DEN2MOEE: RECONSTRUCTING DENSE LLMS INTO EXPERT-SPECIALIZED MIXTURE-OF-EXPERTS FOR EFFICIENT EMBEDDING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Most existing LLM-based embedding models in database semantic search systems rely on dense Transformer architectures. In contrast, Mixture-of-Experts (MoE) embedding models provide a cost-efficient scaling paradigm by activating only a subset of parameters. However, training large-scale MoE embedding models from scratch is prohibitively expensive. Although reconstructing dense LLM embeddings into MoE architectures offers a practical alternative, existing approaches often overlook expert specialization and routing characteristics specific to embedding tasks, resulting in homogeneous experts and inefficient performance recovery. To address these limitations, we propose Den2MoEE, a unified Dense-to-MoE reconstruction framework for embedding models that jointly promotes domain-aware expert specialization and routing-aware adaptation. Den2MoEE reconstructs dense embedding models through TokenScore-weighted neuron domain modeling, expert-diversity-driven Dense-to-MoE conversion, and an efficient two-phase recovery strategy. Furthermore, we exploit MoE router signals as complementary embedding representations, further enhancing the quality of the reconstructed embeddings without increasing activated parameters. Extensive experiments demonstrate that Den2MoEE reduces the number of activated parameters by 33%–55% compared with the original dense LLM embedding models, while consistently outperforming existing embedding baselines under comparable activated parameter budgets. Moreover, Den2MoEE delivers substantial system-level gains, achieving $1.35\times$–$1.8\times$ higher throughput with significantly lower inference latency.

1 INTRODUCTION

High-quality text embeddings are a core primitive in semantic search, retrieval-augmented generation, and in-database vector processing. Recent LLM-based embedding models achieve strong performance (Zhao et al., 2026; Lee et al., 2025; Li et al., 2023b; Zhang et al., 2024), but their dense Transformer architectures scale to billions of parameters (e.g., Qwen3-Embedding (Zhang et al., 2025), Jina-Embedding (Günther et al., 2025), and E5-Mistral (Wang et al., 2024b)), making inference costly. Mixture-of-Experts (MoE) (Jacobs et al., 1991; Cai et al., 2025; Dai et al., 2024) provides a cost-efficient scaling alternative by activating only a subset of parameters. Recent work, such as MoEE (Li and Zhou, 2025), studies MoE routing in embedding learning and finds that router weights encode semantic information complementary to MLP hidden states, suggesting that expert specialization plays a critical role in embedding quality. These findings motivate the exploration of MoE architectures as a cost-effective alternative for large-scale embedding models. Figure 1 shows the comparison between dense LLM embedding and MoE embedding.

However, while MoE-based large language models have demonstrated strong scaling capability, their large parameter footprint and routing overhead make them difficult to deploy in high-throughput and latency-sensitive embedding scenarios, where serving efficiency is often a primary constraint. This mismatch is particularly severe in embedding models, where routing directly affects representation quality, making routing design tightly coupled with semantic modeling rather than a purely computational concern. Training a dedicated MoE embedding model from scratch is costly, so recent work instead converts dense LLMs into MoE architectures (e.g., LLaMA-MoE

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

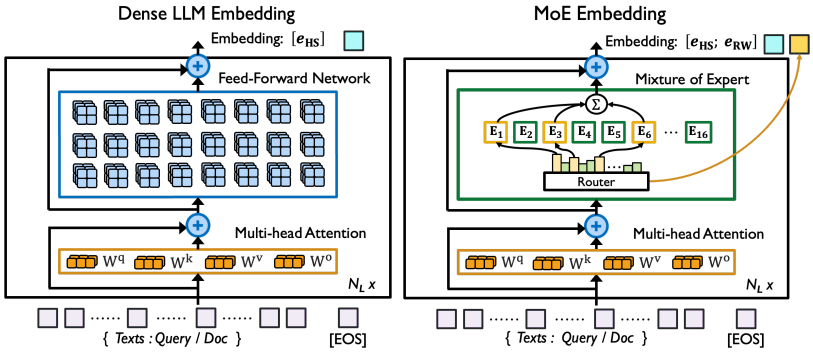


Figure 1: Comparison between dense and MoE embedding.

(Zhu et al., 2024)). Although effective in reducing training overhead, existing conversion strategies, such as FFN duplication (Komatsuzaki et al., 2023) and random neuron partitioning (Zhu et al., 2024), often produce homogeneous experts with limited semantic specialization. As a consequence, substantial additional training is required to recover performance, diminishing the practical benefits of Dense-to-MoE conversion.

Building on these observations, we revisit Dense-to-MoE conversion for embedding models and ask: *How can dense LLM-based embedding models be efficiently reconstructed into MoE architectures while jointly ensuring domain-aware expert specialization and effective routing for embedding generation?* Answering this question requires addressing three tightly coupled challenges. (1) Ambiguous neuron partitioning: existing methods typically rely on activation- or weight-based heuristics to partition neurons into experts. but noisy calibration data and weak semantic supervision often lead to unstable or overlapping expert boundaries. (2) Lack of structural specialization bias: most Dense-to-MoE conversions instantiate homogeneous experts and rely on post-hoc fine-tuning to induce specialization, limiting functional diversity and data efficiency. (3) Routing–representation coupling: in embedding models, routing directly affects representation quality, yet existing methods fail to explicitly account for this coupling during initialization and retraining.

To address these challenges, we propose Den2MoEE[1], a unified Dense-to-MoE reconstruction framework that explicitly enforces domain coherence across expert formation and utilization. Rather than treating partitioning, architecture design, and routing optimization as independent modules, Den2MoEE couples them under a shared domain-aware learning principle. First, we introduce a Neuron Domain Vector-based clustering strategy to construct semantically coherent experts. To obtain reliable neuron-level semantic signals, we propose TokenScore-weighted aggregation, which emphasizes semantically informative tokens while suppressing noisy activations, yielding more stable domain preference estimates. Second, we design a heterogeneous expert architecture consisting of shared experts that capture domain-agnostic knowledge, SVD experts that enforce low-rank domain-specific subspaces, and null experts that improve routing efficiency by absorbing low-information tokens. Third, we propose a two-phase retraining strategy with neuron-anchored router initialization and domain coverage-aware routing bias, enabling more stable expert selection patterns under limited data. Inspired by MoEE, we further incorporate multi-layer router signals as complementary representations, enhancing robustness without increasing activated parameters.

In summary, our contributions are as follows:

- We propose Den2MoEE, a unified Dense-to-MoE reconstruction framework that enforces domain coherence across expert partitioning, structural specialization, and routing optimization for embedding models, enabling near-dense representation quality with significantly reduced activated computation.
- We identify three tightly coupled failure modes in Dense-to-MoE embedding conversion, including noisy partitioning, homogeneous expert collapse, and routing–representation coupling, and propose corresponding solutions via TokenScore-based clustering, heterogeneous expert design, and router-aware retraining.

[1]Den2MoEE is available at https://anonymous.4open.science/r/Den2MoEE-4C70/

- Extensive experimental results show that Den2MoEE consistently outperforms existing embedding baselines under comparable activated parameter budgets, with the number of activated parameters reduced by 33%–55% compared to original dense LLM embedding models. Moreover, Den2MoEE achieves substantial system-level gains including 1.35×–1.8× higher throughput and significantly lower inference latency.

2 Related Work

2.1 LLM as Embedding Model

Text embeddings are a core representation for retrieval (Bajaj et al., 2016), reranking (Liu et al., 2018), and classification (McAuley and Leskovec, 2013). Early transformer-based models such as BERT (Devlin et al., 2019) and SBERT (Reimers and Gurevych, 2019) provide strong contextual representations, while encoder-decoder models like T5 (Raffel et al., 2020) also produce transferable sentence embeddings. Large-scale benchmarks such as MTEB (Muennighoff et al., 2023; Xiao et al., 2024) have standardized evaluation and accelerated progress in general-purpose embedding models. More recently, LLM-based embeddings, including E5 (Wang et al., 2022), GTR (Ni et al., 2022), and INSTRUCTOR (Su et al., 2023), have advanced the field through instruction tuning, contrastive learning, and large-scale pretraining.

Modern models such as BGE (Li et al., 2025), NV-Emb (Lee et al., 2025), E5-Mistral (Wang et al., 2024b), and GTE (Li et al., 2023b), are increasingly built on decoder-only backbones like Mistral (Jiang et al., 2023) and Qwen (Yang et al., 2025), achieving strong semantic understanding and cross-task generalization. However, their growing parameter scales incur substantial computational and memory costs at inference, which is particularly problematic in latency- and throughput-sensitive retrieval systems. This motivates parameter-efficient embedding architectures and sparse computation paradigms.

2.2 Dense-to-MoE Reconstruction

Existing dense-to-MoE methods can be divided into two categories. (1) FFN partition-based methods construct experts by splitting dense FFN parameters without increasing the total parameter budget. MoEBERT (Zuo et al., 2022) distributes salient neurons across experts, MoEfication (Zhang et al., 2022) infers functional partitions from sparse FFN activations, and LLaMA-MoE (Zhu et al., 2024) directly splits FFN weights into non-overlapping experts. These methods rely on heuristic or structure-agnostic partitioning signals, providing limited guidance for capturing fine-grained semantic structure, especially for embedding-oriented representations. (2) FFN upcycling-based methods duplicate pretrained FFN modules to initialize multiple experts, e.g., Sparse Upcycling (Komatsuzaki et al., 2023) and Skywork-MoE (Wei et al., 2024). They increase expert diversity but also enlarge model size and computation, making them less suitable for high-throughput, latency-sensitive deployment. Recent large-scale upcycling work (He et al., 2024; Liew et al., 2025) follows a similar capacity-expanding paradigm.

Overall, existing dense-to-MoE reconstruction primarily targets generative LLMs and overlooks the unique requirements of embedding models. For embedding tasks, the utility of router weights as complementary representations depends on domain-coherent expert specialization, which existing methods rarely encourage. This gap motivates reconstruction strategies that jointly improve expert specialization and deployment efficiency for MoE embedding models.

3 Methodology

Figure 2 illustrates the overall Den2MoEE pipeline. It consists of three stages: (i) TokenScore-weighted neuron domain modeling, which constructs semantically coherent neuron representations; (ii) heterogeneous expert construction, which converts dense FFN layers into specialized experts; and (iii) efficient two-phase retraining, which stabilizes routing and restores embedding quality with minimal activated computation.

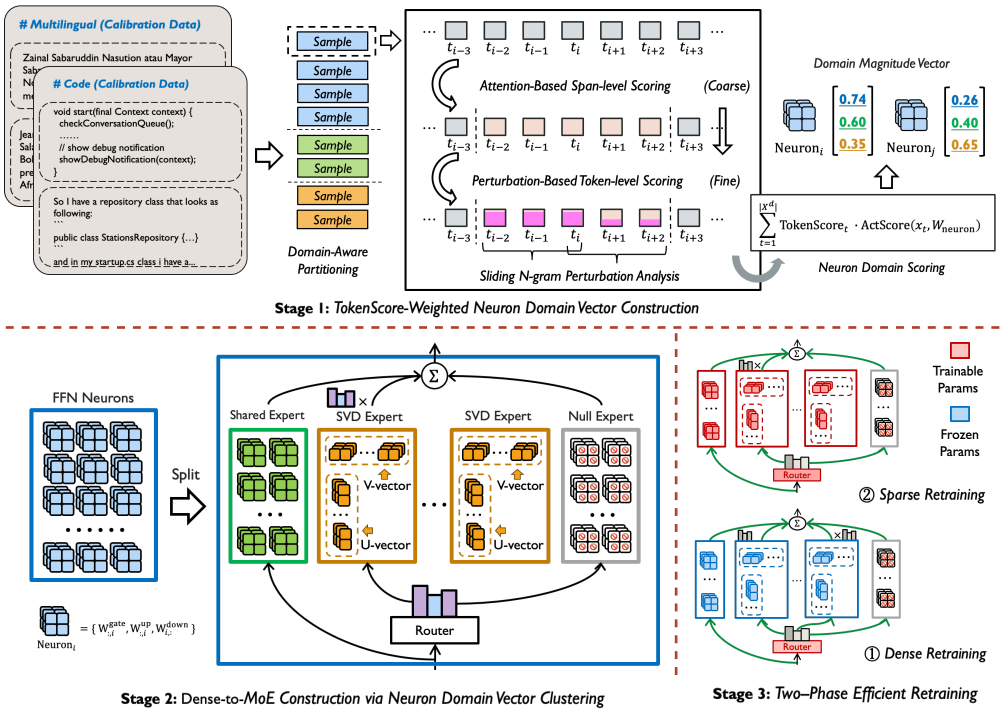


Figure 2: The overall workflow of Den2MoEE.

3.1 TokenScore-weighted Neuron Domain Modeling

Embedding models usually encode the semantics of an input sequence into the hidden state of the terminal token, e.g., [EOS]. Therefore, not all tokens contribute equally to the final embedding. We introduce TokenScore to estimate token-level importance and use it to weight neuron activations when constructing neuron domain representations.

TokenScore. Given an input sequence, we first compute a coarse-grained span-level score via attention accumulation in the final decoder layer, and then refine token-level importance through perturbation-based semantic shift. For a token x_j in span s_i, the final TokenScore is defined as

$$\text{TokenScore}(x_j) = \frac{1}{2}\left(\text{SCS}(s_i) + \text{RSS}(x_j)\right). \tag{1}$$

where SCS measures span-level contribution and RSS measures the relative semantic shift of an n-gram after masking. This additive fusion ensures that a token is prioritized only when it belongs to a semantically significant span and demonstrates high individual necessity. This mechanism effectively penalizes redundant tokens (e.g., fillers) even within high-scoring spans. The detailed computation of SCS and RSS, including span partitioning, perturbation windows, and the computational overhead of TokenScore computation, is provided in Appendix A.

DomainScore. Following prior work (Sun et al., 2024; An et al., 2024), we use the post-gating activation magnitude within the FFN to capture neuron importance. For a token x_t at layer ℓ, the raw activation of the j-th neuron is defined as:

$$\text{ActScore}(x_t, W_j^\ell) = \left|\text{SiLU}\left(x_t W_{:,j}^{\ell,\text{gate}}\right) \cdot \left(x_t W_{:,j}^{\ell,\text{up}}\right)\right|. \tag{2}$$

To reflect token-level importance, we weight this activation by TokenScore. For a sample X^d from domain $\mathcal{D}_d$, the TokenScore-weighted neuron activation score is

$$\text{DomainScore}(X^d) = \frac{1}{|X^d|}\sum_{t=1}^{|X^d|} \text{TokenScore}(x_t) \cdot \text{ActScore}(x_t, W_j^\ell). \tag{3}$$

Neuron domain vector. We partition the calibration set $\mathcal{D}_{\text{cal}}$ into D disjoint domains, following the MTEB taxonomy (Appendix B), and compute the domain-level magnitude of each neuron:

$$s_j^{\ell,d} = \underset{X^d \in \mathcal{D}_d}{\text{Mean}} \left[\text{DomainScore}(X^d)\right], \; \hat{s}_j^{\ell,d} = \frac{s_j^{\ell,d} - \mu_{\ell,d}}{\sigma_{\ell,d}}, \tag{4}$$

where $\mu_{\ell,d}$ and $\sigma_{\ell,d}$ are the mean and standard deviation of all neuron scores within domain d at layer ℓ. The normalized vector

$$\mathbf{v}_j^{\ell} = [\hat{s}_j^{\ell,1}, \ldots, \hat{s}_j^{\ell,D}]^{\top} \tag{5}$$

represents the relative domain preference of neuron j. Compared with raw activations or weight-based heuristics, this TokenScore-weighted domain vector suppresses noisy tokens and provides more stable semantic signals for expert construction.

3.2 Heterogeneous Expert Construction

Given the neuron domain vectors, we reconstruct each dense FFN layer into a sparse MoE layer with specialized experts.

Expert construction via clustering. For each layer ℓ, we partition the N neurons into E disjoint experts of equal size using constrained K-means clustering on the cosine similarity of their domain vectors $\mathbf{v}^{\ell}$. Cosine similarity emphasizes relative domain preference patterns rather than absolute activation scales.

Expert-level profiling. Let $\mathcal{N}_e$ denote the set of neurons assigned to expert e. To characterize the semantic preference of each expert, we compute its Expert Domain Vector:

$$\mathbf{V}_e^{\ell} = [S_e^{\ell,1}, S_e^{\ell,2}, \ldots, S_e^{\ell,D}], \text{ where } S_e^{\ell,d} = \frac{1}{|\mathcal{N}_e|} \sum_{j \in \mathcal{N}_e} \hat{s}_j^{\ell,d}. \tag{6}$$

$S_e^{\ell,d}$ represents the average importance score of expert e within domain d. We further define DomainCoverage to quantify how broadly an expert contributes across domains:

$$\text{DomainCoverage}(e) = \sum_{d=1}^{D} \mathbb{I}[S_e^{\ell,d} > \tau_d], \tag{7}$$

where τ_d is the median of the expert scores in domain d. Experts with high DomainCoverage are generalists, while those with low coverage are more specialized.

Hybrid expert architecture. Based on DomainCoverage and spectral properties, we categorize experts into three functional types.

- **Shared Experts.** Experts in the top 25% of DomainCoverage are kept as shared experts. They capture domain-agnostic knowledge and preserve the core semantic space of the dense model, ensuring a stable structural transition.
- **Low-Rank SVD Experts.** For the remaining experts, we apply singular value decomposition to the up-projection and gate-projection matrices $W_e \in \{W^{\text{up}}, W^{\text{gate}}\}$, and construct low-rank approximations. We truncate the singular spectrum to approximately 60% of the rank while preserving 80% of the total energy. This suppresses redundant high-dimensional directions and encourages SVD experts to focus on informative low-rank semantic subspaces.
- **Null Experts.** Following Zeng et al. (2024), we introduce Null Experts to occupy routing slots vacated by shared experts. They absorb low-information tokens at no additional cost, improve routing resolution, and allow more specialized expert allocation.

3.3 Efficient Two-Phase Retraining

After structural conversion, we recover performance through stable routing initialization, dynamic expert routing bias, and two-phase retraining.

Stable routing initialization. Under full expert activation, the MoE forward pass should be numerically equivalent to the original dense FFN to preserve pre-trained knowledge. Let E denote

the number of experts. In the early stage, routers initialized randomly with small variance tend to produce near-uniform gating outputs ($g_j(x) \approx 1/E$), which attenuates the MoE output magnitude. We compensate by scaling the MoE output by E:

$$\text{MoE}(x) = E \cdot \sum_{j=1}^{E} g_j(x) \cdot \text{Expert}_j(x) \approx \text{FFN}_{\text{dense}}(x). \tag{8}$$

To accelerate expert specialization, we compute routing anchors as the mean up-projection weights of each expert's neurons:

$$\mathbf{A}_e^{\ell} = \frac{1}{|\mathcal{N}_e|} \sum_{j \in \mathcal{N}_e} W_{:,j}^{\ell,\text{up}}. \tag{9}$$

and progressively align router weights:

$$\mathbf{W}_{\text{router}} = \mathbf{W}_{\text{rand}} + \alpha(t) \cdot \mathbf{A}^{\ell},\ \alpha(t) \in [0, 1]. \tag{10}$$

This progressive semantic alignment first stabilizes output magnitude and then encourages semantic-aware expert specialization.

Dynamic expert routing bias. Conventional MoE training often relies on auxiliary load-balancing losses, which may force excessive routing uniformity and suppress expert specialization. Instead, we introduce a domain-coverage-aware expert bias b_i. For SVD experts, the bias is updated according to the difference between the expected activation ratio, derived from DomainCoverage, and the actual load ratio in the current batch. The update increment is

$$\Delta b_i = \begin{cases} \mu \left(\frac{K_e}{K} \cdot \frac{c_i}{\sum_{j=1}^{N} c_j} - \frac{T_i}{KT_{all}} \right), & i \in E_{\text{svd}} \\ 0, & i \notin E_{\text{svd}} \end{cases}. \tag{11}$$

where K_e is the expected number of activated experts ($\leq K$), K is the router's Top-K setting, and $c_i = \text{DomainCoverage}(i)/D$, with larger c_i implying a higher expected activation frequency. T_i is the number of tokens routed to i-th expert in the current batch T_{all} is the total number of tokens. $\frac{K_e}{K} \cdot \frac{c_i}{\sum_{j=1}^{N} c_j}$ gives the expected activation ratio and $\frac{T_i}{KT_{all}}$ represents the actual load ratio per batch, μ controls the bias update step size.

This bias is updated only for SVD experts; Null Experts remain a fixed baseline ($\Delta b_i = 0$). As a result, routing is biased toward domain-specialized experts during inference, while routing to Null Experts remains possible but rare, typically for low-information tokens such as stopwords.

Two-Phase retraining. Due to the structural reorganization of the LLM, retraining is essential to recover overall performance. Given that the top-k operation within the router introduces sparse gradients and non-differentiability, we propose a two-stage retraining strategy:

- **Dense Training Phase.** The router is trained on a small subset of data while all experts remain activated. This phase learns a proper initial representation for experts within the router and integrates the routing initialization described above.
- **Sparse Training Phase.** Both experts and the router are trained simultaneously on large-scale data, with the dynamic expert routing bias enabled. A low-weight load balancing loss is applied only to avoid extreme routing skew while minimally affecting expert specialization.

Algorithm 1 in Appendix C provides a detailed description of the entire Den2MoEE pipeline.

4 Experiments

4.1 Experimental Settings

Datasets. We retrain Den2MoEE on a subset of KaLM-Embedding data (Zhao et al., 2026), covering code and multilingual domains. For calibration, we use the 30-domain partition detailed in Appendix B and sample 15K instances from StarCoder (Li et al., 2023a) and MIRACL (Zhang et al., 2023), with 500 instances per domain. The calibration data is used to evaluate neuron domain

Table 1: Performance on MTEB(Code, v1). ◇ / * denote Den2MoEE achieving the best average results with 0.4B / 1.8B activated parameters, respectively, and † indicates that Qwen3-Embedding-MoE is reconstructed following LLaMA-MoE.

Model	Act-Size	Mean (All)	Mean (CoIR)	Apps	COIR-CodeSearch-Net	Code-Edit-Search	Code-Feedback-MT	Code-Feedback-ST	Code-SearchNet-CCR	Code-SearchNet	Code-Trans-Ocean-Contest	Code-Trans-Ocean-DL	CosQA	Stack-Overflow-QA	Synthetic-Text2SQL
BGE-M3	0.6B	58.22	51.68	14.77	58.07	59.83	47.86	69.27	53.55	61.98	86.22	29.37	27.36	80.71	49.65
Qwen3-Embedding-0.6B-MoE (2/8, A0.4B)†	0.4B	67.66	66.96	68.45	74.06	59.19	82.68	75.97	83.36	83.12	78.38	26.72	31.27	81.54	67.14
EmbeddingGemma-300M	0.3B	68.76	67.29	84.39	75.54	62.10	51.42	80.26	73.71	90.15	85.51	33.52	43.60	86.47	58.42
CodeXEmbed-400M	0.4B	-	68.42	45.91	81.23	-	61.51	83.36	93.74	-	82.72	40.81	41.28	92.35	61.29
Den2MoEE-0.6B-A0.4B (Ours)	0.4B	72.68◇	71.62◇	72.32	78.85	69.32	86.43	84.51	87.09	86.62	81.82	30.04	34.71	89.11	71.32
Gte-Qwen2-1.5B-instruct	1.5B	61.98	59.31	28.91	71.56	59.60	49.92	81.92	72.08	91.08	79.02	32.73	32.23	90.27	54.49
Gte-Qwen2-7B-instruct	7.6B	62.17	59.21	28.39	71.79	67.06	57.66	85.15	66.24	86.96	81.83	32.17	31.26	84.34	53.22
INF-Retriever-v1	7.1B	69.70	67.59	47.37	72.27	71.79	77.64	86.63	75.71	88.77	89.16	35.18	34.18	94.22	63.51
Jina-Embedding-v4	3.8B	-	72.87	78.32	84.05	-	70.61	85.06	83.70	-	89.34	44.19	31.49	91.43	70.45
Qwen3-Embedding-4B-MoE (4/16, A2B)†	2.0B	74.16	73.23	81.08	80.78	71.65	87.76	85.65	87.35	86.54	82.76	30.39	34.52	91.18	71.25
CodeXEmbed-2B	2.0B	-	75.65	76.86	87.87	-	65.51	86.36	97.66	-	90.30	38.57	40.47	94.47	78.42
Den2MoEE-4B-A1.8B (Ours)	1.8B	**76.54***	**75.89***	**84.82**	84.93	72.10	88.18	86.52	89.11	87.45	86.05	34.38	36.52	93.70	74.68

Table 2: Performance on MTEB (Multilingual, v2). Same notation as Table 1.

Model	Act-Size	Mean (Type)	Bitext Mining	Class-ification	Clus-tering	Inst. Retrieval	Multilabel Class.	Pair Class.	Rerank	Retrieval	STS
BGE-M3	0.6B	52.18	79.11	60.35	40.88	-3.11	20.10	80.76	62.79	54.60	74.12
Qwen3-Embedding-0.6B-MoE (2/8, A0.4B)†	0.4B	52.73	62.63	59.21	49.56	4.12	23.26	79.73	61.84	61.10	73.12
EmbeddingGemma-300M	0.3B	54.31	64.40	60.90	51.17	5.61	24.82	81.40	63.25	62.49	74.73
Multilingual-E5-Large-Instruct	0.6B	55.08	80.13	64.94	50.75	-0.40	22.91	80.86	62.61	57.12	76.81
Den2MoEE-0.6B-A0.4B (Ours)	0.4B	55.14◇	71.35	65.96	51.46	4.22	23.72	79.95	60.54	63.77	75.30
Gte-Qwen2-1.5B-instruct	1.5B	52.69	62.51	58.32	52.05	0.74	24.02	81.58	62.58	60.78	71.61
Linq-Embed-Mistral	7.1B	54.14	70.34	62.24	50.60	0.94	24.77	80.43	64.37	58.69	74.86
Qwen3-Embedding-4B-MoE (4/16, A2B)†	2.0B	55.42	73.15	67.77	52.41	6.42	21.59	80.24	58.68	64.82	73.68
Gte-Qwen2-7b-Instruct	7.6B	55.93	73.92	61.55	52.77	4.94	25.48	85.13	65.55	60.08	73.98
Den2MoEE-4B-A1.8B (Ours)	1.8B	**57.52***	75.54	69.32	54.41	7.86	24.13	82.05	60.98	67.10	76.33

specialization and guide expert construction during the calibration phase.The details of the retraining dataset is present in Appendix D.1.

Training setup. We use the full training dataset during the sparse training stage, while in the dense training stage only 15% of the training data is used to warm up the router. All training stages are performed for a single epoch, and training instances are optimized using the InfoNCE loss. Our implementation is based on the HuggingFace Transformers (Wolf et al., 2020) and Swift (Zhao et al., 2024) frameworks. All experiments are conducted on 2 nodes with 8 H20 (96GB) GPUs per node. Detailed hyperparameter configurations are summarized in Appendix D.2.

Baseline models. We compare against same-scale open-source embedding models ($\leq$2B activated parameters) and larger representative models. Code baselines include BGE-M3 (Chen et al., 2024), EmbeddingGemma-300M (Vera et al., 2025), Gte-Qwen2 (Li et al., 2023b), CodeXEmbed (Liu et al., 2025), INF-Retriever-v1 (infly-ai, 2025), and Jina-Embedding-v4 (Günther et al., 2025). Multilingual baselines include BGE-M3 (Chen et al., 2024), EmbeddingGemma-300M (Vera et al., 2025), Multilingual-E5-Large-Instruct (Wang et al., 2024a), Gte-Qwen2 (Li et al., 2023b), and Linq-Embed-Mistral (Choi et al., 2024). We also compare against Qwen3-Embedding and its LLaMA-MoE variant (i.e., Qwen3-Embedding-MoE). Den2MoEE-0.6B and Den2MoEE-4B are derived from Qwen3-Embedding-0.6B and Qwen3-Embedding-4B, respectively. All baseline results are reported in the original papers or MTEB benchmark (except for Qwen3-Embedding-MoE).

Evaluation metric. All evaluations use bfloat16 weights and follow the official Qwen3-Embedding evaluation scripts. We report results on MTEB (Code, v1) and MTEB (Multilingual, v2). For efficiency, we measure end-to-end latency, throughput, and memory footprint with sequence length 512 and batch size 128. The calculation of throughput is in Appendix E.

4.2 Overall Performance

Table 1 and Table 2 show the performance on MTEB Code and Multilingual, and Table 3 compares deployment efficiency against the original dense backbones and LLaMA-MoE reconstructions.

Performance. Den2MoEE achieves SOTA average performance with 0.4B and 1.8B activated parameters, significantly surpassing the 0.3B EmbeddingGemma model, while remaining competitive at larger scales. This indicates that Den2MoEE effectively preserves and reorganizes the representational capacity of the original dense model.

Table 3: Performance and deployment efficiency comparison of Den2MoEE against the original Qwen3-Embedding models and LLaMA-MoE baselines. †: LLaMA-MoE reconstruction.

Model	Act-Size	Multilingual	Code	Latency (ms)	Throughput (fps)	Memory Footprint (MB)
Qwen3-Embedding-0.6B	0.6B	56.00	75.41	833.96	153.49	1136.37
Qwen3-Embedding-0.6B-MoE (2/8, A0.4B)†	0.4B (-0.2B)	52.73 (-3.27)	67.66 (-7.75)	652.15 (↓ 22%)	196.27 (↑ 28%)	1268.73 (↑ 12%)
Den2MoEE-0.6B-A0.4B	0.4B (-0.2B)	55.14 (-0.86)	72.68 (-2.73)	617.13 (↓ 26%)	207.41 (↑ 35%)	1224.55 (↑ 8%)
Qwen3-Embedding-4B	4.0B	60.86	80.06	4361.44	29.35	7756.94
Qwen3-Embedding-4B-MoE (4/16, A2B)†	2.0B (-2.0B)	55.42 (-5.44)	74.16 (-5.90)	2585.12 (↓ 40%)	49.51 (↑ 69%)	8914.83 (↑ 15%)
Den2MoEE-4B-A1.8B	1.8B (-2.2B)	57.52 (-3.34)	76.54 (-3.52)	2398.79 (↓ 45%)	53.36 (↑ 82%)	8510.45 (↑ 10%)

Table 4: Ablation study on Den2MoEE Design. $\mathrm{Act.Experts} = N_{\mathrm{share}} + N_{\mathrm{route}} * \mathrm{Sparsity}_{\mathrm{svd}}$. GFLOPs ↓ represents the reduction compared to the dense model.

Model	MoE Conf. (N_{exp})	MTEB (Code)	MTEB (Multi.)	Act. Experts (per layer)	GFLOPs ↓ (per layer)
Den2MoEE-4B-A1.8B	$\mathrm{Exp}_{\mathrm{share}}$=2, $\mathrm{Exp}_{\mathrm{svd}}$=14, $\mathrm{Exp}_{\mathrm{null}}$=2, Top-K=2	76.54	57.52	3.24 =(2.00+1.84*0.68)	0.1191 (79.75% ↓)
w/o SVD Experts and Null Experts	$\mathrm{Exp}_{\mathrm{share}}$=2, $\mathrm{Exp}_{\mathrm{base}}$=14, $\mathrm{Exp}_{\mathrm{null}}$=0, Top-K=2	74.59 (-1.95)	56.33 (-1.19)	4.00 =(2.00+2.00*1.00)	0.1121 (75.00% ↓)
w/o Shared Experts	$\mathrm{Exp}_{\mathrm{share}}$=0, $\mathrm{Exp}_{\mathrm{svd}}$=16, $\mathrm{Exp}_{\mathrm{null}}$=2, Top-K=4	74.26 (-2.28)	56.10 (-1.42)	2.68 =(0.00+3.88*0.69)	0.1244 (83.25% ↓)
Naive MoE Design	$\mathrm{Exp}_{\mathrm{share}}$=0, $\mathrm{Exp}_{\mathrm{base}}$=16, $\mathrm{Exp}_{\mathrm{null}}$=0, Top-K=4	73.49 (-3.05)	55.64 (-1.88)	4.00 =(0.00+4.00*1.00)	0.1121 (75.00% ↓)

Comparison with prior MoE reconstruction. Den2MoEE also outperforms the LLaMA-MoE reconstruction of Qwen3-Embedding under the same activation budget. This indicates that domain-aware neuron scoring, heterogeneous expert design, and router-aware retraining are more effective than weight-based expert splitting for embedding-oriented reconstruction.

Efficiency. Table 3 shows that Den2MoEE-0.6B-A0.4B reduces activated parameters by 33% compared with Qwen3-Embedding-0.6B, while achieving 26% lower latency and 35% higher throughput. Den2MoEE-4B-A1.8B reduces activated parameters by 45%, with 45% lower latency and 82% higher throughput. Memory overhead remains manageable in both settings. These results show that MoE conditional computation translates parameter sparsity into system-level acceleration.

Overall, Den2MoEE preserves near-dense embedding quality while substantially reducing activated computation and improving serving efficiency.

4.3 Ablation Studies

Expert architecture design. In Table 4, removing shared experts leads to the largest performance degradation, with Code and Multilingual scores dropping by 2.28 and 1.42 points, respectively. This confirms that shared experts preserve common knowledge inherited from the dense model and facilitate cross-domain knowledge sharing. Replacing the SVD-specialized and null experts with standard dense experts (w/o SVD & Null Experts) also results in noticeable performance loss, showing that heterogeneous expert decomposition is important for capturing domain-specific patterns while maintaining sparsity. The Naive MoE design, which directly splits the dense FFN into homogeneous experts, performs worst despite using a comparable number of experts. Overall, Den2MoEE achieves the best effectiveness-efficiency trade-off, reducing the average number of activated experts from 4.00 to 3.24 and GFLOPs by approximately 80% while achieving the strongest performance.

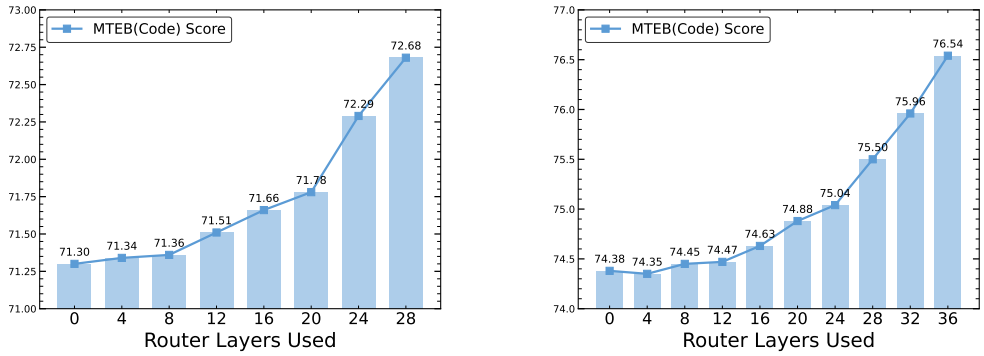
Data efficiency. Table 5 shows that Den2MoEE consistently outperforms LLaMA-MoE across all data scales. In the 0.6B setting, LLaMA-MoE requires about 5M samples to reach the performance Den2MoEE achieves with only 3M samples. In the 4B setting, Den2MoEE reduces the training data requirement by roughly 1–2M samples under comparable performance. These results demonstrate substantially improved reconstruction efficiency, suggesting that Den2MoEE learns more transferable and domain-aligned representations from each training sample.

Dense retraining. We ablate the dense retraining by directly applying sparse routing without router warm-up. As shown in Appendix F.1, dense retraining consistently leads to faster loss reduction and earlier performance gains once sparse routing begins.

Top-K selection. Appendix F.2 presents the impact of Top-K selection on performance and FFN activation rate in MoE. For the 0.6B and 4B models, E8K2 and E16K2 offer the best trade-off between FFN sparsity and model quality.

Table 5: Reconstruction data efficiency comparison on MTEB Multilingual.

Data (M)	Den2MoEE-4B-A1.8B	LLaMA-MoE-4B-A2B	Den2MoEE-0.6B-A0.4B	LLaMA-MoE-0.6B-A0.4B
1.00	46.58	40.79	43.56	36.18
2.00	54.34	49.15	50.86	45.84
3.00	55.11	53.05	52.94	50.80
4.00	55.82	54.64	54.13	51.83
5.00	57.52	55.42	55.14	52.73



(a) Den2MoEE-0.6B-A0.4B (b) Den2MoEE-4B-A1.8B

Figure 3: Impact of router weights. Router weights are taken from the first N layers (e.g., 4, 8, 12).

4.4 In-depth Analysis

Router weights as complementary embeddings. We select the router weights from the first N_{layers} layers, perform mean pooling over every four consecutive layers and concatenate the pooled vectors to form $e_{\text{RW}} \in \mathbb{R}^{d_{\text{RW}}}$ ($d_{\text{RW}} = (N_{\text{layers}}/4) \times N_{\text{experts}}$). As shown in Figure 3, router weights from deeper layers provide complementary semantic information to hidden states. Performance gains are negligible for the first 8 layers (0.6B) and 12 layers (4B), but become pronounced after layer 20 (0.6B) and layer 24 (4B). Den2MoEE-4B (+2.16%) benefits more than Den2MoEE-0.6B (+1.38%), and the last four layers contribute more in the larger model. This suggests that routers with more experts can partition the latent space more finely and capture task-specific features not covered by hidden states. More analysis about the role of router weights is in Appendix F.3.

Clustering metrics. We evaluate different clustering strategies in Appendix F.4. Domain Vector-based clustering outperforms weight-based clustering. Among weight vectors, Up-proj outperforms Gate-proj, as Up-proj prefers detecting input directions while Gate-proj mainly controls linear combinations and output scaling. Removing TokenScore from Domain Vectors degrades performance, confirming its role in suppressing irrelevant tokens during calibration.

Router distribution. Figure 7 in Appendix F.5 visualizes expert activation distributions across representative domains. Den2MoEE produces significantly sparser and more concentrated routing distributions, indicating that the heterogeneous expert architecture and router alignment strategy effectively reduce routing ambiguity and improve expert specialization.

Appendix F.6 demonstrates the generalization capability across other dense embedding backbones such as Harrier-oss-v1 (Microsoft, 2026) and F2LLM-v2 (Zhang et al., 2026). Appendix F.7 further reveals the scalability of Den2MoEE to additional domains (e.g., legal domain).

5 Conclusion

In this work, we propose Den2MoEE, a unified dense-to-MoE reconstruction framework that jointly promotes domain-aware expert specialization and routing-aware adaptation. It converts dense LLM embeddings into efficient MoE architectures while preserving near-dense quality. Extensive experiments demonstrate that Den2MoEE outperforms existing baselines under comparable activated parameter budgets while delivering substantial efficiency gains in both throughput and inference latency. These results suggest that structured Dense-to-MoE reconstruction is a practical path toward efficient, high-quality embedding models.

AI USE STATEMENT

Generative AI tools were used to assist with language refinement. All AI-assisted suggestions were independently reviewed and validated by the authors. The authors retained full responsibility for the research methodology, experimental design, analysis, conclusions, and final content of the work.

REFERENCES

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-Based Adaptive Structured Pruning for Large Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10865–10873. AAAI Press, 2024. doi: 10.1609/aaai.v38i10.28960. URL https://doi.org/10.1609/aaai.v38i10.28960.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv preprint arXiv:1611.09268*, 2016. doi: 10.48550/arXiv.1611.09268. URL https://arxiv.org/abs/1611.09268.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A Survey on Mixture of Experts in Large Language Models. *IEEE Transactions on Knowledge and Data Engineering*, 37(7):3896–3915, 2025. doi: 10.1109/TKDE.2025.3554028. URL https://doi.org/10.1109/TKDE.2025.3554028.

Jianghao Chen, Junhong Wu, Yangyifan Xu, and Jiajun Zhang. LADM: Long-context Training Data Selection with Attention-based Dependency Measurement for LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3076–3090. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.154. URL https://aclanthology.org/2025.acl-long.154/.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. Towards Understanding the Mixture-of-Experts Layer in Deep Learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 23049–23062. Curran Associates, Inc., 2022. doi: 10.52202/068431-1675. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

Chanyeol Choi, Junseong Kim, Seolhwa Lee, Jihoon Kwon, Sangmo Gu, Yejin Kim, Minkyung Cho, and Jy-yong Sohn. Linq-Embed-Mistral Technical Report. *arXiv preprint arXiv:2412.03223*, 2024. doi: 10.48550/arXiv.2412.03223. URL https://arxiv.org/abs/2412.03223.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.70. URL https://aclanthology.org/2024.acl-long.70/.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, Ömer Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Mariya Hendriksen, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Šuppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani, Pranjal Chitale, Simone Tedeschi, Nguyen Tai, Artem Snegirev, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri Krishnakumar, Anna Maksimova, Silvan Wehrli, Maria Tikhonova, Henil Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Hongjin Su, Jimmy Lin, Howard Yen, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muennighoff. MMTEB: Massive Multilingual Text Embedding Benchmark. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/fc0e3f908a2116ba529ad0a1530a3675-Paper-Conference.pdf.

Gregor Geigle, Nils Reimers, Andreas Rücklé, and Iryna Gurevych. TWEAC: Transformer with Extendable QA Agent Classifiers. *arXiv preprint arXiv:2104.07081*, 2021. doi: 10.48550/arXiv.2104.07081. URL https://arxiv.org/abs/2104.07081.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446/.

Michael Günther, Saba Sturua, Mohammad Kalim Akram, Isabelle Mohr, Andrei Ungureanu, Bo Wang, Sedigheh Eslami, Scott Martens, Maximilian Werk, Nan Wang, and Han Xiao. jina-embeddings-v4: Universal Embeddings for Multimodal Multilingual Retrieval. In *Proceedings of the 5th Workshop on Multilingual Representation Learning (MRL 2025)*, pages 531–550. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.mrl-main.36. URL https://aclanthology.org/2025.mrl-main.36/.

Ethan He, Abhinav Khattar, Ryan Prenger, Vijay Korthikanti, Zijie Yan, Tong Liu, Shiqing Fan, Ashwath Aithal, Mohammad Shoeybi, and Bryan Catanzaro. Upcycling Large Language Models into Mixture of Experts. *arXiv preprint arXiv:2410.07524*, 2024. doi: 10.48550/arXiv.2410.07524. URL https://arxiv.org/abs/2410.07524.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. CodeSearchNet Challenge: Evaluating the State of Semantic Code Search. *arXiv preprint arXiv:1909.09436*, 2019. doi: 10.48550/arXiv.1909.09436. URL https://arxiv.org/abs/1909.09436.

infly-ai. inf-retriever-v1. Hugging Face model card, 2025. URL https://huggingface.co/infly/inf-retriever-v1. Revision 5f469d7. Accessed September 25, 2026.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79. URL https://doi.org/10.1162/neco.1991.3.1.79.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023. doi: 10.48550/arXiv.2310.06825. URL https://arxiv.org/abs/2310.06825.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse Upcycling: Training Mixture-of-Experts from Dense Checkpoints. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=T5nUQDrM4u.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/c4bf73386022473a652a18941e9ea6f8-Paper-Conference.pdf.

Chaofan Li, Minghao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Defu Lian, Yingxia Shao, and Zheng Liu. Making Text Embedders Few-Shot Learners. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5eba8556ce2d1afff57591464d48fbc3-Paper-Conference.pdf.

Raymond Li, Loubna Ben allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia LI, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Joel Lamy-Poirier, Joao Monteiro, Nicolas Gontier, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Ben Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason T Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Urvashi Bhattacharyya, Wenhao Yu, Sasha Luccioni, Paulo Villegas, Fedor Zhdanov, Tony Lee, Nadav Timor, Jennifer Ding, Claire S Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro Von Werra, and Harm de Vries. StarCoder: may the source be with you! *Transactions on Machine Learning Research*, 2023a. URL https://openreview.net/forum?id=KoFOg41haE. Reproducibility Certification.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards General Text Embeddings with Multi-stage Contrastive Learning. *arXiv preprint arXiv:2308.03281*, 2023b. doi: 10.48550/arXiv.2308.03281. URL https://arxiv.org/abs/2308.03281.

Ziyue Li and Tianyi Zhou. Your Mixture-of-Experts LLM Is Secretly an Embedding Model for Free. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/aed2049f68827943dda5a63b7c4ba0a2-Paper-Conference.pdf.

Seng Pei Liew, Takuya Kato, and Sho Takase. Scaling Laws for Upcycling Mixture-of-Experts Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 37682–37704. PMLR, 2025. URL https://proceedings.mlr.press/v267/liew25a.html.

Xueqing Liu, Chi Wang, Yue Leng, and ChengXiang Zhai. LinkSO: a dataset for learning to retrieve similar question answer pairs on software development forums. In *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering*, NL4SE 2018, pages 2–5. Association for Computing Machinery, 2018. doi: 10.1145/3283812.3283815. URL https://doi.org/10.1145/3283812.3283815.

Ye Liu, Rui Meng, Shafiq Joty, Silvio Savarese, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. CodeXEmbed: A Generalist Embedding Model Family for Multilingual and Multi-task Code Retrieval. In *Second Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=z3lG70Azbg.

Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, Marek Cygan, and Sebastian Jaszczur. Scaling Laws for Fine-Grained Mixture of Experts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine*

Learning Research, pages 33270–33288. PMLR, 2024. URL https://proceedings.mlr.press/v235/ludziejewski24a.html.

Jan Ludziejewski, Maciej Pióro, Jakub Krajewski, Maciej Stefaniak, Michał Krutul, Jan Małaśnicki, Marek Cygan, Piotr Sankowski, Kamil Adamczewski, Piotr Miłoś, and Sebastian Jaszczur. Joint MoE Scaling Laws: Mixture of Experts Can Be Memory Efficient. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 41056–41073. PMLR, 2025. URL https://proceedings.mlr.press/v267/ludziejewski25a.html.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, RecSys '13, pages 165–172. Association for Computing Machinery, 2013. doi: 10.1145/2507157.2507163. URL https://doi.org/10.1145/2507157.2507163.

Microsoft. Harrier-OSS-v1-0.6B. Hugging Face model card, 2026. URL https://huggingface.co/microsoft/harrier-oss-v1-0.6b. Accessed September 25, 2026.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. MTEB: Massive Text Embedding Benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.eacl-main.148. URL https://aclanthology.org/2023.eacl-main.148/.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. Large Dual Encoders Are Generalizable Retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.669. URL https://aclanthology.org/2022.emnlp-main.669/.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), 2020. URL https://jmlr.org/papers/v21/20-074.html.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One Embedder, Any Task: Instruction-Finetuned Text Embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.71. URL https://aclanthology.org/2023.findings-acl.71/.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A Simple and Effective Pruning Approach for Large Language Models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/14c856c7a41297804de4c4890e846b25-Paper-Conference.pdf.

Nandan Thakur, Jianmo Ni, Gustavo Hernandez Abrego, John Wieting, Jimmy Lin, and Daniel Cer. Leveraging LLMs for Synthesizing Training Data Across Many Languages in Multilingual Dense Retrieval. 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 7699–7724. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.426. URL https://aclanthology.org/2024.naacl-long.426/.

Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, Daniel Cer, Alice Lisak, Min Choi, Lucas Gonzalez, Omar Sanseviero, Glenn Cameron, Ian Ballantyne, Kat Black, Kaifeng Chen, Weiyi Wang, Zhe Li, Gus Martins, Jinhyuk Lee, Mark Sherwood, Juyeong Ji, Renjie Wu, Jingxiao Zheng, Jyotinder Singh, Abheesht Sharma, Divyashree Sreepathihalli, Aashi Jain, Adham Elarabawy, AJ Co, Andreas Doumanoglou, Babak Samari, Ben Hora, Brian Potetz, Dahun Kim, Enrique Alfonseca, Fedor Moiseev, Feng Han, Frank Palma Gomez, Gustavo Hernández Ábrego, Hesen Zhang, Hui Hui, Jay Han, Karan Gill, Ke Chen, Koert Chen, Madhuri Shanbhogue, Michael Boratko, Paul Suganthan, Sai Meher Karthik Duddu, Sandeep Mariserla, Setareh Ariafar, Shanfeng Zhang, Shijie Zhang, Simon Baumgartner, Sonam Goenka, Steve Qiu, Tanmaya Dabral, Trevor Walker, Vikram Rao, Waleed Khawaja, Wenlei Zhou, Xiaoqi Ren, Ye Xia, Yichang Chen, Yi-Ting Chen, Zhe Dong, Zhongli Ding, Francesco Visin, Gaël Liu, Jiageng Zhang, Kathleen Kenealy, Michelle Casbon, Ravin Kumar, Thomas Mesnard, Zach Gleicher, Cormac Brick, Olivier Lacombe, Adam Roberts, Qin Yin, Yunhsuan Sung, Raphael Hoffmann, Tris Warkentin, Armand Joulin, Tom Duerig, and Mojtaba Seyedhosseini. EmbeddingGemma: Powerful and Lightweight Text Representations. *arXiv preprint arXiv:2509.20354*, 2025. doi: 10.48550/arXiv.2509.20354. URL https://arxiv.org/abs/2509.20354.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text Embeddings by Weakly-Supervised Contrastive Pre-training. *arXiv preprint arXiv:2212.03533*, 2022. doi: 10.48550/arXiv.2212.03533. URL https://arxiv.org/abs/2212.03533.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual E5 Text Embeddings: A Technical Report. *arXiv preprint arXiv:2402.05672*, 2024a. doi: 10.48550/arXiv.2402.05672. URL https://arxiv.org/abs/2402.05672.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving Text Embeddings with Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.acl-long.642. URL https://aclanthology.org/2024.acl-long.642/.

Tianwen Wei, Bo Zhu, Liang Zhao, Cheng Cheng, Biye Li, Weiwei Lü, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Liang Zeng, Xiaokun Wang, Yutuan Ma, Rui Hu, Shuicheng Yan, Han Fang, and Yahui Zhou. Skywork-MoE: A Deep Dive into Training Techniques for Mixture-of-Experts Language Models. *arXiv preprint arXiv:2406.06563*, 2024. doi: 10.48550/arXiv.2406.06563. URL https://arxiv.org/abs/2406.06563.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed Resources For General Chinese Embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pages 641–649. Association for Computing Machinery, 2024. doi: 10.1145/3626772.3657878. URL https://doi.org/10.1145/3626772.3657878.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger

Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3687–3692. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1382. URL https://aclanthology.org/D19-1382/.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. AdaMoE: Token-Adaptive Routing with Null Experts for Mixture-of-Experts Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6223–6235. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.361. URL https://aclanthology.org/2024.findings-emnlp.361/.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. mGTE: Generalized Long-Context Text Representation and Reranking Models for Multilingual Text Retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1393–1412. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-industry.103. URL https://aclanthology.org/2024.emnlp-industry.103/.

Xinyu Zhang, Xueguang Ma, Peng Shi, and Jimmy Lin. Mr. TyDi: A Multi-lingual Benchmark for Dense Retrieval. In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pages 127–137. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.mrl-1.12. URL https://aclanthology.org/2021.mrl-1.12/.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. MIRACL: A Multilingual Retrieval Dataset Covering 18 Diverse Languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131, 2023. doi: 10.1162/tacl_a_00595. URL https://aclanthology.org/2023.tacl-1.63/.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176*, 2025. doi: 10.48550/arXiv.2506.05176. URL https://arxiv.org/abs/2506.05176.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. MoEfication: Transformer Feed-forward Layers are Mixtures of Experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.71. URL https://aclanthology.org/2022.findings-acl.71/.

Ziyin Zhang, Zihan Liao, Hang Yu, Peng Di, and Rui Wang. F2LLM-v2: Inclusive, Performant, and Efficient Embeddings for a Multilingual World. *arXiv preprint arXiv:2603.19223*, 2026. doi: 10.48550/arXiv.2603.19223. URL https://arxiv.org/abs/2603.19223.

Xinping Zhao, Xinshuo Hu, Zifei Shan, Shouzheng Huang, Yao Zhou, Xin Zhang, Zetian Sun, Zhenyu Liu, Dongfang Li, Xinyuan Wei, Youcheng Pan, Yang Xiang, Meishan Zhang, Haofen Wang, Jun Yu, Baotian Hu, and Min Zhang. KaLM-Embedding-V2: Superior Training Techniques and Data Inspire A Versatile Embedding Model. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/6082359ffba504842a41da0d6a3cffe2-Paper-Conference.pdf.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Hong Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. SWIFT:

A Scalable lightWeight Infrastructure for Fine-Tuning. *arXiv preprint arXiv:2408.05517*, 2024. doi: 10.48550/arXiv.2408.05517. URL https://arxiv.org/abs/2408.05517.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. OpenCodeInterpreter: Integrating Code Generation with Execution and Refinement. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12834–12859. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.762. URL https://aclanthology.org/2024.findings-acl.762/.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. LLaMA-MoE: Building Mixture-of-Experts from LLaMA with Continual Pre-Training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15913–15923. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.890. URL https://aclanthology.org/2024.emnlp-main.890/.

Simiao Zuo, Qingru Zhang, Chen Liang, Pengcheng He, Tuo Zhao, and Weizhu Chen. MoEBERT: from BERT to Mixture-of-Experts via Importance-Guided Adaptation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1610–1623. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.116. URL https://aclanthology.org/2022.naacl-main.116/.

A Hierarchical Token Importance Scoring

A.1 Attention-based Span-level Scoring

Inspired by Chen et al. (2025), given an input sequence $\mathbf{X} = \{x_0, x_1, \ldots, x_{T-1}, x_{\text{last}}\}$, we partition the prefix into N non-overlapping spans $\mathcal{S} = \{s_0, s_1, \ldots, s_{N-1}\}$. For the h-th attention head in the final decoder layer, let $A^{(h)}_{\text{last},t}$ denote the attention weight assigned by the *last token* to a preceding token x_t. We define the Span Attention Accumulation (SAA) for span s_i as the aggregated attention across all H heads.

$$\text{SAA}(s_i) = \sum_{h=1}^{H} \sum_{t \in s_i} A^{(h)}_{\text{last},t} \tag{12}$$

$$A^{(h)}_{\text{last},t} = \text{Softmax}\left(\frac{Q^{(h)}_{\text{last}} (K^{(h)}_{0:T-1})^\top}{\sqrt{d_k}} \right)[t] \tag{13}$$

To ensure cross-sequence comparability, the final Span Contribution Score (SCS) is obtained by applying Min-Max normalization over all candidate spans.

$$\text{SCS}(s_i) = \frac{\text{SAA}(s_i) - \min_{s \in \mathcal{S}} \text{SAA}(s)}{\max_{s \in \mathcal{S}} \text{SAA}(s) - \min_{s \in \mathcal{S}} \text{SAA}(s)}. \tag{14}$$

This score serves as a coarse-grained proxy for global significance. We retain the top-50% spans $\mathcal{S}_{\text{top-50\%}}$ for further fine-grained analysis, effectively reducing the weights of non-informative tokens.

A.2 Perturbation-based Token-level Scoring

Within the salient spans $\mathcal{S}_{\text{top-50\%}}$, we introduce the Relative Semantic Shift (RSS) to quantify the displacement in the embedding space. For an n-gram $\mathbf{g}$, we generate a perturbed sequence $\mathbf{X}_{\setminus \mathbf{g}}$ by masking its tokens. The RSS of $\mathbf{g}$ is defined as:

$$\text{RSS}(\mathbf{g}) = \frac{\left\| \phi(\mathbf{X}) - \phi(\mathbf{X}_{\setminus \mathbf{g}}) \right\|_2}{\|\phi(\mathbf{X})\|_2 + \left\| \phi(\mathbf{X}_{\setminus \mathbf{g}}) \right\|_2}, \tag{15}$$

where $\phi(\cdot)$ is the embedding function. To derive a robust token-level metric, we utilize a sliding window of size n with stride s. The Token-level RSS for token x_j, denoted as $\text{RSS}(x_j)$, is computed as the mean RSS of all n-grams $\mathcal{G}_j$ that cover x_j.

$$\text{RSS}(x_j) = \frac{1}{|\mathcal{G}_j|} \sum_{\mathbf{g} \in \mathcal{G}_j} \text{RSS}(\mathbf{g}). \tag{16}$$

Each RSS computation requires an additional forward pass on a perturbed sequence. The overall complexity scales as $\mathcal{O}(N_{\text{cal}} \cdot L\alpha/s)$, where N_{cal} is the number of calibration samples, L is the sequence length, α denotes the retained salient-region ratio, and s is the sliding-window stride. In practice, the computational overhead of this stage is modest. For example, in our longest sequence setting ($L = 512$), the sequence is partitioned into 16 spans of size 32, from which the top 50% salient spans are retained for token-level perturbation, yielding at most 256 candidate tokens. Using a 4-gram masking window with stride $s = 4$, the number of perturbed forward passes per sample is bounded by $256/4 = 64$, representing the maximum calibration cost in our implementation (e.g, the RSS calculation stage accounts for 3%–4% of the full pipeline time).

B Domain Partitioning

We partition the calibration set $\mathcal{D}_{\text{cal}}$ into D disjoint domains: $\mathcal{D}_{\text{cal}} = \bigcup_{d=1}^{D} \mathcal{D}_d$. Here, $N_d = |\mathcal{D}_d|$ denotes the cardinality of each domain, and x^d_n represents the n-th sample within domain d. Following the MTEB taxonomy, we use 30 fine-grained domains for calibration, comprising 14 programming-language domains and 16 natural-language domains. The complete domain list is provided in Table 6. This domain partitioning enables comprehensive domain profiling and provides the basis for the subsequent calibration phase.

Table 6: Domain partitioning of the calibration dataset $\mathcal{D}_{\text{cal}}$.

Category	**Specific Domains (d)**	**Domain Count (D)**
Code	c, c++, go, java, js, php, py, rb, rs, scala, sh, sql, swift, ts	14
Multilingual	ar, bn, en, es, fa, fi, fr, hi, id, ja, ko, ru, sw, te, th, zh	16
Total		**30**

C Overall Framework

Algorithm 1 shows the detailed pipeline of the proposed framework, which describes how token-level semantic importance is first computed, how neuron domain representations are constructed, and how these representations are used to derive structured expert specialization in the resulting MoE architecture.

As summarized in Algorithm 1, Den2MoEE is a unified domain-aware framework that couples token-level importance modeling, neuron specialization, and expert routing initialization under a shared semantic coherence principle. Token importance scores serve not only for filtering, but also as the driving signal for neuron grouping and expert formation, enabling a consistent abstraction hierarchy from tokens to neurons to experts.

Algorithm 1: Den2MoEE: Dense-to-MoE Conversion for Embedding Models

Input: Dense embedding model $\mathcal{M}$; calibration dataset $\mathcal{D}_{cal}$; number of experts E
Output: MoE embedding model $\mathcal{M}_{moe}$
Partition $\mathcal{D}_{cal}$ into semantic domains $\{\mathcal{D}_1, \ldots, \mathcal{D}_D\}$;

// Step 1: Hierarchical Token Importance Scoring
foreach *sample* $X \in \mathcal{D}_{cal}$ **do**
  Compute span-level score SCS(s_i);
  Compute token-level score RSS(x_j);
  Obtain TokenScore(x_j);

// Step 2: Neuron Domain Vector Construction
foreach *layer* ℓ **do**
  foreach *neuron* j **do**
    Compute ActScore(x_t, W_j^ℓ);
    Compute DomainScore(X^d);
    Construct neuron domain vector $\mathbf{v}_j^\ell$;

// Step 3: Expert Construction
foreach *layer* ℓ **do**
  Cluster neurons using constrained K-Means on $\{\mathbf{v}_j^\ell\}$;
  Form E experts $\{\mathcal{N}_1, \ldots, \mathcal{N}_E\}$;

// Step 4: Expert Profiling and Categorization
Compute expert domain vectors $\mathbf{V}_e^\ell$;
Compute expert DomainCoverage(e);
Assign experts as $\{$Shared Experts, SVD Experts, Null Expert$\}$;

// Step 5: Router Initialization
Construct routing anchors $\mathbf{A}_e^\ell$;
Initialize router using progressive semantic alignment;

// Step 6: Two-Phase Retraining
Dense warmup: activate all experts and train router;
Sparse retraining: enable Top-K routing and jointly optimize experts and router;
Update routing bias b_i for SVD experts during training;
return $\mathcal{M}_{moe}$

D Additional Experiment Settings

D.1 Training Dataset Composition

Table 7 details the composition of the 5.20M-instance retraining dataset across the code and multilingual categories. Specifically, the Code category contains data from CodeSearchNet (Husain et al., 2019), StackOverflow (Geigle et al., 2021), and CodeFeedback (Zheng et al., 2024), totaling 2.42M instances. The Multilingual category includes Swim-IR (Monolingual) (Thakur et al., 2024), MS-MARCO (Bajaj et al., 2016), MSMARCO-v2 (Enevoldsen et al., 2025), Mr. TyDi (Zhang et al., 2021), and PAWS-X (Yang et al., 2019), comprising 2.78M instances.

Table 7: Training dataset composition categorized by the code and multilingual data.

Category	Dataset	Size (M)
Code	CodeSearchNet	1.38
Code	StackOverflow	1.00
Code	CodeFeedBack	0.04
Multilingual	Swim-IR (Monolingual)	2.17
Multilingual	MSMARCO	0.17
Multilingual	MSMARCO-v2	0.26
Multilingual	Mr.TyDi	0.05
Multilingual	PAWS-X	0.13
	Total	**5.20**

D.2 Training Hyperparameters

Table 8 summarizes hyperparameter configurations used in the dense and sparse retraining stages.

Table 8: Hyperparameter configurations for dense and sparse training stages.

Hyperparameter	Dense Phase	Sparse Phase
#GPUs	8	16
Weight dtype	Bfloat16	Bfloat16
Learning rate	5e-5	1e-5
Learning rate scheduler	Cosine	Cosine
Warmup ratio	0.10	0.03
Optimizer	AdamW	AdamW
Global batch size	2048	2048
DeepSpeed	Zero-2	Zero-2

E Efficiency Evaluation Metrics

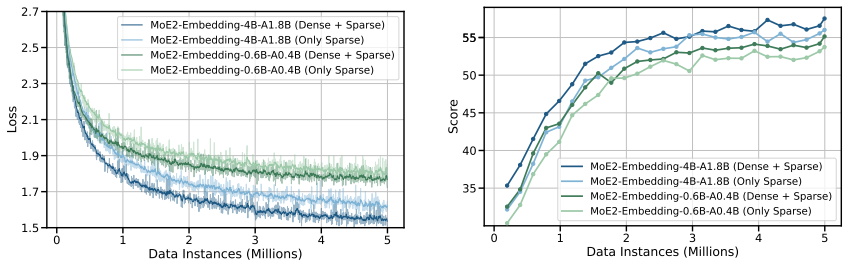
We evaluate deployment efficiency under a standard MTEB inference setting with sequence length 512 and batch size 128. We report end-to-end latency, throughput, and memory footprint, which provides a more practical assessment of the system-level benefits brought by MoE conversion beyond parameter reduction alone. Specifically, the throughput is calculated as:

$$\text{Throughput} = \frac{\text{BatchSize} \times 1000}{\text{avg latency (ms)}}. \tag{17}$$

F Additional Experiments

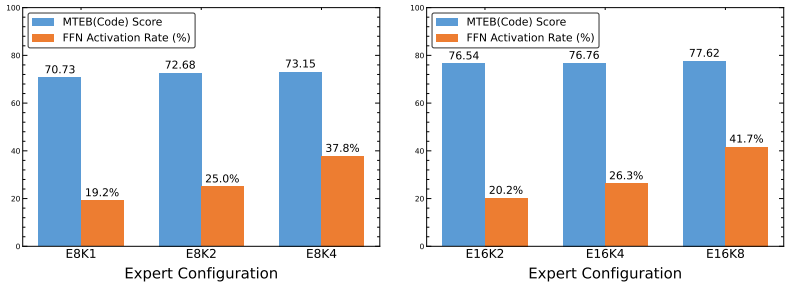
F.1 Effectiveness of Dense Retraining

To evaluate the role of the dense retraining phase in the Den2MoEE retraining pipeline, we perform a controlled ablation in which this stage is removed and the model is directly retrained under sparse



(a) Retraining loss (b) Perf. on MTEB(Multilingual)

Figure 4: Den2MoEE with and without dense retraining.



(a) Den2MoEE-0.6B-A0.4B (E8) (b) Den2MoEE-4B-A1.8B (E16)

Figure 5: Impact of Top-K Selection on performance and FFN activation rate. E8K2 denotes 8 experts with Top-2 routing.

routing (Only Sparse Retraining). We compare the full strategy (Dense $\rightarrow$ Sparse) with the ablated variant on two model scales, Den2MoEE-4B-A1.8B and Den2MoEE-0.6B-A0.4B. Using checkpoints saved every 100 steps, we track both training loss and MTEB multilingual performance, resulting in the curves shown in Figure 4.

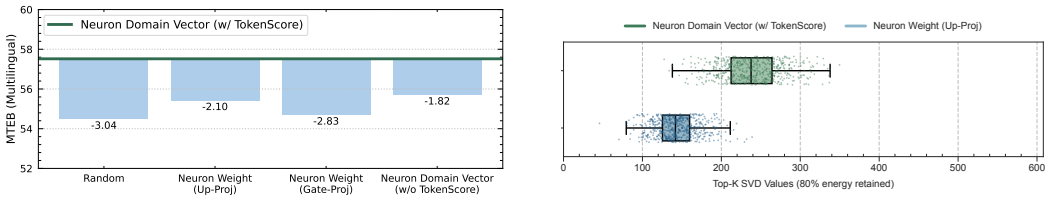
From Figure 4a and Figure 4b, we observe that introducing dense retraining consistently leads to faster loss reduction and earlier performance gains once sparse routing begins. Models with router warm-up start the sparse phase from a lower loss region and maintain a steeper descent trajectory, ultimately converging to both lower final loss and higher downstream performance. These improvements become increasingly pronounced with model scale, suggesting that larger Den2MoEE models are more sensitive to router initialization and early routing dynamics (4B-A1.8B $>$ 0.6B-A0.4B).

We attribute these improvements to the optimization role of router warm-up. Dense retraining enables experts and representations to co-adapt under full gradient flow, yielding a better-aligned feature–expert space before sparsification. It mitigates routing cold-start, reduces early-stage instability and expert load imbalance, and provides a smooth transition from dense sharing to sparse selection.

F.2 Impact of Top-K Selection on Performance and FFN Activation Rate in MoE

For a fixed computation budget, increasing the number of experts generally improves performance when model size and training data are appropriately scaled (Ludziejewski et al., 2025). However, prior work (Ludziejewski et al., 2024) shows that overly fine-grained expert partitioning under a fixed LLM scale may lead to training instability and performance degradation. Therefore, expert design requires balancing specialization and expert capacity.

To achieve such a balance, we split the FFN into 8 experts for the 0.6B model and 16 experts for the 4B model. These configurations provide a moderate level of expert granularity while maintaining sufficient parameter capacity within each expert. Based on these backbones, we further conduct ablation studies on the number of selected experts K (Figure 5). While larger K yields better performance by activating a broader set of experts, it also increases FFN activation rate and computational



(a) Comparison of clustering metrics

(b) SVD value distributions (80% energy retained)

Figure 6: Clustering metrics analysis on 4B MoE embedding model.

cost. Therefore, the objective is not to maximize sparsity alone, but rather to identify configurations that offer a favorable balance between FFN sparsity and model quality.

From Figure 5a, we observe that under the 0.6B setting, E8K2 achieves a strong trade-off between performance and FFN activation rate. Compared with E8K1, it increases the FFN activation rate by only 5.8% while already reaching performance close to E8K4. Although larger values of K provide additional gains, the improvement becomes progressively smaller relative to the increase in activated FFN parameters.

A similar trend can be observed for the 4B model (Figure 5b). E16K2 represents a more suitable operating point among the evaluated configurations. Increasing K from 2 to 4 introduces an additional 6.1% activation rate, whereas the corresponding performance improvement remains relatively limited. This indicates diminishing returns from activating more experts beyond a certain point. Consequently, E16K2 provides a reasonable compromise between computational efficiency and representation quality for the 4B backbone.

F.3 On the Role of Router Weights

For Den2MoEE-0.6B and Den2MoEE-4B, the RW embedding dimensions are 56 ($= 28/4 \times 8$) and 144 ($= 36/4 \times 16$), respectively.

Incremental concatenation and evaluation at the four-layer level, as shown in Figure 3, reveal a non-linear contribution profile, with a "shallow-layer latency" effect: the first 8 layers (0.6B) and 12 layers (4B) provide negligible gains, while performance inflection points occur in deeper layers (after layer 20 for 0.6B, layer 24 for 4B), indicating that deep routers encode high-level semantic information whereas shallow routers rely more on low-level syntactic or structural priors (Geva et al., 2021; Chen et al., 2022). Figure 3 also shows that RW embeddings complement hidden states by capturing more task-specific features in deeper layers, consistent with prior findings (Li and Zhou, 2025). And scaling model depth and expert count significantly enhances e_{rw}'s embedding power: Den2MoEE-4B (+2.16%) outperforms 0.6B (+1.38%), with a larger contribution from the last four layers (4B: +0.58% vs. 0.6B: +0.39%).

This shows that routers with more experts in MoE can partition the latent space more finely and better capture features not covered by hidden states.

F.4 Comparison with Different Clustering Metrics

To study the impact of clustering metrics on MoE reconstruction, we construct four 4B MoE embedding models using different clustering strategies, as shown in Figure 6a. All variants share the same architecture and training setup, differing only in the metric used to group neurons into experts. Thus, the observed differences stem mainly from the clustering signals.

Figure 6a shows that Domain Vector-based clustering outperforms weight-based clustering. Among weight vectors, Up-proj outperforms Gate-proj, as Up-proj prefers detecting input directions, while Gate-proj primarily controls linear combinations and output scaling, leading to near-random performance when used for clustering (Zhu et al., 2024). Removing TokenScore from Domain Vectors degrades performance, highlighting its role in suppressing irrelevant tokens in calibration data. Although the gains are modest, the relative order remains consistent, suggesting that the clustering metrics have a stable effect on expert reconstruction quality.

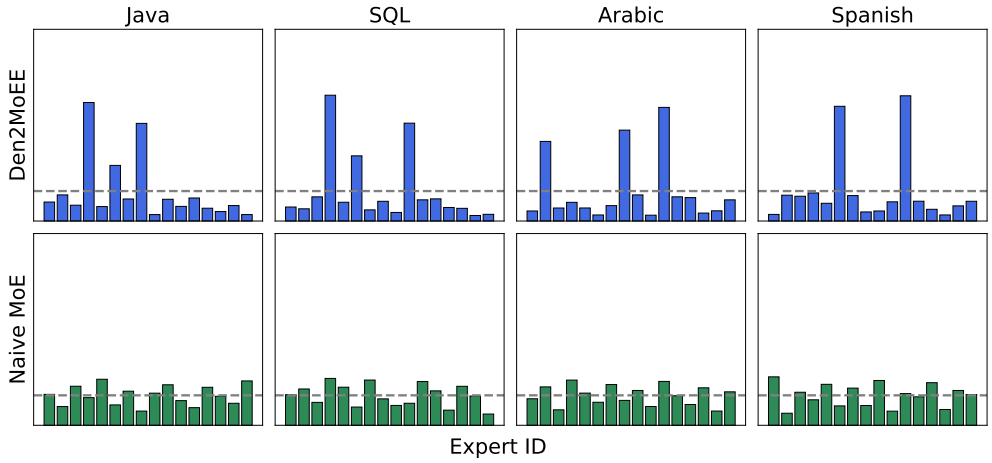


Figure 7: Expert routing activation probability distributions on the 4B (E16) model across selected domains. Native MoE's routing activation distribution tends to be uniform.

Table 9: MTEB (Multilingual v2) performance across different dense embedding backbones. † indicates models reconstructed using the LLaMA-MoE framework.

Model	Act-Size	Mean (Type)	Bitext Mining	Class-ification	Clus-tering	Inst. Retrieval	Multilabel Class.	Pair Class.	Rerank	Retrieval	STS
Harrier-oss-v1-0.6B	0.6B	59.00	82.85	73.88	54.00	0.81	26.37	82.07	63.16	70.75	77.09
Harrier-oss-v1-0.6B-MoE†	0.4B	54.63 (-4.37)	76.74	68.36	50.32	0.65	24.42	76.12	58.86	65.59	70.64
Den2MoEE-harrier-A0.4B	0.4B	56.19 (-2.81)	78.15	70.68	52.93	0.73	24.87	77.41	59.57	68.73	72.71
F2LLM-v2-0.6B	0.6B	55.02	70.31	64.06	56.58	1.39	25.23	78.95	65.14	59.30	74.22
F2LLM-v2-0.6B-MoE†	0.4B	53.17 (-1.85)	67.94	62.15	54.37	1.18	24.30	76.36	62.93	57.69	71.57
Den2MoEE-F2LLM-A0.4B	0.4B	53.83 (-1.19)	68.82	62.91	55.12	1.26	24.63	77.25	63.79	58.28	72.40

We further analyze the number of singular values required to retain 80% energy in expert Up-proj matrices clustered by Neuron Weight (Up-proj) and Domain Vectors (w/ TokenScore), as shown in Figure 6b. Box plots summarize the minimum number of singular values preserving 80% spectral energy for each expert. Weight-based clustering yields experts with pronounced low-rank structures, naturally arising from grouping neurons with similar weight patterns. In contrast, Domain Vector-based clustering produces experts with higher effective rank and broader spectral profiles. While effective rank does not directly determine downstream performance, the broader spectra of Domain Vector-based experts suggest richer representational variations and potentially greater capacity to preserve heterogeneous semantic patterns inherited from the original dense model during subsequent retraining.

F.5 Router Distribution Analysis

To further investigate the source of these gains, Figure 7 visualizes expert activation distributions across representative domains, including Java, SQL, Arabic, and Spanish. Compared with the Naive MoE design, which exhibits diffuse and ambiguous activation patterns across multiple experts, Den2MoEE produces significantly sparser and more concentrated routing distributions. The observed behavior indicates that the proposed heterogeneous expert architecture and router alignment strategy effectively reduce routing ambiguity and improve expert specialization. Consequently, semantically related tokens are more consistently assigned to domain-relevant experts, while redundant computation can be skipped through null experts. This leads to clearer sparsity patterns, more stable routing decisions, and ultimately a superior efficiency-performance trade-off.

F.6 Generalization Across Dense Embedding Backbones

To evaluate the generalization capability of Den2MoEE beyond a single backbone, we further conduct experiments on two representative open-source multilingual embedding models, Harrier-oss-

Table 10: Performance across code, multilingual, and legal domains. † indicates models reconstructed using the LLaMA-MoE framework.

Model	Law	Code	Multilingual
Qwen3-Embedding-4B	69.54	80.06	60.86
Qwen3-Embedding-4B-MoE†	66.92 (-2.62)	74.08 (-5.98)	55.47 (-5.39)
Den2MoEE-4B-A1.8B	68.19 (**-1.35**)	76.52 (**-3.54**)	57.63 (**-3.23**)

v1-0.6B (Microsoft, 2026) and F2LLM-v2-0.6B (Zhang et al., 2026), which achieve strong performance on recent multilingual MTEB benchmarks. Table 9 reports the results after converting the original dense models into sparse MoE architectures with an activation size reduced from 0.6B to 0.4B.

Across both backbones, Den2MoEE consistently achieves higher overall performance than the corresponding LLaMA-MoE-based reconstructions under the same activation budget. For Harrier-oss-v1, the average score decreases from 59.00 to 56.19 after conversion, corresponding to a performance drop of 2.81 points, whereas the LLaMA-MoE baseline incurs a larger decrease of 4.37 points. A similar trend can be observed on F2LLM-v2, where Den2MoEE reduces the performance degradation from 1.85 to 1.19 points. These results suggest that the proposed Dense-to-MoE conversion strategy is able to preserve a larger portion of the original model capability while maintaining the same level of sparsity.

Beyond the overall mean score, Den2MoEE exhibits improvements over the baseline across most evaluation categories. On both backbones, consistent gains are observed in Bitext Mining, Classification, Clustering, Pair Classification, Reranking, Retrieval, and STS tasks. Such improvements indicate that the semantic knowledge encoded in the dense model can be transferred to the reconstructed experts more effectively, leading to stronger retention of multilingual representation quality after conversion. It is also worth noting that the magnitude of performance degradation varies across backbones. Compared with Harrier-oss-v1, F2LLM-v2 experiences a smaller accuracy loss after conversion for both methods, suggesting that the effectiveness of Dense-to-MoE reconstruction may be influenced by properties of the original dense model. Nevertheless, Den2MoEE consistently provides a more favorable performance–efficiency tradeoff than the baseline across both model families.

F.7 Scalability to Additional Domains

To evaluate the scalability of Den2MoEE to additional domains, we extend the original code and multilingual reconstruction corpus with an additional legal domain. Specifically, we construct a mixed-domain dataset consisting of code, multilingual, and legal data, where the legal corpus contains approximately 1M samples collected from Law-GPT, THUCNews, and Zhihu-KOL legal content. The reconstructed models are evaluated on MTEB (Law), MTEB (Code), and MTEB (Multilingual), and the results are reported in Table 10.

Compared with LLaMA-MoE, Den2MoEE consistently achieves stronger performance across all domains, reducing the performance gap to the dense model from 2.62 to 1.35 on Law, from 5.98 to 3.54 on Code, and from 5.39 to 3.23 on Multilingual. These results indicate that the proposed domain-aware reconstruction framework remains effective as the number of target domains increases. More importantly, introducing approximately 1M additional legal samples does not noticeably compromise performance on the original Code and Multilingual tasks, suggesting that Den2MoEE can effectively incorporate new domain knowledge while maintaining strong knowledge recovery across existing domains. The improved Multilingual performance further suggests that expanding domain coverage may provide complementary training signals beyond the newly introduced domain.

These findings highlight the importance of recovery data design in MoE reconstruction. Beyond data volume, domain coverage and data composition can substantially influence reconstruction quality and cross-domain knowledge transfer. Understanding how reconstruction data should be selected and balanced to maximize knowledge recovery and expert specialization remains an important direction for future work.