss	1.176 (1.05×)	34.8 ± 0.0	**34.0** ± 0.4	16.6 ± 0.0	**62.4** ± 0.3	**44.2** ± 0.1	51.6 ± 0.1	42.6 ± 0.4	18.0 ± 0.8	**26.6** ± 2.3	25.4 ± 0.2	**3.2**	**35.61** ± 0.12	*ref.*
GPT2-m	✗	—	—	33.3	40.4	18.6	66.3	40.1	54.8	50.2	19.9	**29.1**	25.3	5.4	37.80	$p < 0.0001$
	✓	$\mathcal{L}_{\text{train}}$	2.541 (1.00×)	33.1 ± 0.6	43.2 ± 0.3	19.1 ± 0.0	67.7 ± 0.4	40.5 ± 1.3	55.5 ± 1.0	**54.2** ± 2.7	18.9 ± 0.7	28.2 ± 2.3	25.1 ± 0.3	4.5	38.55 ± 0.18	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + noisy embedding	2.580 (1.01×)	33.3 ± 0.1	44.0 ± 0.3	18.6 ± 0.6	66.7 ± 0.1	**44.5** ± 0.6	52.1 ± 1.6	51.7 ± 1.6	19.5 ± 1.8	25.5 ± 0.1	25.8 ± 0.3	4.8	38.16 ± 0.25	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + active forgetting	2.580 (1.01×)	33.4 ± 0.3	43.0 ± 0.6	18.9 ± 0.1	66.7 ± 0.1	44.3 ± 0.1	50.8 ± 1.4	51.8 ± 2.5	19.8 ± 1.4	26.5 ± 0.1	**26.1** ± 0.0	4.4	38.13 ± 0.29	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + Decorrelation	2.760 (1.09×)	33.4 ± 0.0	**45.4** ± 1.3	18.9 ± 0.8	66.6 ± 0.1	42.2 ± 0.1	56.2 ± 0.4	53.8 ± 3.3	18.0 ± 0.8	28.2 ± 3.0	25.4 ± 0.2	4.0	38.81 ± 0.12	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + ℓ_2-repel	2.675 (1.05×)	**33.6** ± 0.8	44.2 ± 0.8	18.5 ± 0.6	66.2 ± 0.1	42.4 ± 0.1	54.4 ± 1.1	54.0 ± 2.0	18.6 ± 0.6	28.9 ± 2.1	25.3 ± 0.4	4.7	38.61 ± 0.03	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + Orthogonalization	2.692 (1.06×)	33.0 ± 0.4	45.2 ± 0.4	18.6 ± 0.6	**68.1** ± 0.4	41.4 ± 0.2	55.1 ± 0.6	53.6 ± 2.7	18.4 ± 0.4	29.0 ± 0.4	25.0 ± 0.1	4.8	38.74 ± 0.30	$p < 0.05$
	✓	$\mathcal{L}_{\text{train}}$ + Dispersion loss	2.673 (1.05×)	**33.6** ± 0.1	45.2 ± 0.8	**19.2** ± 0.2	67.5 ± 0.1	43.4 ± 0.2	**56.4** ± 1.1	53.8 ± 1.1	**20.1** ± 0.9	28.6 ± 0.1	25.7 ± 0.3	**2.2**	**39.35** ± 0.15	*ref.*
GPT2-l	✗	—	—	33.6	47.8	19.6	71.6	38.9	58.4	54.0	22.4	26.2	25.6	—	39.81	—
GPT2-xl	✗	—	—	36.4	49.4	22.2	71.8	38.0	57.2	58.0	23.6	27.0	25.2	—	40.89	—
Qwen3-0.6B	✗	—	—	35.0	43.0	19.5	66.5	40.7	60.5	67.5	32.5	26.5	49.4	5.7	44.11	$p < 0.0001$
	✓	$\mathcal{L}_{\text{train}}$	4.676 (1.00×)	32.5 ± 0.3	**52.0** ± 0.3	21.5 ± 0.5	**67.5** ± 1.4	44.3 ± 0.9	**61.0** ± 0.8	68.0 ± 2.8	33.0 ± 1.3	29.5 ± 0.9	**50.0** ± 0.1	4.1	45.93 ± 0.37	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + noisy embedding	4.839 (1.03×)	34.0 ± 0.7	48.8 ± 0.4	20.5 ± 0.7	66.0 ± 0.7	49.4 ± 0.1	57.5 ± 0.7	67.2 ± 1.1	34.5 ± 0.7	35.0 ± 1.4	48.8 ± 0.0	5.3	46.17 ± 0.28	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + active forgetting	4.829 (1.03×)	35.0 ± 0.7	49.0 ± 4.2	20.0 ± 2.1	67.2 ± 0.4	48.6 ± 0.9	58.8 ± 1.8	69.0 ± 1.4	34.2 ± 1.1	36.8 ± 0.4	49.3 ± 0.0	4.0	46.79 ± 0.27	$p < 0.05$
	✓	$\mathcal{L}_{\text{train}}$ + Decorrelation	4.939 (1.06×)	35.0 ± 0.0	50.5 ± 0.4	19.5 ± 1.8	**67.5** ± 1.1	47.0 ± 2.3	59.5 ± 1.8	68.5 ± 3.2	**35.0** ± 1.8	34.0 ± 0.4	49.8 ± 0.2	3.5	46.62 ± 0.25	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + ℓ_2-repel	4.864 (1.04×)	33.5 ± 0.0	46.0 ± 2.1	19.0 ± 0.7	66.0 ± 0.4	47.4 ± 0.2	59.5 ± 1.4	69.0 ± 3.2	**35.0** ± 1.8	**40.0** ± 0.4	46.9 ± 0.2	4.8	46.23 ± 0.04	$p < 0.001$
	✓	$\mathcal{L}_{\text{train}}$ + Orthogonalization	4.936 (1.06×)	**35.5** ± 1.1	50.0 ± 0.7	22.0 ± 0.7	65.5 ± 0.4	49.1 ± 0.1	56.5 ± 2.1	71.5 ± 3.2	33.0 ± 1.1	36.0 ± 0.7	48.9 ± 0.2	4.2	46.80 ± 0.20	$p < 0.05$
	✓	$\mathcal{L}_{\text{train}}$ + Dispersion loss	4.878 (1.04×)	**35.5** ± 0.8	49.5 ± 0.8	**22.5** ± 0.6	65.0 ± 1.6	**49.8** ± 1.1	58.5 ± 1.6	**72.5** ± 2.3	34.5 ± 0.8	37.5 ± 1.3	49.2 ± 0.2	**3.2**	**47.45** ± 0.16	*ref.*
Qwen3-1.7B	✗	—	—	39.5	53.0	29.0	71.5	47.6	60.5	72.0	50.0	45.5	63.1	5.1	53.18	$p < 0.0001$
	✓	$\mathcal{L}_{\text{train}}$	9.148 (1.00×)	40.0 ± 0.9	60.7 ± 0.3	28.5 ± 0.9	74.3 ± 0.6	49.1 ± 0.2	61.3 ± 0.3	71.3 ± 1.4	47.7 ± 1.2	49.7 ± 1.2	63.1 ± 0.0	4.3	54.57 ± 0.17	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + noisy embedding	9.345 (1.02×)	35.0 ± 0.0	60.5 ± 2.8	28.0 ± 0.0	70.5 ± 0.7	**50.3** ± 0.9	60.8 ± 3.2	73.2 ± 0.4	48.5 ± 0.0	48.5 ± 2.1	62.5 ± 0.6	5.0	53.78 ± 0.07	$p < 0.0001$
	✓	$\mathcal{L}_{\text{train}}$ + active forgetting	9.298 (1.02×)	35.2 ± 0.4	59.5 ± 1.4	27.5 ± 0.7	72.5 ± 0.0	49.9 ± 0.2	60.5 ± 1.4	**74.2** ± 0.4	47.2 ± 1.1	49.0 ± 1.4	63.2 ± 0.2	5.0	53.89 ± 0.07	$p < 0.001$
	✓	$\mathcal{L}_{\text{train}}$ + Decorrelation	9.535 (1.04×)	39.5 ± 1.5	60.7 ± 1.0	28.3 ± 1.2	74.8 ± 1.0	49.7 ± 0.5	61.5 ± 0.9	72.7 ± 1.5	49.7 ± 0.8	49.7 ± 1.0	63.5 ± 0.2	3.1	55.01 ± 0.13	$p < 0.01$
	✓	$\mathcal{L}_{\text{train}}$ + ℓ_2-repel	9.440 (1.03×)	33.5 ± 0.0	47.8 ± 0.6	23.0 ± 0.0	67.8 ± 1.2	42.1 ± 0.2	57.5 ± 0.5	68.3 ± 1.9	41.3 ± 1.8	38.5 ± 2.6	42.3 ± 0.5	8.0	46.22 ± 0.57	$p < 0.0001$
	✓	$\mathcal{L}_{\text{train}}$ + Orthogonalization	9.565 (1.05×)	**40.8** ± 1.3	61.0 ± 0.5	**29.2** ± 0.3	74.8 ± 1.3	49.5 ± 0.2	61.3 ± 0.8	72.3 ± 1.9	48.2 ± 1.6	49.5 ± 1.3	63.7 ± 0.0	3.0	55.04 ± 0.26	$p < 0.05$
	✓	$\mathcal{L}_{\text{train}}$ + Dispersion loss	9.455 (1.03×)	40.2 ± 2.0	**62.2** ± 0.8	29.0 ± 1.0	**75.2** ± 0.8	50.2 ± 0.4	**62.7** ± 1.2	73.0 ± 1.3	**49.3** ± 0.8	**51.0** ± 1.8	**64.1** ± 0.2	**1.7**	**55.68** ± 0.21	*ref.*
Qwen3-4B	✗	—	—	41.5	60.5	27.5	75.0	52.8	67.0	80.0	58.5	55.5	73.1	—	59.14	—
Qwen3-8B	✗	—	—	49.5	67.5	33.0	77.5	54.7	71.0	86.0	63.5	56.0	78.2	—	63.68	—
Qwen3-14B	✗	—	—	54.0	65.5	35.5	77.0	54.7	74.5	86.0	68.5	62.0	81.7	—	65.93	—

Empirical results

Dispersion loss is effective in **mid-training**

Table 3. Dispersion loss is more effective when applied to deeper layers, where embedding condensation is more pronounced. Experiments are performed under the GPT2 mid-training setting.

	Zero-shot						Few-shot				Average↑	t-test
Dispersion location	ANLI_{R2}↑	$\text{LAMBADA}_{\text{openai}}$↑	OpenbookQA↑	PIQA↑	TruthfulQA↑	WinoGrande↑	ARC_{easy}↑	$\text{ARC}_{\text{challenge}}$↑	MedMCQA↑	MMLU↑		
Layers $\in [1, \frac{L}{2}]$	34.6 ± 0.3	33.0 ± 0.4	16.7 ± 0.6	60.9 ± 0.4	43.3 ± 0.5	**52.4** ± 0.3	43.7 ± 0.3	**17.9** ± 1.3	21.9 ± 1.0	25.3 ± 0.6	34.96 ± 0.20	$p < 0.05$
Layers $\in [\frac{L}{2}, L]$	**34.9** ± 0.4	**33.3** ± 0.6	**16.9** ± 0.1	**61.3** ± 0.8	**43.9** ± 0.1	52.2 ± 1.1	**43.8** ± 1.2	17.8 ± 0.9	**23.7** ± 2.4	**25.8** ± 0.1	**35.37** ± 0.06	*ref.*

Empirical results

Dispersion loss is effective in **mid-training**

Table 3. Dispersion loss is more effective when applied to deeper layers, where embedding condensation is more pronounced. Experiments are performed under the GPT2 mid-training setting.

	Zero-shot						Few-shot				Average↑	t-test
Dispersion location	ANLI_{R2}↑	$\text{LAMBADA}_{\text{openai}}$↑	OpenbookQA↑	PIQA↑	TruthfulQA↑	WinoGrande↑	ARC_{easy}↑	$\text{ARC}_{\text{challenge}}$↑	MedMCQA↑	MMLU↑		
Layers $\in [1, \frac{L}{2}]$	34.6 ± 0.3	33.0 ± 0.4	16.7 ± 0.6	60.9 ± 0.4	43.3 ± 0.5	**52.4** ± 0.3	43.7 ± 0.3	**17.9** ± 1.3	21.9 ± 1.0	25.3 ± 0.6	34.96 ± 0.20	$p < 0.05$
Layers $\in [\frac{L}{2}, L]$	**34.9** ± 0.4	**33.3** ± 0.6	**16.9** ± 0.1	**61.3** ± 0.8	**43.9** ± 0.1	52.2 ± 1.1	**43.8** ± 1.2	17.8 ± 0.9	**23.7** ± 2.4	**25.8** ± 0.1	**35.37** ± 0.06	*ref.*

Table 4. Effect of hyperparameters on the dispersion loss. Ablation experiments are performed under the GPT2 mid-training setting.

Loss	Coefficient λ_{disp}	Temperature τ	**Average** (same 10 tasks) ↑
$\mathcal{L}_{\text{train}}$ + Dispersion loss	0.1	1.0	**35.61**
	0.01	1.0	35.36
	0.5	1.0	35.37
	1.0	1.0	35.29
	0.1	0.1	35.33
	0.1	0.5	35.42
	0.1	2.0	35.27

Empirical results

Dispersion loss is effective in **pre-training**

Table 5. Using dispersion loss during pre-training improves performance on language tasks. Experiments are performed under the `Qwen3-0.6B` pre-training setting.

Loss	Zero-shot						Few-shot				Average↑
	ANLI_{R2}↑	$\text{LAMBADA}_{\text{openai}}$↑	OpenbookQA↑	PIQA↑	TruthfulQA↑	WinoGrande↑	ARC_{easy}↑	$\text{ARC}_{\text{challenge}}$↑	MedMCQA↑	MMLU↑	
$\mathcal{L}_{\text{train}}$	**34.0**	24.0	**15.5**	64.5	37.8	**58.0**	**41.5**	22.0	26.5	24.6	34.84
$\mathcal{L}_{\text{train}}$ + Dispersion loss	32.0	**27.5**	13.5	**68.5**	**45.2**	54.5	40.0	**24.5**	**29.5**	**24.9**	**36.01** (+1.17)

Conclusion

Conclusion

What makes LLMs better than small LMs?
Data? Parameters? Geometry might play a role!

Larger language models are better than smaller language models, but might **not merely because they have more parameters**.

It can be partially attributed to
how they organize the information in the latent representations.

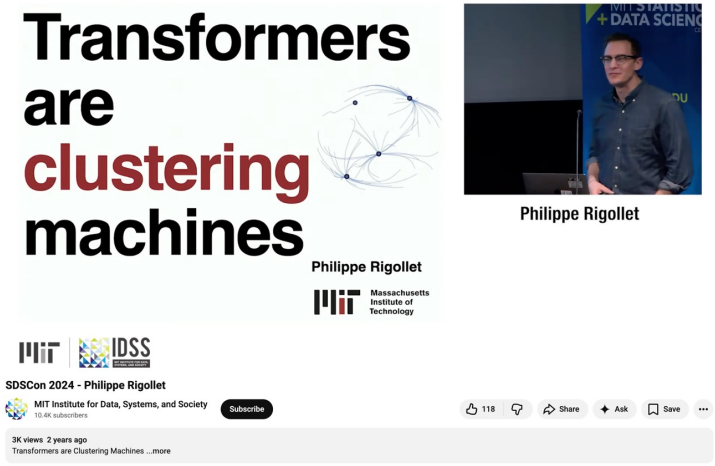
We hope to see future efforts along this interesting direction.

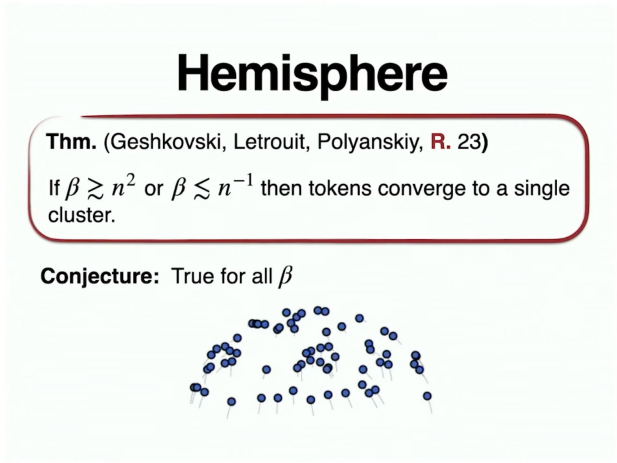
Acknowledgements

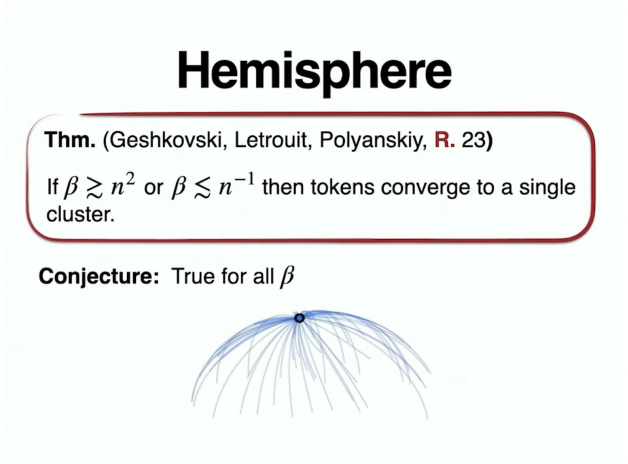
Acknowledgements

1. This work was initially motivated by the paper "A mathematical perspective on Transformers". We started this project in early Apr 2025 after watching a talk on that paper. We were intrigued by a theoretical result in that paper stating that if we stack Transformer layers infinitely, all the token embeddings will cluster to the same point, and we were curious whether that behavior can be observed empirically. That led to the key observations of embedding condensation in our paper.

https://www.youtube.com/watch?v=3McmEtA3t_0







Acknowledgements

2. The design of the dispersion loss was largely inspired by Runqian and Kaiming's paper "Diffuse and Disperse: Image Generation with Representation Regularization". Their paper came out right after when we completed the initial observations of embedding condensation and was thinking about ways to mitigate the phenomenon. When I read that paper, I immediately felt it was highly relevant and could be used as a reasonable solution.

https://arxiv.org/pdf/2506.09027

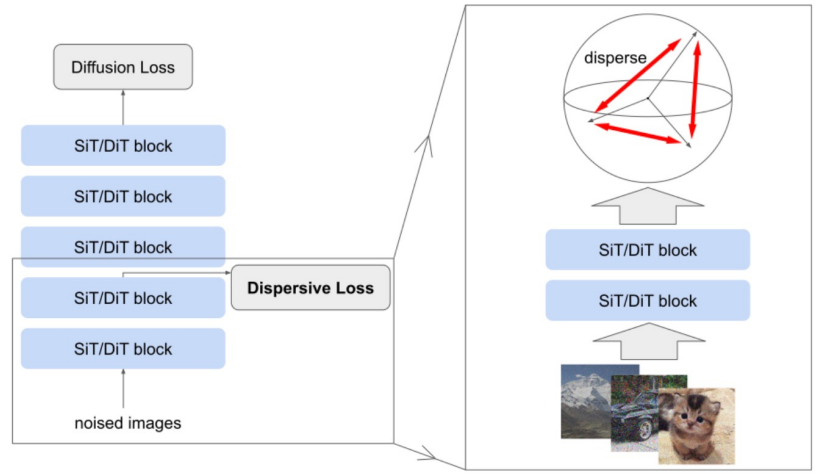


Table 1: **Variants of Dispersive Loss functions**, compared against their contrastive counterparts. Each variant of Dispersive Loss can be viewed as the contrastive loss without positive pairs.

variant	contrastive	dispersive
InfoNCE	$D(z_i,z_i^+)/\tau+\log\sum_j\exp(-D(z_i,z_j)/\tau)$	$\log\mathbb{E}_{i,j}[\exp(-D(z_i,z_j)/\tau)]$
Hinge	$D(z_i,z_i^+)^2+\mathbb{E}_j[\max(0,\epsilon-D(z_i,z_j))^2]$	$\mathbb{E}_{i,j}[\max(0,\epsilon-D(z_i,z_j))^2]$
Covariance	$(1-\mathrm{Cov}_{mm})^2+w\sum_{n\neq m}\mathrm{Cov}_{mn}^2$	$\sum_{m,n}\mathrm{Cov}_{mn}^2$

variant	baseline	contrastive (independent noise)	contrastive (restricted noise)	dispersive
none	36.49	–	–	–
InfoNCE, ℓ_2	–	43.66 (+19.65%)	36.57 (+0.22%)	**32.35** (–11.35%)
InfoNCE, cosine	–	41.62 (+14.06%)	34.83 (–4.55%)	34.33 (–5.92%)
Hinge	–	43.02 (+17.89%)	35.14 (–3.70%)	33.93 (–7.02%)
Covariance	–	37.85 (+3.73%)	35.87 (–1.70%)	35.82 (–1.84%)

Table 2: **Variants of Dispersive Loss**, compared against their contrastive counterparts. We report FID-50k of SiT-B/2 on ImageNet, trained for 80 epochs, without CFG. The "baseline" entry refers to original SiT-B/2 without any contrastive or dispersive loss. The contrastive loss involves two views per training sample, with "independent" or "restricted" noise (the latter case restricts the noise level to differ by up to 0.005). Values in brackets denote the relative improvement or degradation with respect to the baseline.

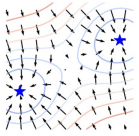
Runqian (Ray) Wang

I am an undergraduate student at MIT double majoring in AI and Math. I am currently advised by Prof. Kaiming He at MIT and Prof. Yilun Du at Harvard. Previously, I was advised by Dr. Zhirong Wu, Dr. Rogerio Feris, and Prof. Polina Golland. I also interned as a student researcher at Microsoft Research Asia and MIT-IBM Watson AI Lab

Email / CV / Github / Google Scholar

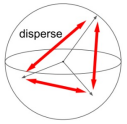
Research

My current focus is on generative modeling. I used to work on computer vision, optimization, LLM finetuning, and medical imaging. I am most interested in simple yet effective methods that can generalize and scale.



Equilibrium Matching: Generative Modeling with Implicit Energy-Based Models
Runqian Wang, Yilun Du
Preprint, 2025
website / code / paper

New generative model with equilibrium dynamics that exceeds Flow Matching in performance, supports optimization-based sampling, and naturally performs multiple downstream tasks.



Diffuse and Disperse: Image Generation with Representation Regularization
Runqian Wang, Kaiming He
Preprint, 2025
code / paper

Plug-and-play representation regularizer for generative modeling that brings consistent improvement across different settings.

Kaiming He 何恺明

Associate Professor, EECS, MIT

Distinguished Scientist, Google DeepMind

kaiming@mit.edu

I am an Associate Professor with tenure in the Department of EECS at MIT. I also work part-time as a Distinguished Scientist at Google DeepMind. My research areas include computer vision and deep learning.

I am best known for my work on Deep Residual Networks (ResNets), which is the most-cited paper of the twenty-first century, according to a *Nature* article. The proposed residual connections are now being used everywhere in modern deep learning models, including Transformers (e.g., ChatGPT), AlphaGo Zero, AlphaFold, and virtually all GenAI models today, among many others. I am also known for my work on visual perception (e.g., Faster R-CNN, Mask R-CNN) and self-supervised learning (e.g., MoCo, MAE). My publications have over 700,000 citations (as of May 2025).

I am a recipient of several prestigious awards, including the 10-year Test of Time Award in NeurIPS 2025, ICCV 2025, the PAMI Young Researcher Award in 2018, the Best Paper Award in ICCV 2017, CVPR 2016, CVPR 2009, the Best Student Paper Award in ICCV 2017, the Best Paper Honorable Mention in CVPR 2021, ECCV 2018, and the Everingham Prize in ICCV 2021.

Before joining MIT in 2024, I was a research scientist at Facebook AI Research (FAIR) from 2016 to 2024, and a researcher at Microsoft Research Asia (MSRA) from 2011 to 2016. I received my PhD degree from the Chinese University of Hong Kong in 2011, and my B.S. degree from Tsinghua University in 2007.

Future Directions

- **Better regularizers.** The dispersion loss is a very simple and straightforward solution, likely with benefits and drawbacks. A more carefully designed method to counteract embedding condensation might be more helpful.
- **Beyond pre-training.** Track how embedding condensation evolves in later stages of training, such as supervised fine-tuning (SFT) and reinforcement learning (RL). It remains unclear whether condensation re-emerges, stabilizes, or interacts differently with alignment objectives.
- **Mechanism and causality.** Pin down the root causes of embedding condensation and establish stronger causal links between condensation and downstream behavior such as generalization.
- **Better architectures.** Design model famlies or modules that inherently resist condensation, complementing or replacing purely loss-based regularization.
- **Better initialization.** Develop initialization schemes that start models in a less condensed regime, potentially reducing the burden on the training objective to counteract the geometric collapse.

Thank you!