CAEM-MoE: Complexity-Aware Expert Merging for Efficient Multi-Task Learning in Mixture of Experts

Anonymous ACL submission

Abstract

The Mixture-of-Experts (MoE) architecture plays a crucial role in scaling Large Language Models to trillions of parameters without incurring excessive computational costs. Although utilizing these powerful models for Multi-Task Learning (MTL) is an attractive objective, training a single model on diverse tasks without proper consideration can often result in performance degradation due to task interference and negative transfer. To tackle this issue, we came up with Complexity-Aware Expert Merging (CAEM), a novel strategy that uses the entropy of expert utilization as an indicator of task complexity. This approach allows for a strategic allocation of expert resources, overcoming common MTL bottlenecks. Our method achieves a significant improvement over standard MTL baselines, exemplified by a 6.47% ROUGE-L gain on the complex XSum task with only negligible trade-offs on other simpler tasks (8-experts case) because of the superior starting point in the loss landscape and path dependency in optimization. This observation led us to identify a more general principle: a "Founder Effect" in model merging. CAEM not only provides a resource-efficient path to high-performance MTL but also provides insights into the mechanisms of model merging.

1 Introduction

In recent years, the scaling of large language models has become a central factor in the development of artificial intelligence capabilities. A significant component of this trend is the Mixture-of-Experts (MoE) architecture (Shazeer et al., 2017), such as the Switch Transformer model proposed by Fedus et al. (2022). This architecture primarily increases the total parameter capacity of the model through sparse conditional computation, while not significantly raising the number of floating-point operations (FLOPs) needed for inference. Additionally, this approach has led to the creation of many highly specialized models that are fine-tuned for specific tasks. This trend of model specialization naturally makes people try to integrate dispersed expertise to fit more complex and diverse real-world demands. Task Arithmetic (Ilharco et al., 2023), a pioneering research direction, posits that one can combine or remove model capabilities through simple arithmetic operations on task vectors, the weight difference between fine-tuned and pre-trained models.

However, subsequent research found that a direct linear combination of task vectors often leads to performance degradation due to destructive interference from issues such as parameter conflicts. To address this, later studies have proposed more sophisticated strategies, such as Hi-Merging (Yuan et al., 2024), which relieves parameter conflicts through pruning and scaling specific parameters, or AdaMerging (Jiang et al., 2024), which automatically learns the merging coefficients λ. Recent work, such as FedMoE (Mei et al., 2024), has also demonstrated the great potential of modular approaches, successfully achieving efficient, personalized federated learning by intelligently combining expert subsets for different clients.

Beyond these strategies, Multi-Task Learning offers a valuable framework, aiming to solve multiple related tasks with a single model, thereby using inter-task commonalities to improve learning efficiency and generalization (Caruana, 1997; Liu et al., 2019). A common approach is to jointly fine-tune a general-purpose MoE pre-trained model on a mixture of data from all target tasks. However, this approach often neglects the intrinsic differences between tasks, potentially leading to negative transfer (Yosinski et al., 2014) or parameter conflicts, which force experts to achieve less optimal specialization for specific tasks. The other extreme, training a complete MoE model independently for each task while ensuring optimal task specialization contradicts the ethos of MTL and results in a tremendous waste of resources.

Therefore, the core of this research focuses on an efficient merging strategy to integrate the knowledge from multiple single-task MoE models into a single multi-task model and, through fine-tuning, achieve performance that surpasses a standard from-scratch training baseline. More importantly, we also explore the underlying mechanism.

To solve this problem, this study proposes the Complexity-Aware Expert Merging strategy. Our core insight is the use of the entropy of expert utilization as a quantitative indicator of task complexity. The intuition behind this is that more complex tasks, such as summarization, require the model to master more diverse and detailed sub-skills. This phenomenon causes the router to invoke a more diverse set of experts when processing inputs, resulting in a more uniform usage pattern with higher entropy. Conversely, simpler tasks, like text classification, may rely on a few key experts, leading to a concentrated, low-entropy usage pattern. Based on this metric, CAEM reframes the model merging from a simple performance maximization problem to a strategic resource trade-off process, thus accomplishing a bottom-up, task-driven allocation of expert capacity.

Our main contributions are as follows.

- We propose and validate the CAEM strategy, demonstrating its ability to achieve an improvement by significantly boosting performance on bottleneck tasks with minimal trade-offs on simpler ones.
- We show that both the CAEM model and the standard baseline model exhibit high parametric compatibility after fine-tuning, with a functional decoupling: encoder representations homogenize, while decoder representations have a capacity-dependent divergence.
- We demonstrate that the optimization trajectory of a merged model is constrained by its initial position in the loss landscape, resulting in the model's final representational characteristics being dominated by those of its original components.

2 Background

2.1 Mixture of Experts

The core strategy of MoE is to assign input tokens to a set of experts dynamically by a router to reduce the FLOPs for inference. A popular example of an MoE architecture is the Switch Transformer (Fedus et al., 2022). This architecture modifies the standard Transformer by replacing the Feed-Forward Network with the MoE layer. Its most distinctive feature is the adoption of a top-1 routing strategy. Compared to other MoE models that select multiple experts at once, which significantly reduces computational complexity while maintaining model performance.

A common finding is that during the training of MoE models, different experts exhibit functional specialization (Du et al., 2022). Consequently, each expert develops proficiency for different sub-tasks, and the router's objective is to learn how to dispatch the current input token to the suited experts accurately. This characteristic provides a potential solution for multi-task learning and offers a solid foundation for the ever-growing field of generative general-purpose AI.

2.2 Multi-Task Learning

The core idea of MTL is to enable a single model to learn multiple different tasks simultaneously, with the expectation that knowledge can be shared across related tasks during training to improve the model's performance on each task. In standard Transformer architectures, MTL often develops a shared encoder to build a common understanding of input tokens while using task-specific decoders to handle diverse generation formats (Zhang and Yang, 2022). The success of MTL hinges on effective knowledge transfer, where features learned for one task provide useful inductive biases for others (Sodhani et al., 2021).

Despite its potential, MTL still faces significant challenges, with task interference and negative knowledge transfer. These issues arise when learning objectives conflict (e.g., their gradient directions are opposite), causing shared parameters to degrade performance (Yu et al., 2020). The sparse activation of experts in MoE architectures strategically solves this problem by distributing the computational load of different sub-tasks across distinct experts, thus reducing direct parameter conflicts. However, while this relieves interference, the question of how to best allocate resources (i.e., experts) in a MTL setting remains largely unexplored. This question led us to a core motivation for our research: to investigate whether the knowledge of several highly specialized experts can be effectively combined and handle several tasks perfectly.

2.3 Model Similarity and the Loss Landscape

The loss landscape of neural networks directly influences the training process and results. Previously, people believed that optimal solutions found from different random initializations were isolated in separate loss basins, separated by high loss barriers. However, Garipov et al. (2018) demonstrated that low-loss paths often connect these disparate solutions. This discovery induced new research in model merging, including the influential empirical work on Model Soups (Wortsman et al., 2022). They found that when models fine-tuned from the same pre-trained weights are averaged, the resulting model is often more robust and performant, implying these solutions likely reside within the same connectable low-loss basin.

However, these studies usually focused on structurally identical, single-task dense models. While they provide a valuable method for probing weight-space symmetries, MTL on MoE has greater complexity with its modular architecture and diverse task-specific initializations. Therefore, in addition to applying the concept of Model Soups as a macroscopic test to probe whether different models inhabit compatible solution spaces, we also analyze the internal mechanisms of models using Centered Kernel Alignment (CKA) to quantify representational similarity (Kornblith et al., 2019). CKA is a method for comparing representations across network layers, as it is invariant to feature rotation and isotropic scaling, making it more reliable than normal vector similarity. These two analytical tools, model-averaged fusion and CKA, allow us to systematically investigate the intrinsic properties of multi-task learning on MoE models from both the weight and representation space perspectives.

3 CAEM-MoE

3.1 Overview

The purpose of CAEM is to merge a set of source models, $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$, into a single MTL MoE model, M_{MTL}. Each S_i is a standard encoder or decoder of Switch Transformer, with n_s experts per MoE layer, which has been fine-tuned on a single task. The resulting M_{MTL} is also a Switch Transformer architecture with n_m experts per layer. The entire procedure is detailed in Algorithm 1 and illustrated schematically in Figure 1. It is important to note that we perform the CAEM process independently for the encoder and the decoder, treating them as separate modules to be merged.

Algorithm 1: Complexity-Aware Expert Merging (CAEM) for Encoder and Decoder

1 Initialize $\boldsymbol{N}_{\text{total}} \leftarrow 0$
2 **for** each MoE layer $l \in \mathcal{L}_{MoE}$ **do**
3 **for** each source model $S_i \in \mathcal{S}$ **do**
4 $\boldsymbol{P}_i^{(l)} \leftarrow \text{GetUsageDist}(S_i, l)$
5 $\boldsymbol{H}_{\text{norm},i}^{(l)} \leftarrow \text{CalcNormEntropy}\left(\boldsymbol{P}_i^{(l)}\right)$
6 $\boldsymbol{H}_{\text{norm, total}}^{(l)} \leftarrow \sum_{i=1}^{M} \boldsymbol{H}_{\text{norm},i}^{(l)}$
7 $\boldsymbol{N}_{\text{alloc}}^{(l)} \leftarrow \left[\frac{n_m \boldsymbol{H}_{\text{norm},1}^{(l)}}{\boldsymbol{H}_{\text{norm, total}}^{(l)}}, \ldots, \frac{n_m \boldsymbol{H}_{\text{norm},M}^{(l)}}{\boldsymbol{H}_{\text{norm, total}}^{(l)}}\right]$
8 $\boldsymbol{N}_{\text{alloc}}^{(l)} \leftarrow \text{AdjustAlloc}\left(\boldsymbol{N}_{\text{alloc}}^{(l)}, n_m, n_s\right)$
9 $\boldsymbol{N}_{\text{total}} \leftarrow \boldsymbol{N}_{\text{total}} + \boldsymbol{N}_{\text{alloc}}^{(l)}$
10 *target_idx* $\leftarrow 1$
11 **for** each source model $S_i \in \mathcal{S}$ **do**
12 *top_idxs* $\leftarrow \text{TopKIndices}\left(S_i, l, \boldsymbol{N}_{\text{alloc}}^{(l)}\right)$
13 **for** each *source_idx* in *top_idxs* **do**
14 $\boldsymbol{E'}_{\textit{target\_idx}}^{(l)} \leftarrow \boldsymbol{E}_{i,\textit{source\_idx}}^{(l)}$
15 $\boldsymbol{r'}_{\textit{target\_idx}}^{(l)} \leftarrow \boldsymbol{r}_{i,\textit{source\_idx}}^{(l)}$
16 *target_idx* $\leftarrow$ *target_idx* $+ 1$
17 $\boldsymbol{w} \leftarrow \boldsymbol{N}_{\text{total}} / \sum(\boldsymbol{N}_{\text{total}})$
18 **for** each non-MoE parameter θ in M_{MTL} **do**
19 $\theta_{M_{MTL}} \leftarrow \sum_{i=1}^{M} \boldsymbol{w}_i \cdot \theta_{S_i}$
20 **return** M_{MTL}

3.2 Complexity-Aware Expert Allocation

The primary purpose of this section is to allocate expert capacity based on the intrinsic complexity of each task. To achieve this, we first run inference with each source model S_i on its corresponding validation dataset to record the expert usage counts. These counts are then transformed to a probability distribution $\boldsymbol{P}_i^{(l)} = \left[p_{i,1}^{(l)}, p_{i,2}^{(l)}, \ldots, p_{i,n_s}^{(l)}\right]$ for each MoE layer l. Next, we compute the normalized entropy of this distribution, $\boldsymbol{H}_{\text{norm},i}^{(l)}$. A value close to 1 signifies a uniform distribution of expert usage, indicating high task complexity, while a value approaching 0 implies a skewed distribution, suggesting lower complexity. The normalized entropy is calculated as:

$$\boldsymbol{H}_{\text{norm},i}^{(l)} = \frac{-\sum_{j=1}^{n_s} p_{i,j}^{(l)} \log p_{i,j}^{(l)}}{\log(n_s)} \in [0, 1] \qquad (1)$$

With $\boldsymbol{H}_{\text{norm},i}^{(l)}$ for each layer, we proportionally allocate the n_m target experts among the source models (Algorithm 1, Line 7). Subsequently, an

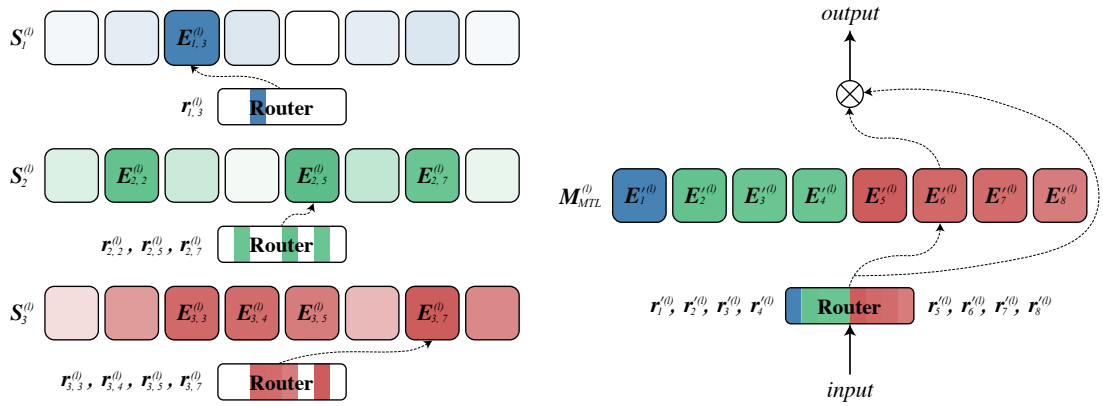

Figure 1: An illustration of the Complexity-Aware Expert Merging (CAEM) process. The case with $M = 3$ source models (S_1, S_2, S_3) and a target MTL model (M_{MTL}), where $n_s = n_m = 8$ and $\boldsymbol{N}_{\text{alloc}} = [1, 3, 4]$. The color hue represents different tasks, and the color intensity represents its usage frequency (darker indicates higher).

adjustment function, AdjustAlloc(), is applied to handle integer rounding and boundary conditions. If there are remaining experts, we allocate them based on the decimal part before rounding. (e.g. ensuring each task contributes between 1 and n_s experts and the total number of allocated experts precisely equals n_m) (Algorithm 1, Line 8).

3.3 Selective Expert Merging

A standard MoE layer of the Switch Transformer operates as follows: a router, with weight matrix $\boldsymbol{W}_{\text{router}} = \left[\boldsymbol{r}_1^{(l)}, \ldots, \boldsymbol{r}_{n_m}^{(l)}\right]$, maps an input token x to a vector of logits. These logits are then passed through a softmax function to produce a probability distribution over the experts, $g(x)$. Subsequently, the expert $\boldsymbol{E}_{i,j}^{(l)}$ corresponding to the highest probability $g(x)$ is selected to process the token.

With the number of experts to be allocated from each source model S_i determined for each layer l, we proceed with the selection. Based on the expert usage frequencies calculated during the allocation phase, we employ the TopKIndices() function to identify the indices of the top-$\boldsymbol{N}_{\text{alloc},i}^{(l)}$ most frequently used experts for each S_i. Subsequently, the selected expert modules $\boldsymbol{E}_{i,j}^{(l)}$ and their corresponding weight vectors $\boldsymbol{r}_{i,j}^{(l)}$ in router are copied from S_i and placed into their new positions within the target model M_{MTL} (Algorithm 1, Lines 11-16).

For the remaining non-MoE, shared parameters θ, we employ a weighted averaging scheme. The weight vector, $\boldsymbol{w} \in \mathbb{R}^M$, is determined by the total proportion of experts that each source model S_i contributed across all MoE layers, denoted by $\boldsymbol{N}_{\text{total}}$ (Algorithm 1, Line 17-19). This ensures that source models contributing more experts also have a greater influence on the shared backbone of the final architecture. This concludes the whole initialization process for our CAEM process.

4 Experiments

4.1 Datasets and Preprocessing

We evaluate our CAEM method on three different types of English NLP tasks: classification (AG News; Zhang et al., 2015), extractive question answering (SQuAD v1.1; Rajpurkar et al., 2016), and summarization (XSum; Narayan et al., 2018). To ensure a unified input-output format for the Switch Transformer, all instances are converted into a seq2seq format, prepended with a task-specific prompt—"Classify:", "Question:"(and "Context:"), and "Summarize:". For AG News, we map the original labels to their string representations (e.g. "World", "Sports") and evaluate performance using accuracy. For SQuAD, we adapt the task by requiring the model to generate the full answer text rather than predicting start and end indices, and evaluate using the standard exact match (EM) score. Finally, performance on XSum is evaluated using ROUGE-L (Lin, 2004). See Appendix A.1 for details.

4.2 Models and Baselines

For our experiments, we benchmark CAEM against a standard MTL baseline at two distinct expert capacities. Furthermore, to validate our design, we conduct two ablation studies. Table 1 summarizes these seven experimental configurations.

Baseline Cases. These models are initialized with the google/switch-base-{8, 16} pretrained weights and then fine-tuned on the multi-task data mixture, as described in Section 4.3.

Configuration Name	Source Models (n_s)	Target Model (n_m)	Allocation Method
Baseline-8	N/A (Directly fine-tuned)	MTM (8)	N/A
Baseline-16	N/A (Directly fine-tuned)	MTM (16)	N/A
CAEM-8	3 × STM (8)	MTM (8)	Entropy-based (Ours)
CAEM-8-to-16	3 × STM (8)	MTM (16)	Entropy-based (Ours)
CAEM-16	3 × STM (16)	MTM (16)	Entropy-based (Ours)
CAEM-8-Avg	3 × STM (8)	MTM (8)	Fixed Ratio (2:3:3)
CAEM-16-Avg	3 × STM (16)	MTM (16)	Fixed Ratio (5:5:6)

Table 1: An overview of the seven experimental configurations. Models include baselines, our proposed CAEM, and ablations with fixed allocation ratios. STM: Single-Task Model. MTM: Multi-Task Model.

CAEM Cases. Before the CAEM process, we use the same pretrained weights to fine-tune single-task models on each of the three tasks independently. The resulting three models are then merged using the CAEM algorithm to create the initial multi-task model, which is subsequently fine-tuned with the same procedure as the baselines.

Ablation Cases. To test the effectiveness of our entropy-based expert allocation, we include two ablation variants that use a uniform-like allocation ratio (2:3:3 for 8 experts; 5:5:6 for 16 experts) instead of the complexity-aware allocation.

4.3 Implementation Details

All models are based on the Switch Transformer architecture. For optimization, we use AdamW with a constant learning rate of 3×10^{-4}. Further hyperparameters are detailed in Appendix A.3.

Training Procedure. For all multi-task fine-tuning, we adopt a mixed-batch strategy. Each optimization step processes three batches (one per task, with a batch size of 24), and gradients are accumulated, yielding an effective batch size of 72. In addition, due to the different dataset sizes, we define an epoch as a fixed number of 87,599 training samples extracted from each task, matching the size of the smallest dataset (SQuAD). Data is sampled alternately across epochs to ensure that all samples are utilized throughout the training process. Finally, we train for 10 epochs and report the performance from the epoch that achieves the best Overall Performance on the validation datasets.

Overall Performance Metric. To provide a single, normalized score across the three distinct tasks, we define the Overall Performance as follows:

$$\text{OverallPerf} = \frac{1}{M} \sum_{i=1}^{M} \frac{\text{Score}_{\text{MT},i}}{\text{Score}_{\text{ST},i}} \qquad (2)$$

where $M = 3$ is the number of tasks, $\text{Score}_{\text{MT},i}$ is the multi-task model's score on task i, and $\text{Score}_{\text{ST},i}$ is the best score achieved by a single-task model of the same expert capacity on that task.

5 Results and Analysis

5.1 Performance Comparison

Table 2 presents the performance results of all seven configurations, demonstrating the efficacy of our CAEM method. All three core CAEM models (CAEM-8, CAEM-16, and CAEM-8-to-16) consistently and significantly outperform their corresponding baselines in Overall Performance.

Focusing on the 8-expert setup, CAEM-8 achieves the highest overall performance among all models, marking a substantial 1.97% relative improvement over Baseline-8. This success highlights our method's ability to perform strategic resource trade-offs, which achieves a remarkable 6.47% gain on the complex bottleneck task, XSum, at the cost of only a negligible 0.16% decrease on AG News. A similar pattern of superiority over the baseline is observed in the 16-expert case. The results of our ablation study (CAEM-8-Avg) further validate our approach. Although the case also surpasses the baseline, its 1.9% performance drop on XSum compared to CAEM-8 confirms that allocating experts based on task complexity (entropy-based) is crucial for maximizing MTL performance.

Interestingly, we found that the performance of the CAEM-8-to-16 case falls between that of CAEM-8 and CAEM-16 for all tasks. Among these, the performance trend CAEM-8 > CAEM-8-to-16 > CAEM-16 is particularly evident on the Overall and XSum metrics. We attribute this phenomenon to the efficiency of expert utilization during the inference process. Table 3 shows the av-

Configuration Name	Overall Perf.	AG News (Acc.)	SQuAD (EM)	XSum (ROUGE-L)
Baseline-8	0.9862	**94.72**	76.50	29.36
Baseline-16	0.9881	94.33	76.64	29.88
CAEM-8	**1.0056**	94.57	76.51	**31.26**
CAEM-8-to-16	1.0012	94.37	76.90	31.02
CAEM-16	1.0007	94.03	**77.15**	30.98
CAEM-8-Avg	0.9986	94.20	76.59	30.68
CAEM-16-Avg	0.9990	94.39	76.58	30.94

Table 2: Performance comparison of experimental configurations. The Overall Performance is normalized by single-task performance (>1.0 is better). The **best** result is in bold and the second-best is underlined.

erage normalized entropy of expert utilization in the final models. The performance trend directly correlates with it. CAEM-8, with the highest encoder and decoder entropy, utilizes its experts more collaboratively and effectively. In contrast, the lower entropy of CAEM-16 suggests that its router adopts a more conservative strategy, concentrating computation on fewer proportions of experts. For CAEM-8-to-16, although its encoder entropy is the lowest of the three, its decoder entropy, which we argue is more critical for these muti-task models (Section 5.2), remains significantly higher than the CAEM-16 case. These results suggest that a smaller pool of experts with uniform used can be more effective than a larger pool of experts sparsely used in a multitasking setting.

Model	En-Entropy	De-Entropy
CAEM-8	**0.4524**	**0.5923**
CAEM-8-to-16	0.3345	0.5312
CAEM-16	0.3966	0.4677

Table 3: Average normalized entropy of expert usage. Higher values indicate more uniform expert utilization.

5.2 Analysis of Inter-Model Similarity

To understand the internal mechanisms of different MTL models, we first investigate the functional similarity between our CAEM models and the baselines. We employ CKA to compare the layer-wise representations of the models before and after fine-tuning, as shown in Figure 2. We can find that in both {8, 16}-expert configurations, the encoder CKA similarity between CAEM and baseline models systematically increases after fine-tuning. This demonstrates that, regardless of their different initializations, the models are guided towards a homogeneous subspace for semantic understanding. In contrast, the decoder exhibits a capacity-dependent dynamic. In the constrained 8-expert setting, the decoders also become similar. However, in the high-capacity 16-expert setting, the decoders show significant divergence, particularly for the simple AG News task. This indicates that sufficient expert capacity allows the models to explore and develop distinct, specialized pathways for text generation.

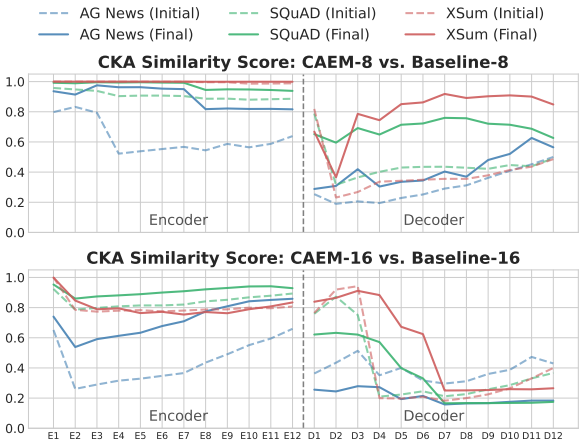

Figure 2: Layer-wise CKA similarity between CAEM and Baseline models, before (Initial, dashed lines) and after (Final, solid lines) multi-task fine-tuning.

To further validate whether this representational similarity corresponds to functional compatibility in the weight space, we adopt the "Model Soup" methodology (Wortsman et al., 2022). As shown in Table 4, averaging the weights of any two models within the same expert capacity (e.g., CAEM-8 and Baseline-8) still maintains good performance. This provides strong evidence that these models, despite their different initialization strategies, ultimately converge into the same broad, functionally connected low-loss basin.

Model Pair for Averaging	Overall Perf.
CAEM-8, Baseline-8	0.8531
CAEM-8, CAEM-8-Avg	**0.8880**
CAEM-8-Avg, Baseline-8	0.8592
CAEM-16, Baseline-16	0.9035
CAEM-16, CAEM-16-Avg	**0.9337**
CAEM-16-Avg, Baseline-16	0.8678

Table 4: Overall Performance after applying model soup (averaging weights) between pairs of models with the same architecture. The **highest** result is in bold.

Finally, an analysis of the expert routing patterns in the CAEM-8 model reveals a key insight (Figure 3). We found that the fine-tuned model develops a task-agnostic expert utilization strategy, activating a shared pool of experts for all tasks. Therefore, the primary role of CAEM is not to enforce a static, specialized architecture, but to provide a superior starting point. This initial assembly of high-quality specialists enables the multi-task fine-tuning to discover a more optimal and synergistic general-purpose solution than the baseline.

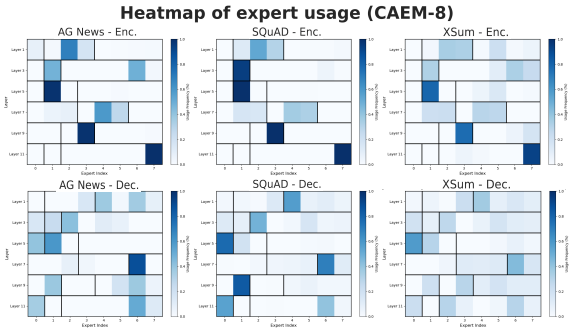

Figure 3: Expert utilization frequency for the CAEM-8 model across all MoE layers and tasks. Color intensity represents the routing frequency (darker indicates higher) for each expert (x-axis) at each layer (y-axis).

Additionally, a notable finding is that, despite the encoders of CAEM-8 and Baseline-8 achieving near-perfect functional equivalence (CKA approaching 1.0 in Figure 2), their performance on XSum differs by a substantial 6.47% (cf. Table 2). This performance gap can be attributed almost entirely to the crucial divergences in their decoder strategies. This confirms that while multi-task learning fosters a shared understanding (Encoder), the ultimate performance on complex generative tasks is dictated by the specialized generation strategies developed in the decoder.

5.3 The Founder Effect

Having investigated the representation similarity of models within the same architecture, we now analyze the special CAEM-8-to-16 configuration to probe the principles of cross-capacity merging. We use CKA to compare the final representations of this model with the CAEM-8 and CAEM-16 models. As illustrated in Figure 4, we found an interesting pattern. Despite undergoing the same multi-task fine-tuning process (mixed-batch), the CAEM-8-to-16 model's encoder representations remain almost perfectly aligned with the model, CAEM-8, achieving CKA scores exceeding 0.999 for tasks like XSum. Conversely, its similarity to the native CAEM-16 model is significantly lower, in fact, than its similarity to the Baseline-16 model (cf. Figure 2).

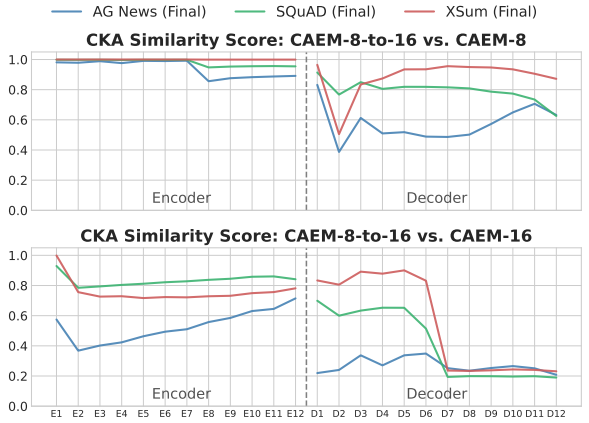

Figure 4: Layer-wise CKA similarity between CAEM-8-to-16 and CAEM-8, CAEM-16 models after multi-task fine-tuning.

In the same manner, we performed model soup experiments. We averaged the weights of the CAEM-8-to-16 model with all other 16-expert models (Baseline-16, CAEM-16, and CAEM-16-Avg) separately. The results presented in Table 5 indicate that all fusion leads to a catastrophic performance collapse, with overall performance approaching zero.

Model Pair for Averaging	Overall Perf.
CAEM-8-to-16, Baseline-16	0.0000
CAEM-8-to-16, CAEM-16	**0.0011**
CAEM-8-to-16, CAEM-16-Avg	0.0010

Table 5: Overall Performance after applying model soup (averaging weights) between CAEM-8-to-16 and other 16-expert configurations. The **highest** result is in bold.

Taken together, these findings provide unequivocal evidence for what we term the "Founder Effect". The CKA analysis reveals a strong representational inertia, where the functional patterns of the source modules are inherited. The model soup experiment proves that this inertia is not merely a superficial similarity but a fundamental constraint; a formidable loss barrier separates the solution basin occupied by the 8-to-16 model from the basin of other 16-expert models, rendering them incompatible in weight space.

This inherited representational structure also perfectly explains the performance interpolation of the CAEM-8-to-16 model. It effectively inherits the encoder's "thinking mode" from the 8-expert model, but its generation process requires adapting a new 16-expert decoder architecture. This architectural mismatch and the compromises made to accommodate it during fine-tuning result in performance that is naturally between the CAEM-8 and CAEM-16.

In conclusion, the Founder Effect is a manifestation of profound path dependence in the optimization of merged models. The initial weight configuration, dominated by the more coherent and composable "founder" modules from the 8-expert source, creates a powerful basin of attraction in the loss landscape. This basin constrains the model's entire optimization trajectory, ensuring the inheritance of the founder's functional properties, irrespective of the target architecture's capacity.

6 Related Work

6.1 Task Interference on MTL

A central challenge in MTL is managing the conflicts among tasks. Prevailing solutions focus on in-training dynamics, either through direct gradient manipulation (Yu et al., 2020) or by dynamically balancing task loss weights. Using heuristics based on uncertainty (Kendall et al., 2018), gradient norms (Chen et al., 2018), or finding Pareto optimal solutions (Sener and Koltun, 2018). In contrast, our work addresses this challenge from a different dimension: model initialization. CAEM is a one-shot merging strategy that provides a superior starting point to enable more efficient MTL.

6.2 Model Merging and the Loss Landscape

Recent interest in model merging is grounded in the understanding that optimal solutions often reside within a connectable low-loss basin. Following foundational work by Garipov et al. (2018) and Frankle et al. (2020), which demonstrated that low-loss paths connect different optimal solutions, Model Soups (Wortsman et al., 2022) empirically verified that averaging models from the same loss basin can lead to better and more robust solutions. To achieve more sophisticated merging, the field has developed advanced techniques such as Fisher-weighted averaging (Matena and Raffel, 2022), interference resolution via TIES-Merging (Yadav et al., 2023), and model editing through Task Arithmetic (Ilharco et al., 2023). However, these studies focus on parameter-level merging of dense models. We extends this to the module-level reorganization of sparse MoE architectures and discovers the "Founder Effect" on multi-task model merging.

6.3 Representation Similarity on MoE

Understanding the internal mechanisms of large models is a core research topic. While prior work has analyzed expert specialization in MoE models (Du et al., 2022) and the hierarchical functions of Transformers using tools like RSA (Kriegeskorte et al., 2008), probing (Tenney et al., 2019), and CKA (Kornblith et al., 2019), BERTology (Rogers et al., 2020), these analytical tools have mostly been applied to dissect a single model after fine-tuning. A key methodological contribution of our work is the application of these tools to tracking the representational trajectories of different models (CAEM vs. Baseline) before and after merging and fine-tuning, revealing the learning dynamics of convergence and divergence in the encoder and decoder from a novel perspective.

7 Conclusion

We propose CAEM, a complexity-aware framework for merging MoE models that provides a superior starting point for efficient multi-task learning. Our method acts as a strategic resource trade-off, it boosts performance on complex bottleneck tasks via minimal trade-offs on simpler ones, and our analysis uncovers a Founder Effect, where a merged model's optimization trajectory is constrained by its source modules. Future work could validate CAEM's generalizability on a wider range of tasks and explore its applicability to different MoE architectures. Another direction is to refine the merging strategy, for instance by normalized its weights while merging, to better adapt the inherited features to models of varying sizes and overcome the scaling limitations observed in this study.

Limitations

Our study is subject to some limitations. First, the generalizability of our findings across tasks and languages could be further explored. Although our experiments cover three diverse NLP tasks, the efficacy of CAEM on other task types (e.g., machine translation) or in non-English contexts remains an open question. Second, our analysis is confined to a specific model architecture and scale. This work focuses exclusively on Switch Transformers with top-1 routing, and our experiments are conducted on models with up to 16 experts. The behavioral patterns of CAEM on models with different routing strategies or at a significantly larger scale may be more complex and warrant further investigation. Finally, while we found the "Founder Effect", our work does not propose a solution to its associated scaling problem. More fundamentally, how the inherited representational patterns from the founder modules can be optimally adapted to unlock the full potential of a new, higher-capacity architecture is a highly valuable avenue for future work.

Ethical Considerations

While our CAEM approach enhances the efficiency of building powerful multi-task models, we have considered its potential ethical implications. On the side of potential risks, our discovery of the Founder Effect offers a critical perspective on bias propagation. If a founder module inherits some biases from its source data, our findings suggest these biases could be stubbornly inherited and exert a dominant influence in the merged model, even after it has undergone extensive multi-task fine-tuning. This underscores the critical importance of auditing source models for bias prior to merging. Conversely, on the positive side, CAEM offers a more economical path for resource-constrained institutions to build high-performance multi-task models, thus helping to democratize access to AI technology. Additionally, the one-shot nature of our merging strategy is potentially more computationally efficient than iterative tuning, reducing the environmental footprint of model development.

References

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 794–803. PMLR.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Chen, Yu-Hui Cheng, Quoc V. Le, Zhifeng Chen, Claire Cui, Jeff Dean, HyoukJoong Lee, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P. Bosma, Zongwei Zhou, and 10 others. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR.

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

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3259–3269. PMLR.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P. Vetrov, and Andrew Gordon Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems 31*, pages 8789–8798.

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Arman Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Enneng Jiang, Zhenyi Liu, Li Liu, Xuanjing Huang, and Henghui Zhu. 2024. Adamerging: Adaptive model merging for multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13589–13597.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR.

Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. 2008. Representational similarity analysis - connecting computational models and the brain. volume 2, page 4.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Stas Bekman, Lewis Tunstall, Mélanie Sclar, Julien Chaumond, Thomas Wolf, Sylvain Gugger, Matthew Carrigan, Lysandre Debut, Victor Sanh, Clara Ma, Canwen Xu, Mario Chui, Narsil Patry, Alexander M. Rush, and 13 others. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the ACL-04 Workshop on Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Michael Matena and Colin Raffel. 2022. Merging models with fisher-weighted averaging. In *arXiv preprint arXiv:2212.09258 (Accessed for survey purposes)*.

Yikai Mei, Lixing Chen, Han Xu, Rui Wang, Xue Lin Wang, and Chen Zhao. 2024. FedMoE: A synergistic solution of communication-efficient and personalized federated learning via mixture-of-experts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12469–12479.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. volume 8, pages 842–866. MIT Press.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems 31*, pages 225–236.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Shubham Sodhani, Aniruddh Master, Dweep Rekhi, and Sze-Wing Liu. 2021. Multi-task learning for tabular data: A survey. In *arXiv preprint arXiv:2109.07279 (Accessed for survey purposes)*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without extra inference cost. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23981. PMLR.

Prateek Yadav, Derek Ram, Yogesh Balaji, Bowen Kopic, Yilin Shen, and Graham Neubig. 2023. TIES-Merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems 36*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27*, pages 3320–3328.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems 33*, pages 5822–5833.

Zhen Yuan, Jiacheng Chen, Zhibin Wu, Jiannan Chen, and Yong Dou. 2024. Hi-Merging: A hierarchical vision transformer merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 28249–28259.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28*, pages 649–657.

Yu Zhang and Qiang Yang. 2022. A survey on multi-task learning. volume 34, pages 5586–5609.

A Appendix

A.1 Dataset Preprocessing

The public datasets used in this research adhere to their original licensing terms. Both the SQuAD v1.1 and XSum datasets are released under the Creative Commons Attribution-ShareAlike 4.0 (CC

BY-SA 4.0) license. The AG News corpus does not have a separate explicit license but is widely made available by its creators for academic research. As for the detailed train, validation, and test splits for the datasets used in our experiments are provided in Table 6.

To adapt the three distinct tasks for a single model within a multi-task learning framework, we reformatted the AG News and SQuAD datasets into a sequence-to-sequence format.

For the **AG News** dataset, which only provides an official training and test set, we follow standard practice and create our own validation set by randomly sampling 10% of the 120k training samples. The remaining 90% is used for training, and the original, untouched test set is used for final evaluation. Each input text was prepended with the prompt "Classify: " and appended with an end-of-sequence token (</s>) to signify the end of the input. For the accuracy calculation, a prediction is considered correct only if it is an exact match with one of the four predefined category strings: "World", "Sports", "Business", or "Science and Technology".

For the **SQuAD** dataset, following the common academic convention due to the absence of a public test set, we use the official development (validation) set for both model selection (i.e., choosing the best epoch) and for reporting the final EM scores. As for the input, which consists of a question and a context, was structured as follows: the prompt "Question: " was added before the question, followed by the prompt "Context: " separating the question and the context. The </s> token was appended to the end of the context. During evaluation, all scores were computed using the official SQuAD script within the Hugging Face evaluate library (v0.4.4), which we accessed via evaluate.load("squad") (Lhoest et al., 2021).

For the **XSum** dataset, which is inherently a seq2seq task, we use the official splits and simply prepended the prompt "Summarize: " to the input document and appended the </s> token at the end. The performance was evaluated using the ROUGE-L score between the generated summary and the reference summary.

Dataset	Train	Validation	Test
AG News	108,000	12,000	7,600
SQuAD v1.1	87,599	10,570	N/A
XSum	204,045	11,332	11,334

Table 6: Statistics for the datasets used in our study.

A.2 Details of the AdjustAlloc() Function

This section elaborates on the AdjustAlloc() function from Algorithm 1. The primary objective of this function is to ensure that each task is allocated at least one expert, while the total number of allocated experts does not exceed the expert capacity of the source model (n_s). The allocation process begins by proportionally assigning experts based on the normalized entropy of each task. The initial allocation, $\boldsymbol{N}_{\text{alloc}}$, is determined by taking the floor of these proportional values, and the corresponding fractional remainders are stored in an array, $\boldsymbol{R}$. Next, we iterate through each task's allocation. If a task has been allocated zero experts, its allocation is adjusted to 1. Conversely, if a task's allocation exceeds n_s, it is capped at n_s. After the constraint enforcement, the sum of experts in $\boldsymbol{N}_{\text{alloc}}$ ($\Sigma \boldsymbol{N}_{\text{alloc},i}$) may be greater or less than the target number of experts for the merged model, n_m. Therefore, we deal with the following two cases separately.

- **Case 1:** $\Sigma \boldsymbol{N}_{\text{alloc},i} > n_m$ **(Excess):**

  To reduce the count, we iteratively remove experts. In each iteration, we identify the task(s) with the maximum number of allocated experts. Among these, we select the one with the smallest remainder in $\boldsymbol{R}$ and decrement its expert count by one. This process is repeated until $\Sigma \boldsymbol{N}_{\text{alloc},i}$ equals n_m.

- **Case 2:** $\Sigma \boldsymbol{N}_{\text{alloc},i} < n_m$ **(Insufficient):**

  To distribute the remaining experts, we identify tasks that have not yet reached their expert capacity. If only one task has available capacity, all remaining experts are assigned to it. If two tasks have available capacity, the remaining experts are split. The task with the higher remainder in $\boldsymbol{R}$ receives the larger portion ($\lceil \text{remaining}/2 \rceil$), and the other receives the smaller portion ($\lfloor \text{remaining}/2 \rfloor$). If all three tasks have available capacity, we employ an iterative, round-robin approach. In each round, one expert is assigned to the task with the current highest remainder in $\boldsymbol{R}$. To prevent re-selection, this task's remainder in $\boldsymbol{R}$ is then set to zero. This continues until all deficit experts are allocated.

A.3 Training Details

Detailed training hyperparameters are provided in Table 7. The multi-task learning setup utilized an effective batch size of 72. This was achieved by constructing each batch from three smaller batches (24), with each smaller batch drawn from one of the three respective tasks.

All experiments were conducted on a server with an NVIDIA RTX A6000. The total estimated computation time for all experiments, including single-task fine-tuning and all multi-task runs, was approximately 600 GPU hours. The number of parameters for the Switch Transformer models are approximately 619.34M for the 8-expert version and 1072.40M for the 16-expert version.

Hyperparameter	**Value**
Pretrained Model	`google/switch-base-{8, 16}`
Optimizer	AdamW
Learning Rate	3×10^{-4} (constant)
Batch Size (per task)	24
Batch Size (MTL)	72
Max Epochs	10
AdamW Optimizer Settings	
Adam β_1	0.9
Adam β_2	0.98
Adam ϵ	1×10^{-16}
Weight Decay	0.005
Training Details	
Max Input Length	1000 tokens
Max Target Length	200 tokens
Software and Versions	
Tokenizer	T5TokenizerFast
Transformers	Hugging Face 4.46.3
Evaluate	Hugging Face 0.4.4

Table 7: Hyperparameter settings used for all experiments, which were chosen based on common practices for fine-tuning Switch Transformer models and were not extensively tuned via a large-scale search due to computational constraints.