AEA: Adaptive Expert Allocation Improves Sentence Embeddings from Mixture-of-Experts LLM

Shufan Yang* **Zifeng Cheng*†** **Zhiwei Jiang†**
Qingfeng Qi **Yafeng Yin** **Cong Wang** **Ao Zhou** **Qing Gu**
State Key Laboratory for Novel Software Technology, Nanjing University
sfyang@smail.nju.edu.cn {chengzf,jzw}@nju.edu.cn
qqf@smail.nju.edu.cn {yafeng,wang.c}@nju.edu.cn
za@smail.nju.edu.cn guq@nju.edu.cn

Abstract

Extracting embeddings directly from Mixture-of-Experts (MoE) models is a promising yet underexplored direction that requires no additional data or fine-tuning. While previous studies have utilized semantic compression prompts or expert routing information to improve sentence embeddings, they typically allocate a fixed number of experts uniformly across all layers and tokens, ignoring inter-layer and inter-token heterogeneity. In this work, we identify two key observations in MoE models: (1) layer-wise variations in expert homogeneity, suggesting that different layers require different expert budgets, and (2) token-wise contribution imbalance, indicating that different tokens should also be allocated different numbers of experts. To address these issues, we propose an **Adaptive Expert Allocation (AEA)** framework that dynamically performs both layer-wise and token-wise expert allocation to enhance embedding quality. Specifically, AEA allocates fewer experts to layers with higher homogeneity and to tokens with lower attention importance, where layer-wise homogeneity is determined by the similarity among embeddings produced by the experts in each layer. Notably, our method is plug-and-play, seamlessly integrates with existing prompt engineering methods, and introduces no additional time overhead. Experiments on the STS tasks demonstrate that AEA consistently improves embedding quality across multiple MoE models.

1 Introduction

Sentence embeddings are essential for a variety of natural language understanding tasks, including semantic retrieval, text matching, and question answering. With the rapid progress of large language models (LLMs) and their Mixture-of-Experts (MoE) extensions, directly eliciting zero-shot sentence embeddings without data and fine-tuning (Liu et al., 2024a; Lei et al., 2024; Zhang et al., 2024) has become an efficient and practical alternative to fine-tuning. This training-free paradigm is appealing, as it eliminates the need for additional training data, avoids the computational overhead of fine-tuning LLMs, and preserves the model's general semantic understanding ability by preventing fine-tuning on specific datasets.

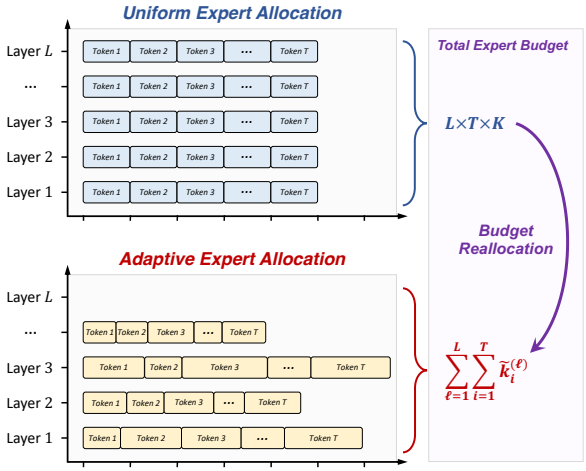

Figure 1: Comparison of uniform expert allocation and our proposed adaptive expert allocation.

Existing studies have shown that prompt engineering can effectively guide LLMs to compress the semantics of an entire sentence into the last token, whose hidden state can then be used as the sentence embedding. PromptEOL (Jiang et al., 2024) first demonstrates this idea, and subsequent methods such as Echo (Springer et al., 2025) and Token Prepending (TP) (Fu et al., 2025) further improve it through refined input designs. These training-free approaches significantly improve embedding quality without requiring any fine-tuning.

Recently, several studies have begun exploring the extraction of sentence embeddings from MoE-based LLMs. MoEE (Li and Zhou, 2025) leverages routing weights as complementary signals to

*Equal contribution.
†Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 28837–28851
July 2-7, 2026

the hidden states to enhance embedding quality. Although this approach effectively integrates expert activation information, it still adopts a uniform expert allocation across all layers and tokens, failing to capture the heterogeneous roles of layers and tokens in semantic representation, as shown in Figure 1. In particular, it overlooks two important characteristics of MoE models: (1) **Layer-wise variation in expert homogeneity**, where layers exhibit different degrees of expert homogeneity, motivating layer-adaptive expert allocation; and (2) **token-wise contribution imbalance**, where tokens contribute unequally to sentence-level semantics and thus should be assigned different numbers of experts.

To this end, we propose an **Adaptive Expert Allocation (AEA)** framework that dynamically allocates experts[1] at both the layer and token levels to enhance sentence embeddings. At the **layer-level**, we redistribute the number of experts across layers according to their degrees of expert homogeneity, assigning fewer experts to layers with higher similarity among experts. Specifically, we use a calibration dataset to compute pairwise embedding similarities among expert outputs within each layer, thereby obtaining a homogeneity score for each layer. Notably, these homogeneity scores are computed only once on the calibration set and reused thereafter. At the **token-level**, we allocate experts based on token importance, assigning fewer experts to less important tokens. Specifically, a token's importance is measured by the maximum attention score it receives across all attention heads. Importantly, the two strategies do not introduce additional experts and only affect expert allocation. As a result, our method incurs no additional time overhead and can be seamlessly integrated with existing methods.

Our main contributions are outlined below:

- We systematically reveal two key phenomena in MoE models: inter-layer expert homogenization and highly sparse attention distributions across tokens.
- We propose an **Adaptive Expert Allocation (AEA)** method that reallocates the expert budget across layers and tokens.
- Extensive experiments on multiple MoE models and STS tasks demonstrate that our method significantly outperforms existing training-free embedding extraction approaches, showcasing strong generalization and efficiency.

[1]We dynamically allocate the number of activated experts for each token.

2 Related Work

Sentence Embeddings Sentence embeddings aim to map sentences into dense vector representations that preserve semantic similarity. Early approaches (Jiang et al., 2022; Chanchani and Huang, 2023; Su et al., 2023) relied heavily on fine-tuning pretrained language models with contrastive objectives (e.g., SimCSE (Gao et al., 2021) and Sentence-T5 (Ni et al., 2022)). With the rise of LLMs, a new trend emerged: fine-tuning LLMs directly as embedding models (Li and Li, 2024; BehnamGhader et al., 2024; Lee et al., 2024; Muennighoff et al., 2024b; Wang et al., 2024; Li et al., 2025; Zhang et al., 2025; Li et al., 2026; Yang et al., 2026). These methods demonstrate strong performance, but they require a large amount of training data and incur high fine-tuning costs. In contrast, this paper focuses on training-free approaches to extracting embeddings from MoE models.

Extracting Sentence Embeddings from LLMs Most recent approaches to sentence embeddings with LLMs rely on prompting strategies, where the hidden state of a designated token is extracted as the sentence representation. A widely used baseline is PromptEOL (Jiang et al., 2024), which introduces a special prompt: *This sentence: "[TEXT]" means in one word:"*, and directly takes the hidden state of the last token as the embedding. Building on this, Echo (Springer et al., 2025) repeats the input twice within the context and extracts the embedding from the second occurrence, allowing early tokens to encode information from subsequent ones. Furthermore, TP (Fu et al., 2025) alleviates the limitation of unidirectional attention by prepending the previous layers sentence embedding as an additional token, enabling each position to access global semantics. In addition, Contrastive Prompting (CP) (Cheng et al., 2025) reduces the impact of semantically irrelevant tokens by introducing auxiliary prompts and contrasting them with the main embedding, thereby filtering out non-core information.

Extracting Sentence Embeddings from MoE Models MoEE (Li and Zhou, 2025) first leverages the routing weights of MoE models and integrates

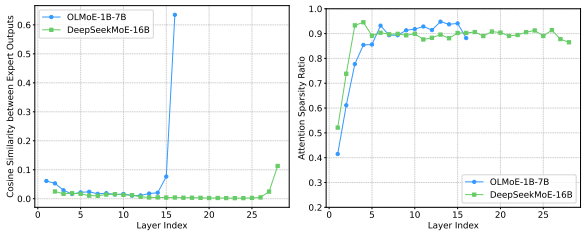

Figure 2: The left part shows the average cosine similarity between the outputs of different experts within each layer, while the right part illustrates the sparsity degree of the attention score matrices.

them with hidden states through combination to obtain improved embeddings. However, this approach adopts a uniform expert allocation across all layers and tokens. Our method differs fundamentally: rather than merely leveraging routing distributions, we focus on exploring how experts can be more effectively reallocated.

3 Preliminary

3.1 Mixture-of-Experts (MoE) Module

In the ℓ-th MoE layer, the standard feed-forward network (FFN) is replaced by a *Mixture-of-Experts (MoE)* module, which comprises a router $G^{(\ell)}(\cdot)$ and N experts $\{E_1^{(\ell)}, E_2^{(\ell)}, \ldots, E_N^{(\ell)}\}$. Given the output of attention module $\mathbf{H}_{\text{att}}^{(\ell)} = [\mathbf{h}_{\text{att},1}^{(\ell)}, \mathbf{h}_{\text{att},2}^{(\ell)}, \ldots, \mathbf{h}_{\text{att},T}^{(\ell)}] \in \mathbb{R}^{T\times d}$, the router assigns each token to a sparse subset of experts. Specifically, for the t-th token representation $\mathbf{h}_{\text{att},t}^{(\ell)} \in \mathbb{R}^d$, the router computes a score $z_{i,t}^{(\ell)}$ for each expert $E_i^{(\ell)}$ and converts these scores into routing weights through a softmax function:

$$g_{i,t}^{(\ell)} = \frac{\exp(z_{i,t}^{(\ell)})}{\sum_{j=1}^{N} \exp(z_{j,t}^{(\ell)})}.$$

To ensure sparsity, the router activates only the Top-K experts with the largest routing weights, while setting the remaining weights to zero. The token representation $\bar{\mathbf{h}}^{(\ell)}$ is then obtained as a weighted sum of the selected expert outputs:

$$\bar{\mathbf{h}}_t^{(\ell)} = \sum_{i=1}^{N} g_{i,t}^{(\ell)}\, E_i^{(\ell)}\big(\mathbf{h}_{\text{att},t}^{(\ell)}\big),$$

where $g_{i,t}^{(\ell)}$ is nonzero only for the Top-K activated experts.

3.2 Motivation of Adaptive Expert Allocation

MoEE (Li and Zhou, 2025) augments sentence embeddings with routing weights, while maintaining a uniform K-expert allocation across layers and tokens. This raises a natural question: *is uniformly allocating experts truly optimal for embedding extraction?*

Observation 1: Layer-level expert homogenization. We measure the average cosine similarity among expert outputs within each layer on the STS-B validation set for two representative MoE models: DeepSeekMoE-16B[2] (Dai et al., 2024) and OLMoE-1B-7B (Muennighoff et al., 2024a). As shown in the left part of Figure 2, the lower and upper layers of these models exhibit higher expert similarity, while the middle layers show lower similarity. This indicates that expert homogenization is highly layer-dependent, occurring predominantly in the lower and upper layers, while being weak or absent in the middle layers.

Inspiration. This motivates us to **allocate fewer experts to highly redundant layers and redistribute capacity to more diverse ones**, thus enhancing representational efficiency.

Observation 2: Token-level sparsity. We further analyze attention distributions across tokens. Given the normalized attention score matrix, we set the threshold to one-tenth of the maximum value in each row and measure the proportion of entries below this threshold. As shown in the right part of Figure 2, nearly 90% of the attention weights fall below this threshold, revealing pronounced sparsity and suggesting that only a small subset of tokens dominate the information flow.

Inspiration. This motivates us to **allocate more experts to high-attention tokens and fewer to low-impact ones**, thereby better exploiting model capacity and improving embedding quality.

4 Methodology

Our empirical findings suggest that assigning an identical number of experts uniformly across all layers and tokens can be suboptimal for effectively utilizing model capacity. To address this limitation, we propose an adaptive expert allocation strategy that operates at both the layer and token

[2]DeepSeekMoE-16B uses a dense FFN in the first Transformer layer, while all other layers adopt MoE layers.

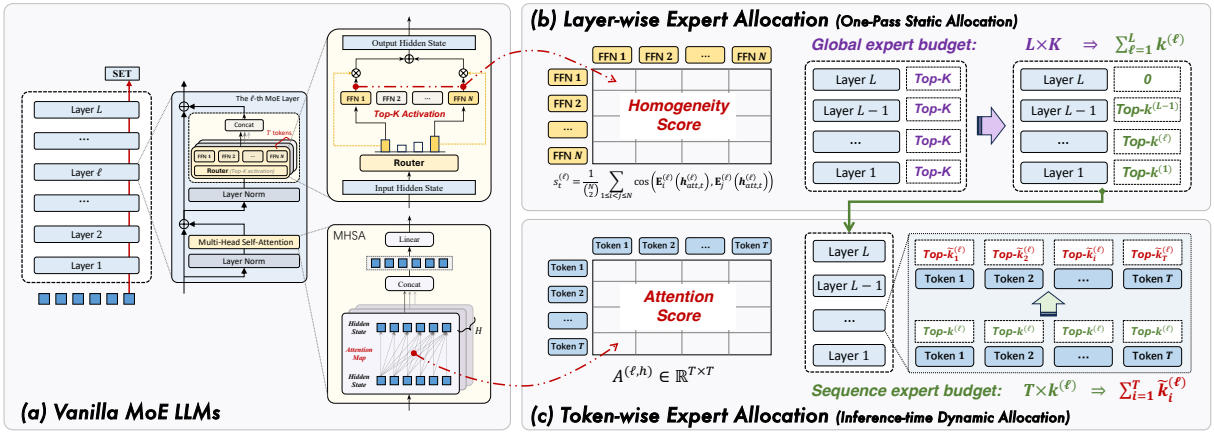

Figure 3: Illustration of the Adaptive Expert Allocation framework.

levels, aiming to enhance the quality of hidden state representations in MoE models.

Specifically, our method first computes a per-layer expert homogeneity score on a calibration dataset to allocate an expert budget to each layer of the LLM. This process only needs to be performed once in advance to determine the expert allocation for each layer. Then, during inference, we adaptively assign a number of experts to each token based on the layer-wise expert budget and token importance. The overall framework is illustrated in Figure 3. Notably, the total expert budget is kept fixed, and the strategy focuses solely on expert allocation.

4.1 Layer-Wise Expert Allocation

We design a layer-wise expert allocation scheme that allocates varying numbers of experts across layers according to their degrees of expert homogeneity. Intuitively, when experts within a layer exhibit high homogeneity, meaning that they produce highly similar outputs, fewer experts are required for that layer. Notably, this allocation process requires only a single additional computation and can be reused thereafter without extra cost. The layer-wise expert allocation module consists of three parts: homogeneity score computation, expert allocation, and early-exit strategy.

Homogeneity Score Computation We first introduce a homogeneity score to quantify the degree of similarity among experts within each layer, computed as the average pairwise similarity between the outputs of all experts in that layer. Specifically, we define the homogeneity score $s_t^{(\ell)}$ of the t-th token in layer ℓ as follows:

$$s_t^{(\ell)} = \frac{1}{\binom{N}{2}} \sum_{1 \le i < j \le N} \cos\big(\mathbf{E}_i^{(\ell)}(\mathbf{h}_{\text{att},t}^{(\ell)}), \mathbf{E}_j^{(\ell)}(\mathbf{h}_{\text{att},t}^{(\ell)})\big).$$

In practice, we use a calibration dataset and compute the average score of the final token across the dataset to obtain the final layer-wise homogeneity score. Notably, we use the validation set of STS-B as the calibration set, and only a single precomputation is required to determine the homogeneity scores for all layers. A larger $s^{(\ell)}$ indicates higher homogeneity, whereas a smaller value indicates greater diversity.

Expert Allocation Then, we map the sequence of homogeneity scores $\mathbf{s} = (s^{(1)}, \ldots, s^{(L)})$ to an expert allocation sequence $\mathbf{k} = (k^{(1)}, \ldots, k^{(L)})$, where each $k^{(\ell)}$ represents the number of experts assigned to the ℓ-th layer. Concretely, for the ℓ-th layer we compute

$$\widehat{k}^{(\ell)} = K_{\text{total}} \cdot \frac{(1 - s^{(\ell)})^{\alpha}}{\sum_{j=1}^{L} (1 - s^{(j)})^{\alpha}},$$

$$k^{(\ell)} = \text{round}\Big(\widehat{k}^{(\ell)}\Big)$$

where L is the total number of layers, K denotes the original number of activated experts per layer, $K_{\text{total}} = K \times L$ denotes the global expert budget, and α is a scaling parameter that controls the relative contrast across layers. Here, round$(\cdot)$ denotes rounding to the nearest integer.

Since rounding may cause the summed result to slightly deviate from the original total budget, i.e., $\sum_{\ell=1}^{L} k^{(\ell)} \neq K_{\text{total}}$, we introduce a correction step to ensure consistency. Let $S = \sum_{\ell=1}^{L} k^{(\ell)}$ and $\delta^{(\ell)} = \widehat{k}^{(\ell)} - k^{(\ell)}$ be the residual for layer ℓ.

To ensure budget consistency, we iteratively adjust the layer-wise allocations until the total allocation equals the predefined global expert budget, as follows. If $S < K_{\text{total}}$, we increment the expert count of the layer with the largest residual $\delta^{(\ell)}$, i.e.,

$$k^{(\ell)} \leftarrow k^{(\ell)} + 1, \quad \ell = \arg\max_{\ell} \delta^{(\ell)}.$$

Conversely, if $S > K_{\text{total}}$, we decrement the expert count of the layer with the smallest residual:

$$k^{(\ell)} \leftarrow k^{(\ell)} - 1, \quad \ell = \arg\min_{\ell} \delta^{(\ell)}.$$

This procedure guarantees that the final allocation satisfies $\sum_{\ell=1}^{L} k^{(\ell)} = K_{\text{total}}$, while assigning fewer experts to layers with high redundancy (large $s^{(\ell)}$) and more experts to layers with high diversity (small $s^{(\ell)}$).

Early-Exit and Reallocation Strategy Prior work indicates that the final-layer representations of LLMs are primarily used for next-token prediction rather than general semantic encoding (Lei et al., 2024; Li and Li, 2024; Liu et al., 2024b), often resulting in inferior embedding quality. To mitigate this issue, we introduce an *early-exit strategy* that extracts embeddings from the m-th layer. Since layers beyond the m-th layer are no longer utilized, we reallocate their expert budget to earlier layers to improve embedding quality.

4.2 Token-Wise Expert Allocation

We further design a token-wise strategy that, during inference, adaptively distributes experts across tokens according to their relative importance, as reflected by attention scores. The intuition is that tokens receiving higher attention scores should be assigned more experts, since they contribute more substantially to the hidden representations.

Formally, given the input hidden states $\mathbf{H}^{(\ell)} = [\mathbf{h}_1^{(\ell)}, \ldots, \mathbf{h}_T^{(\ell)}]$ at the ℓ-th layer, we compute the attention score matrix of the h-th head, denoted as $\mathbf{A}^{(\ell,h)} \in \mathbb{R}^{T \times T}$, where rows correspond to query tokens and columns correspond to key tokens. For the i-th token, we define its attention strength as

$$a_i^{(\ell)} = \max_h \max_j \mathbf{A}_{j,i}^{(\ell,h)},$$

which measures the strongest attention signal that token i receives from any other token across all attention heads. This operation emphasizes the most influential interaction instead of diluting it through averaging, thereby enabling more accurate identification of tokens that play a dominant role in the information flow.

After computing the attention strength $a_i^{(\ell)}$ for each token, we normalize them over the sequence of length T to obtain their relative importance:

$$p_i^{(\ell)} = \frac{a_i^{(\ell)}}{\sum_{j=1}^{T} a_j^{(\ell)}}.$$

Based on the layer-wise allocation described above, each layer ℓ is assigned an expert budget $k^{(\ell)}$, which yields a total expert budget for the sequence of $B^{(\ell)} = k^{(\ell)} \cdot T$. This total budget is then proportionally distributed among tokens according to their normalized attention scores:

$$\tilde{k}_i^{(\ell)} = \left\lfloor B^{(\ell)} \cdot p_i^{(\ell)} \right\rfloor.$$

Here, $\lfloor \cdot \rfloor$ denotes the floor operation, which rounds a value down to the nearest integer. This procedure ensures that tokens receiving higher attention weights are allocated more experts, while less informative tokens are assigned fewer experts.

5 Experiments

5.1 Datasets and Experimental Settings

We conduct evaluation on seven benchmark semantic textual similarity (STS) datasets, covering STS 2012-2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016), STS-B (Cer et al., 2017), and SICK-R (Marelli et al., 2014). In these corpora, each pair of sentences is labeled with a human-assigned semantic similarity score ranging from 0 to 5. To obtain model predictions, we compute cosine similarity between sentence embeddings and assess performance using Spearmans rank correlation, which quantifies the consistency of ranking between predicted scores and gold annotations.

We perform grid search on the STS-B validation set to determine the exit layer m and the scaling factor α. For all three models, we select the penultimate layer as the exit layer. We select scaling factors of 9 for DeepSeekMoE-16B, 10 for Qwen1.5-MoE-A2.7B (Team, 2024), and 6 for OLMoE-1B-7B. Since the inter-expert similarity across layers in Qwen1.5-MoE-A2.7B is generally low, we amplify the differences among layers by using a shifted transformation, i.e., replacing $1 - s^{(\ell)}$ with $0.1 - s^{(\ell)}$. The expert allocation results across layers for the three models are shown in Figure 6.

Baseline	STS12	STS13	STS14	STS15	STS16	SICK-R	STS-B	Avg.
HS	52.13	69.33	54.75	57.99	68.72	63.83	65.27	61.72
HS + Echo	53.17	75.82	66.22	74.99	76.19	69.79	69.59	69.39 ↑ 7.67
HS + TP	59.32	77.35	64.85	71.40	74.80	65.98	70.29	69.14 ↑ 7.42
HS + AEA (**Ours**)	65.38	80.72	68.77	**77.04**	**77.45**	69.68	**73.66**	**73.24** ↑ 11.52
MoEE (concat)	66.83	77.53	63.55	64.61	71.04	66.39	68.57	68.36 ↑ 6.64
MoEE (concat) + Echo	56.28	69.01	56.53	61.49	67.55	66.10	64.18	63.02 ↑ 1.30
MoEE (concat) + TP	67.06	78.08	65.51	67.10	72.34	67.20	70.32	69.66 ↑ 7.94
MoEE (concat) + AEA (**Ours**)	67.48	78.83	65.82	67.53	72.52	66.95	70.37	69.93 ↑ 8.21
MoEE (sum)	66.68	80.95	67.94	66.84	73.56	69.64	72.58	71.17 ↑ 9.45
MoEE (sum) + Echo	56.08	68.89	56.35	61.46	67.57	66.05	64.06	62.92 ↑ 1.20
MoEE (sum) + TP	66.63	80.65	62.73	68.71	71.99	70.74	67.55	69.86 ↑ 8.14
MoEE (sum) + AEA (**Ours**)	**70.38**	**81.33**	**69.81**	69.93	74.81	**70.75**	73.56	72.94 ↑ 11.22

Table 1: Results on the STS task using DeepSeekMoE-16B.

Baseline	STS12	STS13	STS14	STS15	STS16	SICK-R	STS-B	Avg.
HS	55.01	77.49	63.64	73.57	73.62	66.94	67.64	68.27
HS + Echo	55.93	77.60	66.27	**76.92**	**75.56**	69.29	67.41	69.85 ↑ 1.58
HS + TP	57.66	77.86	64.94	73.78	74.35	68.12	69.65	69.48 ↑ 1.21
HS + AEA (**Ours**)	**72.10**	78.91	69.20	72.34	70.69	**76.37**	72.78	**73.20** ↑ 4.93
MoEE (concat)	64.28	77.27	63.78	66.97	71.31	67.11	69.28	68.57 ↑ 0.30
MoEE (concat) + Echo	54.95	56.84	49.93	51.43	64.76	66.71	60.65	57.89 ↓ 10.38
MoEE (concat) + TP	66.25	77.54	66.00	68.54	72.36	68.30	71.65	70.09 ↑ 1.82
MoEE (concat) + AEA (**Ours**)	64.53	77.90	63.60	67.78	70.46	65.88	67.80	68.28 ↑ 0.01
MoEE (sum)	65.20	**81.87**	**70.42**	71.72	74.78	70.77	**74.49**	72.75 ↑ 4.48
MoEE (sum) + Echo	54.48	56.58	49.63	51.13	64.55	66.60	60.27	57.60 ↓ 10.67
MoEE (sum) + TP	66.55	80.27	68.94	70.24	74.02	71.50	71.65	71.88 ↑ 3.61
MoEE (sum) + AEA (**Ours**)	67.59	80.34	67.50	70.48	71.65	69.79	71.11	71.21 ↑ 2.94

Table 2: Results on the STS task using Qwen1.5-MoE-A2.7B.

5.2 Baselines

We integrate our approach with several baseline methods to demonstrate its effectiveness. **MoEE** (Li and Zhou, 2025), the only existing method specifically designed to extract embeddings from MoE models, integrates routing weights with sentence embedding to improve performance. Additionally, we compare our method with two plug-and-play techniques originally proposed for LLMs, both of which aim to mitigate the limitations of unidirectional attention in LLMs. **Echo** (Springer et al., 2025) repeats the text twice, while **TP** (Fu et al., 2025) prepends the sentence embedding token. Except for **Echo**, which obtains sentence embeddings by applying average pooling over the second occurrence of the text, all other methods use the **PromptEOL** (Jiang et al., 2024) prompt to extract sentence embeddings. Details of the prompt are presented in Appendix A.

5.3 Main Results

The experimental results on the STS tasks across three MoE LLMs are presented in Table 1, 2, and 3. Among all methods applied to HS, our approach achieves the most significant improvements over the hidden state baseline, consistently outperforming all other baselines, including MoEE, Echo, and TP. This demonstrates that our method can effectively outperform baselines. Notably, the most substantial gain is observed on DeepSeekMoE-16B, with an improvement of 11.52.

Interestingly, HS + AEA even outperforms MoEE + AEA, despite the latter explicitly incorporating routing weights. This phenomenon can be attributed to our expert allocation strategy, which implicitly embeds routing information into the hidden states. Consequently, it suggests that explicitly extracting routing weights as a supplement to hidden states may be unnecessary, thereby simplifying the embedding extraction process.

Moreover, the consistent improvements across different MoE LLMs further demonstrate the effectiveness and generalizability of our method.

Baseline	STS12	STS13	STS14	STS15	STS16	SICK-R	STS-B	Avg.
HS	65.60	81.93	69.44	77.60	77.31	66.68	71.59	72.88
HS + Echo	56.75	74.92	65.51	77.72	75.83	69.95	71.14	70.26 ↓ 2.62
HS + TP	64.57	81.88	68.99	77.42	**78.21**	68.78	73.18	73.29 ↑ 0.41
HS + AEA (**Ours**)	66.64	82.37	72.44	**80.61**	78.13	**70.99**	**76.89**	**75.44** ↑ 2.56
MoEE (concat)	67.42	80.11	68.52	69.08	73.31	67.75	70.63	70.97 ↓ 1.91
MoEE (concat) + Echo	57.66	70.93	59.58	67.01	70.54	66.52	68.28	65.79 ↓ 7.09
MoEE (concat) + TP	67.86	80.12	68.68	70.24	74.04	68.74	71.36	71.58 ↓ 1.30
MoEE (concat) + AEA (**Ours**)	67.24	81.22	69.79	71.62	74.04	69.11	72.67	72.24 ↓ 0.64
MoEE (sum)	68.08	**83.70**	**72.83**	71.90	75.71	70.22	74.00	73.78 ↑ 0.90
MoEE (sum) + Echo	57.46	75.38	59.31	66.80	70.40	70.24	70.38	67.14 ↓ 5.74
MoEE (sum) + TP	**69.23**	77.32	71.13	69.53	73.94	69.60	70.90	71.66 ↓ 1.22
MoEE (sum) + AEA (**Ours**)	66.11	81.79	71.77	80.19	77.87	70.71	76.50	74.99 ↑ 2.11

Table 3: Results on the STS task using OLMoE-1B-7B.

Dataset	STS12	STS13	STS14	STS15	STS16	SICK-R	STS-B	Avg.
`OLMoE-1B-7B`								
HS	65.60	81.93	69.44	77.60	77.31	66.68	71.59	72.88
HS + token-wise	**67.58**	81.85	71.27	78.97	**78.49**	68.47	74.34	74.42 ↑ 1.54
HS + layer-wise	66.23	81.88	70.78	79.72	76.72	70.26	75.19	74.40 ↑ 1.52
HS + AEA (**Ours**)	66.64	**82.37**	**72.44**	**80.61**	78.13	**70.99**	**76.89**	**75.44** ↑ 2.56
`DeepSeekMoE-16B`								
HS	52.13	69.33	54.75	57.99	68.72	63.83	65.27	61.72
HS + token-wise	55.34	67.49	55.18	61.03	70.26	63.16	65.03	62.50 ↑ 0.78
HS + layer-wise	64.94	81.67	68.66	76.69	76.80	69.60	73.10	73.07 ↑ 11.25
HS + AEA (**Ours**)	**65.38**	**80.72**	**68.77**	**77.04**	**77.45**	**69.68**	**73.66**	**73.24** ↑ 11.52
`Qwen1.5-MoE-A2.7B`								
HS	55.01	77.49	63.64	**73.57**	**73.62**	66.94	67.64	68.27
HS + token-wise	63.43	75.92	65.14	69.34	67.57	74.88	70.55	69.55 ↑ 1.28
HS + layer-wise	**72.87**	**79.15**	65.63	72.38	70.60	76.07	**73.17**	72.84 ↑ 4.57
HS + AEA (**Ours**)	72.10	78.91	**69.20**	72.34	70.69	**76.37**	72.78	**73.20** ↑ 4.93

Table 4: Ablation study of our proposed method on the STS task.

5.4 Ablation Study

We compare the effects of applying only token-wise allocation and only layer-wise allocation on embedding quality. As shown in Table 4, using token-wise allocation or layer-wise allocation alone can both improve embedding quality, with layer-wise allocation yielding larger performance gains in particular. This demonstrates the effectiveness of both allocation strategies. Notably, combining the two leads to more substantial performance improvements.

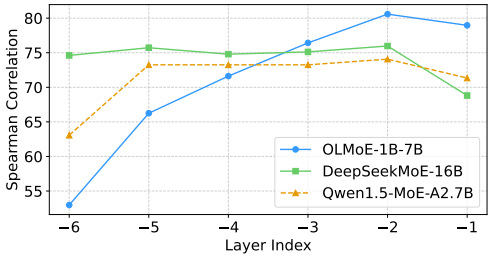

Figure 4: Effect of exit layer m on embeddings.

5.5 Effects of Token-Wise Allocation

We further examine how attention strength should be aggregated when assigning experts. There are two dimensions to consider: (i) whether to average or take the maximum over all tokens that attend to a given token, and (ii) whether to average or take the maximum across attention heads. Combining these choices results in four variants in total. As shown in Table 5, the variant that takes the maximum across both tokens and heads achieves the best embedding quality. This indicates that the strongest attention signal is more informative than aggregated weaker ones, making it a more effective criterion for guiding expert allocation.

Dataset	STS12	STS13	STS14	STS15	STS16	SICK-R	STS-B	Avg.
HS	65.60	81.93	69.44	77.60	77.31	66.68	71.59	72.88
HS + T-mean + H-mean	39.40	70.00	57.85	71.62	65.85	58.89	64.12	61.10 ↓ 11.78
HS + T-mean + H-max	61.20	79.41	68.39	78.46	74.05	68.89	74.46	72.12 ↓ 0.76
HS + T-max + H-mean	64.98	81.49	70.89	80.01	77.02	69.97	75.86	74.32 ↑ 1.44
HS + T-max + H-max	**66.64**	**82.37**	**72.44**	**80.61**	**78.13**	**70.99**	**76.89**	**75.44** ↑ 2.56

Table 5: Effects of aggregation strategies for token-wise expert allocation on OLMoE-1B-7B. T and H denote aggregation over tokens and heads, respectively, where "mean" and "max" indicate average and max pooling.

Dataset	STS12	STS13	STS14	STS15	STS16	SICK-R	STS-B	Avg.
HS	65.6	81.93	69.44	77.60	77.31	66.68	71.59	72.88
HS + token-wise (inverse)	39.88	59.79	48.32	56.17	44.16	53.98	46.76	49.87 ↓ 23.01
HS + token-wise	**67.58**	**81.85**	**71.27**	**78.97**	**78.49**	**68.47**	**74.34**	**74.42** ↑ 1.54
HS + layer-wise$_{final}$ (inverse)	**67.97**	82.07	**69.21**	**77.69**	**76.66**	66.11	71.26	72.75 ↓ 0.13
HS + layer-wise$_{final}$	65.21	**82.21**	**69.21**	77.50	76.60	**67.39**	**72.67**	**73.13** ↑ 0.25

Table 6: Results under different allocation strategies using OLMoE-1B-7B.

5.6 Effects of Exit Layers

We further evaluate the impact of the exit layer m using the STS-B validation set. To ensure a consistent comparison across models with different numbers of layers, we adopt negative indexing, where -1 represents the final layer, -2 represents the penultimate layer, and so on.

As shown in Figure 4, embeddings extracted from the penultimate layer consistently achieve the best performance across all three models, which aligns with previous findings. Compared to the other two models, OLMoE-1B-7B is more sensitive to variations in early-exit layers.

5.7 Effect of Scaling Factor

We investigate the effect of the scaling factor α using the validation set of STS-B. As shown in Figure 5, the embedding quality first improves and then degrades as α increases. This is because a larger α amplifies the inter-layer differences in the number of experts, enabling a more effective allocation of experts across layers, thereby improving embedding quality. However, when the inter-layer disparity becomes excessively large, highly redundant layers may be assigned too few experts, which is insufficient to properly process the input.

5.8 Validation of Allocation Assumptions

We further validate our hypothesis that layers with higher redundancy and tokens with lower attention scores should be allocated fewer experts. Notably, to avoid the effect of early-exit mechanisms on expert allocation, *we use the embeddings from the final layer as the output in the layer-wise setting*.

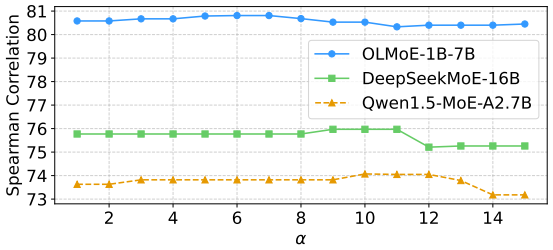

Figure 5: Effect of the scaling factor α.

Prompt	Baseline	STS-B
Prompt	HS	43.27
Prompt	HS + AEA (**Ours**)	**48.72** ↑ 5.45
Pretended CoT	HS	74.68
Pretended CoT	HS + AEA (**Ours**)	**78.43** ↑ 3.75
Knowledge	HS	74.36
Knowledge	HS + AEA (**Ours**)	**76.95** ↑ 2.59

Table 7: Results under different prompt settings using OLMoE-1B-7B.

As shown in Table 6, when the assumption is reversed, assigning more experts to highly redundant layers and to tokens with lower attention scores often leads to performance degradation, especially in the token-wise setting. This further confirms the effectiveness of our assumption.

5.9 Generalization across Different Prompts

We further explore the generalization of our method across different prompts on the test set of the STS-B dataset. Details of the prompt are presented in Appendix A.

As shown in Table 7, our method achieves consistent improvements across all three prompts, demonstrating its strong generalization ability.

6 Conclusion

In this work, we present an **Adaptive Expert Allocation** framework for training-free extraction of high-quality sentence embeddings from MoE LLMs. Our analysis reveals that different layers exhibit varying degrees of expert homogeneity, and different tokens contribute unequally to the final representation. Based on these insights, AEA adaptively redistributes expert capacity at the layer and token levels, achieving consistent improvements on the STS tasks over existing baselines. Moreover, AEA incurs no additional time overhead and can be integrated into existing MoE-based LLMs as a plug-and-play component.

Limitations

Our work has two main limitations. Firstly, our method requires layer-wise access to expert outputs and attention maps to perform adaptive allocation, which may not be feasible for certain closed-source MoE-based LLMs. Secondly, our evaluation is limited to English-only benchmarks, and the applicability of our method to multilingual datasets remains to be explored.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by the JiangSu Natural Science Foundation under Grant No. BK20251989; the National Natural Science Foundation of China under Grants Nos. 62172208, 62441225, 61972192; the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No.JYB2025XDXM118). This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015*, pages 252–263.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014*, pages 81–91.

Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016*, pages 497–511.

Eneko Agirre, Daniel M. Cer, Mona T. Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2012*, pages 385–393.

Eneko Agirre, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013*, pages 32–43.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. *CoRR*, abs/1708.00055.

Sachin Chanchani and Ruihong Huang. 2023. Composition-contrastive learning for sentence embeddings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15836–15848.

Zifeng Cheng, Zhonghui Wang, Yuchen Fu, Zhiwei Jiang, Yafeng Yin, Cong Wang, and Qing Gu. 2025. Contrastive prompting enhances sentence embeddings in llms through inference-time steering. *arXiv preprint arXiv:2505.12831*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Yuchen Fu, Zifeng Cheng, Zhiwei Jiang, Zhonghui Wang, Yafeng Yin, Zhengliang Li, and Qing Gu.

2025. Token prepending: A training-free approach for eliciting better sentence embeddings from llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025*, pages 3168–3181.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2024. Scaling sentence embeddings with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3182–3196.

Ting Jiang, Jian Jiao, Shaohan Huang, Zihan Zhang, Deqing Wang, Fuzhen Zhuang, Furu Wei, Haizhen Huang, Denvy Deng, and Qi Zhang. 2022. Promptbert: Improving bert sentence embeddings with prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8826–8837.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Yibin Lei, Di Wu, Tianyi Zhou, Tao Shen, Yu Cao, Chongyang Tao, and Andrew Yates. 2024. Meta-task prompting elicits embeddings from large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024*, pages 10141–10157.

Junxian Li, Beining Xu, Simin Chen, Jiatong Li, Jingdi Lei, Haodong Zhao, and Di Zhang. 2025. Iag: Input-aware backdoor attack on vlm-based visual grounding. *arXiv preprint arXiv:2508.09456*.

Pengyu Li, Lingling Zhang, Zhitao Gao, Yanrui Wu, Yuxuan Dong, Huan Liu, Bifan Wei, and Jun Liu. 2026. $\mathbf{AGT}^{AO}$: Robust and stabilized llm unlearning via adversarial gating training with adaptive orthogonality. *arXiv preprint arXiv:2602.01703*.

Xianming Li and Jing Li. 2024. Bellm: Backward dependency enhanced large language model for sentence embeddings. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 792–804.

Ziyue Li and Tianyi Zhou. 2025. Your mixture-of-experts LLM is secretly an embedding model for free. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*.

Tian Yu Liu, Matthew Trager, Alessandro Achille, Pramuditha Perera, Luca Zancato, and Stefano Soatto. 2024a. Meaning representations from trajectories in autoregressive models. In *The Twelfth International Conference on Learning Representations, ICLR 2024*.

Zhu Liu, Cunliang Kong, Ying Liu, and Maosong Sun. 2024b. Fantastic semantics and where to find them: Investigating which layers of generative llms reflect lexical semantics. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 14551–14558.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014*, pages 216–223.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. 2024a. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024b. Generative representational instruction tuning. *arXiv preprint arXiv:2402.09906*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. 2025. Repetition improves language model embeddings. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121.

Qwen Team. 2024. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters".

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024*, pages 11897–11916.

Shufan Yang, Zifeng Cheng, Zhiwei Jiang, Yafeng Yin, Cong Wang, Shiping Ge, Yuchen Fu, and Qing Gu. 2026. Regionmarker: A region-triggered semantic watermarking framework for embedding-as-a-service copyright protection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 34313–34321.

Bowen Zhang, Kehua Chang, and Chunping Li. 2024. Simple techniques for enhancing sentence embeddings in generative language models. *CoRR*, abs/2404.03921.

Hao Zhang, Mengsi Lyu, Zhuo Chen, Xingrun Xing, Yulong Ao, and Yonghua Lin. 2025. Pdtrim: Targeted pruning for prefill-decode disaggregation in inference. *arXiv preprint arXiv:2509.04467*.

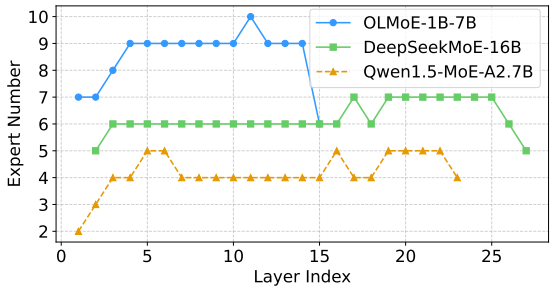

Figure 6: Layer-wise visualization of expert allocation counts for three MoE models.

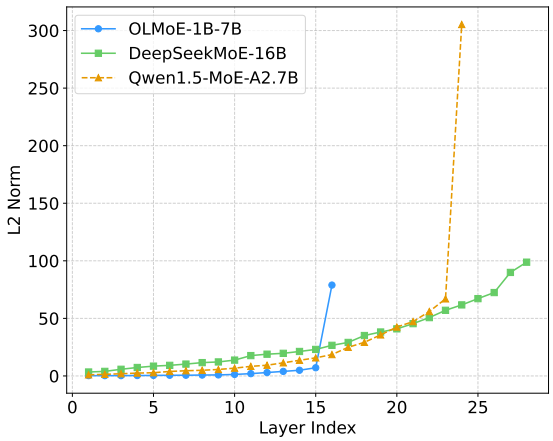

Figure 7: L2 norm results of embeddings across different layers.

A Baselines

Detailed prompt settings for the baseline models are summarized as follows:

PromptEOL: This sentence: "[TEXT]" means in one word: "
Prompt: In one word, describe the style of the following sentence -"[TEXT]": "
Pretended CoT: After thinking step by step, this sentence: "[TEXT]" means in one word: "
Knowledge: The essence of a sentence is often captured by its main subjects and actions, while descriptive terms provide additional but less central details. With this in mind , this sentence: "[TEXT]" means in one word: "

B Analysis of the Exit Layer

To further justify our choice of the penultimate layer as the exit layer, we conduct additional analyses by computing the L2 norm of embeddings extracted from each layer across different models.

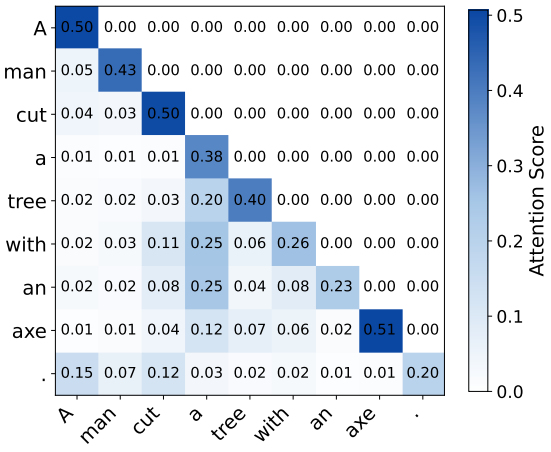

Figure 8: Visualization of the attention score matrix for the sentence *"A man cut a tree with an axe."* from the STS-B dataset, computed using the OLMoE-1B-7B.

As illustrated in Figure 7, the final-layer embeddings exhibit the largest norms, with a sharp surge observed particularly in OLMoE-1B-7B and Qwen1.5-MoE-A2.7B. This suggests that extracting embeddings from the final layer may excessively amplify its activations, thereby diluting representations from preceding layers and degrading embedding quality. These results further confirm the rationale for selecting the penultimate layer as the exit point.

C Analysis of Attention Sparsity Across Tokens

To further illustrate the sparsity of the token-level attention score matrix, we sample a sentence from the STS-B dataset: *"A man cut a tree with an axe."* We compute its attention score matrix using the OLMoE-1B-7B model, and Figure 8 visualizes part of the attention weights from the first attention head. A clear sparsity pattern can be observed, confirming the rationale behind our token-wise expert allocation strategy.

Baseline	Time
OLMoE-1B-7B	
HS	1.00 ×
HS + AEA (**Ours**)	0.99 ×
MoEE (concat)	1.00 ×
MoEE (concat) + AEA (**Ours**)	1.00 ×
MoEE (sum)	1.00 ×
MoEE (sum) + AEA (**Ours**)	0.94 ×

Table 8: Inference time overhead under the PromptEOL setting.

Baseline	CLF	Clust.	PairCLF	Rerank	STS	Summ.	Avg.
			DeepSeekMoE-16B				
HS	58.20	25.26	67.71	49.53	61.72	25.00	47.90
HS + Echo	55.70 ↓ 2.50	**34.63** ↑ 9.37	64.51 ↓ 3.20	53.26 ↑ 3.73	69.39 ↑ 7.67	27.65 ↑ 2.65	50.86 ↑ 2.96
HS + TP	59.75 ↑ 1.55	33.03 ↑ 7.77	59.66 ↓ 8.05	48.16 ↓ 1.37	69.14 ↑ 7.42	27.08 ↑ 2.08	49.47 ↑ 1.57
HS + AEA (**Ours**)	**59.81** ↑ 1.61	31.38 ↑ 6.12	**72.16** ↑ 4.45	**57.45** ↑ 7.92	**73.24** ↑ 11.52	**28.10** ↑ 3.10	**53.69** ↑ 5.79
MoEE (concat)	52.58	25.52	69.83	53.30	68.36	29.46	49.84
MoEE (concat) + Echo	48.17 ↓ 4.41	24.54 ↓ 0.98	61.87 ↓ 7.96	51.64 ↓ 1.66	63.02 ↓ 5.34	27.93 ↓ 1.53	46.20 ↓ 3.64
MoEE (concat) + TP	53.41 ↑ 0.83	**29.86** ↑ 4.34	62.79 ↓ 7.04	**55.62** ↑ 2.32	69.66 ↑ 1.30	28.85 ↓ 0.61	50.03 ↑ 0.19
MoEE (concat) + AEA (**Ours**)	**53.42** ↑ 0.84	26.13 ↑ 0.61	**72.02** ↑ 2.19	55.59 ↑ 2.29	**69.93** ↑ 1.57	**30.11** ↑ 0.65	**51.20** ↑ 1.36
MoEE (sum)	53.57	33.00	72.03	54.36	71.17	29.22	52.23
MoEE (sum) + Echo	50.70 ↓ 2.87	**35.06** ↑ 2.06	70.53 ↓ 1.50	56.83 ↑ 2.47	62.92 ↓ 8.25	26.05 ↓ 3.17	50.35 ↓ 1.88
MoEE (sum) + TP	**54.43** ↑ 0.86	34.39 ↑ 1.39	**73.31** ↑ 1.28	57.10 ↑ 2.74	69.86 ↓ 1.31	**32.04** ↓ 2.82	53.52 ↑ 1.29
MoEE (sum) + AEA (**Ours**)	53.62 ↑ 0.05	34.52 ↑ 1.52	72.78 ↑ 0.75	**57.86** ↑ 3.50	**72.94** ↑ 1.77	29.53 ↑ 0.31	**53.54** ↑ 1.31

Table 9: MTEB results under the PromptEOL setting.

Baseline	Twenty Newsgroups Clustering	Medrxiv ClusteringS2S	Biorxiv ClusteringS2S	Biorxivv ClusteringP2P	Medrxiv ClusteringP2P	StackExchange ClusteringP2P	Avg.
			DeepSeekMoE-16B				
HS	28.04	22.47	20.04	15.52	18.40	25.47	21.66
HS + AEA (**Ours**)	36.87	25.88	29.81	27.33	24.13	25.79	28.30
HS + Echo	**40.92**	**28.34**	27.38	29.68	28.39	30.73	30.91
HS + Echo + AEA (**Ours**)	36.87	25.88	30.43	**34.21**	**30.62**	**32.19**	**32.65**
HS + TP	38.18	27.89	23.47	18.88	20.53	25.78	25.79
HS + TP + AEA (**Ours**)	36.87	25.88	**31.38**	30.04	25.79	26.46	30.30

Table 10: Clustering results, including combinations of our method with Echo and TP.

D Computational Overhead Analysis

We analyze the computational overhead introduced by the layer-wise and token-wise score computation.

The layer-wise score only needs to be computed once per MoE LLM and can be reused across all downstream tasks. We perform the computation on the STS-B validation set, which contains 1,500 samples, and the entire process only takes 1 to 2 minutes on a single V100 GPU. Once the layer-wise expert allocation is determined, it can be directly applied without introducing any additional time or memory overhead.

The token-wise score needs to be computed for each sample. In practice, this introduces only a minimal additional cost, resulting in about a 1.5% increase compared to the baseline. We evaluate this overhead on the STS-B dataset using the OLMoE-1B-7B model. As shown in Table 8, integrating AEA leads to negligible time overhead. This is because, although token-wise scoring incurs extra computation, the early-exit strategy offsets this cost by reducing part of the multi-head attention computation, yielding no observable increase in overall inference time. The memory overhead is also negligible, as token-wise scoring only stores a single vector of length T (where T denotes the number of tokens in the input sample) per layer and does not cache any attention matrices.

E MTEB Results

We further conduct a comprehensive evaluation on the MTEB benchmark (Muennighoff et al., 2022), which encompasses a broad range of natural language processing tasks, including Classification, Clustering, Pair Classification, Re-ranking, Retrieval, Summarization, and STS. Following the standardized evaluation framework of MTEB, we adopt task-specific metrics such as Accuracy, V-Measure, Average Precision, Mean Average Precision, nDCG, and Spearmans correlation. We also present detailed evaluation results of task types, including clustering (Table 11), classification (Table 12), pair classification (Table 13), and re-ranking (Table 14) tasks.

As shown in Table 9, our method also achieves the best average performance across six tasks. Notably, under the HS and MoEE (concat) settings, our method surpasses the strong baseline TP by 4.22 and 1.17 percentage points, respectively. Moreover, our method achieves improvements in all 18 cases, while Echo and TP obtain gains in

only 6 and 12 cases, respectively. This demonstrates the generalizability of our approach. Notably, HS + Echo achieves better performance than HS + AEA on the clustering datasets. This may be because clustering datasets benefit more from bidirectional attention, which enhances textual understanding.

To this end, we further conduct experiments on additional clustering datasets to explore whether AEA can be combined with Echo and TP to improve performance. As shown in Table 10, HS + Echo + AEA achieves the best average performance on clustering datasets, further demonstrating that our method is plug-and-play. This indicates that Echo and our method improve embedding quality from two different perspectives: Echo leverages bidirectional attention, whereas our approach focuses on high-attention tokens. These two perspectives are not mutually exclusive but rather complementary.

F Task Independence of Homogeneity Score Computation

We use a task-agnostic calibration dataset, namely the STS-B validation set consisting of 1,500 samples, to compute the homogeneity scores. We further use task-specific datasets as calibration sets to compute homogeneity scores for each task and obtain layer-wise expert allocation.

As shown in Table 15, the task-shared allocations achieve performance comparable to the task-specific allocation. We further observe that the expert distributions obtained from different tasks are largely consistent, as shown in Table 16. Notably, for the reranking task, the resulting expert allocation is exactly the same as the one computed on the STS-B validation set. This indicates that the calibration set is not dependent on a specific task and is general.

Baseline	Twenty Newsgroups Clustering	Medrxiv ClusteringS2S
DeepSeekMoE-16B		
HS	28.04	22.47
HS + Echo	**40.92**	28.34
HS + TP	38.18	27.89
HS + AEA (**Ours**)	36.87	25.88
MoEE (concat)	28.68	22.36
MoEE (concat) + Echo	26.10	22.99
MoEE (concat) + TP	34.67	25.05
MoEE (concat) + AEA (**Ours**)	28.75	23.50
MoEE (sum)	34.78	31.21
MoEE (sum) + Echo	38.05	32.06
MoEE (sum) + TP	37.13	31.65
MoEE (sum) + AEA (**Ours**)	36.88	**32.16**

Table 11: Clustering results.

Baseline	Tweet Sentiment Extraction	Emotion Classification	Toxic Conversations Classification
DeepSeekMoE-16B			
HS	60.10	49.06	65.45
HS + Echo	55.60	43.71	67.78
HS + TP	**61.54**	51.07	66.73
HS + AEA (**Ours**)	61.00	**51.85**	66.59
MoEE (concat)	59.13	42.81	55.80
MoEE (concat) + Echo	53.37	36.74	54.41
MoEE (concat) + TP	59.97	44.49	55.78
MoEE (concat) + AEA (**Ours**)	59.46	43.76	57.05
MoEE (sum)	56.60	39.06	65.05
MoEE (sum) + Echo	47.97	32.88	**71.24**
MoEE (sum) + TP	57.59	40.50	65.21
MoEE (sum) + AEA (**Ours**)	55.61	39.42	65.83

Table 12: Classification results.

Baseline	TwitterURLCorpus	TwitterSemEval2015
DeepSeekMoE-16B		
HS	74.67	60.75
HS + Echo	76.57	52.44
HS + TP	75.31	44.02
HS + AEA (**Ours**)	78.79	65.52
MoEE (concat)	77.97	61.69
MoEE (concat) + Echo	66.79	56.95
MoEE (concat) + TP	76.12	49.47
MoEE (concat) + AEA (**Ours**)	**80.21**	63.83
MoEE (sum)	79.19	64.87
MoEE (sum) + Echo	79.17	61.90
MoEE (sum) + TP	80.10	**67.32**
MoEE (sum) + AEA (**Ours**)	80.13	65.43

Table 13: Pair classification results.

Baseline	AskUbuntu DupQuestions	SciDocsRR	StackOverflow DupQuestions
DeepSeekMoE-16B			
HS	50.04	66.23	32.32
HS + Echo	52.18	67.26	40.33
HS + TP	45.84	68.85	29.80
HS + AEA (**Ours**)	**53.65**	78.52	40.17
MoEE (concat)	50.84	73.03	36.02
MoEE (concat) + Echo	46.16	71.50	37.26
MoEE (concat) + TP	51.95	74.09	40.82
MoEE (concat) + AEA (**Ours**)	51.46	76.87	38.44
MoEE (sum)	51.84	74.68	36.55
MoEE (sum) + Echo	50.93	78.36	**41.20**
MoEE (sum) + TP	53.26	77.99	40.04
MoEE (sum) + AEA (**Ours**)	53.59	**80.39**	39.61

Table 14: Re-ranking results.

Baseline	CLF	PairCLF	Rerank
DeepSeekMoE-16B			
HS + AEA (task-shared)	59.81	72.16	57.45
HS + AEA (task-specific)	59.78	72.08	57.45

Table 15: Results using task-shared and task-specific homogeneity scores.

Layer	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
DeepSeekMoE-16B																										
Task-shared	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	7	6	7	7	7	7	7	7	7	6	5
CLF	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	7	7	6	7	7	7	7	7	7	6	5
PairCLF	5	6	5	6	6	6	6	6	6	6	6	6	6	6	7	7	6	7	7	7	7	7	7	7	6	5
Rerank	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	7	6	7	7	7	7	7	7	7	6	5

Table 16: Expert allocation results using task-shared and task-specific homogeneity scores.