# Beyond instruction-conditioning, MoTE: Mixture of Task Experts for Multi-task Embedding Models

**Miguel Romero**[1,2,*], **Shuoyang Ding**[1],
**Corey D. Barret**[1], **Georgiana Dinu**[1], **George Karypis**[1,2,*]

[1]Amazon, [2]University of Minnesota

## Abstract

Dense embeddings are fundamental to modern machine learning systems, powering Retrieval-Augmented Generation (RAG), information retrieval, and representation learning. While instruction-conditioning has become the dominant approach for embedding specialization, its direct application to low-capacity models imposes fundamental representational constraints that limit the performance gains derived from specialization. In this paper, we analyze these limitations and introduce the Mixture of Task Experts (MoTE) transformer block, which leverages task-specialized parameters trained with Task-Aware Contrastive Learning (TA-CL) to enhance the model ability to generate specialized embeddings. Empirical results show that MoTE achieves $64\%$ higher performance gains in retrieval datasets ($+3.27 \rightarrow +5.21$) and $43\%$ higher performance gains across all datasets ($+1.81 \rightarrow +2.60$). Critically, these gains are achieved without altering instructions, training data, inference time, or number of active parameters.

## 1 Introduction

Semantic text representations are a key component to many real-world applications such as search, recommendation systems, and spam classification. These representations are commonly obtained using embedding models that map unstructured text into a dense $n$-dimensional vector referred to as dense embedding. Contemporary embedding models are trained by simultaneously optimizing the model towards a wide range of downstream tasks including Retrieval Augmented Generation (RAG) (Gao et al., 2023; Lewis et al., 2020), search (Wise et al., 2020), Semantic Text Similarity (STS) (Chandrasekaran and Mago, 2021), classification (O'Neill et al., 2021; Wang et al., 2021), and clustering (Xu et al., 2015). As a result, a single embedding model is able to generate a dense text representation that can be used in all these downstream tasks.

Historically, dense embedding models have been trained using a multi-task curriculum, where a single model is optimized across multiple datasets, each corresponding to a specific task such as clustering or retrieval. In this framework, a shared embedding space is used across all tasks, allowing a single representation to serve multiple downstream applications. However, different tasks impose distinct semantic similarity requirements, often leading to conflicting objectives. For instance, in a semantic textual similarity (STS) task, the questions “Who was Isaac Newton?" and “Who was the father of Calculus?" should be close in the embedding space since they refer to the same person. In contrast, a retrieval task requires these queries to be far apart, as they can not answer each other (Gao et al., 2021). This presents a fundamental challenge in embedding model development, where a single embedding space must be carefully tuned to support multiple tasks without compromising performance on any of them (Muennighoff et al., 2022; Neelakantan et al., 2022). To address this, modern embedding models have shifted[1] toward generating task-specific embeddings, allowing representations to be adapted based on the downstream application. The consistent performance gains observed from embedding specialization, coupled with its widespread adoption in state-of-the-art models (Wang et al., 2022; Su et al., 2022) have provide strong evidence of the limitations of generic embeddings in downstream tasks.

The predominant approach for embedding specialization is Instruction-Conditioning (IC) (Wang et al., 2022; Su et al., 2022), where the input text

[1]Corresponding authors, emails: romer333@umn.edu, karypis@umn.edu

[1]`https://huggingface.co/spaces/mteb/leaderboard`

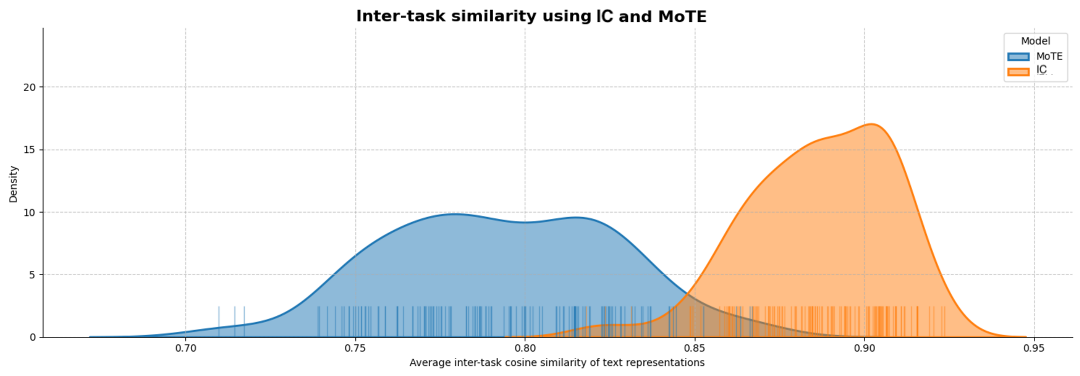

Figure 1: Average inter-task cosine similarity between embedding representations for the same sequence. Results show that compared to Instruction-Conditioning (IC), Mixture-of-Task Experts (MoTE) provides more flexibility (lower cosine-similarity) generating specialized embeddings.

is enriched with task-specific instructions to steer the generation of specialized embeddings. For instance, Wang et al. (2022) employs the instruction "query:" for queries and "passage:" for passages in retrieval tasks, providing the model with information to distinguish between different roles in the retrieval tasks. With this approach, the specialization signal is entirely encoded in the instruction tokens, which propagate through the model to produce the final embeddings.

However, theoretical properties of neural networks, such as Lipschitz continuity, suggest that small changes in input—like adding short instruction tokens—produce only minor shifts in output embeddings (Tang et al., 2024), limiting a model's ability to create disentangled, task-specific representations. This constraint is amplified in smaller models and with short prompts, where low-dimensional embeddings face capacity bottlenecks (Tishby et al., 2000), forcing trade-offs between preserving semantic and task-relevant information. These limitations are compounded during multi-task contrastive training, where gradient interference (Yu et al., 2020) and conflicting objectives (Ravi et al., 2020) saturate model capacity, and where hyper-parameters like batching strategy and contrastive temperature must be carefully tuned to align with downstream task semantics, further complicating effective task-specific embedding specialization.

In this paper, we study these limitations in the context of multi-task embedding models and propose alternative methods to enhance embedding specialization. Our key contributions include:

1. We provide empirical evidence that relying solely on instruction-conditioning can constrain the adaptability of embeddings across diverse tasks. We analyze task-specific performance trade-offs and demonstrate how instruction-conditioned models struggle to fully disentangle task representations in low-capacity settings (Figure 1).

2. We introduce *Mixture of Task Experts (MoTE)*, a Mixture of Experts (MoE) block with task-specialized experts.

3. We propose *Task-Aware Contrastive Learnign (TA-CL)*, a novel training curriculum to tailor the training of MoTE's experts to their associated downstream task.

Experiments conducted on 56 datasets across 7 tasks (Muennighoff et al., 2022) demonstrate that leveraging MoTE and TA-CL enables more effective utilization of task information for embedding specialization (Figure 1). This approach leads to 50% higher performance gains in critical tasks such as retrieval and 31% higher gains in other tasks compared to instruction-conditioned specialization. Critically, these improvements are achieved with identical input information, training data, latency, and number of active parameters. Furthermore, exploiting the observation that embedding models are typically used for one downstream task at a time (e.g., retrieval-indexing, retrieval-querying, classification), MoTE can maintain the same GPU memory footprint as instruction-conditioned models by offloading inactive task experts to CPU or disk. This ensures computational efficiency without sacrificing performance.

## 2 Background

### 2.1 Text Embedding Models

Embedding models $f_\Theta$ map elements from the sequence space $\mathcal{S}$ to $\mathbb{R}^N$. Typically initialized from encoder-only pre-trained checkpoints like BERT (Devlin et al., 2018) or RoBERTa (Liu et al., 2019), they are trained via contrastive learning on multiple datasets containing pairs of sequences $(a, p)$ that should be close to each other in the embedding space (e.g, query-passage pairs in retrieval, passage-passage pairs in clustering, etc.). In this context a common contrastive loss is InfoNCE (Oord et al., 2018):

$$\mathcal{L}(f_\Theta; a, p, P^*) = -\log \frac{e^{\gamma(u_a, u_p)}}{\sum_{d \in \{p, P^*\}} e^{\gamma(u_a, u_d)}}$$

where $P^*$ denotes the other positive in the mini-batch, $u_s$ is the embedding obtained by embedding the sequence $s$, and $\gamma$ is a similarity measure such as cosine-similarity (Han et al., 2012). This optimizes $\Theta$ by pulling similar sequences closer and pushing dissimilar ones apart, with the definition of similar being task-dependent.

### 2.2 Instruction-Conditioned Embeddings

IC is a technique (Wang et al., 2024; Muennighoff et al., 2024; Wang et al., 2023; Zhang et al., 2023) to generate specialized embeddings with a single model by concatenating to the input sequence a text string that describes the downstream use case. This concatenation happens during training and inference. Thus, IC models can be defined as functions $f_\Theta^I = f_\Theta \circ \phi$ over $\mathcal{I} \times \mathcal{S}$ of the set of instructions $\mathcal{I}$ and sequences $\mathcal{S}$ where $\phi$ is the concatenation operation in the sequence space and $f_\Theta$ is an embedding model.

Methods leveraging this strategy vary in the level of detail provided in the instructions. Small models (Wang et al., 2022) use a pre-defined set of instructions with task level information such such as “query:" and “passage:" in retrieval use cases. Large models (Su et al., 2022) use a free-form set of instructions with dataset-specific information such as “Represent the Amazon comment for classifying the sentence as positive or negative:".

### 2.3 Mixture-of-Experts

MoE has been a long-standing technique (Yuksel et al., 2012) that proposes a divide-and-conquer approach by leveraging multiple expert modules and an routing mechanism to identify the most appropriate expert for any given input. A recent application of this approach is the transformer MoE block (Lepikhin et al., 2020; Jiang et al., 2024; Gao et al., 2022) which leverages multiple Multi-Layer Perceptrons (MLP) as experts and a learned token-level mechanism to route individual tokens through multiple experts. Specifically, for each token, the routing mechanism $g$ uses the information in the intermediate token representation to identify the subset of experts that should process the representation (Lepikhin et al., 2020). Replicas of the intermediate token representation are then dispatched to the selected experts where they are processed independently. Lastly, a pooling layer aggregates the experts' output to generate a unique representation for each of the tokens. During training, an auxiliary expert-balancing objective is leveraged to learn the routing mechanism while ensure a consistent training of and a balanced workload across the experts modules (Jiang et al., 2024; Gao et al., 2022).

## 3 Method

While instruction-conditioning enables the generation of specialized embeddings $f_\Theta^{\mathcal{I}}(i, s)$, relying solely on the input instruction to steer the specialization of the embedding while the other parameters are shared can limit the models ability to generate specialized embeddings (Ravi et al., 2020; Yu et al., 2020). To overcome this constraints we propose to introduce task specialized parameters to increase the model's capacity to specialize embeddings while maintaining the overall number of active parameters. Formally, we propose a function $\Psi^I_{[\Theta_0|\Theta_I]}$ such that:

$$\begin{aligned} \Psi^I_{[\Theta_0|\Theta_I]} : I \times S &\longrightarrow \mathbb{R}^N \\ (\mathbf{i}, \mathbf{s}) &\longmapsto f^i_{[\Theta_0|\Theta_i]}(i, s) = v_i \end{aligned}$$

where

$$\begin{aligned} f^i_{[\Theta_0|\Theta_i]} : \{i\} \times S &\xrightarrow{\phi} S \xrightarrow{f_{[\Theta_0|\Theta_i]}} \mathbb{R}^N \\ (\mathbf{i}, \mathbf{s}) &\longmapsto \mathbf{i}|\mathbf{s} \longmapsto v_i \end{aligned}$$

Consistent with IC, $\Psi^I_{[\Theta_0|\Theta_I]}$ accepts tuples $(i, s)$ where $i \in \mathcal{I}$ represents the task instruction providing contextual information about the downstream task (e.g., “query: ”, “passage: ”) and $s$ denotes the input sequence. However, unlike standard instruction-conditioning, $\Psi^{I}_{[\Theta_0|\Theta_I]}$ dynamically selects the appropriate set of specialized parameters $\Theta_i$ to route the input sequence $[i|s]$ and enhancing task specialization.

## 3.1 Architecture

We implement $\Psi^{I}_{[\Theta_0|\Theta_I]}$ using the Mixture of Task Experts (MoTE) transformer block (Figure 2a). MoTE implements the specialized parameter sets $\Theta_I$ with different expert modules and leverages a task-based routing mechanism to process input sequences through different experts based on its intended downstream use.

MoTE's instruction-based routing mechanism $R : \mathcal{I} \rightarrow \mathcal{E}$ maps an instruction $i \in \mathcal{I}$ to an expert module $e \in \mathcal{E}$, which is then applied to the intermediate state of $[i|s]$. This approach offers two key benefits for embedding specialization. First, unlike traditional MoE routing, experts are trained exclusively on task-relevant examples, leading to more efficient training and better specialization. Second, MoTE enhances the model's capacity for embedding specialization without incurring additional latency costs by routing full sequences through a single expert rather than distributing tokens across multiple experts. Furthermore, in most practical applications, MoTE avoids increased GPU memory usage, as inactive experts can be offloaded to the CPU or disk, ensuring efficient resource utilization.

Each expert block $e \in \mathcal{E}$ consists of a MultiLayer Perceptron (MLP) and two normalization layers (Figure 2b). The MLP enables specialized transformations for different tasks, while the normalization layers allow MoTE to learn distinct centroids and variances for each task (Ba, 2016), improving task adaptability.

### 3.1.1 Initialization

MoTE's task-routing mechanism requires task-augmented inputs to train the task experts. However, datasets associated with downstream embedding task are only available during contrastive learning while generic pre-training relies on generic task-agnostic sequences.

To enable MoTE to enhance the model's specialization capacity while leveraging existing pre-trained checkpoints we up-cycle (Komatsuzaki et al., 2022; He et al., 2024) the dense pre-trained blocks into MoTE blocks before the contrastive learning stage. Specifically, we follow a three-step process. First, we select the dense transformer blocks to be converted into MoTE blocks. Second, for each selected block, we instantiate $|\mathcal{E}|$ experts using the same MLP configuration as the original dense transformer block. Last, we initialize all experts in $\mathcal{E}$ with a copy of the dense transformer's MLP weights. This approach ensures that the model retains knowledge from its pre-trained checkpoint while gaining the flexibility of task-specialized experts.

## 3.2 Training

With MoTE's instruction-based routing, training examples are directed to their corresponding task experts, ensuring that task-specific gradients are only backpropagated through the relevant expert. However, different downstream tasks have distinct representation requirements, influencing training nuances such as batching strategy and contrastive temperature (Appendix A). This poses a challenge because contrastive learning relies on negatives from the same mini-batch to compute the loss, yet different experts require slightly different training configurations.

To overcome this challenge, we introduce Task-Aware Contrastive Learning (TA-CL), which dynamically adjusts training configurations based on the selected expert. TA-CL leverages a hierarchical data structure that organizes training samples by task and dataset. During mini-batch construction, a single task is selected, and examples are drawn from one or more of the task's datasets according to that task's batching strategy. Additionally, task metadata is passed with the mini-batch to the training loop to adjust the tailor the contrastive objective to the task. When combined with MoTE, this approach ensures that each task expert is trained with a configuration suited to its specific downstream task.

## 4 Experimental Methodology

We evaluate performance using the Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2022), which spans 7 tasks and 56 datasets (see Table 1). Following MTEB's evaluation protocol, we use the same metrics and procedures, reporting the average performance per task and overall across all tasks.

We evaluate the embedding specialization benefits derived from Instruction-Conditioning (IC) and MoTE on Nussbaum et al. (2024)'s model[2].

[2]Within top-10 of the 100-250M parameter models as of 02/08/2025

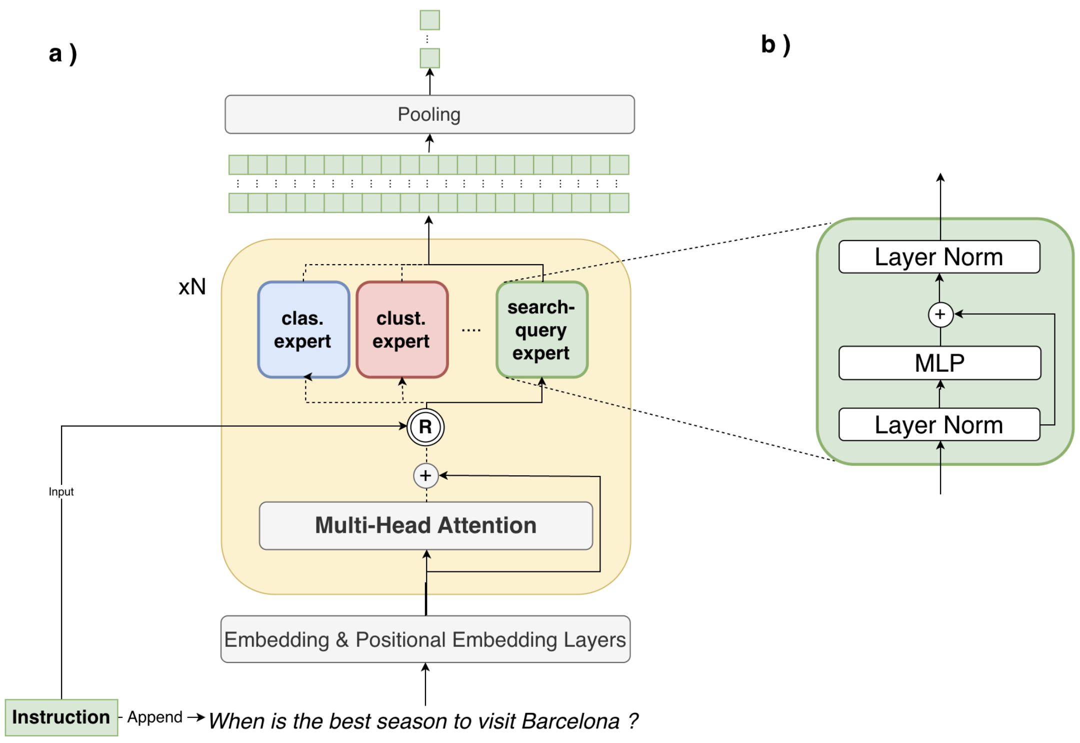

Figure 2: a) Overview of the MoTE block, which replaces the standard transformer block in MoTE. Each MoTE block contains multiple task-specific experts and a routing mechanism that selects the appropriate expert based on the task instruction. b) Detailed view of a single expert's internal structure, consisting of a layer normalization followed by a MLP network and a layer normalization, mirroring the standard transformer feedforward configuration but applied independently within each expert.

Table 1: Metrics and distribution of datasets across tasks in the MTEB benchmark.

| Task | # Datasets | Metric |
|---|---|---|
| Retrieval | 15 | NDCG@10 |
| Class. | 12 | Accuracy |
| Clustering | 11 | V-Measure |
| Re-ranking | 4 | MAP |
| Pair Class. | 3 | AP |
| STS | 10 | Spearman corr. |
| Summ. | 1 | Spearman corr. |

For more information about the metrics please refer to Appendix C.

Both candidates are initialized using the same pre-trained checkpoint[3], and trained on the same classification, clustering and retrieval datasets [4] during contrastive learning [5]. All candidates are trained for a single epoch using the InfoNCE objective (Oord et al., 2018) and the AdamW optimizer with a learning rate of $5 \times 10^{-6}$, a weight decay of $0.1$ and a batch size of $6,144$.

The IC and MoTE candidates leverage the same instructions: "classification: ", "clustering: ", "search query: ", and "search document: ". Additionally, the MoTE candidate is implemented by using MoTE blocks in every transformer block and four experts in each transformer block, each of which is associated with one of the above instructions. Further implementation details are provided in Appendix D.

Additionally, to assess the performance benefits of multi-task embedding models with task-specific experts compared to specialized single-task models, we train and evaluate three Specialized Embedding Models (SEM). Each SEM uses the same underlying transformer architecture as the EM and IEM candidates but is trained independently on one of the classification, clustering, or retrieval subsets of the training data. Performance is then reported for each SEM on its corresponding downstream task.

[3]https://huggingface.co/nomic-ai/nomic-bert-2048

[4]https://github.com/nomic-ai/contrastors/blob/main/src/contrastors/configs/data/contrastive_pretrain.yaml

[5]Dataset source: https://huggingface.co/datasets/sentence-transformers/embedding-training-data

Table 2: Comparison of performance gains from different embedding specialization approaches.

| Task | EM | SEM | IC | MoTE |
|---|---|---|---|---|
| Retrieval [1] | 42.31 | -8.86 | +3.27 | **+5.21** |
| Class. [2] | 65.37 | -1.29 | +3.37 | **+3.79** |
| Clustering [3] | 42.95 | -3.84 | **+0.14** | -0.06 |
| Average[4] | 49.78 | -5.02 | +2.35 | **+3.23** |

Comparison of performance gains of Specialized Embedding Models (SEM), Instruction-Conditioning (IC) and MoTE with respect to the non-specialized Embedding Model (EM) on tasks seen during training. [1] NDCG@10; [2] accuracy; [3] validity measure; [4] Score average across 38 datasets.

## 5 Results

### 5.1 Performance across seen tasks

This section explores the performance of MoTE in tasks that were used during training, namely retrieval, classification and clustering (we will refer to those as *seen* tasks). Table 2 shows the absolute average dataset performance of the generic embedding model as well as the relative gains derived from each of the specialization methods at the task level and across all seen tasks.

In retrieval, MoTE achieves a +5.21 improvement, surpassing IC's +3.27. Similarly, in classification, MoTE shows a +3.79 gain compared to IC's +3.37. Overall, MoTE achieves the highest average dataset performance improvement (+3.23), exceeding IC's +2.35, while maintaining latency and memory usage. Additionally, while MoTE improves performance on clustering tasks relative to SEM, its gains are modest compared to other tasks. This appears to stem from the high similarity between search-document and clustering embedding spaces and is further investigated in Section 5.4.

Notably, the Specialized Embedding Models (SEM), which are independently trained for each task, consistently underperform relative to multi-task embedding models. These findings highlight that MoTE 's task-specialized expert architecture provides more effective embedding specialization than instruction-conditioning approaches, improving performance across diverse downstream tasks.

### 5.2 Inter-task similarity analysis

Our initial hypothesis was that relying solely on instructions constrains the model's ability to disentangle specialized embeddings, resulting in greater similarity between embeddings across tasks. To test this, we analyze and compare the specialized embeddings produced by MoTE and IC when trained on the same data and optimization process. If IC has enough capacity to disentangle the specialized embeddings, increasing the model's specialization capacity (MoTE) would not lead to a significant different in the similarities between embedding specializations. On the other hand, if IC constrained the model's capacity to generate disentangled specialized embeddings, training the model with the same data and optimization process but with higher specialization capacity (MoTE) will lead to larger differences (lower similarity) between the specialized embeddings.

To test this hypothesis we randomly 128 randomly generic Wikipedia articles[6] and compute their specialized embeddings for clustering, classification, retrieval-document, and retrieval-query tasks. We then measure the pairwise (inter-task) cosine similarity among these embeddings for both MoTE and IC where higher cosine similarity values indicate higher similarity between task-specialized embeddings. To assess statistical significance, we conduct a one-sided Welch's $t$-test for each pairwise task comparison, following the hypothesis formulated in Expression 1, where $\mu_{\mathcal{M}}^{(T_1,T_2)}$ represents the true cosine similarity between tasks $T_1$ and $T_2$ for method $\mathcal{M}$..

$$\begin{aligned} H_0 &: \mu_{MoTE}^{(T_1,T_2)} = \mu_{\text{IC}}^{(T_1,T_2)} \\ H_A &: \mu_{MoTE}^{(T_1,T_2)} < \mu_{\text{IC}}^{(T_1,T_2)} \end{aligned} \quad (1)$$

Figure 3 shows the inter-task similarity distributions for MoTE and IC alongside their $t$-test p-value for each of the tasks combinations. We observe that MoTE is able to consistently achieves a lower degree of similarity between its specialized embeddings than IC leading to up to 0.3 lower cosine similarity in some scenarios. These results, alongside the performance improvements in Section 5.1, highlights the limitations of IC to tailor embeddings to a specific downstream use and its impact in downstream task performance. Interestingly we observe that on comparison between search document and clustering the inter-task similarity of MoTE and IC is not significant different which indicate that the representation for both downstream uses is not significantly different and therefore leveraging the same expert with IC might

[6]Dataset available at: https://huggingface.co/datasets/sentence-transformers/embedding-training-data/resolve/main/SimpleWiki.jsonl.gz

Table 3: Performance of different methods across tasks not used during training

| Task | EM | SEM | IC | MoTE |
|---|---|---|---|---|
| Re-ranking[1] | **55.38** | -7.35 | -2.01 | -0.45 |
| Pair Class.[2] | 81.52 | -8.22 | -0.8 | **+0.29** |
| STS[3] | 77.91 | -2.82 | +2.11 | **+2.15** |
| Summa.[3] | 32.31 | -5.86 | +1.45 | **+2.01** |
| Average [4] | 70.97 | -4.90 | +0.67 | **+1.25** |

Comparison of performance gains of instruction-conditioning (IC) and MoTE over the non-specialized Embedding Model (EM). [1] MAP; [2] AP; [3] Spearman correlation; [4] Score average across 18 datasets.

Table 4: Comparison of MoTE's performance when trained with static or task aware training.

| Task | Static | TA-CL |
|---|---|---|
| Retrieval[1] | 46.48 | **47.52** |
| Classification[2] | 69.00 | **69.16** |
| Clustering[3] | 42.82 | **42.89** |
| Re-ranking[4] | 54.69 | **54.93** |
| Pair Classification [5] | **81.93** | 81.81 |
| STS[6] | 79.70 | **80.03** |
| Summarization[6] | 34.06 | **34.32** |
| Avg. dataset performance | 58.78 | **59.19** |

TA-CL employs homogeneous batching for retrieval mini-batches and heterogeneous batching in classification and clustering mini-batches and a contrastive temperature of 0.03 for retrieval and classification mini-batches and 0.06 for clustering mini-batches. Static training leverages heterogeneous batching and a 0.03 contrastive temperature. [1] NDCG@10; [2] accuracy; [3] validity measure; [4] MAP; [5] AP; [6] Spearman correlation.

suffice. We leave the exploration of the consolidation of experts for future research.

### 5.3 Generalization to unseen tasks

Conditioning embedding models on specific instructions to tailor text representations for a finite set of tasks risks creating overly specialized embedding spaces. This risk is further accentuated by MoTE, which leverages specialized parameters for each task. This section explores the performance of MoTE in tasks that were not used during training, namely re-ranking, pair-classification, Semantic Text Similarity (STS), and summarization (we will refer to those as *unseen* tasks). Similar to (Nussbaum et al., 2024), we associate unseen tasks to tasks seen during training. Specifically for this experiment we associate re-ranking dataset to the retrieval task and pair classification, STS and summarization datasets to the classification task.

Results in Table 3 shows that MoTE outperforms IC in most cases, achieving the highest gains in STS (+2.15) and Summarization (+2.01), demonstrating its adaptability to unseen tasks. In Pair Classification, MoTE maintains a slight advantage (+0.29), while IC shows a small drop (-0.8) with respect to the un-specialized model. Re-ranking is the only task where both approaches underperform relative to EM, likely due to a distribution shift between retrieval training data and the re-ranking evaluation datasets. Notably, several re-ranking datasets include atypical query and passage formats—such as unusually long queries or question deduplication samples—which differ from standard retrieval training scenarios. This suggests that re-ranking benefits from the broader, non-specialized embedding space of EM over task-specialized models. Overall, MoTE achieves the highest average performance gains (+1.25), highlighting its effectiveness in generalizing to new tasks while maintaining strong performance.

### 5.4 Ablation: Impact of TA-CL

To assess the benefits of TA-CL we consider ablate both training regimes when considering the batching strategy and contrastive temperature as task-aware configurations (Appendix A).

Table 4 shows that TA-CL leads to consistent performance improvements of the resulting model with the higher relative gains on tasks in which the regime had to be compromised in the static curriculum such as retrieval where using heterogeneous sampling resulted on 46.48 but by leveraging homogeneous sampling in TA-CL the performance rises to 47.52. Overall, we observe that TA-CL improves average dataset performance by +0.41 across all 56 MTEB datasets. To assess the statistical significance of these gains, we conducted a one-sided Welch's t-test comparing TA-CL and static training outcomes across datasets, obtaining a $p$-value of $1 \times 10^{-4}$, well below the standard 0.05 threshold. This provides strong evidence that TA-CL 's improvements over static contrastive learning are statistically significant.

## 6 Ablation: MoTE vs MoE routing

Unlike MoE's learned Token-Level Routing (TLR), MoTE employs Sequence-Level Routing (SLR), directly utilizing task information to select the appropriate expert for processing the entire input sequence. This approach allows MoTE to maintain

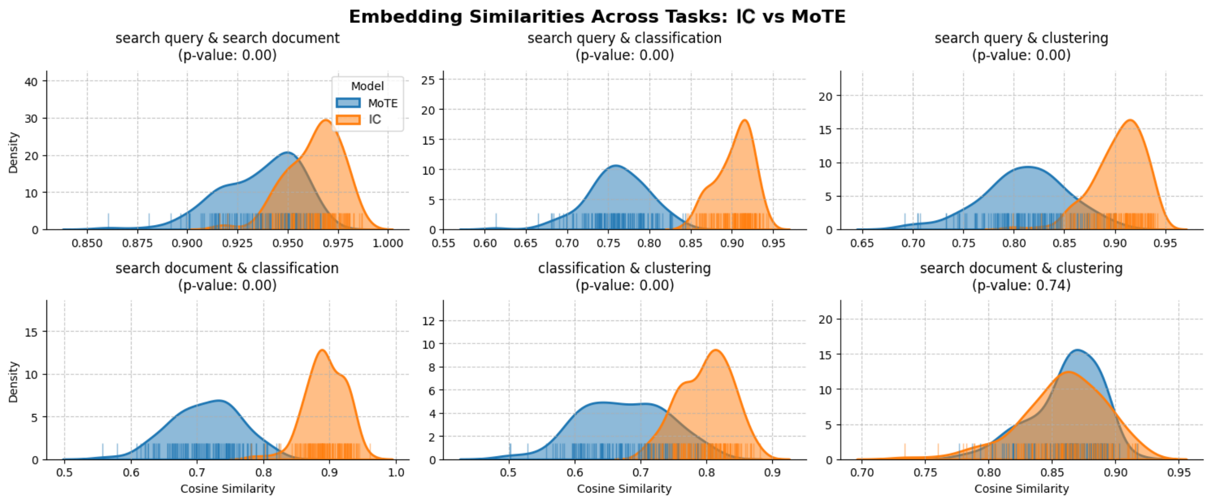

Figure 3: Inter-task similarity of text representation when using IC and MoTE. We observe that in most cases MoTE is able to reach a higher degree independence across tasks after the multi-task contrastive learning stage.

Table 5: Performance of different routing mechanisms

| Task | TLR | SLR |
|---|---|---|
| Retrieval[1] | 45.17 | **47.52** |
| Classification[2] | 67.98 | **69.16** |
| Clustering[3] | 42.69 | **42.89** |
| Reranking[4] | 54.40 | **54.93** |
| Pair Classification[5] | 80.84 | **81.81** |
| STS[6] | 78.47 | **80.06** |
| Summarization[6] | 32.67 | **34.32** |
| Average[7] | 57.86 | **59.19** |

Performance comparison between MoE's Task-Level Routing (TLR) and MoTE's Sequence-Level Routing (SLR).
[1] NDCG@10; [2] accuracy; [3] validity measure; [4] MAP; [5] AP; [6] Spearman correlation;
[7] Average performance across 56 datasets.

the processing efficiency of IC while training experts on task-specific data. In this ablation study, we compare the effectiveness of SLR against TLR by modifying the routing mechanism in each transformer block while keeping the number of experts constant. To ensure a fair comparison, we augment the input with task-specific instructions ("classification: ", "clustering: ", "search query: ", and "search document: ") so that TLR can also incorporate downstream task information during routing.

Table 5 shows that SLR consistently outperforms TLR across all tasks, with the largest improvements in Retrieval (+2.35), Semantic Textual Similarity (+1.59), Summarization (+1.65), and Pair Classification (+0.97). By leveraging task information at the sequence level, SLR enables more effective expert specialization, resulting in an overall average performance increase of +1.33 across all 56 MTEB datasets. To assess the robustness of this improvement, we conducted a one-sided Welch's t-test comparing SLR and TLR results across datasets, obtaining a $p$-value of $3 \times 10^{-2}$, confirming that SLR's performance gains over TLR are statistically significant at the $0.05$ level.

## 7 Ablation: MoE architecture design

Contemporary MoE literature introduces two strategies for integrating MoE blocks into transformer architectures, each offering a different balance between local and global parametrizations. Specifically, MoE blocks can be incorporated into Every transformer Block (EB) (Fedus et al., 2022) or at Every Other Block (EOB) (Lepikhin et al., 2020). In this experiment, we evaluate the performance impact of these design choices when extending embedding architectures with MoTE blocks.

Table 6 reveals that integrating MoTE blocks at EB versus EOB leads to minimal performance differences across tasks. While EB achieves a slightly higher average dataset performance (59.19 vs. 59.07), the gains are marginal across retrieval, classification, clustering, re-ranking, and summarization tasks. Notably, EOB slightly outperforms EB in pair classification, whereas EB leads in retrieval, STS, and summarization. These results suggest that while marginal, introducing MoTE blocks at EB results in overall better performance compared to doing so at EOB.

Table 6: Performance comparison across different integrations of MoTE blocks

| **Task** | **EB** | **EOB** |
|---|---|---|
| Retrieval[1] | **47.52** | 47.20 |
| Classification[2] | 69.16 | **69.17** |
| Clustering[3] | **42.89** | 42.81 |
| Re-ranking[4] | **54.93** | 54.91 |
| Pair Classification[5] | 81.81 | **81.83** |
| STS[6] | **80.06** | 79.96 |
| Summarization [6] | **34.32** | 33.97 |
| Average[7] | **59.19** | 59.07 |

Comparison between integrating MoTE blocks in Every transformer Block (EB) or in Every-Other transformer Block (EOB). [1] NDCG@10; [2] accuracy; [3] validity measure; [4] MAP; [5] AP; [6] Spearman correlation;
[7] Average performance across 56 datasets.

## 8 Conclusion

In this paper, we identified the limitations of instruction conditioning for embedding specialization and introduced MoTE and TA-CL as alternative approaches to overcome these constraints. Our results demonstrate that MoTE significantly improves overall performance—particularly in critical tasks such as retrieval—without increasing computational costs, inference latency, or the number of active parameters. By leveraging task-aware contrastive learning and task-based routing, MoTE enhances embedding specialization while maintaining computational efficiency.

Additionally, our findings suggest that a hybrid approach—combining MoTE with instruction conditioning—could be more effective than either method alone. While MoTE improves task disentanglement, we observed that in certain cases, preserving synergies between related tasks is equally important for performance. For example, Table 2 shows that MoTE negatively impacts clustering performance, while Figure 3 indicates that separating clustering from search document tasks does not yield significant benefits. These results highlight the need for architectural modifications that balance task specialization with synergy. We leave the study of these relationships amongst downstream tasks and exploration of hybrid approaches integrating single and multi-task experts for future research.

## Limitations

While our approach does not increase inference latency or active computational costs, scaling MoTE to larger architectures or broader task distributions may introduce new challenges in expert selection and routing efficiency. Future work will explore the generalization of MoTE to larger embedding models and further our understanding of task disentanglement and synergy.

By systematically studying these trade-offs, we aim to develop embedding models that maximize specialization while maintaining computational efficiency. Exploring adaptive expert selection mechanisms and dynamic task-aware routing could further enhance the versatility and robustness of multi-task embedding models, paving the way for more effective real-world applications.

## References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada. Association for Computational Linguistics.

Jimmy Lei Ba. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 89–96, New York, NY, USA. Association for Computing Machinery.

Dhivya Chandrasekaran and Vijay Mago. 2021. Evolution of semantic similarity—a survey. *ACM Computing Surveys (CSUR)*, 54(2):1–37.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Ze-Feng Gao, Peiyu Liu, Wayne Xin Zhao, Zhong-Yi Lu, and Ji-Rong Wen. 2022. Parameter-efficient mixture-of-experts architecture for pre-trained language models. *arXiv preprint arXiv:2203.01104*.

Jiawei Han, Micheline Kamber, and Jian Pei. 2012. 2 - getting to know your data. In Jiawei Han, Micheline Kamber, and Jian Pei, editors, *Data Mining (Third Edition)*, third edition edition, The Morgan Kaufmann Series in Data Management Systems, pages 39–82. Morgan Kaufmann, Boston.

Ethan He, Abhinav Khattar, Ryan Prenger, Vijay Korthikanti, Zijie Yan, Tong Liu, Shiqing Fan, Ashwath Aithal, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Upcycling large language models into mixture of experts. *arXiv preprint arXiv:2410.07524*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2022. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

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

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. 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*.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*.

Zach Nussbaum, John X Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*.

James O'Neill, Polina Rozenshtein, Ryuichi Kiryo, Motoko Kubota, and Danushka Bollegala. 2021. I wish I would have loved this one, but I didn't – a multilingual dataset for counterfactual detection in product review. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7092–7108, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Sachin Ravi, Sebastian Musslick, Maia Hamin, Theodore L Willke, and Jonathan D Cohen. 2020. Navigating the trade-off between multi-task learning and learning to multitask in deep neural networks. *arXiv preprint arXiv:2007.10527*.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic. Association for Computational Linguistics.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2022. One embedder, any task: Instruction-finetuned text embeddings. *arXiv preprint arXiv:2212.09741*.

Eric Tang, Bangding Yang, and Xingyou Song. 2024. Understanding llm embeddings for regression. *arXiv preprint arXiv:2411.14708*.

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.

Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of ndcg type ranking measures. In *Conference on learning theory*, pages 25–54. PMLR.

Zichen Wang, Steven A Combs, Ryan Brand, Miguel Romero Calvo, Panpan Xu, George Price, Nataliya Golovach, Emmanuel O Salawu, Colby J Wise, Sri Priya Ponnapalli, et al. 2021. Lm-gvp: A generalizable deep learning framework for protein property prediction from sequence and structure. *bioRxiv*, pages 2021–09.

Colby Wise, Vassilis N Ioannidis, Miguel Romero Calvo, Xiang Song, George Price, Ninad Kulkarni, Ryan Brand, Parminder Bhatia, and George Karypis. 2020. Covid-19 knowledge graph: accelerating information retrieval and discovery for scientific literature. *arXiv preprint arXiv:2007.12731*.

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Short text clustering via convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 62–69.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.

Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. 2012. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

## A Hyperparameter Sensitivity Analysis

### A.1 Batching Strategy

The selection of in-batch negative samples is inherently task-dependent, with varying embedding characteristics influencing optimal sampling approaches. Tasks emphasizing *local* semantic nuances, such as Semantic Textual Similarity (STS) (Agirre et al., 2012), require negatives that capture proximal semantic distinctions. Consider paraphrase identification, where sentences like "Vivendi shares closed 1.9 percent at 15.80 euros in Paris after falling 3.6 percent on Monday" and "in new york, vivendi shares were 1.4 percent down at $18.29" demand fine-grained contextual differentiation. Conversely, tasks focused on *global* semantic representations, including classification and clustering, benefit from more diverse negative sampling strategies that capture broader semantic variations.

To empirically validate this hypothesis, we conducted a controlled experiment training an embedding model under two distinct sampling regimes: homogeneous and heterogeneous. By maintaining all other experimental parameters constant, we isolated the impact of sampling strategy.

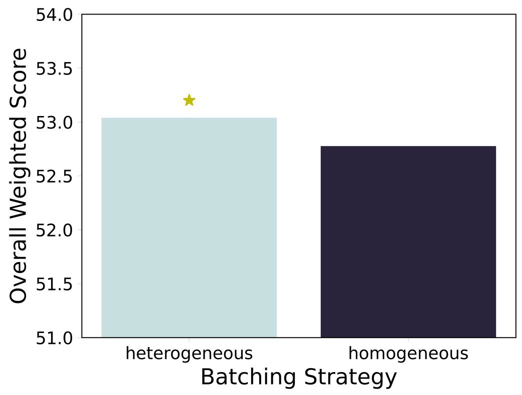

Figure 4: Aggregate Performance: Batching Strategy

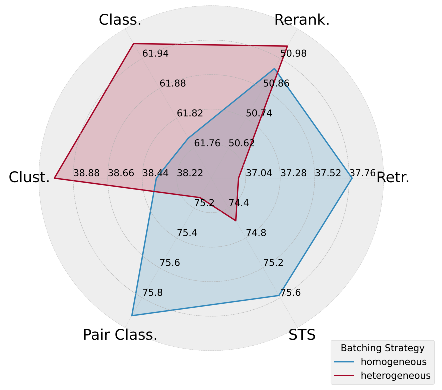

Figure 5: Task-Specific Performance: Batching Strategy

Figure 4 demonstrates that heterogeneous sampling yields the highest aggregate performance metric. However, a granular analysis in Figure 5 reveals nuanced task-specific variations: homogeneous sampling optimizes performance for local semantic tasks, while heterogeneous sampling proves superior for global embedding objectives. This underscores the critical insight that a universal sampling strategy cannot uniformly optimize performance across diverse downstream tasks.

### A.2 Contrastive Temperature

Similar to batching strategies, different contrastive temperatures capture different requirements across downstream task. Lower contrastive temperatures place a higher relative weight on samples that are more semantically similar to the anchor, thus capturing *local* semantic nuances. In contrast, higher temperatures provide a more uniform distribution of the negative weights to capture difference across a more diverse set of negatives.

To empirically validate this hypothesis, we conducted a controlled experiment training an embedding model under two distinct contrastive temperatures: 0.03 and 0.06. By maintaining all other experimental parameters constant, we isolated the impact of contrastive temperature.

Figure 6 show that the overall optimal configuration is to choose a temperature of 0.03 but a more detailed analysis in Figure 7 shows that this configuration, while beneficial to the overall performance, hurts clustering tasks.

## B EA: Expert Averaging

The main drawback of MoTE when compared to current dense alternatives is its higher memory consumption which can lead to lower throughput in real world applications such as RAG-indexing or classification which require bulk inference. To alleviate this problem we introduced an Expert Averaging (EA) compression mechanism to the MoTE architecture which alleviated the increased memory footprint at no additional

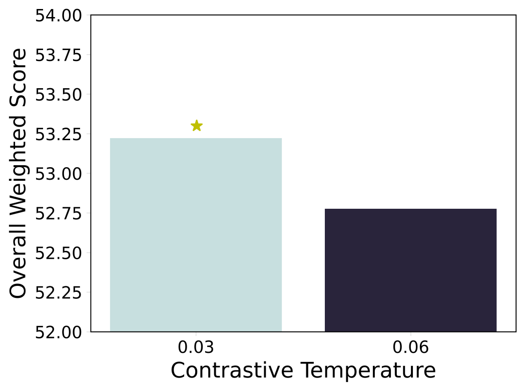

Figure 6: Overall optimal contrastive temperature.

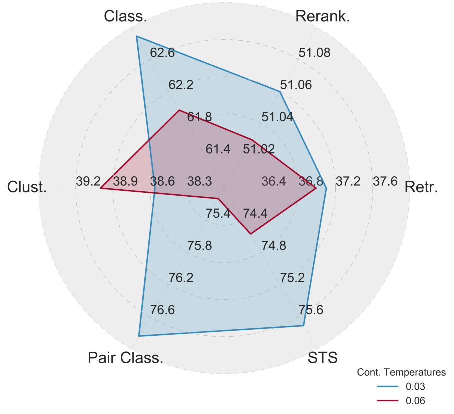

Figure 7: Optimal contrastive temperature per downstream task.

training cost by averaging the expert blocks. In this section we systematically study the performance retention of this technique compared to both MoTE and IEM.

Table 7: EA performance compared to IEM and MoTE across tasks

| | **IEM** | **MoTE** | **EA** |
|---|---|---|---|
| Retrieval[1] | 45.58 | 47.20 | 46.93 |
| Classification[2] | 68.74 | 69.17 | 68.82 |
| Clustering[3] | 43.09 | 42.81 | 42.70 |
| Reranking[4] | 53.37 | 54.91 | 54.91 |
| Pair Classification[5] | 80.72 | 81.83 | 81.59 |
| STS[6] | 80.02 | 79.96 | 79.13 |
| Summarization[6] | 33.76 | 33.97 | 33.38 |
| Average dataset performance | 58.43 | 59.07 | 58.60 |

[1] NDCG@10; [2] accuracy; [3] validity measure;
[4] MAP; [5] AP; [6] Spearman correlation

EA mitigates MoTE's increased memory footprint while retaining $0.17$ average dataset performance over IEM. The average performance retention is largely driven by retrieval and re-ranking tasks with of $1.35$ NDCG@10 and $1.54$ MAP performance improvements of EA over IEM, respectively.

## C Evaluation Metrics

### C.1 Normalized Discounted Cumulative Gain at 10

The Normalized Discounted Cumulative Gain (NDCG) (Burges et al., 2005; Wang et al., 2013) is a metric used to evaluate the ranking quality of a search engine or recommendation system by comparing the relevance of retrieved documents to the ideal ranking. The NDCG at position 10 is defined as:

$$\text{NDCG@10} = \frac{\text{DCG@10}}{\text{IDCG@10}}$$

where:

- DCG@10 (Discounted Cumulative Gain) is calculated as:

$$\text{DCG@10} = \sum_{i=1}^{10} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}$$

- IDCG@10 (Ideal Discounted Cumulative Gain) is the DCG@10 of the ideal (perfectly ranked) ordering of results.
- $\text{rel}_i$ represents the relevance score of the result at position $i$.

The NDCG@10 value ranges from 0 to 1, where 1 indicates a perfect ranking.

## C.2 Accuracy

Accuracy is a metric used to evaluate the overall correctness of a classification model. It is defined as the ratio of correctly predicted instances to the total number of instances:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where $TP$, $TN$, $FP$, and $FN$ denote the True Positives, True Negatives, False Positives and False Negatives, respectively. The accuracy value ranges from 0 to 1, with 1 representing perfect classification.

## C.3 V-Measure

V-Measure (Rosenberg and Hirschberg, 2007) is a clustering evaluation metric that assesses the quality of a clustering solution by measuring its homogeneity and completeness. It is defined as the harmonic mean of these two metrics:

$$\text{V-Measure}_\beta = (1+\beta) \cdot \frac{\text{Homogeneity} \cdot \text{Completeness}}{\beta \cdot \text{Homogeneity} + \text{Completeness}}$$

where:

- Homogeneity ensures that each cluster contains only members of a single class. It is defined as:
$$\text{Homogeneity} = 1 - \frac{H(C|K)}{H(C)}$$
- Completeness ensures that all members of a given class are assigned to the same cluster. It is defined as:
$$\text{Completeness} = 1 - \frac{H(K|C)}{H(K)}$$
- If $\beta$ is greater than 1 completeness is weighted more strongly in the calculation, if $\beta$ is less than 1, homogeneity is weighted more strongly.
- $H(C|K)$ is the conditional entropy of the class distribution ($C$) given the clustering distribution ($K$).
- $H(K|C)$ is the conditional entropy of the clustering distribution given the class.
- $H(C)$ and $H(K)$ are the entropies of the class distribution and clustering distribution, respectively.

$\text{V-Measure}_1$ ranges from 0 to 1, where 1 indicates perfect clustering.

## C.4 Average Precision

Average Precision (AP) evaluates the model's ability to rank positive pairs (relevant pairs) higher than negative pairs (non-relevant pairs). It is defined using the ranking of positive pairs:

$$\text{AP} = \frac{1}{P} \sum_{k=1}^{P} \frac{k}{\text{rank}(k)}$$

where:

- $P$ is the total number of positive pairs.
- $\text{rank}(k)$ is the rank position of the $k$-th positive pair in the sorted list of predictions.

AP ranges from 0 to 1, where 1 indicates perfect ranking of all positive pairs above negative pairs.

### C.5 Mean Average Precision

Mean Average Precision (MAP) is a metric used to evaluate the performance of information retrieval systems, ranking models, or classification tasks with multiple relevance levels. It is defined as the mean of the average precision scores for all queries:

$$\text{MAP} = \frac{1}{Q}\sum_{q=1}^{Q}\text{AP}(q)$$

where:

- $Q$ is the total number of queries.
- $\text{AP}(q)$ is the Average Precision for query $q$.

MAP ranges from $0$ to $1$, where $1$ indicates perfect precision across all queries.

### C.6 Spearman correlation

Spearman's Rank Correlation Coefficient ($\rho$) measures the monotonic relationship between two ranked variables. In the context of STS and Summarization tasks, it evaluates how well the predicted similarity scores or summary scores correlate with human-annotated scores.

$$\rho = 1 - \frac{6\sum_{i=1}^{N} d_i^2}{N(N^2-1)}$$

where:

- $N$ is the total number of instances (e.g., pairs of sentences or summaries).
- $d_i = \text{rank}(x_i) - \text{rank}(y_i)$ is the difference between the rank of the predicted score $x_i$ and the rank of the human-annotated score $y_i$.
- $\sum_{i=1}^{N} d_i^2$ is the sum of the squared rank differences.

Specifically, on STS Tasks $x_i$ represents predicted similarity scores, and $y_i$ represents human-assigned similarity scores while in summarization tasks $x_i$ represents predicted summary quality scores, and $y_i$ represents human-assigned summary ratings.

Spearman's $\rho$ ranges from $-1$ to $1$, where $\rho = 1$ indicates a perfect positive correlation, meaning the predicted rankings match the human-assigned rankings exactly. A value of $\rho = 0$ signifies no correlation, implying that there is no monotonic relationship between the predicted and human rankings. Conversely, $\rho = -1$ represents a perfect negative correlation, where the predicted rankings are the exact inverse of the human rankings.

## D Implementation Details

### D.1 Model Architecture

The MoTE architecture replaces standard transformer blocks with MoTE blocks, each containing four experts assigned to specific instruction types: classification, clustering, search query, and search document. Our implementation builds upon the design proposed by (Nussbaum et al., 2024), with two key modifications:

1. The standard FeedForward (FF) block is replaced with an `nn.ModuleList` containing multiple FF blocks, one per expert.
2. The routing mechanism is adapted as described in Section 3, directing tokens to appropriate experts based on task-specific instructions.

## D.2 Training Procedures

Mini-batches are constructed by sampling tasks, with data drawn from their corresponding datasets. While the total batch size is kept constant, both the batch construction strategy and the contrastive temperature differ by task type:

- Batch Construction Strategy
  - For classification and clustering tasks, mini-batches are formed by randomly sampling data points across all datasets associated with the task.
  - For retrieval tasks, each mini-batch is built by sampling data points from a single dataset, randomly selected for each batch.
- Contrastive Temperature:
  - A temperature of $0.03$ is used for classification and retrieval mini-batches.
  - A higher temperature of $0.06$ is applied for clustering mini-batches.

## D.3 Batch Size and Infrastructure

Training is performed with a global batch size of $6144$, made possible using DeepSpeed Stage 2 across a 16-node cluster, with each node equipped with 8 Nvidia A100 GPUs (40GB). DeepSpeed enables efficient gradient synchronization while broadcasting embedding representations across devices for contrastive loss computation. Notably, gradient caching is not used due to the reliance on in-batch negatives for contrastive learning.