Weight Tying Biases Token Embeddings Towards the Output Space

Antonio Lopardo[1], Avyukth Harish[2], Catherine Arnett[1], Akshat Gupta[2]
[1]EleutherAI, [2]University of California Berkeley
Correspondence: antonio.lopardo@outlook.com

Abstract

Weight tying, i.e. sharing parameters between input and output embedding matrices, is common practice in language model design, yet its impact on the learned embedding space remains poorly understood. In this paper, we show that tied embedding matrices align more closely with output (unembedding) matrices than with input embeddings of comparable untied models, indicating that the shared matrix is shaped primarily for output prediction rather than input representation. This unembedding bias arises because output gradients dominate early in training. Using tuned lens analysis, we show this negatively affects early-layer computations, which contribute less effectively to the residual stream. Scaling input gradients during training reduces this bias, providing causal evidence for the role of gradient imbalance. This is mechanistic evidence that weight tying optimizes the embedding matrix for output prediction, compromising its role in input representation. These results help explain why weight tying can harm performance at scale and have implications for training smaller LLMs, where the embedding matrix contributes substantially to total parameter count.

1 Introduction

Weight tying has become standard practice in language model design since Press and Wolf (2017) showed that sharing parameters between the input embedding matrix and the output projection matrix improves performance while reducing parameter count in recurrent models. This practice is often motivated by parameter efficiency, implicit regularization, and the intuitive symmetry between reading and predicting tokens.

However, more recent work has challenged the applicability of these findings to modern architectures. Chung et al. (2020) showed that weight tying can be harmful to performance in deeper transformer-based language models (Vaswani et al., 2017), where the benefits observed in shallow networks no longer hold.

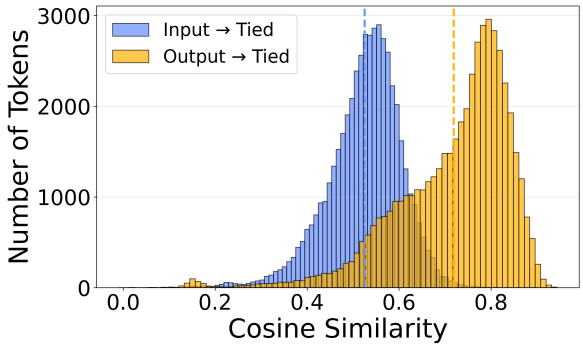

Figure 1: Per-token cosine similarity (after linear alignment) between the tied embedding matrix and the untied input (blue) and output (orange) matrices for two OLMo-1B training runs (tied and untied). Dashed lines indicate means. The tied matrix is substantially more aligned with the untied output (mean: 0.719) than the untied input (mean: 0.525).

This finding is reflected in contemporary model design choices: the Qwen3 family (Qwen Team, 2025) explicitly ties embeddings for smaller models but unties them for larger ones, suggesting that the costs of weight tying outweigh its benefits as models grow. Similarly, OLMo 2 and 3 (Team OLMo et al., 2024; Team Olmo et al., 2025) explicitly untie embeddings for all model sizes. Beyond these architectural trends, Machina and Mercer (2024) found evidence that Pythia models, which use untied embeddings, exhibit less anisotropy in their representation spaces compared to tied models, suggesting that untying may yield more uniform representations, a property shown to improve downstream performance (Gao et al., 2019; Ethayarajh, 2019; Biś et al., 2021; Rajaee and Pilehvar, 2022; Wang et al., 2019, 2020; Stollenwerk and Stollenwerk, 2025).

These observations collectively suggest untying is beneficial at scale. However, the underlying

Findings of the Association for Computational Linguistics: ACL 2026, pages 40795–40810

mechanism behind why this is the case remains poorly understood. In this paper, we provide an explanation to this exact question, grounded in training dynamics. We show that in weight-tied models, the shared matrix is dominated early in training by gradients from the output layer and thus receives stronger learning signals from the next-token prediction objective. As a consequence, **the shared embedding matrix is shaped primarily for output projection, leaving early transformer layers to operate with input embeddings optimized for a different purpose** (Figure 1). We support these findings with embedding similarity analysis, analysis with the tuned lens (Belrose et al., 2023), gradient flow tracking, and gradient manipulation experiments. Together, these experiments show that the tied embedding matrix is largely shaped by its role in the output layer, due to gradient imbalances early in training.[1]

In summary:

- We show that tied embedding matrices align more closely with output (unembedding) matrices than with input embeddings for comparable untied models.
- This unembedding bias affects early-layer computations, as shown by elevated KL divergence in tuned lens analysis.
- We show that output-layer gradients dominate input-layer gradients early in training, resulting in the unembedding bias in weight-tied models.
- Finally, we show that scaling input gradients during training reduces this bias, confirming the causal role of gradient imbalance.

2 Background & Related Work

In transformer-based language models, the input embedding matrix $W_E \in \mathbb{R}^{V \times d}$ (where V is the vocabulary size and d is the hidden dimension of the model) maps discrete token indices to continuous vector representations that serve as input to the first transformer layer. These initial representations encode semantic and syntactic information in a form that subsequent layers can process (Jawahar et al., 2019). The unembedding matrix $W_U \in \mathbb{R}^{d \times V}$ serves a different function: it projects the final hidden states back to vocabulary-sized logits for next-token prediction. This matrix is tasked with transforming the refined representations from the last layer into logits over the vocabulary, where each logit reflects the likelihood of the corresponding token given the context.

The input embedding matrix should thus produce representations useful for the model's internal computations, while the output matrix must discriminate between tokens based on the final hidden state, optimizing directly for cross-entropy loss. With tied embeddings, a single set of parameters must satisfy both objectives simultaneously (Bertolotti and Cazzola, 2024). We show that in this scenario one objective dominates: that of the output.

2.1 Weight Tying in Language Models

Weight tying for embedding and unembedding matrices was introduced by Press and Wolf (2017) and Inan et al. (2017), who showed that sharing the input embedding matrix with the unembedding matrix, such that $W_U = W_E^\top$, improves perplexity while reducing parameter count in Recurrent Neural Networks (RNNs; Elman, 1990). The technique became standard practice in transformer language models, adopted by GPT-2 (Radford et al., 2019), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and Gemma (Gemma Team et al., 2024), *inter alia*.

The parameter savings from weight tying are substantial, particularly for smaller models. For example, for Pythia 70M, the embedding matrices account for over 73.4% of all parameters; even at 2.8B scale, they still represent approximately 9.2% of the model, so tying can still yield meaningful savings. See Appendix A for elaboration. However, Chung et al. (2020) demonstrated that weight tying can harm performance: decoupling input and output embeddings improves average performance on the XTREME benchmark (Hu et al., 2020; more details in Appendix F). This shows models benefit from specialized input and output embedding spaces. Currently, untying is more common for models at multi-billion parameters scale, e.g. Llama 3 (Grattafiori et al., 2024), OLMo 2 (Team OLMo et al., 2024), DeepSeek-V3 (DeepSeek-AI, 2024), and Qwen3 (Qwen Team, 2025). In this work, we aim to understand the effect of weight tying on learned representations, as this becomes a more common design choice.

[1] Code and artifacts: https://github.com/AntonioLopardo/weight-tying-bias

2.2 Embedding Space Alignment

The alignment of embedding spaces has a long history in NLP, starting with work on cross-lingual word embeddings. Mikolov et al. (2013) first showed that crosslingual embedding spaces share similar geometric structure, enabling alignment via simple linear transformations. Xing et al. (2015) demonstrated that constraining this mapping to be orthogonal significantly improves alignment quality, and Artetxe et al. (2016) further refined this approach with principled bilingual mappings that preserve monolingual structure. Conneau et al. (2020) applied these techniques to contextual representations, demonstrating that even independently trained monolingual BERT models can be aligned post-hoc with linear mappings. We adopt a similar approach, using different transformations, to measure alignment difficulty between matrices across tied and untied models.

2.3 The Logit Lens and the Tuned Lens

The logit lens, introduced by nostalgebraist (2020), projects hidden states from intermediate layers into vocabulary distributions by applying the model's unembedding matrix W_U. For a model with L total layers, the logit lens operator transforms an intermediate layer representation $h_\ell \in \mathbb{R}^d$ ($\ell < L$) as shown below:

$$\text{LogitLens}(h_\ell) = \text{LayerNorm}(h_\ell)\, W_U$$

This technique revealed that transformer predictions converge roughly monotonically across layers as the residual stream accumulates updates from each layer. However, the logit lens assumes intermediate representations are in the same basis as the final layer, which is often not the case. The tuned lens (Belrose et al., 2023) addresses this limitation by learning an affine transformation for each layer:

$$\text{TunedLens}_\ell(h_\ell) = \text{LogitLens}(A_\ell h_\ell + b_\ell)$$

where $A_\ell \in \mathbb{R}^{d\times d}$ is a learned linear map and $b_\ell \in \mathbb{R}^d$ is a bias term. The translators compensate for cross-layer representational differences. The difficulty of learning these translators, measured by the residual KL divergence between the tuned lens prediction and the model's final output distribution, indicates how compatible each layer's representations are with the final layer. In our experiments, therefore, we use the tuned lens over logit lens.

3 Models

Our analysis uses open-weights models with both tied and untied configurations at the billion-parameter scale.

OLMo-1B The Open Language Model (Groeneveld et al., 2024) provides fully open-source models with publicly released training code, data, and intermediate checkpoints. Two independent OLMo-1B runs are available: `OLMo-1B`[2] (tied, February 2024) and `OLMo-1B-0724`[3] (untied, July 2024), sharing the same core architecture but trained on different snapshots of the Dolma dataset (Soldaini et al., 2024). The data-version mismatch is a confound, but the pair remains the cleanest open comparison for isolating weight tying at billion-parameter scale.

Qwen3 The Qwen3 family uses scale-dependent weight tying: smaller models (0.6B, 1.7B, and 4B) tie embeddings while larger models (8B and above) untie them. All models share training methodology, tokenizer, and architectural design principles, varying only in parameter count[4]. We compare Qwen3-4B (tied) with Qwen3-8B (untied) to replicate our OLMo findings across different model families. While model size is a confound, the shared training setup helps isolate the effect of weight tying.

GPT-Neo and Pythia We have found no open-weight, billion-scale model families that were designed to isolate weight tying, though OLMo-1B comes closest with its tied and untied releases. The next closest pair matching parameter size is GPT-Neo-2.7B (Black et al., 2021) and Pythia-2.8B (Biderman et al., 2023). Both have 32 layers, hidden size 2560, FFN width 10240, and context 2048, so their transformer blocks contain essentially the same parameter count. They differ in positional encoding (learned absolute vs. partial RoPE), attention pattern (alternating sparse/dense vs. fully dense), and transformer block structure (sequential vs. parallel). Both are trained on the Pile (Gao et al., 2020), both ship intermediate checkpoints, and crucially, GPT-Neo ties its embeddings while Pythia does not. The ∼130M parameter gap between the two models corresponds to the size of the untied

[2] https://huggingface.co/allenai/OLMo-1B-hf

[3] https://huggingface.co/allenai/OLMo-1B-0724-hf

[4] We note that unlike GPT-Neo, Pythia, and OLMo, training data are closed and may not be the same across models.

output matrix. We report this less-controlled comparison because the central pattern largely holds despite these differences.

4 Embedding Space Alignment Analysis

4.1 Representational Alignment

To test whether tied embeddings more closely resemble input or output representations, we compare the tied embedding matrix to both input and output embedding matrices from corresponding untied models. Following prior work (Conneau et al., 2020), we learn transformations to align embeddings between matrix pairs and evaluate alignment quality using mean cosine similarity between corresponding token vectors after alignment.

We consider three transformation types of increasing expressiveness:

- **Identity**: Without transformations, we measure whether embeddings are already aligned.
- **Orthogonal**: With a Procrustes analysis (Schönemann, 1966), which preserves distances and angles, we test whether the spaces differ only by a rotation transformation.
- **Linear**: Using an unconstrained linear mapping via least squares and allowing arbitrary rescaling, we ask if a linear relationship exists between the spaces.

Table 1 shows the alignment results for OLMo-1B and GPT-Neo/Pythia. Because Qwen models have different embedding dimensions, we instead use KNN overlap and spectral distance analysis (Appendix B); the same output-bias pattern holds, with the tied matrix sharing 71.0% of its nearest neighbors with the untied output versus only 36.6% with the untied input.

For each transformation, we align embedding matrices pairwise: *Input (U)* $\rightarrow$ *Output (U)* aligns the untied input and output matrices, *Output (U)* $\rightarrow$ *Tied* aligns the untied output to the tied matrix, and *Input (U)* $\rightarrow$ *Tied* aligns the untied input to the tied matrix. We compare OLMo-1B (tied) against OLMo-1B-0724 (untied), and GPT-Neo-2.7B (tied) against Pythia-2.8B (untied). If output-layer gradients dominate the shared matrix, we expect that aligning the untied output matrix to the tied matrix should yield higher similarity than aligning the untied input matrix to the tied matrix. We find:

The tied embedding matrix is more aligned with the output embedding matrix of corresponding untied models than the input embedding matrix. For both OLMo and Pythia models, the cosine similarity score for *Output (U)* $\rightarrow$ *Tied* is much larger than the similarity scores for *Input (U)* $\rightarrow$ *Tied* for all three transformations (Table 1). This shows that the shared embedding matrix in tied models resemble what an untied output matrix would look like much more than an untied input matrix. Supplemental KNN overlap and spectral distance analyses corroborate this finding across all three model families (see Appendix B).

Comparison	Identity	Orthogonal	Linear
OLMo-1B (tied) vs OLMo-1B-0724 (untied)			
Input (U) $\rightarrow$ Output (U)	0.012	0.440	0.574
Output (U) $\rightarrow$ Tied	**0.014**	**0.669**	**0.719**
Input (U) $\rightarrow$ Tied	0.001	0.420	0.525
GPT-Neo-2.7B (tied) vs Pythia-2.8B (untied)			
Input (U) $\rightarrow$ Output (U)	-0.001	0.456	0.518
Output (U) $\rightarrow$ Tied	**0.001**	**0.507**	**0.637**
Input (U) $\rightarrow$ Tied	0.000	0.376	0.507

Table 1: Geometric alignment between embedding spaces under identity, orthogonal (rotation), and linear (unconstrained least squares) transformations. We report mean cosine similarity between corresponding token embeddings from tied and untied models.

Untied input and output matrices exhibit limited similarity The alignment between untied embeddings yields only moderate similarity even after learning a linear transformation between the two spaces (0.574 for OLMo, 0.440 for GPT-Neo/Pythia; *Input (U)* $\rightarrow$ *Output (U)* rows in Table 1). This suggests that input and output embedding representations do not converge when embeddings are untied. Thus, we argue that tying forces a compromise between representational demands.

4.2 Tuned Lens Reveals "First Layer Penalty"

The alignment analysis above reveals that tied embeddings structurally resemble output embeddings. But does this structural similarity have mechanistic consequences for how the model processes information? To investigate, we turn to the tuned lens.

The tuned lens (Section 2.3) trains an affine translator at each layer to minimize the KL divergence between the translated hidden state's prediction distribution and the model's final output distribution. The residual KL divergence after training indicates how well each layer's representations align with

the output space, lower values suggest better alignment, making this metric particularly informative for our analysis.

Method We specifically focus on the residual KL-divergence after training translators in early layers. Because weight tying makes the input and output embedding matrices identical, one might expect reduced input–output mismatch and therefore lower tuned-lens KL divergence at early layers. On the contrary, if weight tying forces the embedding matrix to prioritize output prediction, we instead can expect early layers in tied models to produce representations that are *less* compatible with the output space, since these early layers must work with input embeddings optimized for a different purpose. This would manifest as higher KL divergence at early layers compared to untied models. We train per-layer translators on WikiText-103 (Merity et al., 2017) with step budgets matched within each comparison pair (full training details in Appendix C).

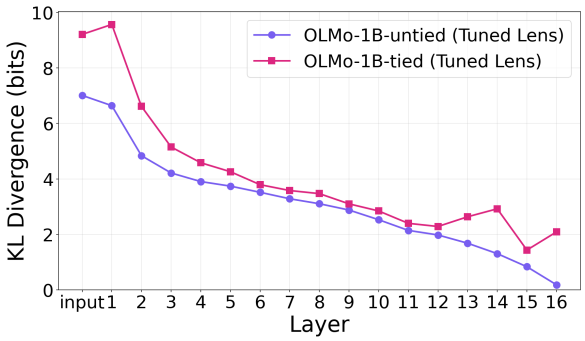

Figure 2: Tuned lens KL divergence for tied vs untied OLMo-1B. Lower values indicate better alignment between a layer's representations and the output space. The clearest separation appears in the early layers, where the tied model shows higher KL divergence.

Results The results for residual KL-divergence after training translators, for the same number of steps, for OLMo-1B (tied vs untied) are shown in Figure 2. We additionally report results for Pythia-2.8B vs GPT-Neo-2.7B and Qwen3-4B vs Qwen3-8B in Appendix C. We see that **tied models exhibit systematically higher KL divergence than untied counterparts in the early layers.** For OLMo-1B, the tied model starts at approximately 9.2 bits at the input layer compared to 7.0 bits for the untied model, a gap of over 2 bits. This gap persists across the early layers before gradually narrowing, indicating that the first few layers of the tied model

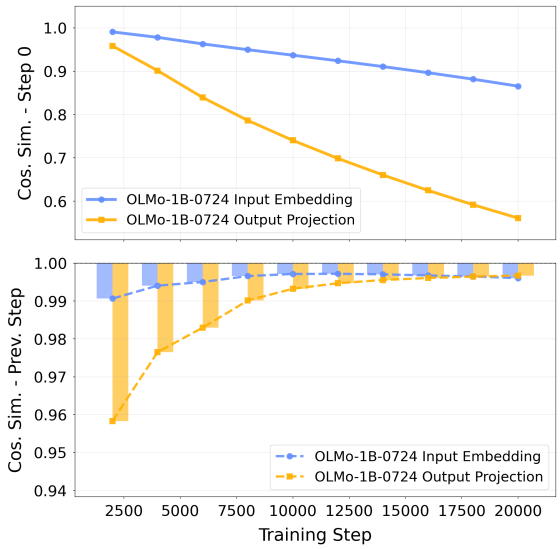

Figure 3: For OLMo-1B-0724 (untied). Top: cosine similarity to initial embeddings (step 0), measuring cumulative drift. Bottom: cosine similarity between neighbouring checkpoints (sampled every 2000 training steps).

operate with representations that are harder to align with the output space.

This elevated divergence suggests that early-layer activations in tied models are harder to align with the output space. The first transformer layer, which operates on representations produced by the tied embedding matrix, contributes less effectively to the residual stream. Later layers must compensate for this initial mismatch rather than building predictions incrementally from the start.

5 Output Gradient Dominance

Our alignment analysis reveals a clear unembedding bias: tied embeddings more closely resemble untied output embeddings than untied input embeddings. And our tuned lens analysis shows the implications of this mismatch, where the early layers in tied models contribute less effectively to the residual stream. In this section, we explore the reason behind the dominance of the tied output embeddings in learning the shared embedding vectors by examining gradient flow in both untied and tied models.

5.1 Output-Embedding Evolve Faster than Input-Embeddings in Untied Models

In untied models, the input and output embedding matrices are separate and receive gradients independently. Due to the direct connection be-

tween the output layer and the cross-entropy loss, the output embedding matrix may change more rapidly during training. By tracking how each matrix evolves across model checkpoints, we can establish whether this asymmetry exists even when the matrices are not tied.

This analysis requires access to model weights at multiple points during training. **Pythia** (Biderman et al., 2023) released many checkpoints for all models, saved at regular intervals during training; **OLMo** (Groeneveld et al., 2024) releases checkpoints every 1000 training steps, too. Both families have models with untied embeddings, allowing us to track input and output matrices independently.

Method To analyze embedding evolution dynamics, we compute two complementary metrics: (1) the mean cosine similarity between each checkpoint and the initial (step 0) embeddings, which measures cumulative drift from initialization, and (2) the mean cosine similarity between each token's embedding vector at neighbouring checkpoints, which measures how rapidly the matrix is changing. Figure 3 shows these metrics for OLMo-1B-0724 (untied) across the first 20000 steps. The plots for Pythia can be found in Figure 9 (Appendix D). We sample checkpoints at intervals of 2000 steps for OLMo to ensure meaningful differences between samples.

Results Across both models, we observe a consistent pattern: the output embedding matrix undergoes substantially larger updates than the input embedding matrix, especially early in training.

For OLMo-1B-0724 (untied), by step 20K the output matrix cosine similarity with its initialization is approximately 0.56, while for the input matrix it is 0.87 (Figure 3). The bottom panel reveals that this difference originates in the first few thousand steps. The output matrix shows cosine similarity of 0.96 between neighbouring checkpoints, indicating substantial change over each 2000-step interval. By contrast, that of the input matrix remains near 1.0, indicating minimal change. Both matrices converge to similar rates later in training, but the early asymmetry permanently transforms the output matrix. For Pythia-1B, this is even more pronounced (Figure 9). The output matrix shifts dramatically from the first checkpoint, with cosine similarity between neighbouring checkpoints as low as 0.78 in early training.

We hypothesize that this early-training asymmetry is critical for tied models: **when input and output embeddings share parameters, the dominant output-layer gradients in the early phases of training shape the shared matrix**, potentially at the expense of early-layer compatibility.

5.2 Norm-Frequency Relationship

We also qualitatively compare the tied and untied embedding vectors using norm-frequency plots, which show how embedding norms vary with token frequency. Here we calculate the norm-distribution of the embedding vectors as a function of the frequency of occurrence of tokens in a 300MB English-text sample from OSCAR-2301 (Abadji et al., 2022).[5]

Figure 4 plots the L2 norm of each token's embedding against its log-frequency for OLMo-1B after 10k steps (20B tokens). With fixed y-axis scales across all three panels, it is easy to see that the untied input embedding matrix (left) has not moved far from initialization, with norms clustered tightly around 0.8–1.0. In contrast, the untied output (center) and tied (right) matrices have evolved substantially and closely resemble each other, both showing a characteristic pattern where norms increase with token frequency up to approximately 10^4 occurrences before exhibiting a downward-sloping trend for very high frequency tokens. The visual resemblance between the tied and untied output matrices, at this early stage of training, supports our finding that the tied matrix is shaped predominantly by output-layer gradients from the start. This serves as additional support for our quantitative alignment and model checkpoints analyses above.

5.3 Output Gradients Dominate Training in Weight-Tied Models

The checkpoint analysis in Section 5.1 demonstrates that output embeddings evolve faster than input embeddings when they are separate parameters. But does this gradient asymmetry persist when the matrices are tied? To answer this directly, we measure where gradients originate in weight-tied models. In weight-tied architectures, the same matrix serves both as input embeddings and output projection, with gradients from both pathways accumulating into the shared parameters. To disentangle these contributions without altering training dynamics, we use gradient hooks that observe—but do not modify—the separate gradient flows. We

[5]We use the pre-assembled file `eng_Latn_300mb.txt` from https://huggingface.co/datasets/catherinearnett/monolingual-tokenizer-data.

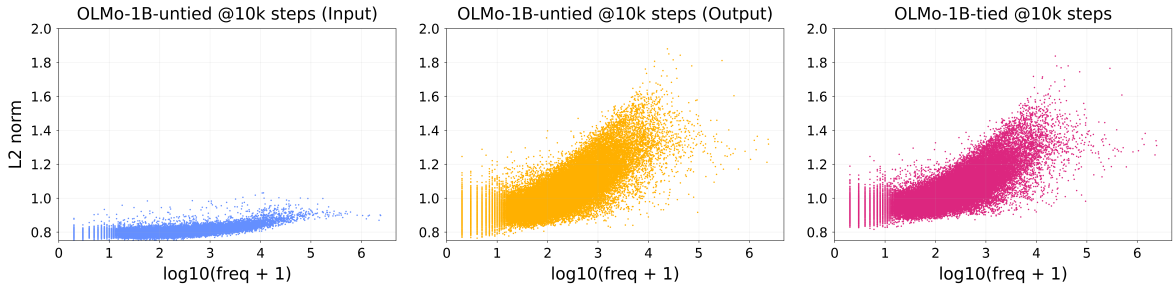

Figure 4: Norm-frequency relationship for OLMo-1B after 10k steps (20B tokens). Left: untied model's input (blue) and output (orange) matrices. Right: tied model's shared matrix (pink).

log the L2 norms of gradients through each pathway during early training, quantifying how much learning signal originates from each side.

Method We train OLMo-1B with tied embeddings from scratch[6] using standard OLMo training configuration and record gradient norms at every training step for the first 1000 steps. To reduce noise, we apply rolling average with a window of 20 steps. We do not modify the original training recipe and just track the contributions of gradient flows from output and input embedding paths.

Results Figure 5 shows the results. The left panel (log scale) reveals a persistent gap between the two pathways: output-layer gradients (orange) consistently exceed input-layer gradients (blue) throughout the first 1000 training steps. The right panel shows the relative contribution of each pathway (in percent of gradient norm). **Output-layer gradients account for approximately 70% of the total signal during early training**, with input-layer contributions making up the remaining 30%.

This persistent imbalance confirms that in weight-tied architectures, the shared embedding matrix receives the majority of its learning signal from the output layer, particularly during the critical early phase of training when representations are first being shaped. This explains the embedding alignment results from Section 4 and sheds light on the source of the weight tying penalty: **the shared matrix in weight-tied models is optimized primarily for output prediction from early training stages**, leaving the early transformer layers to work with a less compatible representational space.

[6] https://github.com/allenai/OLMo/tree/main/configs

6 Causal Ablation : Selective Gradient Scaling in Tied Models

Our analysis so far has established that (1) tied embeddings structurally resemble untied output embeddings (Section 4.1), (2) early layers in tied models contribute less effectively to the residual stream (Section 4.2), and (3) output-layer gradients dominate the shared embedding matrix during training (Section 5.3). In order to establish a causal link showing that gradient flows from output embedding dominate training in tied models, we intervene directly on the shared embedding gradients during training. If output-gradient dominance *causes* the tied matrix to develop output-like structure, then artificially amplifying input-layer gradients should shift the resulting embeddings toward input-like structure. Conversely, if the gradient imbalance is merely a side effect, scaling should have little impact on the final representations.

We train OLMo-1B with tied embeddings from scratch with the original training configuration, using gradient hooks to multiply input-layer gradients by a scaling factor before they accumulate into the shared embedding matrix. We compare a baseline model with no scaling against one with input gradients scaled by 5×. Since training these models from scratch is expensive, we pause our training for both models after 10000 steps (20B tokens) when the results become evident. We then measure cosine similarity (after alignment) between each tied model's embedding matrix and the input/output matrices from OLMo-1B-0724 (untied) at the same training step. The results are in Table 2. We report additional results in Appendix E.

Scaling input gradients shifts embedding structure predictably towards untied input embeddings and away from untied output embeddings. With 5× input gradient scaling, the resulting em-

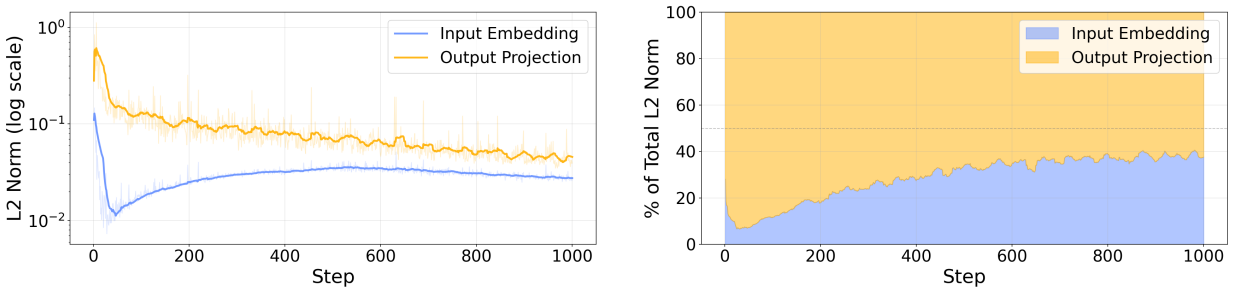

Figure 5: Gradient flow to the shared embedding matrix in tied OLMo-1B during the first 1000 training steps. Left: L2 norm of gradients from the input embedding (blue) and output projection (orange) pathways on a log scale. Right: relative contribution of each pathway as a percentage of total gradient norm.

Model	vs Untied Input	vs Untied Output
Tied (no scaling)	0.216	0.384
Tied (input $\times 5$)	0.222	0.369

Table 2: Cosine similarity (after alignment) between tied OLMo-1B embedding matrices and the input/output matrices of OLMo-1B-0724 (untied) at step 10K. Higher gradient scaling shifts mean more input-like structure.

bedding matrix becomes more aligned with the untied input matrix (0.216 $\rightarrow$ 0.222) but is *less* aligned with the untied output matrix (0.384 $\rightarrow$ 0.369). Additional scaling factors and earlier checkpoints show the same trend (Appendix E).

This trade-off is inherent to weight tying. Shifting the shared matrix toward one role necessarily moves it away from the other, and the gradient balance causally determines which role dominates. Under this simple constant-scaling intervention, there is no "free" equilibrium, improving input compatibility comes at the cost of output compatibility. Downstream evaluation shows no consistent performance gains from this structural shift (Appendix E), which is in line with the zero-sum nature of this trade-off. Whether more sophisticated interventions can navigate this trade-off remains an open question.

7 Discussion

7.1 When to Tie vs. Untie

The decision to tie or untie embeddings ultimately reduces to a trade-off between parameter efficiency and representational capacity. Above, we highlighted that in smaller models, embedding parameters may constitute over 70% of total parameters (Pythia 70M; see Table 4). At this scale, these parameter savings are substantial and justify the representational cost we have documented. At billion-parameter scale, the efficiency advantage largely disappears while the representational disadvantage remains. This trade-off is reflected in contemporary models. The Qwen3 developers explicitly cite this as a consideration: smaller models (0.6B, 1.7B, and 4B parameters) use tied embeddings, while larger models (8B and above) untie them. Here, we provide a mechanistic justification for this threshold: at scale, the cost of adding a separate output matrix is negligible, but the benefit of allowing the input embeddings to evolve independently is present.

7.2 Gradient Rebalancing as a Training Intervention

Our causal intervention experiments (Section 6) establish that gradient imbalance causally determines the structure of the tied embedding matrix. We also evaluated whether rebalancing gradients improves downstream performance and find that it does not: the scaled model shows no consistent gains over the baseline. This is in line with the trade-off we document, as shifting the shared matrix toward input-like structure necessarily reduces its effectiveness as an output projection.

Future work could explore whether more targeted interventions, beyond simple constant scaling, can navigate this trade-off. If such strategies exist, they could offer a practical middle ground: tied embeddings with intervention-corrected training dynamics that mitigate the representational cost we have documented while retaining the parameter efficiency of weight tying. *This can have important implications for smaller language models where embeddings form a large portion of the parameters.*

8 Conclusion

We have presented evidence that weight tying in language models incurs a hidden cost. By analyzing training dynamics through gradient flow, we show that the shared embedding matrix in tied models is shaped predominantly by output-layer gradients during early training, leaving input representations optimized for output prediction rather than for compatibility with early-layer computations.

This gradient imbalance manifests as a representational mismatch. We show that tied embedding matrices are structurally closer to untied output matrices than to untied input matrices, confirming that output-layer optimization dominates the shared representation. Furthermore, in our tuned lens analysis across three model families, tied models exhibit systematically higher KL divergence in their early layers compared to untied counterparts, indicating that the first layers contribute less effectively to the residual stream, a pattern consistent with input embeddings that have been trained suboptimally compared to the untied models. Untying allows the input matrix to evolve more slowly than the output matrix, free from the dominant output-layer gradients that would otherwise shape it.

Our findings provide a mechanistic explanation for the empirical observation that untying embeddings improves performance at scale. At billion-scale and beyond, the parameter savings from weight tying become negligible while the representational cost remains. For practitioners designing large language models, we recommend untying embeddings when parameter budget permits, as the representational benefits outweigh the modest increase in model size. For smaller models where parameter savings matter most, our work motivates training interventions that could preserve the efficiency benefits of weight tying while reducing its representational cost.

Limitations

Our analysis relies on several assumptions that merit discussion. While the tuned lens reveals that tied models exhibit systematically higher KL divergence in early layers compared to untied models, we cannot be certain this difference reflects a meaningful representational cost. The elevated divergence may be compensated for in later layers.

For each setting, we test only a single training run, as training runs at the billion-parameter scale are very costly, therefore we were not able to conduct statistical tests across multiple training runs, random seeds, etc.

Our experiments are also constrained to specific model families (OLMo, Pythia, and Qwen3), chosen for their availability of tied/untied variants or intermediate checkpoints. While these models span different training configurations and scales, our findings may not generalize to architectures with substantially different designs, such as mixture-of-experts models or non-autoregressive language models.

Finally, we do not quantify the magnitude of the representational penalty in downstream-performance terms, nor do we identify a parameter-count or training-budget threshold above which untying becomes preferable. Practical guidance therefore remains qualitative: at billion-parameter scale and beyond, parameter savings from tying shrink while the representational cost persists, but the precise crossover point is left to future work.

Acknowledgments

We thank Luca Malagutti for helpful discussions and feedback throughout this project. Experiments were conducted using compute resources from EleutherAI.

References

Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. 2022. Towards a cleaner document-oriented multilingual crawled corpus. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4344–4355, Marseille, France. European Language Resources Association.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294. Association for Computational Linguistics.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Francesco Bertolotti and Walter Cazzola. 2024. By tying embeddings you are assuming the distributional hypothesis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 3584–3610. PMLR.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*.

Daniel Biś, Maksim Podkorytov, and Xiuwen Liu. 2021. Too much in common: Shifting of embeddings in transformer language models and its implications. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5117–5130. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Hyung Won Chung, Thibault Févry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. 2020. Rethinking embedding coupling in pre-trained language models. *arXiv preprint arXiv:2010.12821*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2924–2936.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Alexis Conneau, Shijie Wu, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6022–6034. Association for Computational Linguistics.

DeepSeek-AI. 2024. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Brandon Duderstadt, Hayden S. Helm, and Carey E. Priebe. 2023. Comparing foundation models using data kernels. *arXiv preprint arXiv:2305.05126*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of EMNLP-IJCNLP*, pages 55–65.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Representation degeneration problem in training natural language generation models. In *Proceedings of ICLR*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2023. A framework for few-shot language model evaluation.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024. Gemma: Open models based on Gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, and 1 others. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *arXiv preprint arXiv:2003.11080*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proceedings of ICLR*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Anemily Machina and Robert Mercer. 2024. Anisotropy is not inherent to transformers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Association for Computational Linguistics.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

nostalgebraist. 2020. interpreting GPT: the logit lens. LessWrong.

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

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Sara Rajaee and Mohammad Taher Pilehvar. 2022. An isotropy analysis in the multilingual BERT embedding space. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, and 1 others. 2024. Dolma: An open corpus of three trillion tokens for language model pretraining research. *arXiv preprint arXiv:2402.00159*.

Felix Stollenwerk and Tobias Stollenwerk. 2025. Better embeddings with coupled adam. *arXiv preprint arXiv:2502.08441*.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, and 1 others. 2025. Olmo 3. *arXiv preprint arXiv:2512.13961*.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, and 1 others. 2024. Olmo 2: The best fully open language model to date. *arXiv preprint arXiv:2501.00656*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Dilin Wang, Chengyue Gong, and Qiang Liu. 2019. Improving neural language modeling via adversarial training. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6555–6565. PMLR.

Lingxiao Wang, Jing Huang, Kevin Huang, Ziniu Hu, Guangtao Wang, and Quanquan Gu. 2020. Improving neural language generation with spectrum control. In *International Conference on Learning Representations*.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Appendix

A Parameter Counts

Table 4 shows the proportion of total parameters contained in the embedding matrices for the Pythia

Model	Total (Untied)	Total (Tied)	Saved
Pythia-70M	70.4M	44.7M	25.8M
Pythia-160M	162.3M	123.7M	38.6M
Pythia-410M	405.3M	353.8M	51.5M
Pythia-1B	1011.8M	908.8M	103.0M
Pythia-2.8B	2775.2M	2646.4M	128.8M

Table 3: Total parameter counts with and without weight tying for Pythia models. For an untied model, tying removes one embedding matrix, saving Vd parameters (half of the untied embedding total $2Vd$). Totals are computed by summing checkpoint parameter counts.

Model	Hidden	Embed Params	% of Total
Pythia-70M	512	51.4M	73.4%
Pythia-160M	768	77.2M	48.3%
Pythia-410M	1024	103.0M	25.1%
Pythia-1B	2048	206.0M	20.6%
Pythia-2.8B	2560	257.4M	9.2%

Table 4: Embedding parameters for the Pythia suite (untied). Embed Params shows the combined input and output matrices ($2 \times V \times d$ where $V = 50{,}257$). Weight tying would halve these values, saving over 35% of parameters at small scale but less than 5% at billion-scale.

model family (Biderman et al., 2023). This family of models does not use weight tying. For the 70M parameter model, the embedding matrices account for over 70% of all parameters; even at 2.8B scale, they still represent approximately 9% of the model, so tying can still yield meaningful savings.

Table 3 reports total parameter counts with and without weight tying, computed by subtracting Vd parameters from the untied checkpoints.

Table 4 shows that embedding matrices constitute a substantial fraction of total parameters at smaller scales, but this proportion decreases as models grow. At 70M parameters, embeddings account for 73.4% of the model, making weight tying highly impactful for parameter efficiency. By 2.8B parameters, this drops to 9.2%.

B KNN Overlap and Spectral Distance Analysis

As complementary metrics to cosine similarity, we measure (1) **KNN@10 overlap**: the fraction of shared nearest neighbors between token embeddings in two matrices (higher = more similar), and (2) **spectral distance**: the operator-norm distance between the jointly embedded k-NN graphs via the omnibus adjacency spectral embedding of Duderstadt et al. (2023) (lower = more similar).

Comparison	KNN@10 (↑)	Spectral Dist (↓)
OLMo-1B (tied) vs OLMo-1B-0724 (untied)		
Input (U) ↔ Output (U)	0.455	1.349
Output (U) ↔ Tied	**0.733**	**0.588**
Input (U) ↔ Tied	0.496	1.373
Qwen3-4B (tied) vs Qwen3-8B (untied)		
Input (U) ↔ Output (U)	0.366	1.421
Output (U) ↔ Tied	**0.710**	**0.809**
Input (U) ↔ Tied	0.366	1.441
GPT-Neo-2.7B (tied) vs Pythia-2.8B (untied)		
Input (U) ↔ Output (U)	**0.611**	**0.949**
Output (U) ↔ Tied	0.408	1.035
Input (U) ↔ Tied	0.372	1.462

Table 5: KNN@10 overlap (higher = more similar) and spectral distance (lower = more similar) between embedding spaces. Bold indicates the most similar pair within each family. GPT-Neo/Pythia uses aligned vocabulary (36,938 tokens).

Both metrics tell a consistent story. For OLMo-1B, the tied matrix shares 73.3% of its nearest neighbors with the untied output versus only 49.6% with the untied input; spectral distances are 0.588 versus 1.373. Qwen3 shows an even starker contrast: 71.0% overlap with output versus 36.6% with input; spectral distances are 0.809 versus 1.441. The pattern holds for GPT-Neo/Pythia (KNN@10: 0.408 vs. 0.372; spectral distance: 1.035 vs. 1.462), though the effect is smaller, likely because of the bigger differences between the two models.

C Additional Results — Tuned Lens Analysis

Section 4.2 presented tuned lens results for OLMo-1B (tied vs untied). Here we extend this analysis to Pythia-2.8B vs GPT-Neo-2.7B and Qwen3-4B vs Qwen3-8B. The methodology remains identical: we train per-layer affine translators that map hidden states at each layer to the final layer's representation space, minimizing KL divergence. All translators are trained on WikiText-103 (raw, v1) (Merity et al., 2017); step budgets are matched within each comparison pair but differ across pairs (OLMo-1B: 500 steps; Pythia-2.8B / GPT-Neo-2.7B: 100 steps; Qwen3-4B / 8B: 50 steps).

Figure 6 shows results for Pythia-2.8B (untied) and GPT-Neo-2.7B (tied). Both models have 32 layers. GPT-Neo-2.7B (tied) starts at approximately 17 bits at the input layer compared to 7 bits for Pythia-2.8B (untied), a gap of 10 bits. This gap subsequently increases to 15 bits and then narrows, with the curves starting to converge around layer

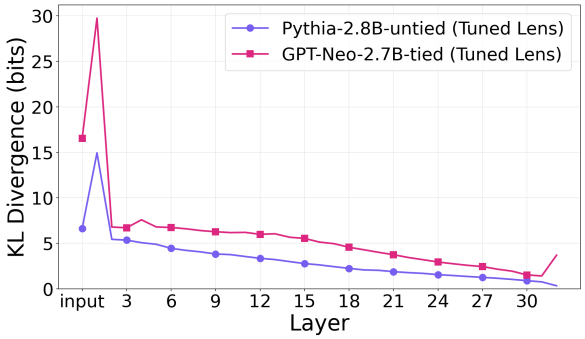

Figure 6: Tuned lens KL divergence across layers for Pythia-2.8B (untied) vs GPT-Neo-2.7B (tied). Both models have 32 layers. GPT-Neo-2.7B shows elevated KL divergence in the early layers.

10. Both models show a spike in KL divergence early on followed by a smooth, monotonic reduction until convergence. Pythia-2.8B (untied) also exhibits a slight increase from the input layer to layer 1, though much smaller than the tied model's spike. The elevated early-layer KL divergence in GPT-Neo-2.7B suggests that early layer activations in tied models are harder to align with the output space and the first layer is contributing less effectively to the residual stream.

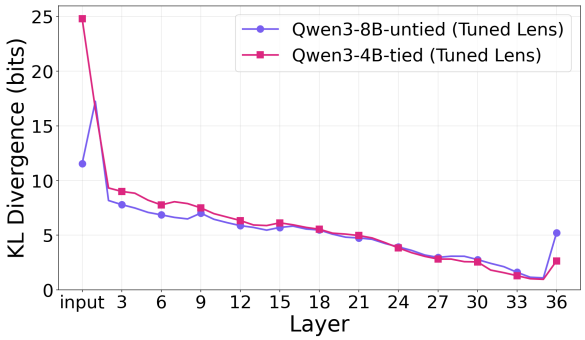

Figure 7: Tuned lens KL divergence for Qwen3-4B (tied) vs Qwen3-8B (untied). Qwen3-4B (tied) displays significantly higher early-layer KL divergence before starting to converge with Qwen3-8B (untied).

Figure 7 presents results for Qwen3-4B (tied, 36 layers) and Qwen3-8B (untied, 36 layers). We again observe tied models exhibiting higher early-layer KL divergence. Qwen3-4B (tied) begins at approximately 25 bits, while Qwen3-8B (untied) starts at 12 bits, a 13-bit gap. Qwen3-8B (untied) also shows a slight increase from the input layer to layer 1, though the gap relative to the tied model remains large. This gap narrows quickly, with the tied model maintaining elevated divergence through layer 10, where the curves start to converge. In the final layers, both models show increasing KL divergence.

The consistency of elevated early-layer KL divergence across three independent model family comparisons in OLMo (same family, different training runs), Pythia/GPT-Neo (similar architecture, same dataset), and Qwen3 (same family, different sizes), provides evidence that weight tying creates an early-layer representational penalty. This pattern holds across different model scales (from 1B to 8B parameters), different training datasets, and different architectural details, even though the exact depth profile beyond the earliest layers varies across architectures.

D Additional Results — Untied Models Matrices Updates

Section 5.1 presented embedding evolution dynamics for OLMo-1B-0724 (untied). Here we extend this to OLMo-7B and Pythia-1B. For each model, we compute cumulative drift (cosine similarity to initial embeddings) and similarity between neighbouring checkpoints. OLMo checkpoints are available every 1000 steps; Pythia has 154 checkpoints spanning the full training run with denser sampling early in training.

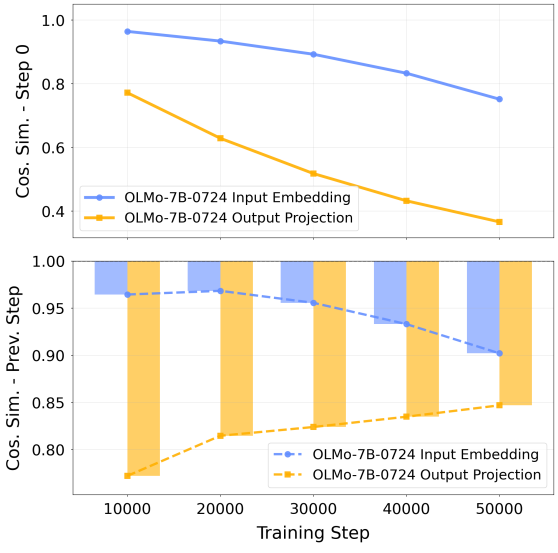

Figure 8: Embedding matrix dynamics for OLMo-7B (untied). Top: cosine similarity to initial embeddings (step 0), measuring cumulative drift. Bottom: cosine similarity between neighbouring checkpoints.

Figure 8 shows OLMo-7B (untied) dynamics. The top panel (similarity vs initial) shows the cumulative effect: by step 50,000, the output matrix

drifts to cosine similarity of approximately 0.38 with initialization, while input only drifts to 0.75. The bottom panel reveals that output embeddings change much more rapidly than input embeddings, particularly in the first 10,000 steps where output similarity drops to approximately 0.78 while input remains above 0.96. This shows that the output embedding matrix undergoes significantly larger updates than the input embedding matrix in the early stages of training.

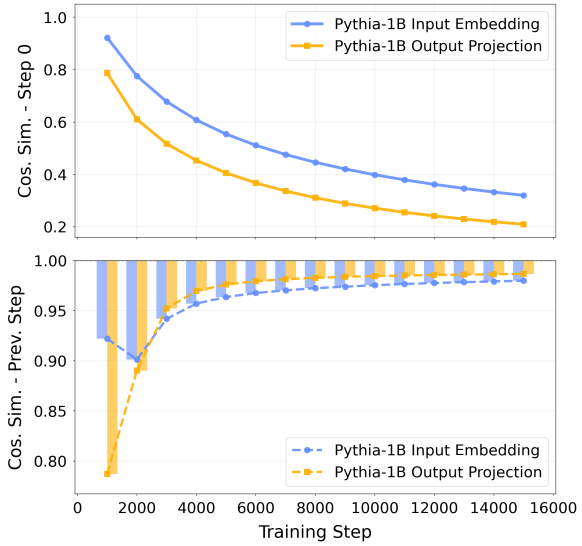

Figure 9: Embedding matrix dynamics for Pythia-1B (untied). Top: cosine similarity to initial embeddings (step 0), measuring cumulative drift. Bottom: cosine similarity between neighbouring checkpoints.

Figure 9 shows Pythia-1B (untied). The top panel shows that output embeddings drift dramatically from initialization in the very first checkpoint, dropping to cosine similarity of approximately 0.79 (21% change) while input only drops to 0.92 (8% change). The bottom panel confirms this explosive early update: output reaches as low as 0.79 cosine similarity between the first two checkpoints, indicating massive restructuring, while input remains relatively stable. Both matrices then stabilize after approximately step 4000. By the end of training (top panel), output has drifted to cosine similarity of 0.21 with initialization, while input has only drifted to 0.32.

The consistent pattern across OLMo-1B-0724 (Section 5.1), OLMo-7B, and Pythia-1B establishes that output embeddings undergo substantially larger updates than input embeddings during early training, even when matrices are untied and receive gradients independently. This reflects the inherent structure of the training objective where output-layer gradients are naturally stronger than input-layer gradients.

E Additional Results — Gradient Scaling

Section 6 presented gradient scaling results at step 10,000 for tied OLMo-1B models trained with input gradient scaling factors of 1× (baseline) and 5×. Here we present additional results at step 1000 with scaling factors of 2× and 10× to demonstrate the robustness of the intervention effect. The experimental setup remains identical: we train OLMo-1B with tied embeddings from scratch using the standard configuration, applying gradient hooks to multiply input-layer gradients by a constant factor before they accumulate into the shared embedding matrix. At checkpoints, we measure cosine similarity (after Procrustes alignment) between the tied model's embedding matrix and the input/output matrices from OLMo-1B-0724 (untied) at matched training steps.

Model	vs Untied Input	vs Untied Output
Tied (no scaling)	0.172	0.197
Tied (input ×2)	0.173	0.197
Tied (input ×10)	0.173	0.190

Table 6: Cosine similarity after Procrustes alignment of tied OLMo-1B models trained with scaled input gradients at step 1000. Even early in training, higher input gradient scaling slightly increases alignment with the untied input matrix and decreases alignment with the untied output matrix.

Table 6 shows alignment results at step 1000, much earlier in training than the step 10,000 results in Table 2 (Section 6). Even at this early stage, the intervention has a measurable effect. With baseline training (no scaling), the tied matrix achieves 0.172 cosine similarity with untied input and 0.197 with untied output, already showing the expected output bias. With 2× input gradient scaling, alignment with untied input increases slightly to 0.173 while alignment with untied output remains at 0.197. With 10× scaling, input alignment remains at 0.173 while output alignment decreases to 0.190.

The effect size is modest at step 1000, reflecting that only 2B tokens have been processed. However, the directionality is already correct: amplifying input gradients shifts the matrix toward input structure and away from output structure. The gradient scaling results across two checkpoints (step 1000 and step 10,000) demonstrate that gradient balance

causally determines embedding structure, and this effect accumulates over training. The modest effect sizes reflect a fundamental constraint: improving input compatibility necessarily reduces output compatibility, confirming that weight tying forces a compromise between distinct representational demands.

Downstream evaluation. A natural follow-up is whether the structural shift from gradient scaling translates into downstream performance gains. We evaluate the baseline tied model and the $5\times$-scaled model using lm-evaluation-harness (Gao et al., 2023) on WikiText-2 (Merity et al., 2017), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2020), ARC (Clark et al., 2018), BoolQ (Clark et al., 2019), OpenBookQA (Mihaylov et al., 2018), and BLiMP (Warstadt et al., 2020), both at step 10,000.

Benchmark	Tied	Tied (input $\times$5)
WikiText-2 PPL ↓	**35.71**	36.64
PIQA	0.620	**0.631**
HellaSwag	**0.329**	0.326
Winogrande	**0.514**	0.509
ARC-Easy	0.396	**0.402**
ARC-Challenge	0.233	**0.241**
BoolQ	**0.576**	0.548
OpenBookQA	0.256	**0.278**
BLiMP	0.755	**0.757**

Table 7: Downstream evaluation of tied OLMo-1B models with and without input gradient scaling at step 10K (20B tokens). Higher is better for all benchmarks except WikiText-2 (perplexity, lower is better). No consistent performance difference is observed.

Table 7 shows that although gradient scaling measurably shifts embedding structure (Table 2), this does not translate into consistent downstream improvements. Perplexity slightly worsens (35.71 $\rightarrow$ 36.64), consistent with the trade-off identified in Section 6: shifting the shared matrix toward input-like structure reduces its effectiveness as an output projection. Downstream results are mixed—the scaled model gains on PIQA, ARC-Challenge, and OpenBookQA, but loses on BoolQ, HellaSwag, and Winogrande, with no clear overall winner. BLiMP, a linguistic minimal-pair benchmark that directly tests grammatical competence, shows the two models are essentially identical (75.5% vs 75.7%). A naive constant scaling factor rebalances gradient contributions but does not resolve the underlying compromise.

F Chung et al. XTREME Benchmark Results

Section 2.1 references Chung et al. (2020) as key prior work demonstrating that weight tying can harm performance in transformer-based models. Table 8 displays their central finding from experiments on multilingual BERT-style models. They trained two configurations: coupled (tied embeddings, 177M parameters for both pre-training and fine-tuning) and decoupled (untied embeddings, 269M parameters during pre-training but pruned to 177M for fine-tuning to ensure fair comparison at evaluation time). Both models were pre-trained on multilingual data and evaluated on the XTREME benchmark, which tests cross-lingual transfer across diverse tasks.

The decoupled model achieves 62.7% average performance compared to 62.3% for the coupled model. While the overall gain is modest (0.4 percentage points), improvements appear consistently across most tasks. XNLI (cross-lingual natural language inference) improves by 0.6 points (71.3 vs 70.7). The largest gains appear in question-answering tasks: XQuAD improves by 0.7/0.6 EM/F1 points (46.9/63.8 vs 46.2/63.2) and TyDi QA by 2.1/1.4 EM/F1 points (42.8/58.1 vs 40.7/56.7). These tasks may particularly benefit from improved input representations that support compositional reasoning.

Chung et al. (2020) established that untying helps but did not explain why. Our work provides the mechanistic explanation: decoupling allows input embeddings to specialize for semantic representation rather than being forced into an output-optimized space shaped by dominant output-layer gradients. Our tuned lens analysis (Section 4.2, Appendix C) connects directly to these empirical results: elevated KL divergence in early layers of tied models indicates less effective compositional representations, which would particularly impact deep semantic understanding required for question answering.

G License Information

GPT-Neo is released under an MIT license and Pythia, OLMo 3, and Qwen3 are all released under Apache 2.0 licenses. The Dolma dataset is released under an ODC-by license. All of these licenses permit use for scientific research and our use of these artifacts is consistent with their licenses.

	# PT params	# FT params	XNLI Acc	NER F1	PAWS-X Acc	XQuAD EM/F1	TyDi EM/F1	Avg
Coupled	177M	177M	70.7	**69.2**	**85.3**	46.2/63.2	40.7/56.7	62.3
Decoupled	269M	177M	**71.3**	68.9	85.0	**46.9/63.8**	**42.8/58.1**	**62.7**

Table 8: Effect of decoupling input and output embedding matrices on XTREME benchmark (Hu et al., 2020) performance (Chung et al., 2020). PT: Pre-training. FT: Fine-tuning. Decoupling adds parameters during pre-training but improves average downstream performance.

H AI Assistants in Research or Writing — Disclosure

We used AI writing assistants (Claude, Cursor) during the preparation of this manuscript for editing and proofreading text, formatting LaTeX code, and implementing experiments. All scientific content, experimental design, analysis, and conclusions are the sole responsibility of the authors.