CAMEx: Curvature-aware Merging of Experts

Anonymous authors
Paper under double-blind review

Abstract

Existing methods for merging experts during model training and fine-tuning predominantly rely on Euclidean geometry, which assumes a flat parameter space. This assumption can limit the model's generalization ability, especially during the pre-training phase, where the parameter manifold might exhibit more complex curvature. Curvature-aware merging methods typically require additional information and computational resources to approximate the Fisher Information Matrix, adding memory overhead. In this paper, we introduce CAMEx (**C**urvature-**A**ware **M**erging of **Ex**perts), a novel expert merging protocol that incorporates natural gradients to account for the non-Euclidean curvature of the parameter manifold. By leveraging natural gradients, CAMEx adapts more effectively to the structure of the parameter space, improving alignment between model updates and the manifold's geometry. This approach enhances both pre-training and fine-tuning, resulting in better optimization trajectories and improved generalization without the substantial memory overhead typically associated with curvature-aware methods. Our contributions are threefold: (1) CAMEx significantly outperforms traditional Euclidean-based expert merging techniques across various natural language processing tasks, leading to enhanced performance during pre-training and fine-tuning; (2) we introduce a dynamic merging architecture that optimizes resource utilization, achieving high performance while reducing computational costs, facilitating efficient scaling of large language models; and (3) we provide both theoretical and empirical evidence to demonstrate the efficiency of our proposed method.

1 Introduction

Sparse Mixture of Experts (SMoE) (Jacobs et al., 1991; Shazeer et al., 2017) is currently a core component for constructing foundation and large language models (LLMs), whose parameters count can rise up to billions and trillions (Devlin et al., 2019; Yang et al., 2019; Liu et al., 2019; Raffel et al., 2020; Fedus et al., 2022; Wei et al., 2022). Nevertheless, Hoffmann et al. (2024); Kaplan et al. (2020), recognized a scaling law that underpins the LLM's evolution, which is larger models require exponentially more computational resources and data to continue improving, and without sufficient scaling in all dimensions, performance gains may plateau. Thus, identifying and implementing efficient methodologies for the sustainable scaling of LLMs is imperative. SMoE addresses this challenge by sparsely activating parameters of large models, which can boost model performance with only minor losses in computational efficiency. The methodology is integrated chiefly into feedforward layers of transformers, processing tokens by selectively activating a small number of experts and hence trimming down the computing memory and FLOPS (Fedus et al., 2022; Lepikhin et al., 2021).

Since its debut in Shazeer et al. (2017), SMoE has gone through numerous explorations and advancements in routing mechanism development and expert architecture design. Dai et al. (2022) proposes a two-phase training strategy for stabilizing the gate function so that the expert's selection of one token does not fluctuate between different inference times. Zhou et al. (2022) changes the perspective of the router to experts with experts choice routing, ensuring a balancing load between experts . Chi et al. (2022) and Chen et al. (2023) address the concern of representation collapse in SMoE by proposing cosine scoring and a fixed random initialized router, respectively. Some other works view the routing mechanism as a reinforcement learning or optimization transport problem. In terms of expert design orientation, Rajbhandari et al. (2022) and Dai et al. (2024) introduce the

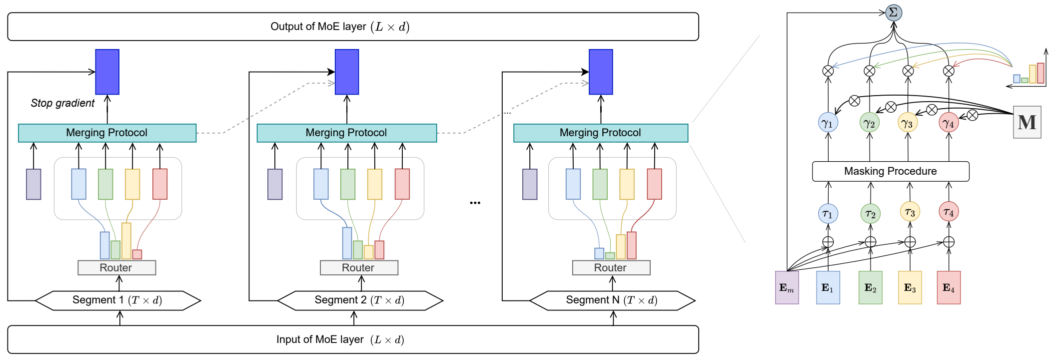

Figure 1: Overview of CAMEx for a causal language modeling SMoE. The experts are merged through the router scores and the curvature-matrix $\mathbf{M}$. During the merging protocol, we can generate the masks for the domain-vectors, denoted as γ_i, such as Ties or Dare. We follow the causal segmenting pipeline from (Zhong et al., 2024) to achieve both memory efficiency and causal information constraints. Note that *stop gradient* operator is applied for the first segment router scores.

concept of shared experts wherein each token is processed by a fixed expert and another selected through gating, achieving two experts engagement per layer without increasing the communication cost beyond that of top-1 gating. Muqeeth et al. (2024) proposes to merge experts by taking the weighted mean of the expert's parameters with respect to router scores. This methodology is then extended in He et al. (2023), Zhong et al. (2024), and Li et al. (2024) for causal language modeling pretraining and fine-tuning tasks.

Among existing rigorous research on SMoE, our work focuses on the experts merging lines of research. Specifically, we systemically integrate natural gradient into task-specific merging protocol for SMoE. To the best of our knowledge, the current merging protocol applied for SMoE still deems the parameter space of the expert's parameters as Euclidean ones. Nevertheless, it has been shown that the space of neural network parameters brings the characteristic of the Riemannian manifold (Amari, 1998). Therefore, it is natural for us to make an effort in such a direction for merging experts. Although some existing works on merging models have already leveraged the Fisher Information Matrix (Matena & Raffel, 2022; Jin et al., 2023), we find that they require large computational space and complicated steps to perform well. In contrast, our merging protocol is simple and straightforward to implement while still taking into account the curvature of the parameters manifold. We discover the superior performance of curvature-aware merging in our method compared to the regular merging procedure applied to SMoE. Our main contributions are three-fold:

1. We introduce a novel rapid and efficient merging technique named Curvature- Aware Merging of Experts (CAMEx) for SMoE that includes information about the curvature of the expert's parameters manifold.
2. We propose a new architecture based on CAMEx, which dynamicalizes the merging protocol along with parameters reduction. Our empirical experiments prove the dominant performance of this architecture on pre-training tasks.
3. We theoretically prove that our CAMEx obtains better alignment between experts and the training task domain.

We empirically demonstrate that 1) our proposed merging method can add in rapidness of convergence speed for pre-training and 2) when combined with other merging protocols, it can boost the model's performance on a variety of practical tasks, including language modeling, text classification, question answering, and image classification.

2 Curvature-aware Merging of Experts

This section aims to give an overview of model merging methods and their integration into SMoE architecture. We then introduce our curvature-aware merging protocol stamping from the natural gradient. Finally, we perform an theoretical analysis to support our proposal.

Table 1: Notations and Definitions.

Symbol	Description	Symbol	Description
T	Number of tokens	k	Number of selected experts
N	Total number of experts	$\mathbf{E}_i \in \mathbb{R}^{d \times h}$	Weights for the ith expert
$\mathbf{h} \in \mathbb{R}^{T \times d}$	Input tokens or hidden states	$\mathbf{G}(\cdot, \cdot) \in \mathbb{R}^{T \times N}$	Gating function output
$\mathcal{S}_t$	Set of top-k experts for token $\mathbf{h}_t$	$\alpha \in [0, 1]$	Rescalling factor

2.1 Background: Expert Merging in Sparse Mixture of Experts

It is convenient to recall the concept of SMoE and a few well-known experts merging methods for SMoE. From this point to the rest of our paper, let us use the notations summarized in Table 1.

Sparse Mixture of Experts. A SMoE layer processes the tokens series as follows:

$$\begin{cases} \mathbf{y}_t = \sum_{i \in \mathcal{S}_t} \mathbf{G}(t, i) \cdot \mathbf{E}_i \mathbf{h}_t \\ \mathbf{G}(t, \cdot) = \text{softmax}(\mathbf{W}_g \mathbf{h}_t) \\ \mathcal{S}_t = \text{top-k}(\mathbf{G}(t, \cdot)) \end{cases} \tag{1}$$

SMEAR. Muqeeth et al. (2024) introduces the ensemble of expert parameters through weighted average computation with the factors are the router scores.

Task-Specific merging in SMoE. Our work will follow the scheme of task-specific merging (Ilharco et al., 2023). In such a setting, we assume the existence of N pre-trained models parameterized by θ_i each was pre-trained on a different task. We then define the task-vector for each pre-trained model through the merged model θ_m as $\tau_i = \theta_i - \theta_m$. The merging protocol will be performed by Eqn. Merg. Under the context of SMoE, each expert learns to handle a particular subset of the input space or specializes in a specific type of feature or pattern (Jacobs et al., 1991; Dai et al., 2024). We believe it is more suitable to reference this technique as **domain-specific merging**. We, therefore, will rename the tensors $\tau_i = \mathbf{E}_i - \mathbf{E}_m$ as *domain-vector*. Additionally, to take the router information into account, we will define the formulation for domain-specific merging in a SMoE layer as follows:

$$\hat{\mathbf{E}}_m = \mathbf{E}_m + \alpha \sum_{i=1}^{N-1} s_i \tau_i \tag{2}$$

where s_i denotes the score of the router for the ith expert. We want to note that with $0 < \alpha < 1$, domain-specific merging aligns with soft merging.

2.2 Background: Other Model Merging Methods

In this section, we discuss other recent and widely-adopted model merging methods outside the context of SMoE that we will combine with our curvature-aware merging method in our experiments in Section 3.

TIES merging. Yadav et al. (2023) improves upon task arithmetic by removing interference between the task vectors. Specifically, TIES zeros out entries in a given task vector with low magnitude and resolves sign conflicts across different task vectors.

DARE merging. Different form TIES, DARE merging randomly zeros out the neurons like a Dropout layer (Yu et al., 2024).

Fisher merging. Existing work on Fisher merging suffers from computational complexity since computing and inverting the Fisher Information Matrix, especially for large neural networks, is often intractable. Even when using approximations like diagonal or block-diagonal Fisher matrices, these methods can still be computationally expensive and challenging to apply at scale. Furthermore, the accuracy of Fisher approximations, such as diagonal or block-diagonal, can be problematic (Matena & Raffel, 2022).

2.3 GRADIENT INTERPRETATION OF MODELS MERGING

We want to emphasize the alignments between the paradigm of gradient descent and model merging. For this, we denote $\theta \in \mathbb{R}^N$, $\mathcal{L}(\theta)$, and η as the model's parameters, the empirical loss function, and the learning rate, respectively. During the training process of a deep learning model, the parameters are updated following the gradient descent formula:

$$\theta_{n+1} = \theta_n + \eta(-\nabla\mathcal{L}(\theta_n)) \tag{GD}$$

In the aspect of deep models merging, we also have an update rule in a similar manner, which is

$$\hat{\theta}^m = \theta^m + \alpha \sum_{i=1}^{n} \underbrace{(\theta^i - \theta^m)}_{\substack{\text{gradient-like} \\ \text{update direction}}} \tag{Merg}$$

where θ^m denotes the merged model's parameters, and θ^i denotes the parameters of the ith expert. Here, we interpret θ^i as the optimal parameters of the model for a specific task or domain, and then the update rule gives us a direction toward optimizing for all tasks.

However, it has been pointed out by Amari (1998) that the parameter space structure of deep learning models has Riemannian characteristics. Therefore, a more *natural* gradient updating scheme was proposed,

$$\theta_{n+1} = \theta_n + \eta \underbrace{G(\theta_n)(-\nabla\mathcal{L}(\theta_n))}_{\text{natural gradient}} \tag{NGD}$$

In this formula, $G(\theta_n) \in \mathbb{R}^{N\times N}$ denotes the *Riemannian metric tensor* (Amari, 1998; Amari & Douglas, 1998), which characterizes the intrinsic curvature of a particular manifold in N-dimensional space (Martens, 2020) or sometimes, the inversed Fisher Information Matrix. The same ideology was introduced for merging large language models in Fisher merging (Matena & Raffel, 2022) and Regmean (Jin et al., 2023). However, both methods suffer from the bottleneck in the computation cost of approximating the Fisher Information. Moreover, these methods are challenging to apply in sparse layers of SMoE since they would introduce huge latency, FLOPS, and memory for computing and storing matrices whose number of entries is proportional to a number of expert parameters.

2.4 EXPERTS MERGING WITH CURVATURE-AWARE

In this section, for the sake of conciseness, we focus on the language modeling task; a similar methodology can be applied to other tasks, such as classification. We introduce an efficient way to merge experts within SMoE layers, based on the causal segmenting approach proposed by (Zhong et al., 2024). The goal of the causal segment routing strategy is to enhance the efficiency of expert merging operations while maintaining the autoregressive nature of language models. More details about this algorithm can be found in Appendix C.1 and Algorithm 1. We then perform the following merging protocols:

$$\hat{\mathbf{E}}^l_m = \mathbf{E}^l_m + \alpha \sum_{i=1}^{N-1} \mathbf{M}_i \cdot (s^l_i * \tau^l_i) \tag{CA-Merg}$$

where $\mathbf{M}_I \in \mathbb{R}^{d_{in}d_{out}\times d_{in}d_{out}}$ denote the curvature matrix which performs matrix product with the gradient-like component. The curvature of the parameters manifold will be learned through these tensors while optimizing the empirical loss. This approach has also proven its effectiveness in meta-learning for few-shot classification (Park & Oliva, 2019). We further explore the computing efficiency of merging experts by proposing a novel dynamic merging formula

$$\begin{cases} \mathbf{E}^{l+1}_m &= \mathbf{E}^l_m + \dfrac{\alpha}{N-1} \displaystyle\sum_{i=1}^{N-1} \mathbf{M}_i \cdot \tau^l_i \\ \hat{\mathbf{E}}^{l+1}_m &= \mathbf{E}^{l+1}_m + \alpha \displaystyle\sum_{i=1}^{N-1} \mathbf{M}_i \cdot (s^{l+1}_i * \tau^{l+1}_i) \end{cases} \tag{Dynamic-Merg}$$

The architecture corresponding to this recurrent representation can be found in Figure 2. The architecture contains a global expert that traverses through the SMoE layers by the updating rule in

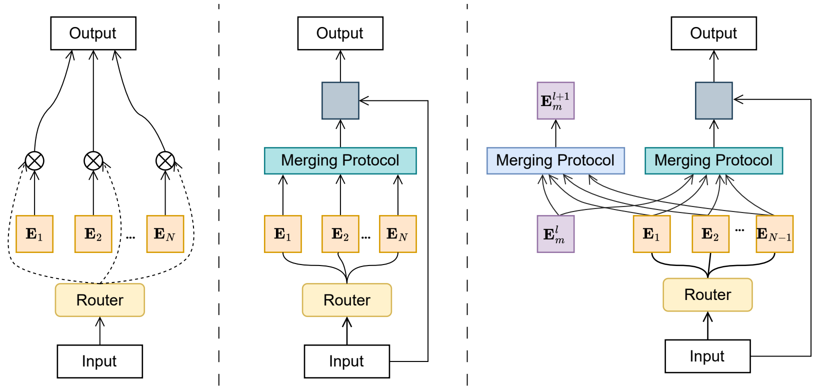

Figure 2: **Overall architecture of different SMoE layers**. The figure presents the vanilla SMoE layer on the left, the merging expert layer in the middle, and our proposed dynamic merging SMoE layer on the right. Our architecture reduces the number of parameters compared to the other two, while maintaining the same number of activated neurons per layer. Importantly, despite the dynamic merging mechanism, our architecture preserves the same number of experts at each layer as the other SMoE architectures, ensuring comparable model capacity, i.e., the number of activated parameters per layer.

Eqn. Dynamic-Merg. Not only will this allow a notable reduction in model size and GFLOPS, but it also ensures the number of experts in each SMoE is the same as in the full-expert setting, where each layer has the same number of experts. We refer to Appendix F for a step-by-step walkthrough of key equations in CAMEx.

2.5 Efficency

Parameter efficient approximation of curvature matrix. Storing and computing a curvature matrix requires a whopping memory and time complexity of $O(n^4)$ and $O(n^4)$, respectively. This is infeasible even for a simple SMoE layer, as one layer can contain many experts. To mitigate this problem, we follow Martens & Grosse (2015) and approximate the curvature matrix using the Kronecker product. It has been proven by Hameed et al. (2022) that we can approximate arbitrary matrix using a finite sum of Kronecker products. For a curvature matrix $\mathbf{M}_i \in \mathbb{R}^{d_{in}d_{out} \times d_{in}d_{out}}$, we present the rank-1 approximation as below:

$$\mathbf{M}_i \approx \mathbf{M}_i^{in} \otimes \mathbf{M}_i^{out} \tag{3}$$

with $\mathbf{M}_i^{in} \in \mathbb{R}^{d_{in} \times d_{in}}$ and $\mathbf{M}_i^{out} \in \mathbb{R}^{d_{out} \times d_{out}}$. Still, this form of approximation is too large to compute and store during training time, so we further decompose $\mathbf{M}_i^{in}$ and $\mathbf{M}_i^{out}$ using Kronecker product because of the efficient computation using tensor algebra. This form of approximation reduces the number of parameters added and only puts negligible memory and computational overhead to the training process at the cost of additional $O(n)$ memory complexity and $O(n^{2.5})$ computational complexity. Although this might limit the representative capacity of the curvature matrix, we empirically find that the performance of our method still surpasses other merging methods.

Efficient test-time inference with reparameterization. We focus on the case where $\alpha = 1$. To further optimize the computation of curvature-aware merging, we embed the curvature matrices into the domain-vectors using the following reparameterization trick:

$$\mathbf{E}_i' \leftarrow \mathbf{E}_m + \mathbf{M}_i \cdot \tau_i \tag{4}$$

In this case, the merging formula at test time becomes:

$$\hat{\mathbf{E}}_m = \mathbf{E}_m + \sum_{i=1}^{N-1} s_i \cdot (\mathbf{E}_i' - \mathbf{E}_m) = \mathbf{E}_m + \sum_{i=1}^{N-1} \mathbf{M}_i \cdot (s_i \cdot \tau_i)$$

Thus, during inference, we avoid storing the curvature matrices and recomputing their product with domain vectors, reducing the total FLOPs. This explains the computational efficiency seen in Section 3.

2.6 THEORETICAL ANALYSIS

In this section, we investigate how optimizing the curvature matrix $\mathbf{M}$ in Eqn. CA-Merg can improve the generalization of the expert merging process for downstream tasks. We first recall the formulation for domain-specific merging

$$\hat{\mathbf{E}}_m = \mathbf{E}_m + \alpha \sum_{j=1}^{N-1} s_j \cdot \tau_j \tag{5}$$

We begin by calculating the gradient of the downstream task loss function, denoted by $\mathcal{L}$, with respect to $\mathbf{M}_j$ at a specific lth layer as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{M}_j} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_m} \cdot \frac{\partial \hat{\mathbf{E}}_m}{\partial \mathbf{M}_j} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_m} \cdot (\alpha s_j * \tau_j) = \alpha s_j * \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_m} \cdot (\mathbf{E}_j - \mathbf{E}_m) \tag{6}$$

This is the outer product of the gradients of the task loss and the domain vectors. It is important to note the connection with the Fisher Information Matrix. For a downstream task, if we define the loss function as the negative log-likelihood, such as in a supervised classification task $\mathcal{L}(\theta) = \mathbb{E}_{(x,y)\sim p}[-\log_\theta p(y|x)]$, then the empirical Fisher Information Matrix can be defined as

$$\mathbf{F} = \mathbb{E}_{(x,y)\sim p}[\nabla_\theta \log_\theta p(y|x) \nabla_\theta \log_\theta p(y|x)^\top]$$

Next, we consider how the gradient of the curvature matrix can contribute to better performance. Given a time step t, the standard gradient descent from an initial point $\mathbf{M}_j$ with learning rate β yields the following update:

$$\mathbf{M}_j^{t+1} = \mathbf{M}_j^t - \beta * \frac{\partial \mathcal{L}}{\partial M_j^t} = \mathbf{M}_j^t - \alpha\beta * s_j^t * \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_m^t} \cdot (\mathbf{E}_j^t - \mathbf{E}_m^t) \tag{7}$$

We assume standard gradient descent for simplicity, but the argument extends to other advanced gradient algorithms, such as momentum and ADAM. We then apply $\mathbf{M}_j$ to the merging process in Eqn. CA-Merg and get

$$\hat{\mathbf{E}}_m = \underbrace{\mathbf{E}_m + \alpha \sum_{j=1}^{N-1} s_j^{t+1} * \mathbf{M}_j^t \cdot \tau_j^{t+1}}_{\text{domain-specific merging with curvature-aware}} - \alpha^2 \beta \sum_{j=1}^{N-1} s_j^t s_j^{t+1} * \left(\tau_j^{t^\top} \cdot \tau_j^{t+1} \right) \cdot \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_m^t} \tag{8}$$

The detail for the derivation can be found in Appendix E. We can see that the first term in Eqn. 8 is the classic domain-specific merging formula with the guidance of the learned curvature. Furthermore, the second term contains the direction from the task loss gradient and the inner product between domain-vectors from two consecutive gradient steps. If $\mathbf{M}_j = \mathbf{I}\ \forall j$, this term can be seen as an auxiliary signal from task loss of the previous update step guiding the merging direction. The term $s_j^t s_j^{t+1} \cdot \left(\tau_j^{t^\top} \tau_j^{t+1} \right)$ modeling the agreement of the merging direction between updating steps: if there are conflicts between current and the previous updating direction, then this signal will be alleviated, thus dampening the harm to the merging direction of the current step; otherwise if they show strong agreement, this amplifies the impact of the updating direction toward minimizing the task loss with respect to the previous step, thus accelerate the training process while implicitly helping current merging direction with additional *experience*.

On the other hand, we can rewrite Eqn. 8 as follows:

$$\hat{\mathbf{E}}_m = \mathbf{E}_m + \alpha \sum_{j=1}^{N-1} s_j^{t+1} * \mathbf{M}_j^t \cdot \tau_j^{t+1} - \alpha^2 \beta \sum_{j=1}^{N-1} s_j^t s_j^{t+1} * \underbrace{\left(\tau_j^{t^\top} \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_m^t} \right)}_{\text{gradient matching}} \cdot \tau_j^{t+1} \tag{9}$$

We now have the inner-product between the gradient of the task loss and the domain-vector. This can be interpreted as the matching between the update from the task loss gradient and the domain-specific direction. We then have the updated domain-specific direction for each expert whose weighting factors are calculated by the inner-product. Therefore, we are performing a soft nearest distance voting to find the experts that agree the most with the task loss and enhance the merged experts with the corresponding domain-vector.

3 EXPERIMENTAL RESULTS

Table 2: **Performance of T5-base variants on the fine-tuning tasks for GLUE**. All SMoE variants have 8 experts per layer. We follow Devlin et al. (2019) in conducting experiments on the GLUE benchmark. Our curvature-aware methods outperform all baselines across tasks, while maintaining the same number of parameters and FLOPs as the SMoE models.

Methods	Params	TFLOPs	SST-2	MRPC	CoLA	QQP	STSB	QNLI	RTE	MNLI
Vanilla	220M	4.65	93.34	89.70	58.06	88.76	89.06	92.34	74.36	86.36
SMoE	1.0B	4.65	94.26	90.87	56.78	88.69	89.44	92.07	70.75	86.38
Domain-Specific	1.0B	4.65	93.57	90.19	58.07	88.77	89.40	92.51	72.56	86.40
Ties	1.0B	4.65	93.92	91.44	58.54	86.47	88.58	91.87	75.54	86.39
Dare	1.0B	4.65	93.80	89.46	58.33	88.72	89.13	92.29	73.64	86.20
Domain-specific-CA	1.0B	4.65	93.80	91.16	58.57	**88.86**	89.47	92.60	74.72	86.44
Dare-CA	1.0B	4.65	94.49	91.15	58.56	88.76	**89.56**	**92.80**	**78.70**	86.34
Ties-CA	1.0B	4.65	**94.61**	**92.49**	**60.06**	88.83	89.54	91.89	75.81	**86.45**

We perform evaluations on **four** major tasks, including language modeling, text classification, question answering, and image classification. For language modeling, we use the Wikitext-2 and Wikitext-103 (Merity et al., 2016) benchmarks. For text classification, we employ a subset of the GLUE (Wang et al., 2019) benchmark, a collection of **eight** diverse tasks designed to test different aspects of language understanding. For question answering, we employ two famous benchmarks: SQuAD (Rajpurkar et al., 2016) and WikiQA (Yang et al., 2015). Finally, the ImageNet-1k (Deng et al., 2009) dataset is chosen for image classification evaluation.

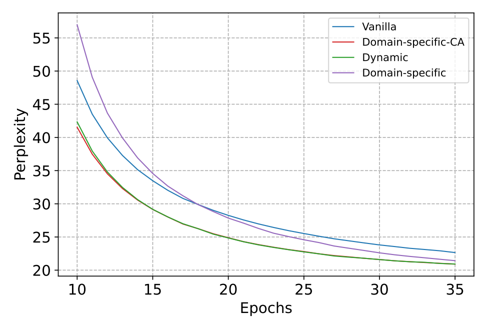

Figure 3: Perplexity of GPT2-small variants starting at the tenth epoch.

We choose GPT-2 (Radford et al., 2019) small and Swin-Transformer small (Liu et al., 2021) as our backbones for language modeling and image classification, respectively. Regarding GLUE and question-answering tasks, T5 base (Raffel et al., 2020) is chosen.
Our experimental results confirm that the proposed merging method accelerates pre-training convergence and, when combined with other merging protocols, enhances model performance across tasks and settings. All results are averaged over 5 runs with different random seeds. Detailed information on the datasets, models, training procedures, and hyperparameters is provided in Appendix B and Appendix D. For additional experiments on different routers and merging methods, we refer to Appendix H.1, and H.2.

3.1 Training and Evaluation Details

We fix the number of epochs for all models on each task. For each text-related task, we first undertake a comprehensive hyper-parameter search. This encompasses batch sizes from $\{8, 16, 32, 64\}$, learning rates from $\{3e{-}4, 1e{-}4, 3e{-}5, 1e{-}5\}$, to pinpoint the optimal fine-tuned models. Regarding image classification tasks, a batchsize of 96 for chosen for all models. In addition, we choose AdamW (Loshchilov & Hutter, 2019) as the default optimizer and conduct all experiments on NVIDIA A100 GPUs. We compare our proposal to three merging baselines, including domain-specific, Ties, and Dare merging. There exists prior works on merging methods with the aid of the Fisher Information Matrix, such as Matena & Raffel (2022), which rely on access to a validation set used to compute the Fisher matrix or fine-tune hyperparameters. To eliminate the need for a validation set, Jin et al. (2023) proposes storing and transmitting inner product matrices derived from the training data for each task, which are of the same size as the original model. However, this approach becomes costly for large models, as storage and transmission demands increase linearly with model size and the number of tasks as well as the number of experts. Therefore, we choose baselines that are needless of extra information and computational cost to perform comparisons. More details about theoretical comparison between CAMEx and Fisher-based merging methods can be found in Appendix A. We want to note that our merging protocol can be easily integrated into other works such as merge then compress protocol Li et al. (2024).

3.2 RESULTS

In the following tables, the row with our method's results is highlighted in *grey*. Results with the best and second best performance are written in **bold** and underline, respectively. In addition, methods with the postfix "-CA" denote the curvature-aware version of the corresponding baseline.

In Table 2, the results demonstrate that CA-augmented models consistently outperform their non-CA counterparts. Ties-CA achieves the highest scores on SST-2 (94.61), MRPC (92.49), CoLA (60.06), and MNLI (86.45), showing considerable improvements over both the vanilla and standard Ties models. Similarly, Dare-CA performs best on RTE (78.70), surpassing Dare (73.64), indicating that CA improves performance on smaller datasets and tasks with higher variability. Furthermore, Domain-specific-CA exceeds the non-CA version on QNLI and MNLI, demonstrating the broader applicability of curvature-aware methods. We provided a significant t-test at Appendix G.

Table 3: Performance of GPT-2 small variants for the pre-training task on Wikitext-103.

Methods	Perplexity↓	Params (M)	GFLOPS ↓
Vanilla	23.03	125	292.5
SMoE	22.42	522	292.5
Domain-specific	21.64	522	292.5
Domain-specific-CA	**21.50**	522	292.5
Dynamic	<u>21.55</u>	**470**	292.5

In Table 3, the Domain-specific-CA and Dynamic outperform the Vanilla, SMoE, and Domain-specific baselines, with lower perplexity values. Domain-specific-CA achieves the lowest perplexity score of 21.50, showcasing superior performance in language modeling tasks when compared to all other methods. The Dynamic architecture follows closely with a perplexity of 21.55 while also reducing the parameter count by 9%, compared to the other methods. This highlights the Dynamic architecture's efficiency in maintaining strong performance with fewer parameters, making it ideal for resource-constrained environments. Moreover, the Dynamic architecture is competitive with Domain-specific-CA and outperforms the rest in terms of convergence speed, which is shown in Figure 3.

Table 4: Performance of GPT-2 small variants for the supervised fine-tuning task on Wikitext-2

Methods	Perplexity↓	Params (M)	GFLOPS ↓
Vanilla	21.84	125	292.5
SMoE	21.60	522	292.5
Domain-specific	21.56	522	292.5
Ties	21.45	522	292.5
Dare	21.60	522	292.5
Domain-specific-CA	**21.06**	522	292.5
Dare-CA	21.42	522	292.5
Ties-CA	<u>21.11</u>	522	292.5

In Table 4 the Vanilla model reach a perplexity of 21.84. Despite increasing parameters, SMoE only slightly improves to 21.60. Domain-specific, Ties, and Dare methods show small gains, with Ties reaching 21.45. However, curvature-aware (CA) methods outperform all others. Domain-specific-CA achieves the best perplexity at 21.06, followed by Ties-CA (21.11) and Dare-CA (21.42), each significantly improving over their non-CA counterparts. All models beyond Vanilla share the same computational cost, indicating that CA methods enhance performance without added complexity. Domain-specific-CA stands out, demonstrating the clear advantage of curvature-aware optimization.

Table 5: Performance of T5-base variants on question answering tasks.

Methods	Params	TFLOPs	SQuAD Em/F1	WikiQA Accuracy
Vanilla	222M	2.86	81.01/88.14	96.06
SMoE	1.0B	2.86	81.25/88.50	96.04
Domain-specific	1.0B	2.86	80.21/87.44	95.32
Ties	1.0B	2.86	80.76/88.11	95.87
Dare	1.0B	2.86	80.88/88.03	96.01
Domain-specific-CA	1.0B	2.86	80.44/87.69	95.72
Ties-CA	1.0B	2.86	<u>81.52</u>/**88.60**	**96.55**
Dare-CA	1.0B	2.86	**81.76/88.60**	<u>96.23</u>

In Table 5, the baseline models, including Vanilla, SMoE, and non-CA versions of Ties and Dare, achieve solid results but show diminishing improvements as model complexity increases. In contrast, our curvature-aware methods significantly outperform their counterparts. For instance, on the SQuAD dataset, Dare-CA achieves the highest Exact Match (EM) score of 81.76% and an F1 score of 88.60%, surpassing all other methods. Similarly, on WikiQA, Ties-CA attains the highest accuracy of 96.55%, with Dare-CA closely following at 96.23%.

In Table 6, while Vanilla and SMoE exhibit solid accuracy scores, they are surpassed by the curvature-aware (CA) enhanced versions of the models. Notably, Ties-CA delivers the best top-1 accuracy at 83.38% and the highest top-5 accuracy at 96.96%, slightly edging out Dare-CA, which achieves 83.38% and 96.94%, respectively.

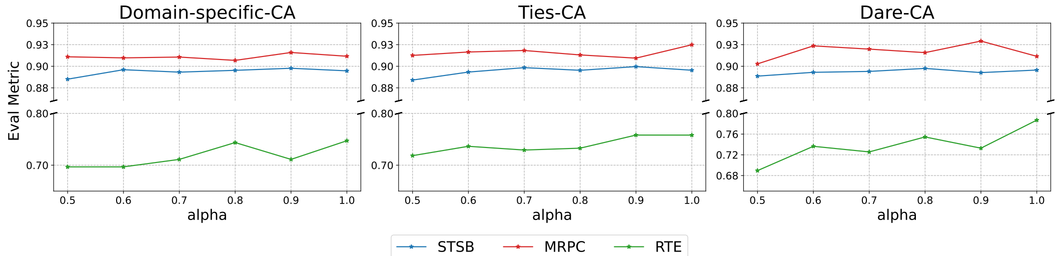

Figure 4: Impact of the α parameter on Curvature-Aware method performance across NLP tasks. We observe that the scaling factors that are within the range $[0.8, 1]$ consistently improve model's performance.

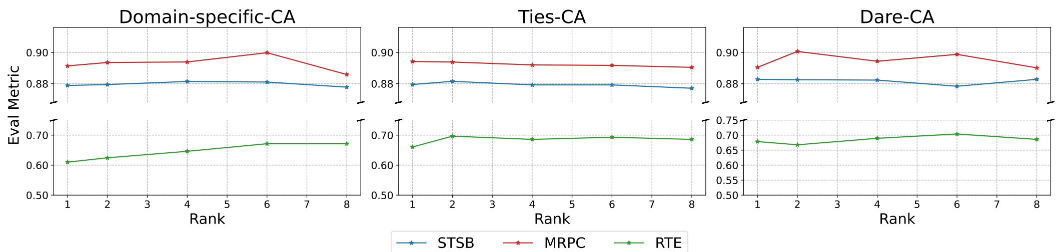

Figure 5: Impact of the Kronecker rank of curvature matrix on model's performance. We observe that as the rank increases the performance drops and then saturates. However, we would like to note that this curve might change depending on the downstream tasks and the merging protocol.

4 Ablation

Impact of the scaling factor. The plot in Figure 4 illustrates the impact of the α parameter on the performance of three curvature-aware (CA) model variants Domain-specific-CA, Ties-CA, and Dare-CA across three natural language processing tasks: STSB, MRPC, and RTE. The α parameter ranges from 0.5 to 1.0. The overall trend suggests that increasing α leads to better generalization, particularly for complex tasks such as RTE, where sentence-level entailment and similarity benefit from stronger curvature-aware representations. Moreover, across all tasks, the model reaches its peak performance when α is inside the range $[0.8, 1]$. This observation aligns with that indicated by Yadav et al. (2023). For a more comprehensive analysis on the impact of α and number of experts, we direct the readers to Appendix H.4.1 and H.4.2.

Table 6: Comparison of Accuracy for Swin-Transformer small variants on ImageNet-1k.

Methods	Params (M)	GFLOPs	Acc@1	Acc@5
Vanilla	50	6.75	83.14	96.90
SMoE	157	6.75	83.15	96.71
Domain-specific	157	6.75	83.15	96.91
Ties	157	6.75	83.28	96.93
Dare	157	6.75	83.13	96.88
Domain-specific-CA	157	6.75	83.29	96.95
Ties-CA	157	6.75	**83.38**	**96.96**
Dare-CA	157	6.75	**83.38**	96.94

Improved performance with higher Kronecker rank. Across all three tasks (STSB, MRPC, and RTE), the evaluation metrics tend to improve as the rank increases from 1 to 8. This indicates that higher-ranked models generally perform better, suggesting a positive correlation between rank and task performance. Notably, the Domain-specific-CA model consistently achieves high performance across all tasks, especially in STSB, where metrics approach 0.90. Although MRPC and RTE show slightly lower metrics, ranging from 0.50 to 0.75, there is a clear improvement in performance as rank increases, particularly in the lower-to-mid ranks. However, we

Table 7: Comparison for Swin-Transformer small variants on corrupted ImageNet.

Methods	ImageNet-O	ImageNet-A	ImageNet-R
Vanilla	45.88	23.68/53.10	37.34/52.34
SMoE	43.34	23.72/53.15	38.02/55.17
Ours	**50.69**	**25.45/54.24**	**38.37/55.42**

observed a decline in performance for Ties-CA and Dare-CA as the rank increases. We hypothesize that this is due to the masking mechanism employed by these methods, which may interfere with the learning process of the curvature matrices.

Robustness against noise. Table 7 demonstrates that curvature-aware models offer superior performance on corrupted ImageNet datasets compared to both Vanilla and SMoE variants. Among the models, our best configuration (Ties-CA) stands out as the best performer, showcasing robustness to corruptions across all datasets. These results suggest that incorporating curvature-awareness can substantially improve model robustness in challenging conditions.

5 Related Work

Sparse Mixture-of-Experts (SMoE). As the demand for model scaling grows increasingly widespread, there is a pressing inquiry into efficient ways to optimize computing costs while minimizing the impact on model performance. To address this need, Sparse Mixture of Experts (SMoE) has emerged and undergone extensive research and exploration (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). Starting with Shazeer et al. (2017), the integration of SMoE into transformer architectures followed shortly after with the works of Lepikhin et al. (2021) and Fedus et al. (2022). The principle of SMoE is based on a simple concept: scaling the horizontal dimension of models (i.e., the number of feedforward blocks) rather than the vertical dimension (i.e., the number of stacked layers). This allows the model to selectively activate units or parameters based on the input tokens, thereby optimizing resource usage while maintaining performance.

SMoE Efficiency Bottlenecks and Emerging Solutions. While it remains controversial whether to use Top-1 or Top-K routing, some research has highlighted the potential performance gains from increasing the number of activated experts (Shazeer et al., 2017; Chen et al., 2023). Other studies have found redundancies among experts in MoE layers (Li et al., 2024; Lu et al., 2024a). Additionally, some work has proposed using low-rank experts (Wu et al., 2024b; Liu et al., 2024; Wu et al., 2024a) inspired by LoRA (Hu et al., 2022). Despite the varying research directions, these studies consistently show that training a robust SMoE requires substantial computational and memory resources. This has motivated researchers such as Li et al. (2024), He et al. (2023), and Zhong et al. (2024) to merge experts within each MoE layer, reducing the number of experts to a single one and significantly improving training and inference efficiency.

Model Merging with curvature-aware. Though numerous methods for merging models have been introduced and developed (Yadav et al., 2023; Cai et al., 2023; Ilharco et al., 2022; Matena & Raffel, 2022; Jin et al., 2022; Don-Yehiya et al., 2022; Rame et al., 2023; Lu et al., 2024b), most of these works consider merging protocols in the Euclidean parameter space. However, it has been noted that the space of deep neural network models is a Riemannian one (Amari, 1998). Matena & Raffel (2022) and Jin et al. (2022) were the first to fuse model weights while accounting for the Fisher Information. Despite their promising results, these methods require massive computation to approximate the inversion of the Fisher matrix. Moreover, the Fisher matrix has a size proportional to the dimension of the model parameters, which significantly increases memory usage. Consequently, these methods are challenging for directly integrating into SMoE layers to fuse expert weights.

6 Limitation and conclusion

In this work, we introduced CAMEx, a curvature-aware approach to expert merging in Mixture of Experts architectures. By leveraging natural gradients to account for the parameter manifold's curvature, CAMEx enhances model performance and reduces computational costs during both pre-training and fine-tuning, outperforming traditional Euclidean-based methods. Additionally, our dynamic merging architecture optimizes resource usage by incorporating a global expert across layers, thus minimizing model size without sacrificing accuracy. Despite the overall improvements, a minor limitation is that curvature-aware merging demonstrates reduced compatibility with Ties and Dare merging at higher Kronecker ranks. Future work could dive deeper into this phenomenon and extend CAMEx to other expert merging methods and explore curvature-aware approaches in broader neural network models to further enhance our dynamic architecture. This research lays the groundwork for developing more efficient and scalable models in large-scale machine learning.

Reproducibility Statement: Source codes for our experiments are provided in the supplementary materials of the paper. The details of our experimental settings and computational infrastructure are given in Section 3 and the Appendix D. All datasets that we used in the paper are published, and they are easy to find in the Internet.

Ethics Statement: Given the nature of the work, we do not foresee any negative societal and ethical impacts of our work.

REFERENCES

S. Amari and S.C. Douglas. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, volume 2, pp. 1213–1216 vol.2, 1998. doi: 10.1109/ICASSP.1998.675489.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998. doi: 10.1162/089976698300017746.

Ruisi Cai, Zhenyu Zhang, and Zhangyang Wang. Robust weight signatures: Gaining robustness as easy as patching weights? *arXiv preprint arXiv:2302.12480*, 2023.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001.

Tianlong Chen, Zhenyu Zhang, AJAY KUMAR JAISWAL, Shiwei Liu, and Zhangyang Wang. Sparse moe as the new dropout: Scaling dense and self-slimmable transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=w1hwFUb_81.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, and Furu Wei. On the representation collapse of sparse mixture of experts. *arXiv preprint arXiv:2204.09179*, 2022.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognizing textual entailment challenge. In *Proceedings of the First PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. StableMoE: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7085–7095, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.489.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2401.06066.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Shachar Don-Yehiya, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Cold fusion: Collaborative descent for distributed multitask finetuning. *arXiv preprint arXiv:2212.01378*, 2022.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL http://jmlr.org/papers/v23/21-0998.html.

Marawan Gamal Abdel Hameed, Marzieh S. Tahaei, Ali Mosleh, and Vahid Partovi Nia. Convolutional neural network compression through generalized kronecker product decomposition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):771–779, Jun. 2022. doi: 10.1609/aaai.v36i1.19958. URL https://ojs.aaai.org/index.php/AAAI/article/view/19958.

Shwai He, Run-Ze Fan, Liang Ding, Li Shen, Tianyi Zhou, and Dacheng Tao. Merging experts into one: Improving computational efficiency of mixture of experts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14685–14691, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.907.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15262–15271, 2021b.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2024. ISBN 9781713871088.

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. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights. *Advances in Neural Information Processing Systems*, 35:29262–29277, 2022.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FCnohuR6AnM.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qrwe7XHTmYb.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient SMoe with hints from its routing policy. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=eFWG9Cy3WK.

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

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Zihan Liu, Hanyi Wang, Yaoyu Kang, and Shilin Wang. Mixture of low-rank experts for transferable ai-generated image detection, 2024. URL https://arxiv.org/abs/2404.04883.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models, 2024a. URL https://arxiv.org/abs/2402.14800.

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=81YIt63TTn.

James Martens. New insights and perspectives on the natural gradient method. *J. Mach. Learn. Res.*, 21(1), jan 2020. ISSN 1532-4435.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 2408–2417. JMLR.org, 2015.

Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Mohammed Muqeeth, Haokun Liu, and Colin Raffel. Soft merging of experts with adaptive routing. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=7I199lc54z. Featured Certification.

Eunbyung Park and Junier B Oliva. Meta-curvature. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/57c0531e13f40b91b3b0f1a30b529a1d-Paper.pdf.

Quora. Quora question pairs, 2017. https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2020.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. *arXiv preprint arXiv:2201.05596*, 2022.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Alexandre Rame, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization. *arXiv preprint arXiv:2212.10445*, 2023.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2019.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pp. 6732–6739, 2019.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, 2018.

Jialin Wu, Xia Hu, Yaqing Wang, Bo Pang, and Radu Soricut. Omni-smola: Boosting generalist multimodal models with soft mixture of low-rank experts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14205–14215, June 2024a.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of loRA experts. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=uWvKBCYh4S.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Resolving interference when merging models. In *NeurIPS*, New Orleans, USA, 2023. Proceedings of Machine Learning Research.

Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2013–2018, Lisbon, Portugal, 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1237. URL https://aclanthology.org/D15-1237.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

Zexuan Zhong, Mengzhou Xia, Danqi Chen, and Mike Lewis. Lory: Fully differentiable mixture-of-experts for autoregressive language model pre-training. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=LKEJPySnlt.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Y Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V Le, and James Laudon. Mixture-of-experts with expert choice routing. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=jdJo1HIVinI.

Supplement to "CAMEx: Curvature-aware Merging of Experts"

Table of Contents

A Comprision of CAMEx and Fisher-based merging methods

The pipeline comparison between CAMEx and Fisher-based merging methods is shown in Figure 6. Both approaches aim to capture the curvature of the parameter space during the merging process. (Diagonal) Fisher Merging Matena & Raffel (2022) applies a diagonal approximation to the Fisher information matrix. In this work, they estimate the diagonal of the Fisher matrix as:

$$\mathbf{F} = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathbf{D_m}} \left[\mathop{\mathbb{E}}_{\mathbf{y}\sim\mathbf{p}_\theta(\mathbf{y}|\mathbf{x})} \left[\nabla_\theta \log \mathbf{p}_\theta(\mathbf{y}|\mathbf{x}) \nabla_\theta \log \mathbf{p}_\theta(\mathbf{y}|\mathbf{x})^\top\right]\right], \tag{10}$$

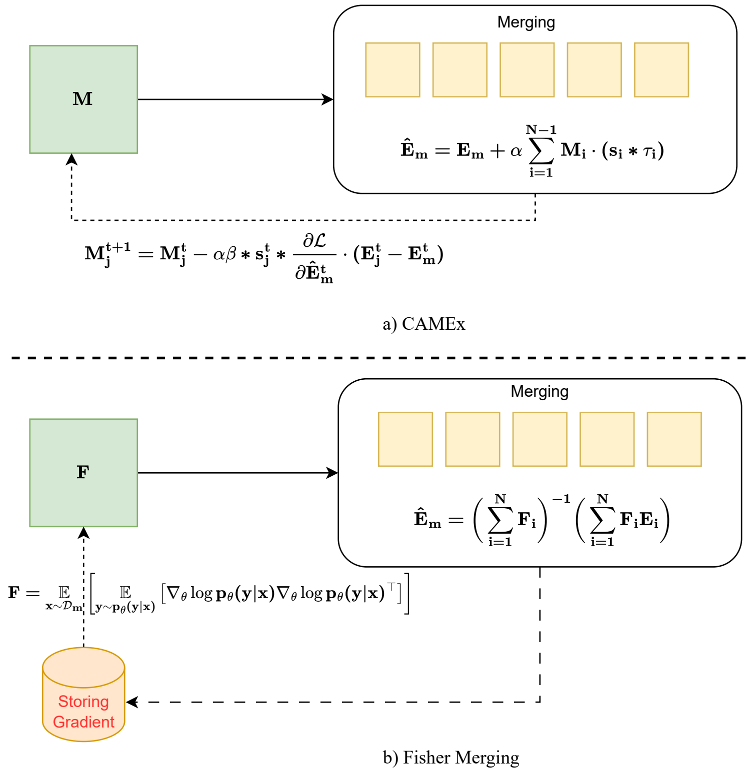

Figure 6: CAMEx merging pipeline vs Fisher-based merging pipeline. Note that Fisher merging requires the storing of the $\nabla_{E_i} \log p_{E_i}(y|x)$ for all experts and all x in training dataset. Furthermore, it has been pointed out that Fisher merging will have poor performance while using fewer examples to estimate the Fisher.

The expectation over y can be estimated via sampling from $p_\theta(y|x_i)$ or computed exactly when the number of classes is small. The closed-form solution for Fisher merging (without necessarily applying the diagonal approximation) is given by:

$$\hat{\mathbf{E}}_{\mathbf{m}}^{\mathbf{l}} = \left(\sum_{\mathbf{m}=1}^{\mathbf{M}} \mathbf{F}_{\mathbf{m}}^{\mathbf{l}}\right)^{-1} \left(\sum_{\mathbf{i}=1}^{\mathbf{N}} \mathbf{F}_{\mathbf{i}}^{\mathbf{l}} \mathbf{E}_{\mathbf{i}}^{\mathbf{l}}\right). \tag{11}$$

Thus, to approximate the Fisher Information Matrix for SMoE models, Fisher merging requires storing $\nabla_{E_i} \log p_{E_i}(y|x)$ for all experts and all x in the training dataset. Additionally, it has been noted that Fisher merging can suffer from poor performance when fewer examples are used to estimate the Fisher matrix (Matena & Raffel, 2022).

In the case of our method (depicted in Figure 6a), by denoting $\mathbf{M}_i$ as the curvature matrix of the i-th expert, CAMEx utilizes the formula for merging experts derived from the natural gradient descent update as:

$$\hat{\mathbf{E}}_{\mathbf{m}}^{\mathbf{l}} = \mathbf{E}_{\mathbf{m}}^{\mathbf{l}} + \alpha \sum_{\mathbf{i}=1}^{\mathbf{N}-1} \mathbf{M}_{\mathbf{i}} \cdot (\mathbf{s}_{\mathbf{i}}^{\mathbf{l}} * \tau_{\mathbf{i}}^{\mathbf{l}}) \tag{CA-Merg}$$

CAMEx implicitly implements the gradient-based matching between the task loss gradient and domain-vector of the corresponding expert to approximate the empirical Fisher through the dynamic of gradient descend update of $\mathbf{M}_\mathbf{i}$:

$$\mathbf{M}_{\mathbf{i}}^{\mathbf{t}+1} = \mathbf{M}_{\mathbf{i}}^{\mathbf{t}} - \beta * \frac{\partial \mathcal{L}}{\partial \mathbf{M}_{\mathbf{i}}^{\mathbf{t}}} = \mathbf{M}_{\mathbf{i}}^{\mathbf{t}} - \alpha\beta * \mathbf{s}_{\mathbf{i}}^{\mathbf{t}} * \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_{\mathbf{m}}^{\mathbf{t}}} \cdot (\mathbf{E}_{\mathbf{i}}^{\mathbf{t}} - \mathbf{E}_{\mathbf{m}}^{\mathbf{t}}), \tag{12}$$

where the term $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_{\mathbf{m}}} \cdot (\mathbf{E}_{\mathbf{i}} - \mathbf{E}_{\mathbf{m}})$ represents the outer product of the gradients of the task loss and the domain vectors. This operation contributes to capturing the curvature of the expert parameter space, ensuring curvature awareness during the merging process. This approach eliminates the need to compute the inversion of the empirical Fisher Information Matrix, thereby reducing computational overhead while maintaining sensitivity to parameter space curvature.

B Additional Details on Datasets

This section provides detailed information on the datasets and evaluation metrics used in the experiments in Section 3.

B.1 Language Modeling on WikiText

The WikiText-103 dataset consists of Wikipedia articles designed to capture long-range contextual dependencies. The training set includes approximately 28,000 articles, totaling around 103 million words. The validation and test sets have 218,000 and 246,000 words, respectively, spread across 60 articles per set, with each set comprising roughly 268,000 words. Our experiments follow the standard procedure described in Merity et al. (2017).

WikiText-2 is a smaller version of WikiText-103, containing 2 million tokens and a vocabulary of 33,000 words.

B.2 Text classification on GLUE benchmark

These tasks include MNLI (Williams et al., 2018), which assesses a model's ability to determine entailment between pairs of sentences; QQP (Quora, 2017) and MRPC (Dolan & Brockett, 2005), which focus on identifying sentence similarity and paraphrase detection; SST-2 (Socher et al., 2013) for sentiment analysis; CoLA (Warstadt et al., 2019) for grammaticality judgment; and QNLI (Wang et al., 2019) for question-answer classification. Additionally, STSB (Cer et al., 2017) evaluates the model's ability to measure sentence similarity, while RTE (Dagan et al., 2006) tests logical reasoning.

B.3 Question-answering on SQuAD and WikiQA

SQuADv1.1 (Rajpurkar et al., 2016) (Stanford Question Answering Dataset) is a widely used benchmark for reading comprehension and question answering tasks. It contains over **100,000 question-answer pairs** sourced from more than 500 Wikipedia articles. Each question is paired with a paragraph from the article, where the answer is a span of text extracted from the passage. The dataset consists of natural language questions that cover a wide range of topics, context paragraphs from Wikipedia, and answers marked by their start and end positions within the context. The primary task is to extract the correct answer span based on the posed question. Key features of the dataset include the need for exact span extraction, the large dataset size, and its task design focused on reading comprehension. Evaluation is typically done using Exact Match (EM), which measures the percentage of predictions that exactly match the ground-truth answers, and the F1 score, which measures the overlap between predicted and ground-truth answers by calculating the harmonic mean of precision and recall.

WikiQA (Yang et al., 2015) is an open-domain question answering dataset designed for answer sentence selection tasks. It consists of natural language questions primarily extracted from search engine queries, with candidate sentences sourced from Wikipedia articles. Each candidate sentence is labeled as either a correct or incorrect answer for the given question. The dataset contains 3,047 questions and 29,258 candidate sentences. The main challenge is selecting the correct sentence from a set of candidates, unlike SQuADv1.1, where the task focuses on extracting a text span. Key features include its real-world query origins, the sentence selection task, and the open-domain nature, which requires models to identify relevant sentences from diverse topics. WikiQA is evaluated using Accuracy.

Algorithm 1 The Overall Procedures of `CAMEx`.

Initialize: A model $\mathcal{M}$ with l SMoE layers, the total number of original experts N.
Let $\mathtt{H}^t \in \mathbb{R}^{B\times L\times N}$ and $\mathtt{T}^t \in \mathbb{R}^{B\times L\times d}$ denote the *router logits* and the *sequence of tokens* at intermediate layer t, respectively.
for layer $t = 1, \ldots, l$ **do**
  $K = L/S, T^l \leftarrow \mathtt{RESHAPE}(T, B*K, S, d)$ ▷ Begin Causal Segmenting
  $\mathtt{H}^l \leftarrow \mathbf{G}\left(T^l\right)$
  $\mathtt{H}^l \leftarrow \mathtt{ROLLandDETACH}\left(\mathtt{H}^l\right)$
  if `TIES-MERGING` **then** ▷ Generate mask for merging
    for expert $i = 1, \ldots, N-1$ **do**
      $\tau_i \leftarrow \mathbf{E}_i - \mathbf{E}_m$
      $\gamma_i \leftarrow sgn(\tau_i)$
    end for
    $\gamma^m = sgn(\sum_{i=1}^{N-1} \tau_i)$
    for expert $i = 1, \ldots, N-1$ **do**
      $\tau_i^m \leftarrow \gamma_i \wedge \gamma^m$
      $\tau_i \leftarrow \tau_i \cdot \mathbf{M}_i$
    end for
  else
    Generate mask for `DARE-MERGING`
  end if
  $\mathbf{E}_m \leftarrow \mathbf{E}_m + \gamma_m * \sum_{i=1}^{N-1} \mathtt{H}_i^l * \tau_i$ ▷ Merge Experts
end for

B.4 Image Classification on Imagenet

ImageNet-1k, the most widely utilized subset of the ImageNet dataset introduced by Deng et al. (2009), comprises 1.28 million images for training and 50,000 images for validation, across 1,000 categories. Performance evaluation is typically based on top-1 and top-5 accuracy metrics.

B.5 Adversarial Examples and Out-of-distribution datasets

ImageNet-A: The ImageNet-A dataset (Hendrycks et al., 2021b) contains real-world images specifically curated to fool ImageNet classifiers. It focuses on 200 classes, a subset of the 1,000 classes in ImageNet-1k. Errors made within these 200 classes are considered particularly significant, as they represent a wide variety of categories from ImageNet-1k.

ImageNet-O: This dataset consists of examples adversarially filtered to challenge out-of-distribution (OOD) detectors on ImageNet (Hendrycks et al., 2021b). It includes images from the larger ImageNet-22k dataset but excludes those present in ImageNet-1k. The selected samples are those that a ResNet-50 model confidently misclassifies as an ImageNet-1k class, and the primary evaluation metric is the area under the precision-recall curve (AUPR).

ImageNet-R: ImageNet-R contains a variety of artistic renditions of object classes found in the original ImageNet dataset (Hendrycks et al., 2021a). This dataset includes 30,000 artistic representations of images from 200 classes, selected from the ImageNet-1k subset. The dataset was created to challenge models with non-standard visual interpretations of the classes.

C Algorithm and implementation details

C.1 Causal segmenting

Background of Causal Segmenting: A significant advancement in SMoE design centers on fully differentiable architectures that eliminate the need for additional loss terms to stabilize training. In Muqeeth et al. (2024), a model was introduced that computes a weighted average of expert feed-forward networks (FFNs). For an input x with corresponding routing weights, the output is defined as:

$$o_x = \text{FFN}\left(h_x; \sum_{i=1}^{N} s_i \cdot \mathbf{E}_i\right), \quad \text{where} \quad s_i = \text{Softmax}(\mathbf{G}(h_x))_i.$$

However, applying this approach to autoregressive language models is computationally costly, as the merged FFN must be computed for each token in the sequence, leading to costs that scale linearly with the number of experts. An alternative based on pooling—routing via the sequence's average representation, as follows:

$$s_i = \text{Softmax}\left(\mathbf{G}\left(\frac{\sum_{j=1}^{L} h_{x_j}}{L}\right)\right)_i.$$

This, however, disrupts the autoregressive property essential for pre-training. To address this, Zhong et al. (2024) introduced causal segment routing. This technique merges FFNs in an MoE layer by utilizing information from the preceding segment to process the current segment. Specifically, given a training instance X consisting of L tokens (e.g., $L = 4096$), we divide the instance into N segments, each containing T (e.g., $T = 256$) consecutive tokens. For the k-th segment S_k, where $k > 1$, we compute the average of the hidden representations from the previous segment S_{k-1}, denoted as $\bar{h}_{k-1}$. By using the average hidden representation, the model can adapt to prompts of varying lengths during inference. The average hidden representation $\bar{h}_{k-1}$ is then employed to determine the routing weights, leading to a merged expert $\bar{\mathbf{E}}$:

$$\bar{h}_{k-1} = \frac{1}{T}\sum_{x \in S_{k-1}} h_x, \quad s_i = \text{Softmax}(\mathbf{G}(\bar{h}_{k-1})), \quad \bar{\mathbf{E}} = \sum_i s_i \cdot \mathbf{E_i}. \tag{13}$$

The merged expert $\bar{\mathbf{E}}$ is then used to process all tokens in the current segment S_k, i.e., $o_x = \text{FFN}(h_x; \bar{\mathbf{E}}), \forall x \in S_k$. This approach ensures that the model's routing decisions rely exclusively on data from preceding positions. For the first segment S_1, the segment's own representation is used to compute the merging weights for its FFN. To prevent information leakage, a stop-gradient operation is applied to $\mathbf{G}(\bar{h}_1)$:

$$\bar{h}_0 = \frac{1}{T}\sum_{x \in S_0}^{T} h_x \tag{14}$$

These tokens are then used to calculate the scores for the merging procedure

$$\begin{aligned} s_0 &= \texttt{DETACH}\left(\mathbf{G}(\bar{h}_1, k)\right) \\ s_i &= \mathbf{G}(\bar{h}_{i-1}), \quad i = 1, \ldots, S-1 \end{aligned} \tag{ROLLandDETACH}$$

C.2 Some implementations

Implementation of Kronecker product We consider the case where experts are linear layers

```
# Calculating domain-specific vectors
taus = weights - weight_m

# output_size = dim_out1 * dim_out2, input_size = dim_in1 * dim_in2
taus = taus.view(1, -1, dim_out1, dim_out2, dim_in1,
    dim_in2).repeat(rank, 1, 1, 1, 1, 1)
# Calculate Kronecker-product
taus = torch.einsum("rbij, rbjklm->rbiklm", curve1_out, taus)
taus = torch.einsum("rbik, rbjklm->rbjilm", curve2_out, taus)
taus = torch.einsum("rbil, rbjklm->rbjkim", curve1_in, taus)
taus = torch.einsum("rbim, rbjklm->rbjkli", curve2_in, taus)
# Summation along the Kronecker rank dimension and reshape
taus = taus.sum(0)
taus = taus.reshape(-1, output_size, input_size)
```

D MORE EXPERIMENT DETAILS

Supervised Fine-Tuning Hyper-Parameters Besides {batch size, learning rate, epoch counts} which vary for each task, we keep other hyper-parameters of supervised fine-tuning fixed for all tasks. These are shown in Table 8.

Table 8: Fine-tuning hyper-parameters of all models in Section 3

Hyper-Parameters	Values
Optimizer	ADAMW
Adam ϵ	$1e{-}6$
Adam β	(0.9, 0.98)
Warm-up steps	16
Weight decay	0.01
LR scheduler	LINEAR DECAY
Scaling factor α	1
Kronecker rank r	1

E DERIVATION

This is the derivation for Eqn 8 in Section 2.6

$$\hat{\mathbf{E}}_m = \mathbf{E}_m + \alpha \sum_{j=1}^{N-1} \mathbf{M}_j^{t+1} \cdot (s_j^{t+1} * \tau_j^{t+1}) \tag{15}$$

$$= \mathbf{E}_m + \alpha \sum_{j=1}^{N-1} \left[\mathbf{M}_j^t - \alpha\beta * s_j^t * \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_m^t} \cdot \tau_j^t \right] \cdot (s_j^{t+1} * \tau_j^{t+1}) \tag{16}$$

$$= \underbrace{\mathbf{E}_m + \alpha \sum_{j=1}^{N-1} s_j^{t+1} * \mathbf{M}_j^t \cdot \tau_j^{t+1}}_{\text{domain-specific merging with curvature-aware}} - \alpha^2 \beta \sum_{j=1}^{N-1} s_j^t s_j^{t+1} * \left({\tau_j^t}^\top \cdot \tau_j^{t+1} \right) \cdot \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_m^t} \tag{17}$$

F STEP-BY-STEP WALKTHROUGH FOR KEY EQUATIONS OF CAMEX

F.1 KEY EQUATION FOR MERGING OF CAMEX

$$\hat{\mathbf{E}}^{\mathbf{l}}_{\mathbf{m}} = \mathbf{E}^{\mathbf{l}}_{\mathbf{m}} + \alpha \sum_{\mathbf{i=1}}^{\mathbf{N-1}} \mathbf{M_i} \cdot (\mathbf{s}^{\mathbf{l}}_{\mathbf{i}} * \tau^{\mathbf{l}}_{\mathbf{i}}) \tag{CA-Merg}$$

In (CA-Merg) equation, we consider the merging of experts at l-th layer of the model. $\mathbf{E}^{\mathbf{l}}_{\mathbf{m}}$ denotes the "base" expert that is not included in the routing process. $\tau_i^l = \mathbf{E}^{\mathbf{l}}_{\mathbf{i}} - \mathbf{E}^{\mathbf{l}}_{\mathbf{m}}$ denotes i-th the domain-vector that adapts the "base" expert to the corresponding domain. Finally, s_i^l denotes the score of the i-th domain vector w.r.t the input. We view the merging of experts as a optimization problem where $\alpha * s_i^l$ acts as the adaptive learning rate. Therefore, it is straightforward to integrate natural gradient approach into this equation by introducing curvature matrices $\mathbf{M_i}$. Due to the challenging tractability of the Fisher Matrix in the intermediate layers of deep models, we proposed to learn them empirically through backpropagation as indicated by Eqn. 6 in the main text and a simmilar approach using meta-learning (Park & Oliva, 2019).

F.2 KEY EQUATION FOR MERGING IN DYNAMIC ARCHITECTURE

$$\begin{cases} \mathbf{E}_{\mathbf{m}}^{\mathbf{l+1}} & = \mathbf{E}_{\mathbf{m}}^{\mathbf{l}} + \dfrac{\alpha}{\mathbf{N-1}} \sum_{\mathbf{i=1}}^{\mathbf{N-1}} \mathbf{M_i} \cdot \tau_{\mathbf{i}}^{\mathbf{l}} \\ \hat{\mathbf{E}}_{\mathbf{m}}^{\mathbf{l+1}} & = \mathbf{E}_{\mathbf{m}}^{\mathbf{l+1}} + \alpha \sum_{\mathbf{i=1}}^{\mathbf{N-1}} \mathbf{M_i} \cdot (\mathbf{s}_{\mathbf{i}}^{\mathbf{l+1}} * \tau_{\mathbf{i}}^{\mathbf{l+1}}) \end{cases} \tag{Dynamic-Merg}$$

The (Dynamic-Merg) system perform two steps which are calculating base expert for the next layer and perform (CA-Merge), respectively. For the first step, we eliminate the score and take the average of curvure-aware domain vector instead to avoid information leakage. The result then takes the role as the base expert for the next layer.

F.3 Curvature update

In the main text, we try to give an explaination of how our method we update the curvature matrix with the curvature information of the parameters space. To achive that, we first take the derivative of equation (CA-Merge) w.r.t the curvature matrix $\mathbf{M_i}$:

$$\frac{\partial \hat{\mathbf{E}}_{\mathbf{m}}}{\partial \mathbf{M_j}} = (\alpha s_j * \tau_j) = \alpha s_j * (\mathbf{E_j} - \mathbf{E_m}) \tag{18}$$

To evaluate the gradient of the task loss $\mathcal{L}$ w.r.t $\mathbf{M_i}$ we apply the chain-rule:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{M_j}} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_{\mathbf{m}}} \cdot \frac{\partial \hat{\mathbf{E}}_{\mathbf{m}}}{\partial \mathbf{M_j}} = \alpha s_j * \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{E}}_{\mathbf{m}}} \cdot (\mathbf{E_j} - \mathbf{E_m}) \tag{19}$$

G Student's t-test for experiments on GLUE dataset

We report the t-test results, beginning with the null hypothesis H_0: *The performance between each pair of T5-Ties-CA vs T5-Ties, and T5 on GLUE SST-2, MRPC, CoLA, and MNLI are the same.*. In this test, we choose the significant value to be 0.05.

Table 9: Evaluation results on SST-2 with different random seeds.

Index	**Ties_CA**	**Ties**	**Vanilla**
1	94.44	93.77	93.31
2	94.86	94.13	93.33
3	94.62	93.90	93.21
4	94.60	94.12	93.46
5	94.54	93.70	93.41
6	94.55	93.87	93.56
7	94.37	94.03	93.67

Table 10: T-statistic and p-value when evaluating on SST-2.

Test	**t-statistic**	**p-value**
Ties-CA vs Vanilla	13.72	1.08e-8
Ties-CA vs Ties	7.36	8.74e-6

Table 11: Evaluation results on MRPC with different random seeds.

Index	Ties_CA	Ties	Vanilla
1	92.35	91.35	89.85
2	92.61	91.30	89.65
3	92.55	91.40	89.74
4	92.40	91.55	89.49
5	92.54	91.62	89.76
6	92.44	91.77	89.85
7	92.33	91.43	89.62

Table 12: T-statistic and p-value when evaluating on MRPC.

Test	t-statistic	p-value
Ties-CA vs Vanilla	42.91	1.67e-14
Ties-CA vs Ties	12.95	2.06e-8

Table 13: Evaluation results on CoLA with different random seeds.

Index	Ties_CA	Ties	Vanilla
1	61.01	57.95	57.74
2	59.53	58.63	57.82
3	60.36	58.90	58.03
4	60.13	58.92	58.23
5	59.41	58.31	58.51
6	59.33	57.38	57.36
7	60.03	58.53	58.40

Table 14: T-statistic and p-value when evaluating on CoLA.

Test	t-statistic	p-value
Ties-CA vs Vanilla	7.14	1.18e-5
Ties-CA vs Ties	5.16	2.00e-4

Table 15: Evaluation results on MNLI with different random seeds.

Index	Ties_CA	Ties	Vanilla
1	86.52	86.25	86.22
2	86.45	86.32	86.31
3	86.37	86.39	86.36
4	86.59	86.46	86.41
5	86.32	86.53	86.50
6	86.54	86.38	86.34
7	86.47	86.41	86.34

Table 16: T-statistic and p-value when evaluating on MNLI.

Test	t-statistic	p-value
Ties-CA vs Vanilla	2.29	0.04
Ties-CA vs Ties	1.49	0.16

Based on the p-values in the tables above, we draw the following conclusions:

- The T5-Ties-CA variant significantly outperforms T5-Ties and T5-Vanilla on SST-2, MRPC, and CoLA.
- While T5-Ties-CA does not statistically outperform T5-Ties on MNLI, it still demonstrates significant improvement over T5-Vanilla.

H ADDITIONAL EXPERIMENTS

H.1 INTEGRATING CAMEX INTO TWIN-MERGING

We expand our experiments to include a broader range of most recent merging expert methods. Specifically, we integrated our CAMEx method with the Twin-Merging approach (Lu et al., 2024b). Key distinctions between CAMEx and Twin-Merging lie in their core mechanisms:

- Our method is a non-Euclidean merging method, which utilizes the curvature-aware matrix, whereas Twin-Merging is a model merging method, which relies on Euclidean merging.
- Our approach is specifically designed for finetuning, in contrast to Twin-Merging, which is intended for post-training.
- Finally, our dynamic mechanism performs inter-layer to form the merged expert, unlike Twin-Merging, which uses within-layer pre-calculations for merging. To integrate our method with Twin-Merging, we first fine-tune the Curvature Aware model for a specific GLUE task. At test time, we apply the Twin-Merging algorithm to merge experts, referring to our approach as Twin-CA. Notably, we found Twin-Merging to be a simple yet powerful technique that is easy to implement and helps reduce memory usage during inference. We adhere to the original implementation settings, using a sparsity density value of 0.2.

Table 17: Performance of Twin-Merging and its Curvature Aware (CA) variant on GLUE tasks.

Method	MRPC	RTE	STSB
Twin-Merging	91.97	72.20	88.56
Twin-CA (**Ours**)	**92.30**	**74.73**	**89.55**

The results in Table 17 demonstrate the effectiveness of our CAMEx approach when integrated with the Twin-Merging mechanism on GLUE tasks, highlighting its strong potential for incorporation into more advanced merging techniques.

H.2 EXPERIMENTS ON TOKEN-CHOICE VS EXPERT-CHOICE ROUTING

We also demonstrate our merging approach with the following routing mechanisms:

- Stable MoE routing (Dai et al., 2022).
- Naive routing (Shazeer et al., 2017).

Note that the Curvature Aware model leverages the segment routing strategy (the causal segmenting strategy) proposed in Lory (Zhong et al., 2024), enabling a direct comparison between our model and the expert choice method.

Table 18: Performance of T5-base variants on the finetuning GLUE tasks.

Method	MRPC	RTE	STSB	SST-2
Expert Choice MoE	**93.10**	66.78	89.19	93.80
Stable MoE routing CA	92.96	**78.76**	**89.64**	**94.63**
Naive routing CA	92.49	78.70	89.56	94.61

H.3 Longer training for Wikitext-103 pre-training

We conduct additional experiments by training for longer iterations on the Wikitext-103 dataset. The performance gaps between methods remain stable starting around epoch 40.

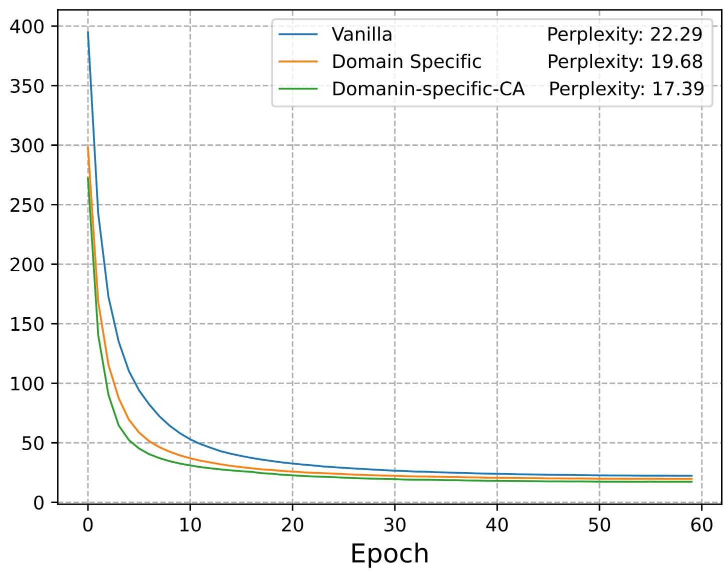

Figure 7: Performance of Vannila, Domain-specific, Domain-specific-CA under longer pre-training.

As shown in Figure 7 the trends demonstrate consistent improvements of our method over the baseline, with the gap remaining significant even after prolonged training.

H.4 More comprehensive ablation study on hyperparameters

H.4.1 Ablation study on α

We extend the range of α for the ablation study, specifically evaluating Dare-CA and Ties-CA with $\alpha \in [0.1, 1.6]$. The evaluation is conducted using 5 different seeds, and the results are averaged.

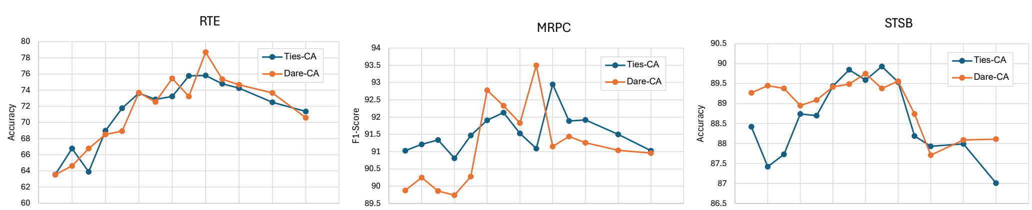

Figure 8: Test performance of Curvature-Aware methods under varying settings of α.

The results in Figure 8 lead to the following observations:

- The performance of the models is suboptimal or even worse than the vanilla baseline when α is either too small ($\alpha \in [0.1, 0.4]$) or too large ($\alpha > 1.1$).
- Dare-CA is more sensitive to the choice of α, showing sharper improvements and declines across the range.
- Ties-CA exhibits more gradual changes, suggesting it is more robust to variations in α. The optimal range for α is $[0.8, 1.0]$.

H.4.2 ABLATION STUDY ON NUMBER OF EXPERTS

We conduct additional studies on our method using different numbers of experts in the T5 backbone.

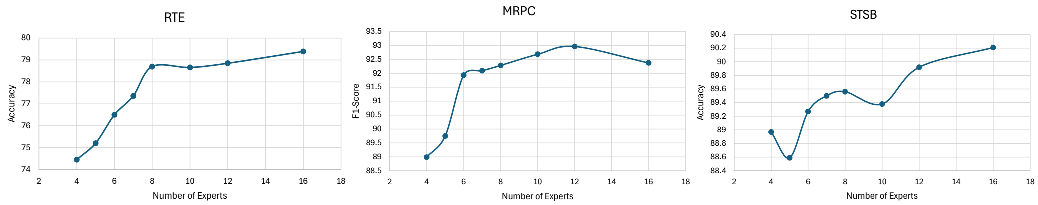

Figure 9: Test performance of Curvature-Aware methods under varying number of experts.

The following conclusions can be drawn from Figures 9:

- Increasing the number of experts generally improves accuracy up to a certain point: Accuracy improves as the number of experts increases, with the most significant gains occurring from 4 to 8 experts.
- After 12 experts, the accuracy either saturates or slightly decreases.
- We suggest using 8 experts as it provides a balanced trade-off between performance and efficiency.