MIXTURES OF SUBEXPERTS FOR LARGE LANGUAGE CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting Large Language Models (LLMs) to a continuous stream of tasks is a critical yet challenging endeavor. While Parameter-Efficient Fine-Tuning (PEFT) methods have become a standard for this, they face a fundamental dilemma in continual learning. Reusing a single set of PEFT parameters for new tasks often leads to catastrophic forgetting of prior knowledge. Conversely, allocating distinct parameters for each task prevents forgetting but results in a linear growth of the model's size and fails to facilitate knowledge transfer between related tasks. To overcome these limitations, we propose a novel adaptive PEFT method referred to as *Mixtures of SubExperts (MoSEs)*, a novel continual learning framework designed for minimal forgetting and efficient scalability. MoSEs integrate a sparse Mixture of SubExperts into the transformer layers, governed by a task-specific routing mechanism. This architecture allows the model to isolate and protect knowledge within dedicated SubExperts, thereby minimizing parameter interference and catastrophic forgetting. Crucially, the router can adaptively select and combine previously learned sparse parameters for new tasks, enabling effective knowledge transfer while ensuring that the model's capacity grows sublinearly. We evaluate MoSEs on the comprehensive TRACE benchmark datasets. Our experiments demonstrate that MoSEs significantly outperform conventional continual learning approaches in both knowledge retention and scalability to new tasks, achieving state-of-the-art performance with substantial memory and computational savings.

1 INTRODUCTION

Large Language Models (LLMs) have significantly advanced the state of natural language processing (NLP), powering systems that perform tasks such as summarization, question answering, dialogue, translation, and reasoning (Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023). These models are typically pretrained on massive corpora using self-supervised learning and subsequently fine-tuned for specific downstream applications. However, their training paradigm is inherently static: once deployed, LLMs cannot easily incorporate new knowledge or adapt to evolving domains without costly retraining or fine-tuning on large datasets.

In real-world scenarios, where data continuously arrives and user demands shift over time, models must be updated efficiently without degrading past performance. *Continual Learning* (CL) addresses this challenge by enabling models to acquire new tasks sequentially while retaining previously learned knowledge (Parisi et al., 2019). Unfortunately, deep neural networks—especially those with shared parameters—suffer from *catastrophic forgetting* when fine-tuned in sequence (McCloskey & Cohen, 1989; Kirkpatrick et al., 2017), as gradients for new tasks often override information critical for earlier ones. Although CL has been extensively studied in smaller-scale settings such as image classification, adapting it to large-scale language models remains an open problem. LLMs introduce unique challenges for CL, including high memory requirements, difficulty in task boundary detection, and interference due to overparameterization.

Several conventional strategies have been proposed to mitigate forgetting, but most struggle with scalability and efficiency. Replay-based methods (Lopez-Paz & Ranzato, 2017; Chaudhry & et al., 2019) maintain exemplar buffers from previous tasks and interleave them during training. However, storing real data raises privacy and compliance concerns, particularly in sensitive domains such as healthcare or finance. Moreover, replay buffers grow linearly with the number of tasks, limiting

their practicality. Regularization-based approaches such as Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) and Synaptic Intelligence (SI) (Zenke & et al., 2017) and Architecture-based approaches such as Supermasks in Superposition (SupSup) (Wortsman et al., 2020) and Winning SubNetworks (WSN) (Kang et al., 2022) restrict updates to parameters deemed important for past tasks. These methods, while effective in small models - convolutinal neural networks (CNNs), often underperform on high-dimensional parameter spaces like transformers due to limited plasticity and reliance on heuristic importance metrics.

In transformer-based LLM fields, Parameter-efficient fine-tuning (PEFT, Figure 1) offers an attractive alternative by modifying only a small subset of model parameters for each task, reducing computational overhead and allowing quick deployment (Houlsby et al., 2019; Hu et al., 2021a; Li & Liang, 2021). Adapter modules, LoRA (Low-Rank Adaptation), and prefix tuning have shown strong performance in multitask and transfer learning scenarios. However, naive application of PEFT to continual learning still leads to severe forgetting, as task-specific parameters can interfere without proper isolation or coordination. Another promising direction is the use of *Mixture of Experts* (MoE) architectures (Shazeer et al., 2017; Lepikhin & et al., 2020; Fedus & et al., 2022), which introduce sparse routing to different expert subnetworks, increasing model capacity without proportionally increasing inference cost. While MoEs allow for modular computation and dynamic routing, they have not yet been successfully adapted for large language continual learning. Without a mechanism to prevent expert overlap or drift, MoE-based models still experience performance degradation in sequential settings (Zhang & et al., 2023).

To overcome these limitations, we propose a novel method for continual learning in LLMs, titled *M*ixture of *S*ub*E*xperts (MoSEs) for parameter-efficient fine-tuning. Our method is designed to integrate the strengths of modular expert-based computation with the efficiency of PEFT techniques, enabling scalable, robust, and interference-rare LLM continual learning across tasks.

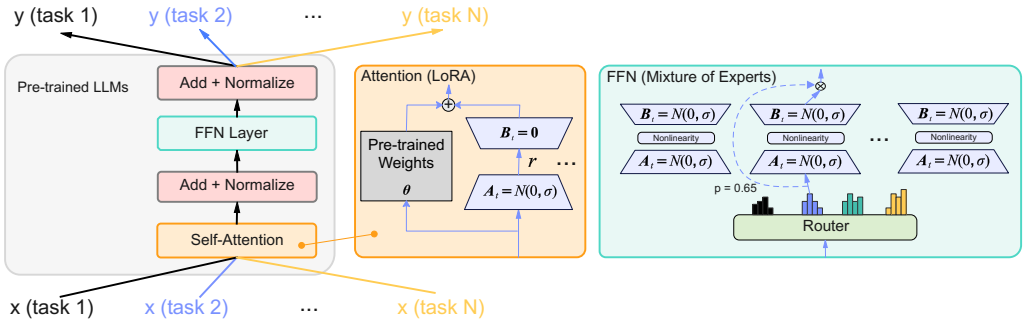


Figure 1: **Continual LLMs**: the objective is to design a fine-tuned transformer that works well across a sequential task: two types of parameter-efficient fine-tuning (PEFT) methods through Low-Rank Adaptation (LoRA) at attention layers and Mixture of Experts (MoEs) at feed forward network (FFN) layer. $\boldsymbol{A}$, $\boldsymbol{B}$ are the learnable parameters, initialized respectively.

Our contributions are summarized as follows:

- We introduce a sparsely-gated Mixture of SubExperts (MoSEs) framework in which sparse experts are selected and their sparse parameters are adaptively overlapped across tasks, thus maximizing the model scalability.
- Adaptively by selecting task-specific sub-experts with prompt embedding in sparse attention layers, our MoSEs achieve minimal catastrophic forgetting while reducing computational overhead, without explicit regularization or replay.
- We evaluate MoSEs on the TRACE benchmark datasets, which include a diverse set of language tasks, demonstrating that MoSEs are the state-of-the-art method in terms of performance, forgetting, and parameter efficiency.

2 Related Work

Continual learning (CL), also known as lifelong learning, aims to enable models to sequentially learn multiple tasks without catastrophic forgetting (Parisi et al., 2019). Classical CL methods are

broadly categorized into regularization-based, replay-based, and dynamic architecture approaches. **Regularization-based methods**, such as Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) and Synaptic Intelligence (SI) (Zenke & et al., 2017), constrain updates to important parameters for past tasks by adding Fisher-based penalties. These methods are effective in small-scale models but degrade under the scale and parameter redundancy of LLMs. **Replay-based methods** mitigate forgetting by storing a subset of past data for rehearsal (Rebuffi et al., 2017; Chaudhry et al., 2019a;b; Saha et al., 2021; Lin et al., 2023; Liang & Li, 2024). Although effective, they raise concerns about data privacy and storage costs, particularly in domains such as healthcare and finance. **Architecture-based models**, such as Progressive Neural Networks (Rusu & et al., 2016), Dynamically Expandable Networks, allocate new sub-networks for each task, and Supermasks in Superposition (Wortsman et al., 2020), which use fixed backbones with binary masks to carve out task-specific subnetworks. While avoiding forgetting, they lead to linear parameter growth and inference inefficiencies, which are prohibitive in large-scale transformer models. To alleviate the linear growth of expanded parameters, Winning SubNetworks (WSN) (Kang et al., 2022) was proposed using a task-adaptive parameter reuse mechanism. However, the effectivness and efficiency of WSN are limited to convolutional neural networks (CNNs). In this work, we design a new subexpert routing continual learning method with computational or memory constraints in LLMs, where retraining or full fine-tuning is costly.

To adapt pretrained language models efficiently, parameter-efficient fine-tuning (PEFT) methods have emerged as practical alternatives to full model updates. These include Adapters (Houlsby et al., 2019), Prefix-Tuning (Li & Liang, 2021), and Low-Rank Adaptation (LoRA) (Hu et al., 2021a). These methods introduce small, trainable modules while keeping most of the backbone frozen. LoRA, for example, inserts low-rank matrices into attention and feed-forward layers, achieving performance comparable to full fine-tuning with significantly fewer trainable parameters. While PEFT methods excel at reducing resource demands and enabling multi-task deployment, they are not designed for sequential learning. In continual learning settings, updating adapters or LoRA weights across tasks can still result in parameter interference unless task-specific modules are carefully isolated. Methods such as AdapterFusion and Prompt Tuning offer some modularity, but fail to scale to long task sequences or handle forgetting explicitly. Recent works have started combining PEFT with continual learning (Zhang & et al., 2023), but often lack routing mechanisms that control expert sharing or protection between tasks.

To address the issues of routing mechanisms, Mixture-of-Experts (MoE) models were introduced to increase capacity without a proportional increase in compute by activating only a subset of experts per input (Shazeer et al., 2017; Lepikhin & et al., 2020; Fedus & et al., 2022). MoEs have shown remarkable scalability in language models such as GLaM (Du & et al., 2022), Switch Transformers (Fedus & et al., 2022), and others. The sparse, conditional computation of MoEs offers a natural avenue for continual learning: individual experts can potentially be assigned or reserved for specific tasks. However, most MoE architectures are trained in multitask settings with joint routing optimization. This leads to entangled expert specialization and makes it difficult to prevent interference during sequential task updates. Additionally, conventional routing mechanisms are typically optimized end-to-end and lack control over expert assignment post-training. As a result, they do not support continual task addition or expert freezing without retraining the full model. Recent works such as task-adaptive routing (Zhang & et al., 2023) or modular continual learning have begun exploring MoE-based solutions, but they remain limited in scalability or require task identity at inference. Our work builds on these insights and thorough observations by combining parameter-efficient fine-tuning with task-aware, interference-free expert assignment, offering a novel, sparse, and scalable solution to continual learning in LLMs.

3 PREREQUISITES

We begin by reviewing conventional fine-tuning approaches, including Low-Rank Adaptation (LoRA) and Mixture-of-Experts (MoE), and discuss the key challenges these methods present for continual learning (CL) in large language models (LLMs).

3.1 PRELIMINARIES

Problem Statement. Continual Learning (CL) involves training deep neural networks (DNNs) on a time-varying data stream represented as a sequence of tasks, $\mathcal{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_\mathcal{T}\}$. Each task

$\mathcal{D}_t = \{(\boldsymbol{x}_i^t, \boldsymbol{y}_i^t)\}_{i=1}^{n_t}$ consists of n_t data points, where $\boldsymbol{x}_i^t \in \mathcal{X}_t$ denotes an input sample, and $\boldsymbol{y}_i^t \in \mathcal{Y}_t$ is the corresponding label. Upon arrival of the t-th task $\mathcal{D}_t$, the model $f_{\boldsymbol{\theta}}$ is updated using only the current task data, as previous data $\mathcal{D}_{<t}$ from earlier tasks are no longer accessible.

This work focuses primarily on task-agnostic incremental learning (TaIL), a challenging CL setting, where the task identity is available during training as the sequence index but is not provided at inference time. The learning objective in this setting is formally as follows:

$$\max_{\boldsymbol{\theta}} \sum_{t=1}^{T} \sum_{(\boldsymbol{x}_i^t, \boldsymbol{y}_i^t) \in \mathcal{D}_t} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_i^t | \boldsymbol{x}_i^t) \tag{1}$$

3.2 Low-Rank Adaptation (LoRA)

Low-Rank Adaptation (LoRA) (Hu et al., 2021a), stated in Figure 1, is a parameter-efficient fine-tuning method tailored to adapt large pre-trained models, such as Transformers, by introducing trainable low-rank matrices into selected layers while keeping the original weights $\boldsymbol{\theta}$ frozen at all attention layers. This strategy has proven effective in continual learning scenarios involving large language models (LLMs), where both computational efficiency and mitigation of catastrophic forgetting are essential.

Rather than updating the full pre-trained weight matrix $\boldsymbol{\theta} \in \mathbb{R}^{d \times d}$ during training, LoRA re-parameterizes it as:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \Delta\boldsymbol{\theta}, \quad \text{where} \quad \Delta\boldsymbol{\theta} = \boldsymbol{B}\boldsymbol{A}, \tag{2}$$

where $\boldsymbol{A} \in \mathbb{R}^{r \times d}$ and $\boldsymbol{B} \in \mathbb{R}^{d \times r}$ are low-rank matrices with rank $r \ll d$. The base matrix $\boldsymbol{\theta}$ is frozen and only low-rank matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are trained. The update $\Delta\boldsymbol{\theta}$ is scaled by a factor α/r to regulate its magnitude. During inference, the low-rank update can either be merged into $\boldsymbol{\theta}$ for efficient execution or retained as a modular adapter to allow flexible task switching.

LoRA offers several benefits for continual learning in LLMs. It significantly reduces the number of trainable parameters, allowing for scalable multi-task training. Since the base model is kept frozen, it effectively preserves prior task knowledge, helping to prevent catastrophic forgetting. Moreover, its modularity facilitates task-specific adaptation and transfer learning while also reducing storage and communication costs by requiring only the lightweight adapter weights to be stored or transferred.

However, LoRA also presents several limitations in real-world scenarios. As the number of tasks increases, managing a large collection of adapters can introduce memory and storage challenges. During deployment, if adapters are not merged into the base model, switching tasks may necessitate loading different sets of weights, adding complexity to the inference process. Additionally, when adapters are reused across tasks, interference between tasks can arise unless special care is taken to manage this sharing.

3.3 Mixture-of-Experts (MoE)

Mixture-of-Experts (MoE), shown in Figure 1, is a modular architecture that enhances neural networks by introducing a set of parallel expert subnetworks, combined with a routing mechanism that selects a subset of these experts for each input (Shazeer et al., 2017) at all feed-forward network (FFN) layers. MoE introduces a set of N expert networks and a routing function that selects a sparse subset of $k \ll N$ experts for each input, allowing conditional computation. For input $\boldsymbol{x}$, the output is:

$$\text{MoE}(\boldsymbol{x}) = \sum_{j \in \text{TopK}(R(\boldsymbol{x}))}^{k} R_j(\boldsymbol{x}) \cdot E_j(\boldsymbol{x}), \tag{3}$$

where $R_j(\boldsymbol{x})$ is the dense routing weight and E_j is the j-th dense expert. This approach has gained significant traction in scaling large language models (LLMs) efficiently and is increasingly adopted in continual learning scenarios.

In the MoE architecture, each expert is a feedforward network, and a trainable gating function determines which k out of N experts should process a given input. The use of sparse gating - activating only a small subset of experts - ensures that the computational overhead remains low, even when the number of experts is large.

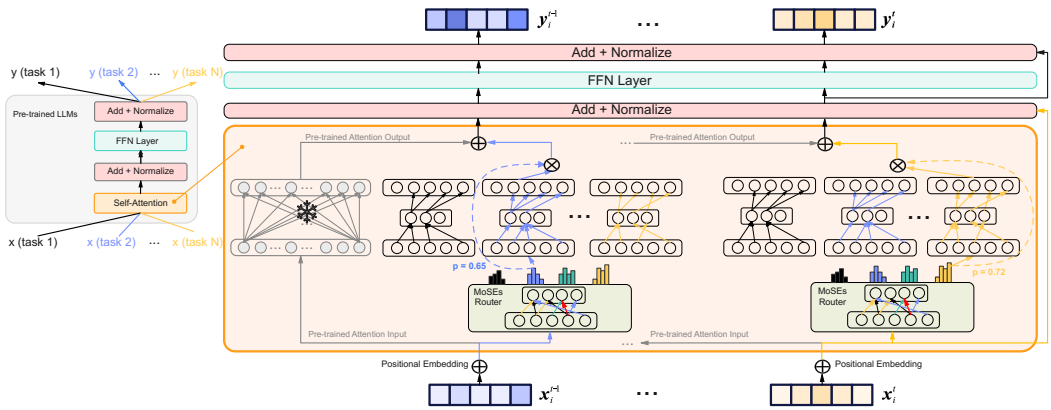


Figure 2: **Mixtures of SubExperts (MoSEs)**: The Self-Attention layer is fine-tuned by MoSEs to operate on task-specific tokens. Given, $\boldsymbol{x}_i^{t-1}$ and $\boldsymbol{x}_i^t$, the MoSEs router adaptively distributes them across N sub-experts. Each token is routed to the most relevant subexperts, and the final output is computed as the weighted sum of the selected subexpert outputs, where the weights correspond to the router gate values (e.g., $p = 0.65$ or $p = 0.72$).

Specifically, conventional MoE models offer compelling advantages for continual learning. Experts can specialize in different tasks or domains, which helps preserve previously learned knowledge and reduces catastrophic forgetting. With sparse activation, MoE allows scaling up the model's capacity without increasing inference cost linearly. Like LoRA, only a fraction of the parameters is updated per input, enabling efficient training on task-specific data and memory reuse. New tasks can be handled by introducing or fine-tuning a subset of experts, facilitating incremental learning without touching the base model.

Despite their promise, MoE poses several challenges in practice: Certain experts may be overused, while others remain underutilized, leading to inefficient learning and poor generalization. The gating mechanism can introduce instability or collapse, i.e., routing to the same experts, requiring careful regularization and training tricks in continual learning. Task-specific routing and expert configurations complicate deployment, especially in constrained environments. Moreover, as the number of tasks increases, managing and storing expert configurations per task becomes a scalability bottleneck.

4 Our Mixtures of SubExperts (MoSEs)

To address the issues led by the classical MoE, we propose a new continual learning framework, *Mixtures of SubExperts (MoSEs)*, designed for large-scale language models, as shown in Figure 2. MoSEs enable scalable and efficient continual learning by combining sparse mixture-of-subexperts' representations, task-specific sparse routing, and parameter-efficient fine-tuning at attention layers. To control model capacity growth, MoSEs select sparse subexperts and adaptively reuse previously learned weights. Unlike traditional methods that require loading separate weights per task, our MoSEs simplify task management through binary subnets and prompt-based control. To ensure a balanced utilization of subexperts, we reinitialize the gating and expert score parameters $\boldsymbol{s}$ at the beginning of each new task. Furthermore, to avoid routing instability and expert collapse, MoSEs apply top-c% subexpert sparsity selection without freezing previously learned parameters, effectively preserving prior knowledge while enabling robust and stable task adaptation.

4.1 Mixtures of SubExperts

Similar to LoRA, sparse Mixture-of-SubExperts (MoSEs) are augmented with pre-trained parameters at selected transformer attention layers. Each MoSE layer consists of a pool of N sub-neural experts and a trainable sparse routing function. For a given input representation $\boldsymbol{x}$, the sparse router computes a score vector $\hat{R}(\boldsymbol{x}) \in \mathbb{R}^N$ that weights the relevance of each sparse expert $\hat{E}(\boldsymbol{x})$. A sparse top-k selection is applied to activate only the most relevant experts:

$$\text{MoSE}^l_{\tilde{\boldsymbol{\theta}} \odot \boldsymbol{m}^t}(\boldsymbol{x}) = \sum_{j \in \text{TopK}(\hat{R}^l(\boldsymbol{x}))} \hat{R}^l_{\tilde{\boldsymbol{\theta}} \odot \boldsymbol{\delta}^t_j}(\boldsymbol{x}) \cdot \hat{E}^l_{\tilde{\boldsymbol{\theta}} \odot \boldsymbol{\xi}^t_j}(\boldsymbol{x}), \tag{4}$$

where $\hat{R}^l_{\tilde{\boldsymbol{\theta}}\odot\boldsymbol{\delta}^t_j}(\boldsymbol{x})$ is the sparse routing weight represented by sparse parameters $\tilde{\boldsymbol{\theta}} \odot \boldsymbol{\delta}^t_j$ and $\hat{E}_{\tilde{\boldsymbol{\theta}}\odot\boldsymbol{\xi}^t_j}$ is the j-th sparse expert calculated by $\tilde{\boldsymbol{\theta}} \odot \boldsymbol{\xi}^t_j$ at t-th task; binary masks $\boldsymbol{m}^t = \{\boldsymbol{\delta}^t, \boldsymbol{\xi}^t\}$ obtained by selecting top-c% of each weight score function $\boldsymbol{s}$ and $\tilde{\boldsymbol{\theta}}$ are learnable parameters, respectively. Only selected subexperts are involved in the forward and backward passes, ensuring computational efficiency and localized updates.

Each task is assigned its sparse routing mask to mitigate forgetting, activating a distinct subset of sparse experts. Upon arrival of a new task t, we assign a layer-wise task-specific expert subset $\text{MoSE}^l_{\tilde{\boldsymbol{\theta}}\odot\boldsymbol{m}^t_j}$, train only the parameters of $\hat{E}_{\tilde{\boldsymbol{\theta}}\odot\boldsymbol{\xi}^t_j}$ and the routing weights $\hat{R}_{\tilde{\boldsymbol{\theta}}\odot\boldsymbol{\delta}^t_j}$. The hidden state $\boldsymbol{h}^l$ at the l-th layer is adapted by MoSE as follows:

$$\boldsymbol{h}^l = \boldsymbol{x}\boldsymbol{\theta}^l + \beta \cdot \text{MoSE}^l_{\tilde{\boldsymbol{\theta}}\odot\boldsymbol{m}_t}(\boldsymbol{x}) \tag{5}$$

where the pre-trained weight $\boldsymbol{\theta}^l \in \mathbb{R}^{m\times n}$ and mixtures of subexports, MoSE^l at layer l; $\beta = \alpha/r$ is determined by the adaptation rate α and rank size r. This approach ensures that each task operates within a minimally interfering subnetwork, enabling task-specific adaptation without overwriting prior knowledge $\boldsymbol{\theta}^l$. Task interference is further reduced by enforcing orthogonality in expert usage patterns via the following task prompts.

4.2 Task Adaptation

The prompt $\boldsymbol{E} = \{\boldsymbol{e}^l_t\}^{\mathcal{T}}_{t=1}$ is a set of task-t dependent parameters of the l-th attention layer, where $\boldsymbol{e}^l_t \in \mathbb{R}^{L_e\times D}$ has as sequence length L_e, and $\mathcal{T}$ is the total number of tasks. Each $\boldsymbol{e}_t$ is associated with a task-specific key $\boldsymbol{k}_t \in \mathbb{R}^D$, which is also a learnable parameter aimed at capturing representative features of a single task. To obtain a task adaptive hidden representation $\tilde{\boldsymbol{h}}$ in the l-th attention layer, we concatenate the prompt $\boldsymbol{e}^l_t$ into the l-th input hidden state $\boldsymbol{x}^l$, as follows:

$$\tilde{\boldsymbol{h}}^l = \tilde{\boldsymbol{x}}\boldsymbol{\theta}^l + \beta \cdot \text{MoSE}^l_{\tilde{\boldsymbol{\theta}}\odot\boldsymbol{m}_t}(\tilde{\boldsymbol{x}}) \tag{6}$$

where $\tilde{\boldsymbol{x}} = [\boldsymbol{e}^l_t; \boldsymbol{x}^l]$.

Algorithm 1 MoSEs at training time

Input: MoSEs $f_{\boldsymbol{\theta}\odot\boldsymbol{m}}$, ,
  Training set $\{\{\boldsymbol{x}_{i,t}, y_{i,t}\}^{n_t}_{i=1}\}^{\mathcal{T}}_{t=1}$, Prompt $\boldsymbol{E} = \{\boldsymbol{e}^l_t\}^{\mathcal{T}}_{t=1}$, Task keys $\boldsymbol{K} = \{\boldsymbol{k}_t\}^{T}_{t=1}$,
  subnetworks $\boldsymbol{M} = \{\boldsymbol{m}_t\}^{\mathcal{T}}_{t=1}$, Learnable parameters $\tilde{\boldsymbol{\theta}}$, Score parameters $\boldsymbol{s}$,
  number of training epochs of the t-th task $\mathcal{K}_t$, fine-tuning layers $[start_e, end_e]$.
Initialize: $\boldsymbol{\theta}, \boldsymbol{E}, \boldsymbol{M}, \boldsymbol{K}$
for task $t = 1, \cdots, \mathcal{T}$ **do**
  Select the task-specific Prompt $\boldsymbol{e}_t$ and corresponding task key $\boldsymbol{k}_t$.
  Generate the MoSEs $f_{\boldsymbol{e}_t,\tilde{\boldsymbol{\theta}}\odot\boldsymbol{m}_t}$: attach $\boldsymbol{e}_t$ to $start_e$-th to end_e-th MSA layers, with $f_{\boldsymbol{\theta}\odot\boldsymbol{m}_t}$.
  for epoch $e = 1, \cdots, \mathcal{K}_t$ **do**
    Draw a mini-batch $B = \{(\boldsymbol{x}_{i,t}, y_{i,t})\}^{n_t}_{i=1}$
    for $(\boldsymbol{x}, \boldsymbol{y})$ in B **do**
      Obtain binary masks $\boldsymbol{m}_t$ of the top-c% scores $\boldsymbol{s}$ at each target layer.
      Calculate the per sample loss $\mathcal{L}_{total} = \mathcal{L}_{task}(\tilde{\boldsymbol{\theta}} \odot \boldsymbol{m}_t; \boldsymbol{x}, \boldsymbol{e}_t, \boldsymbol{y}) + \lambda_{pull} \cdot \mathcal{L}_{pull}(\boldsymbol{K}; \boldsymbol{x})$.
    end for
    Update $\boldsymbol{E}, \boldsymbol{M}, \boldsymbol{K}, \tilde{\boldsymbol{\theta}}, \boldsymbol{s}$ by back-propagation
  end for
end for

4.3 Optimizations of MoSEs

To ensure that the selected prompt keys remain semantically aligned with the input features, we define a *pull constraint loss* that maximizes the cosine similarity between the normalized prompt keys $\hat{\boldsymbol{k}}_t$ and the normalized input embeddings $\hat{\boldsymbol{x}}_{i,t}$. Formally, for a batch of B inputs with one selected prompt key per sample, the loss is defined as:

$$\mathcal{L}_{\text{pull}} = -\frac{1}{B}\sum_{i=1}^{B}\frac{1}{\mathcal{T}}\sum_{t=1}^{\mathcal{T}}\langle\hat{\boldsymbol{x}}_{i,t}, \hat{\boldsymbol{k}}_t\rangle. \tag{7}$$

This term encourages the model to select prompts that are maximally aligned with task-specific representations. During inference, for each input, the model selects the most correlated prompt key $\boldsymbol{k}_t$ based on cosine similarity to the input embedding. The task ID associated with the selected prompt key is then used by the MoSEs to route the input $\tilde{\boldsymbol{x}} = [\boldsymbol{e}_t; \boldsymbol{x}]$ through task-specific subnetworks and generate predictions. This design allows the model to dynamically adapt to task semantics without explicit task labels in the inference step. The total training objective of MoSEs becomes the following:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda_{\text{pull}} \cdot \mathcal{L}_{\text{pull}}, \tag{8}$$

where λ_{pull} is a coefficient that balances the influence of the pull constraint. In all experiments, we set $\lambda_{\text{pull}} = 0.1$ and adopt Algorithm 1 for training and Algorithm 2 for evaluation.

5 Experiments

We validate our method on large language benchmark datasets: a subset (TRACE 0.5K) and a full set (5K), against continuous learning baselines.

5.1 Experimental Settings

Datasets. To evaluate our MoSEs, the TRACE benchmark dataset (Wang et al., 2023) is utilized, which was constructed based on three key principles: TRACE is free from contamination by LLM pretraining corpora, it presents meaningful challenges to current large language models, and it encompasses a diverse range of tasks. To meet these criteria, Wang et al. (2023) curated multiple datasets and provided two standardized versions: (1) the **TRACE 0.5K** subset, with 500 training and 100 testing samples per task, and (2) the **TRACE 5K** full set, comprising 5,000 training and 2,000 testing samples per task. The full set ensures class balance and yields a total of 40,000 training and 16,000 testing examples. We use TRACE 0.5K for the main evaluation of MoSEs, while TRACE 5K is used in the Appendix (see Table 7) further to demonstrate the superiority of MoSEs over baseline methods. TRACE evaluates models across five core capabilities: Factual Knowledge using MMLU (Hendrycks et al., 2020) with 5-shot accuracy based on perplexity; General Reasoning using Big-Bench-Hard (Suzgun et al., 2022) with chain-of-thought prompting and 3-shot exact match (EM); Multilingual Understanding via TyDiQA (Clark et al., 2020) with 0-shot F1 across 11 languages; Commonsense Reasoning using PIQA with 0-shot accuracy; and Reading Comprehension using BoolQ (Clark et al., 2019) with 0-shot accuracy. In addition, TRACE assesses instruction-following ability using two datasets: Self-Instruct (Wang et al., 2022), which provides 175 diverse prompts, and LIMA (Li et al., 2023), which includes 300 high-quality prompts curated from community interactions and expert knowledge. For safety evaluation, TRACE incorporates CoNa (Bianchi et al., 2023), a dataset containing 178 expert-annotated prompts targeting sensitive and potentially harmful instructions, such as those involving hate speech.

Baselines & MoSEs. We evaluate the performance of MoSEs in a continual learning setting, comparing four approaches—three requiring training and one not (ICL):

- In-Context Learning (ICL): Task demonstrations are supplied as part of the language prompt, acting as a form of prompt engineering (Brown et al., 2020). A 6-shot setting is used for our experiments.
- Single Full-Parameter Fine-Tuning (Single FT): This method involves training all model parameters in each single task.
- LoRA-based Sequential Fine-Tuning (LoRA): Only the low-rank LoRA ($r = 8, \alpha = 32$, 4.19M) matrices are finetuned, leaving the LLM backbone fixed (Hu et al., 2021b). This method is chosen based on prior findings of reduced forgetting with "Efficient Tuning" (Liu & Huang, 2023).
- Mixture of Experts-based Sequential Fine-Tuning (MoE): This method (Shazeer et al., 2017; Lepikhin & et al., 2020; Fedus & et al., 2022) introduces sparse routing to different expert subnetworks with fixed model capacity without proportionally increasing inference cost.
- Our **MoSEs** demonstrate its effectiveness on the Trace Benchmark datasets using **E2T2**, which denotes 2 experts with selection of top-2 and the length of the prompt sequence $L_e = 1$.

Training & Inference. We evaluate and analyze the performance and behavior of our MoSEs framework under two distinct continual learning settings: Task-Agnostic Incremental Learning (TaIL) and Task-Incremental Learning (TIL).

Table 1: (TaIL) Performances of MoSEs (W/O[0-1], E2T2 ($r = 2, \alpha = 8$)) across the TRACE benchmark datasets (0.5K). Single FT refers to fine-tuning the model on single task and MT refers to Multi-task training.

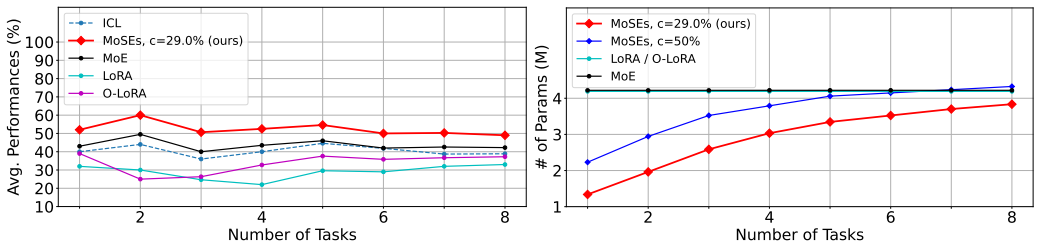
Dataset	C-STANCE	FOMC	MeetingBank	Py150	ScienceQA	NumGLUE-cm	NumGLUE-ds	20Minuten	Avg.	BWT	# Params	Train / Test
ICL	0.40	0.48	0.20	0.52	0.63	0.28	0.20	0.40	39.5	-	-	- / 1.25h
Single FT	0.52	0.71	0.60	0.58	0.79	0.44	0.63	0.28	57.6	-	7.00B	201.8h / 1.25h
LoRA	0.32	0.28	0.14	0.14	0.60	0.26	0.50	0.40	33.1	-22.67 %	4.19M	20.18h / 1.25h
O-LoRA	0.39	0.11	0.29	0.52	0.57	0.27	0.42	0.41	37.3	-12.53 %	4.19M	20.20h / 1.25h
MoE	0.43	0.56	0.21	0.54	0.56	0.22	0.46	0.40	42.2	-11.10 %	4.22M	20.18h / 1.25h
MoSEs, c=29.0%	0.52	0.68	**0.32**	**0.58**	**0.63**	**0.27**	**0.52**	0.40	**49.1**	-0.90 %	**3.82M**	**20.16h / 1.20h**
MoSEs, c=30.0%	**0.58**	**0.71**	0.31	0.54	0.57	0.25	0.52	**0.41**	48.4	**-0.43 %**	3.83M	20.16h / 1.20h
MT of LoRA	0.44	0.68	0.44	0.60	0.72	0.33	0.57	0.39	52.3	-	4.19M	20.18h / 1.25h
MT of MoSEs, c=30.0%	0.54	0.72	0.44	0.60	0.72	0.44	0.58	0.41	55.6	-	3.83M	20.18h / 1.20h

- Task-Incremental Learning (TIL): This is the simpler scenario. The model knows the specific task ID for both training and inference.
- Task-Agnostic Incremental Learning (TaIL): This setting presents a more challenging and realistic scenario. During the training phase, the model is provided with sequential task IDs. In contrast, during inference, it must infer the task on its own by utilizing a task ID that adapts to the given input. To bridge this gap, we leverage prompts designed to be relevant to each specific task ID. This enables the model to implicitly recognize the task from the input data during testing, even without receiving an explicit task ID.

5.2 Performances

Effectiveness of MoSEs. The results in Table 1 demonstrate the effectiveness of MoSEs in task-agnostic continual learning (TaIL) across multiple dimensions. MoSEs achieve superior average performance (49.1 with $c = 29\%$ and 48.4 with $c = 30\%$), outperforming conventional approaches such as LoRA (33.1), O-LoRA (37.3), and MoE (42.4). Importantly, MoSEs significantly mitigates catastrophic forgetting, as shown by its minimal backward transfer (BWT) of -0.90% and -0.43%, compared to much higher forgetting in LoRA (-22.67 %) and MoE (-11.10 %). Despite these gains, MoSEs remain parameter-efficient, using only 3.82M–3.83M trainable parameters - fewer than other baselines - and also reduces test-time latency to 1.20 hours, offering both performance and efficiency advantages. Overall, these results highlight that MoSEs provide superior performance and transfer efficiency while reducing memory and inference overhead, making it a practical and scalable solution for continual learning in large language models.

Performances & Parameter Efficiency. The results presented in Figure 3 highlight the strong performance and parameter efficiency of MoSEs across increasing numbers of tasks in the TRACE 0.5K benchmark. As shown in Figure 3(a), MoSEs with a 29.0% expert selection ratio consistently outperform other baselines - such as LoRA, O-LoRA, and MoE - in terms of average performance, maintaining higher accuracy as the number of tasks grows. Meanwhile, Figure 3(b) demonstrates that MoSEs achieve this superior performance with significantly fewer trainable parameters. Specifically, MoSEs with $c = 29.0\%$ require notably fewer parameters than MoE and scales more efficiently compared to other baselines, even as tasks accumulate. This underscores MoSEs' ability to strike a favorable balance between continual learning effectiveness and computational efficiency.



(a) Average Performances of MoSEs (b) # of Training Parameters (Million, M).

Figure 3: **Average Performances and Model Capasity of MoSEs** on TRACE 0.5K.

5.3 Ablation Studies

Sparse MoSEs Table 2 highlights the effectiveness of sparsity in MoSEs under the task-incremental learning (TIL) setting on the TRACE 0.5K benchmark. Interestingly, the sparse MoSEs configuration

Table 2: (TIL) Performances of MoSEs (W/O[0-1], E2T2 ($r = 2, \alpha = 8$)) across the TRACE benchmark datasets (0.5K).

Dataset	C-STANCE	FOMC	MeetingBank	Py150	ScienceQA	NumGLUE-cm	NumGLUE-ds	20Minuten	Avg.	BWT	# Params	Train / Test
MoSEs, c=30.0%	0.53	**0.74**	0.39	0.57	**0.73**	**0.44**	**0.58**	**0.41**	**54.9**	**+0.05 %**	**3.83M**	**20.16h / 1.20h**
MoSEs, c=40.0%	0.56	0.72	**0.41**	0.58	0.71	0.37	0.54	0.40	53.6	-0.71 %	4.01M	20.17h / 1.20h
MoSEs, c=50.0%	**0.63**	0.71	**0.41**	0.57	0.72	0.41	0.53	0.40	54.7	+0.16%	4.32M	20.18h / 1.20h

Table 3: (TIL) Layer-wise Performances of MoSEs c=30.0%, E2T2 ($r = 2, \alpha = 8$) across the TRACE benchmark datasets (0.5K). W[0-31] denotes finetuning of all layers, while W/O[start-end] denotes skip the layers of [start-end] without using any learnable parameters.

Dataset	C-STANCE	FOMC	MeetingBank	Py150	ScienceQA	NumGLUE-cm	NumGLUE-ds	20Minuten	Avg.	BWT	# Params	Train / Test
MoSEs, W[0-31]	0.51	0.75	0.39	0.57	0.68	0.36	0.54	0.47	52.6	-1.50 %	4.13M	20.45h / 1.21h
MoSEs, W/O[0-1]	0.53	0.74	0.39	0.57	**0.73**	**0.44**	**0.58**	**0.41**	**54.9**	**+0.05** %	3.83M	20.20h / 1.20h
MoSEs, W/O[0-2]	**0.55**	**0.77**	0.37	**0.59**	0.68	0.42	0.55	0.41	54.3	-1.30 %	3.88M	20.16h / 1.19h
MoSEs, W/O[0-3]	0.51	0.66	0.39	**0.59**	0.70	0.36	0.49	0.40	51.2	-1.20 %	3.61M	20.02h / 1.18h
MoSEs, W/O[0-4]	0.50	0.68	**0.40**	0.58	0.71	0.42	0.54	0.39	52.7	-2.30 %	3.48M	19.98h / 1.17h
MoSEs, W/O[0-5]	0.50	0.68	**0.40**	0.58	0.71	0.42	0.54	0.39	51.2	-2.90 %	**3.35M**	**19.97h / 1.16h**

with c=30.0% achieves the highest average performance of 54.9%, outperforming denser variants with c=40.0% and c=50.0%, which yield 53.6% and 54.7%, respectively. In addition to superior accuracy, the c=30.0% model also exhibits favorable backward transfer (BWT) of +0.05%, while maintaining the smallest parameter footprint (3.83M) among the three. Despite slight gains in BWT at higher sparsity (e.g., +0.16% for c=50.0%), the performance does not improve accordingly. These findings demonstrate that properly sparse parameters of previously learned and newly learned parameters in the attention layer (e.g., c=30.0%) strike an optimal balance between accuracy, stability, and efficiency in continual learning.

Layer-wise Efficiency of MoSEs. Table 3 presents the layer-wise performance of MoSEs ($c = 30\%$) and reveals that selective subnet tuning contributes to efficient performance with minimal parameter overhead. The full-layer version, MoSEs W[0-31], achieves an average accuracy of 52.6 with a BWT of -1.50% and parameter count of 4.13M. Interestingly, **MoSEs W/O[0-1]**—which skips tuning the bottom two layers—improves both average performance (54.9) and BWT (+0.05%) while reducing parameters to 3.83M, indicating that early layers may not be essential for task adaptation and can even lead to forgetting. As more lower layers are excluded, e.g., W/O[0-4], W/O[0-5], the model size decreases further to 3.35M, but at the cost of degraded average accuracy (51.2) and worsened forgetting (BWT up to -2.90%). This trend demonstrates a clear trade-off: while excluding lower layers can improve efficiency, overly aggressive layer skipping degrades overall knowledge retention and transfer. These findings confirm that MoSEs are robust and efficient, protecting universal language understanding of transformers, especially when low-level layers are strategically frozen. The optimal balance is achieved by MoSEs W/O[0-1], which combine strong performance, positive transfer, and parameter savings.

Effect of Expert Configuration in MoSEs. Table 4 illustrates how varying the number of experts ($E\#$) and top-K selections ($T\#$) affects the performance and efficiency of MoSEs. The baseline setting **MoSEs, E2T2**, with two experts and top-2 selection, achieves the best average performance (54.9) and positive backward transfer (+0.05%) with only 3.83M parameters and a test time of 1.20h, indicating an optimal trade-off between capacity and knowledge retention. As the number of experts increases to three (E3T2, E3T3) or four (E4T2, E4T3), both the parameter count (4.94M–6.01M) and forgetting grow significantly (BWT drops to as low as -3.20%), while the average accuracy declines (down to 48.8). These results suggest that increasing expert diversity without sufficient selectivity can dilute task-specific knowledge and introduce interference. Additionally, the training time slightly increases with more experts (e.g., from 20.20h to 20.21h), and test time remains relatively constant, showing that runtime cost is not the main bottleneck. Overall, the configuration E2T2 demonstrates the best balance across all metrics, affirming that a minimal yet well-structured mixture of experts is most effective for continual learning.

MoSEs of Low-Rank and Scaling Factor Table 5 indicate that MoSEs with $r = 2, \alpha = 8$ achieve the best trade-off between performance and efficiency. It records the highest average score (54.9), maintains a *positive backward transfer* (+0.05%), and uses the least number of parameters (3.83M) with no increase in training or inference time. Increasing the rank and scaling factor leads to larger models but *worsens BWT* and typically results in *lower performance*, suggesting diminishing returns with higher capacity.

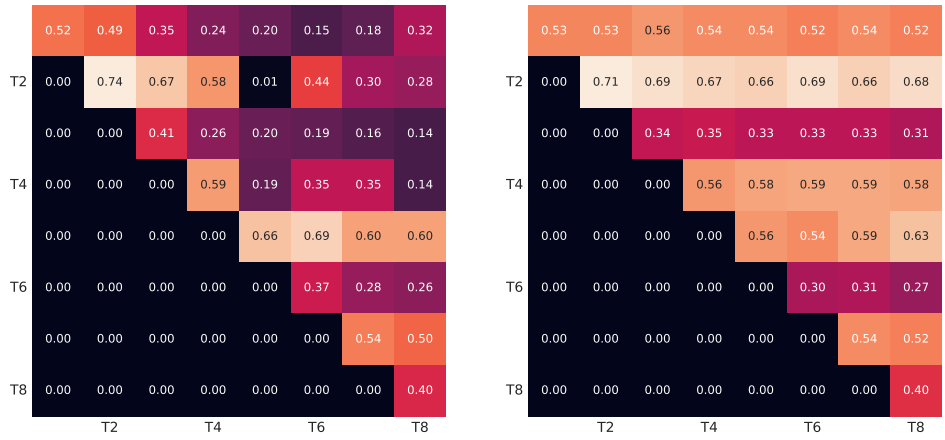
Table 4: (TIL) Expert Performances of MoSEs (W/O[0-1]), c= 30.0% ($r = 2, \alpha = 8$) across the TRACE benchmark datasets (0.5K).

Dataset	C-STANCE	FOMC	MeetingBank	Py150	ScienceQA	NumGLUE-cm	NumGLUE-ds	20Minuten	Avg.	BWT	# Params	Train / Test
MoSEs, E2T2	0.53	**0.74**	**0.39**	**0.57**	**0.73**	**0.44**	**0.58**	**0.41**	**54.9**	**+0.05 %**	**3.83**M	**20.20h / 1.20h**
MoSEs, E3T2	0.52	0.62	0.36	0.54	0.64	0.41	0.52	0.41	50.2	-2.00 %	4.94M	20.20h / 1.20h
MoSEs, E3T3	**0.55**	0.70	0.36	0.56	0.62	0.32	0.51	0.41	50.4	-2.70 %	4.94M	20.20h / 1.20h
MoSEs, E4T2	0.54	0.72	0.32	0.56	0.63	0.36	0.54	0.41	50.9	-2.50 %	6.01M	20.21h / 1.21h
MoSEs, E4T3	0.48	0.68	0.34	0.54	0.60	0.35	0.52	0.40	48.8	-3.20 %	6.01M	20.21h / 1.21h

Table 5: (TIL) Rank-wise Performances of MoSEs (W/O[0-1]) c=30.0%, E2T2 across the TRACE benchmark datasets (0.5K). Note that r is low-rank size and α is a scaling factor.

Dataset	C-STANCE	FOMC	MeetingBank	Py150	ScienceQA	NumGLUE-cm	NumGLUE-ds	20Minuten	Avg.	BWT	# Params	Train / Test
MoSEs, $r = 2, \alpha = 8$	**0.53**	**0.74**	0.39	0.57	0.73	**0.44**	**0.58**	**0.41**	**54.9**	**+0.05 %**	**3.83**M	**20.20h / 1.20h**
MoSEs, $r = 3, \alpha = 12$	0.52	0.70	0.38	0.62	0.66	0.41	0.53	0.41	52.8	-0.99 %	5.03M	20.20h / 1.20h
MoSEs, $r = 4, \alpha = 16$	0.41	0.71	0.40	0.56	0.71	0.35	0.53	0.40	50.8	-1.22 %	5.95M	20.21h / 1.20h
MoSEs, $r = 8, \alpha = 32$	0.26	0.66	0.40	0.61	0.68	0.35	0.55	0.41	48.9	-1.03 %	8.61M	20.22h / 1.20h
MoSEs, $r = 16, \alpha = 64$	0.31	0.64	**0.41**	**0.62**	**0.74**	0.43	0.57	0.40	51.4	-0.34 %	16.90M	20.23h / 1.20h

Backward Transfer Analysis. Figure 4 shows the task-wise transfer matrices of LoRA and MoSEs, where each row represents the performance of a source task after learning a target task. The MoSEs transfer matrix (fig. 4b) demonstrates significantly better retention of past knowledge, indicated by consistently high values in the upper triangular entries (i.e., backward transfer from newer to older tasks). Specifically, after learning new tasks (e.g., T6 or T8), the performance on earlier tasks (e.g., T2 and T4) remains stable, with values above 0.50—a strong indicator of positive or non-destructive backward transfer. In contrast, LoRA (fig. 4a) shows degraded values in many such entries (e.g., below 0.30), suggesting more severe forgetting. This result confirms that MoSE's task-specific subnetwork selection and expert reuse mechanism effectively prevent catastrophic forgetting by preserving prior task knowledge, enabling robust and scalable continual learning in LLMs.



(a) LoRA, $r = 8, \alpha = 32$ (b) MoSEs (ours), $c = 29.0\%$

Figure 4: (TaIL) Transfer Matrixes on the TRACE (0.5K) measured by source and target.

6 Conclusion

Adapting large language models (LLMs) to a continuous stream of tasks remains a challenging problem for continual learning. Conventional parameter-efficient fine-tuning (PEFT) methods suffer from either catastrophic forgetting when reusing a single parameter set or unscalable memory growth when allocating task-specific parameters. To address these limitations, we introduced *Mixtures of SubExperts (MoSEs)*, a continual learning framework that integrates sparse sub-expert mixtures with task-specific routing. This design enables task isolation to prevent forgetting, while also allowing the adaptive reuse of previously learned parameters to facilitate transfer across tasks. Our evaluation on the TRACE benchmark demonstrates that MoSEs not only achieve superior knowledge retention compared to strong baselines, but also scale to new tasks with sublinear growth in capacity. Overall, MoSEs provide a promising direction for building scalable and memory-efficient LLMs capable of robust continual adaptation.

REFERENCES

Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Röttger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. Safety-tuned llamas: Lessons from improving the safety of large language models that follow instructions. *arXiv preprint arXiv:2309.07875*, 2023. 7

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1, 7

Arslan Chaudhry and et al. Tiny episodic memories in continual learning. In *ICML*, 2019. 1

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019a. 3

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and M Ranzato. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486*, 2019b. 3

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019. 7

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. Tydi qa: A benchmark for information-seeking question answering in ty pologically di verse languages. *Transactions of the Association for Computational Linguistics*, 8: 454–470, 2020. 7

Nan Du and et al. Glam: Efficient scaling of language models with mixture-of-experts. In *ICML*, 2022. 3

William Fedus and et al. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. In *ICLR*, 2022. 2, 3, 7

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020. 7

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, 2019. 2, 3

Edward J Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021a. 2, 3, 4

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021b. URL https://arxiv.org/abs/2106.09685. 7

Haeyong Kang, Rusty John Lloyd Mina, Sultan Rizky Hikmawan Madjid, Jaehong Yoon, Mark Hasegawa-Johnson, Sung Ju Hwang, and Chang D Yoo. Forget-free continual learning with winning subnetworks. In *International Conference on Machine Learning*, pp. 10734–10750. PMLR, 2022. 2, 3

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwińska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114 (13):3521–3526, 2017. 1, 2, 3

Dmitry Lepikhin and et al. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020. 2, 3, 7

Xian Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021. 2, 3

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*, 2023. 7

Yan-Shuo Liang and Wu-Jun Li. Loss decoupling for task-agnostic continual learning. *Advances in Neural Information Processing Systems*, 36, 2024. 3

Huiwei Lin, Baoquan Zhang, Shanshan Feng, Xutao Li, and Yunming Ye. Pcr: Proxy-based contrastive replay for online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24246–24255, 2023. 3

Lei Liu and Jimmy Xiangji Huang. Prompt learning to mitigate catastrophic forgetting in cross-lingual transfer for open-domain dialogue generation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, pp. 2287–2292, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394086. doi: 10.1145/3539618.3592043. URL https://doi.org/10.1145/3539618.3592043. 7

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 1

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989. 1

OpenAI. Gpt-4 technical report, 2023. URL https://cdn.openai.com/papers/gpt-4.pdf. 1

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 1, 2

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017. 3

Andrei A Rusu and et al. Progressive neural networks. In *arXiv preprint arXiv:1606.04671*, 2016. 3

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 3

Noam Shazeer et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017. 2, 3, 4, 7

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022. 7

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. Trace: A comprehensive benchmark for continual learning in large language models, 2023. URL https://arxiv.org/abs/2310.06762. 7

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022. 7

Mitchell Wortsman, Vivek Ramanujan, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. Supermasks in superposition. In *Advances in Neural Information Processing Systems*, 2020. 2, 3

Friedemann Zenke and et al. Continual learning through synaptic intelligence. In *ICML*, 2017. 2, 3

Hao Zhang and et al. Adaptive mixture of experts for continual learning. *arXiv preprint arXiv:2302.06665*, 2023. 2, 3

Table 6: (TaIL) Expert Performances of MoE across the TRACE benchmark datasets (0.5K). E# denotes the number of Experts, while T# denotes the number of Top-K selection among E#.

Dataset	C-STANCE	FOMC	MeetingBank	Py150	ScienceQA	NumGLUE-cm	NumGLUE-ds	20Minuten	Avg.	BWT	# Params	Train / Test
MoE, E2T1	0.40	0.12	0.19	**0.55**	0.49	0.26	0.39	0.41	35.1	-17.87 %	4.22M	20.18h / 1.25h
MoE, E2T2	0.43	**0.56**	0.21	0.54	0.56	0.22	0.46	0.40	**42.2**	**-11.10** %	**4.22M**	**20.18h / 1.25h**
MoE, E3T2	0.45	0.40	**0.22**	0.52	**0.58**	0.22	0.44	**0.41**	40.5	-11.31 %	4.49M	20.19h / 1.25h
MoE, E3T3	**0.48**	0.42	0.20	0.50	0.57	**0.32**	**0.48**	0.40	42.1	-11.49 %	4.49M	20.19h / 1.25h

A Appendix

A.1 Training Details

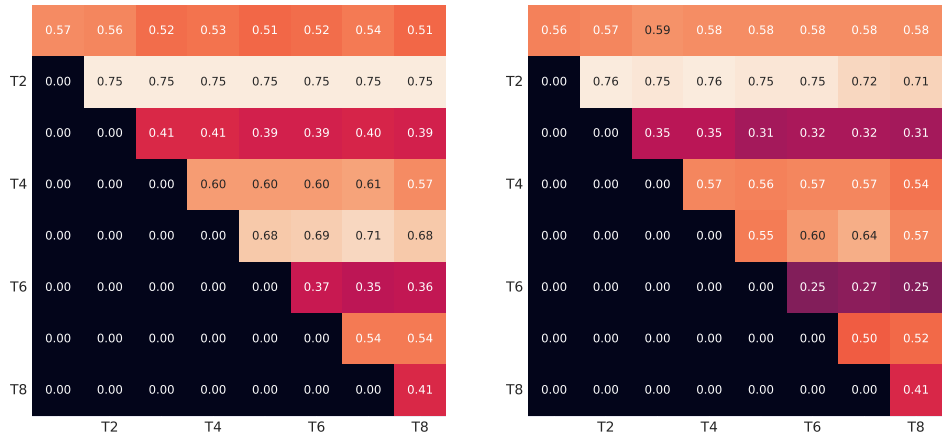
During training, all baselines without LoRA adapters were trained using 5,000 samples for 10 epochs with a constant learning rate of 1e-5. For baselines with LoRA adapters, we used the same number of samples and training epochs, but with a learning rate of 1e-4. Across all experiments—including those on C-STANCE, FOMC, MeetingBank, Py150, ScienceQA, NumGLUE-cm, NumGLUE-ds, and 20Minuten—we used a batch size of 128 and no weight decay. During evaluation, inference was performed with a temperature of 0.1. All training and testing were conducted using the DeepSpeed framework on a single machine equipped with 8×48GB NVIDIA RTX8000 GPUs. Full-parameter fine-tuning was used in all models. Benchmark evaluations were carried out using the OpenCompass toolkit with its default configuration.

A.2 Additional Analysis

Results of MoE Configurations on TRACE (0.5K). Table 6 reports the performance of various Mixture-of-Experts (MoE) configurations on the TRACE benchmark datasets. The configurations vary in the number of experts (E#) and the number of top-k experts selected per input (T#). Among the models, MoE, E2T2 achieves the highest average accuracy of 42.2%, closely followed by MoE, E3T3 at 42.1%. In contrast, MoE, E2T1 shows significantly lower performance at 35.1%, suggesting that activating too few experts underutilizes model capacity. Increasing the number of active experts per input improves both performance and stability. In terms of parameter count, the models with three experts (E3T2, E3T3) require slightly more parameters (4.49M) compared to their two-expert counterparts (E2T1, E2T2, each with 4.22M). However, all configurations maintain comparable training times (approximately 20.18 - 20.19 hours) and a uniform test time of 1.25 hours, thanks to the sparsity in expert selection. Additionally, backward transfer (BWT) metrics indicate that configurations with more activated experts (e.g., E3T3 at -11.49%) exhibit reduced forgetting compared to sparse models like E2T1 (-17.87% BWT). This demonstrates the benefit of wider expert activation for continual learning stability.

Analysis of Transfer Matrices under TIL vs. TaIL. Figure 5 compares the transfer matrices of MoSEs under Task-Incremental Learning (TIL) and standard settings on TRACE 0.5K, with a sparsity ratio $c = 30\%$. In the TIL configuration (fig. 5a), where task-IDs are provided during inference, the model achieves consistently high accuracy along the diagonal (e.g., 0.75 for T2), demonstrating strong task-specific specialization and minimal confusion between tasks. This setting reduces task ambiguity and allows the routing mechanism to activate more relevant sub-experts. In contrast, the TaIL setting (fig. 5b), where task identity is unknown at inference time, leads to more diffuse and lower diagonal values for later tasks (e.g., T6 and T8), indicating degraded performance due to increased interference. The non-zero off-diagonal entries suggest partial task generalization, but also highlight the challenge of identifying task-relevant subnetworks without explicit supervision. Nevertheless, even without access to task-IDs, MoSEs exhibit performance close to that achieved under TIL, suggesting that the model learns sufficiently disentangled and independent representations for each input sample. This property provides evidence that MoSEs can implicitly infer task-relevant structures, enabling robust adaptation and retention. Overall, the TIL setting enhances forward transfer and suppresses interference, as evidenced by clearer task separation in the matrix, while the TaIL setting underscores the effectiveness of MoSEs in learning modular, task-independent representations.

Performances of MoSEs on TRACE (5K). Table 7 reports the TIL performances of various baselines and MoSEs on the TRACE benchmark with 5K training samples. Notably, MoSEs with a sparsity ratio of $c = 29.0\%$ achieve strong overall performance, recording an average score



(a) MoSEs (TIL), $c = 30.0\%$ (b) MoSEs (TaIL), $c = 30.0\%$

Figure 5: **Comparisions of Transfer Matrixes** on the TRACE (0.5K). TIL denotes that task ID is given in inference.

Table 7: (TIL) Performances of MoSEs (W/O[0-1], E2T2 ($r = 2, \alpha = 8$)) across the TRACE benchmark datasets (5K).

Dataset	C-STANCE	FOMC	MeetingBank	Py150	ScienceQA	NumGLUE-cm	NumGLUE-ds	20Minuten	Avg.	BWT	# Params	Train / Test
SeqFT-LoRA	0.32	0.56	0.13	0.36	0.25	0.23	0.27	0.40	31.5	-29.67 %	4.19M	8.60d / 6.11h
O-LoRA	0.43	0.27	0.26	0.30	0.67	0.30	0.14	0.41	34.6	-23.00 %	4.19M	9.29d / 6.11h
MoE	0.43	0.45	0.21	0.40	0.69	0.24	0.06	0.40	36.0	-22.34 %	4.22M	8.90d / 6.11h
MoSEs, c=29.0%	**0.47**	0.63	0.45	**0.57**	0.76	**0.49**	0.51	0.40	53.5	-0.90 %	**3.82M**	8.32d / 6.04h
MoSEs, c=30.0%	0.45	**0.69**	**0.45**	0.56	**0.79**	0.48	**0.51**	**0.40**	**54.1**	**+0.36 %**	3.83M	**8.32d / 6.04h**
MoSEs, c=30.0% (TIL)	0.50	**0.71**	**0.48**	0.56	**0.80**	0.49	**0.53**	**0.41**	**56.0**	**-0.81 %**	3.83M	**8.32d / 6.04h**

of 53.5 and maintaining competitive backward transfer (-0.9%), which highlights the efficiency of using fewer active experts while preserving knowledge. Although MoSEs with $c = 30.0\%$ slightly outperform in terms of average accuracy (54.1) and positive BWT, the results at $c = 29.0\%$ demonstrate a favorable trade-off between parameter efficiency and robustness. Importantly, the overall performance trends remain consistent with those observed in the 0.5K setting, indicating that MoSEs scale effectively with larger datasets without altering their relative advantages across sparsity levels.

Peudo Codes The overall process of the MoSEs during training and testing is described as Algorithm 1 and Algorithm 2. At the test time stated in Algorithm 2, MoSEs utilize a pre-trained transformer backbone together with task-specific prompts and subnetworks. Given an input example $\boldsymbol{x}$, the model first generates a query feature $q(\boldsymbol{x})$, which is matched against the set of task keys $\{\boldsymbol{k}_t\}$ to identify the most relevant task index t_x. The corresponding prompt e_{t_x} and sparse expert subnetwork m_{t_x} are then selected. These components are combined to form the prompted MoSEs function $f_{\boldsymbol{e}_{t_x}, \tilde{\boldsymbol{\theta}} \odot \boldsymbol{m}_{t_x}}$, where the prompt $\boldsymbol{e}_{t_x}$ is attached to the designated MSA layers. Finally, the model produces a prediction $f_{\boldsymbol{e}_{t_x}, \tilde{\boldsymbol{\theta}} \odot \boldsymbol{m}_{t_x}}(\boldsymbol{x})$, enabling task-aware inference without explicit task-ID supervision.

Algorithm 2 MoSEs at test time

Given components: Pre-trained transformer-based backbone $f_{\boldsymbol{\theta}}$, trained
  $\boldsymbol{K} = \{\boldsymbol{k}_t\}_{t=1}^{\mathcal{T}}, \boldsymbol{M} = \{\boldsymbol{m}_t\}_{t=1}^{\mathcal{T}}, start_e, end_e$, prompting function $f_{\boldsymbol{\theta} \odot \boldsymbol{m}}$
Input: test example $\boldsymbol{x}$ from mini-batch $\boldsymbol{b}$
  Generate query feature $q(\boldsymbol{x})$
  Matching for the index of Prompt via $t_{\boldsymbol{x}} = \text{argmin}_t \gamma(q(\boldsymbol{x}), \boldsymbol{k}_t)$
Select $\boldsymbol{e}_{t_{\boldsymbol{x}}}$ and $\boldsymbol{m}_{t_{\boldsymbol{x}}}$
Generate $f_{\boldsymbol{e}_{t_{\boldsymbol{x}}}, \tilde{\boldsymbol{\theta}} \odot \boldsymbol{m}_{t_{\boldsymbol{x}}}}$:
  Attaching $\boldsymbol{e}_{t_{\boldsymbol{x}}}$ to $start_e$-th to end_e-th MSA layers respectively with $f_{\tilde{\boldsymbol{\theta}} \odot \boldsymbol{m}}$.
Prediction: $f_{\boldsymbol{e}_{t_{\boldsymbol{x}}}, \tilde{\boldsymbol{\theta}} \odot \boldsymbol{m}_{t_{\boldsymbol{x}}}}(\boldsymbol{x})$

Public Source Code All official source codes will be available soon.