Automatic Expert Discovery in LLM Upcycling via Sparse Interpolated Mixture-of-Experts

Shengzhuang Chen
Thomson Reuters
Foundational Research
shengzhuang.chen@thomsonreuters.com

Ying Wei*
Zhejiang University
ying.wei@zju.edu.cn

Jonathan Richard Schwarz*
Thomson Reuters
Foundational Research
jonathan.schwarz@thomsonreuters.com

Abstract

We present Sparse Interpolated Mixture-of-Experts (SIMoE) instruction-tuning, an end-to-end algorithm designed to fine-tune a dense pre-trained Large Language Model (LLM) into a MoE-style model that possesses capabilities in multiple specialized domains. During instruction-tuning, SIMoE automatically identifies multiple specialized experts under a specified sparsity constraint, with each expert representing a structurally sparse subset of the seed LLM's parameters that correspond to domain-specific knowledge within the data. SIMoE simultaneously learns an input-dependent expert merging strategy via a router network, leveraging rich cross-expert knowledge for superior downstream generalization that surpasses existing baselines. Empirically, SIMoE consistently achieves state-of-the-art performance on common instruction-tuning benchmarks while maintaining an optimal performance-compute trade-off compared to all baselines.

1 Introduction

The rapid advancement of large language models (LLMs) (Meta, 2024; OpenAI, 2024) has revolutionized natural language processing, cementing their role as foundational tools across disciplines such as engineering (Hou et al., 2024), mathematics (Romera-Paredes et al., 2023), humanities (Ziems et al., 2024), and the life sciences (Lin et al., 2022). While pre-trained LLMs demonstrate impressive general-purpose capabilities, their adaptation to specialized tasks often demands extensive instruction-tuning (Zhang et al., 2023). This process typically involves supervised fine-tuning on domain-specific instruction datasets to align outputs with task requirements. Recent research highlights the critical interplay between scaling instruction-tuning data diversity, volume, and model capacity to achieve robust generalization (Wei et al., 2022; Longpre et al., 2023; Meta, 2024). However, balancing these dimensions efficiently without incurring prohibitive computational costs remains an open challenge.

This tension has spurred growing interest in Sparse Mixture-of-Experts (SMoE) architectures (Shazeer et al., 2017), which offers a promising pathway to flexible scaling of model capacity while maintaining inference efficiency by dynamically activating subsets of parameters per input for both training and inference. Yet, practical adoption of SMoEs in instruction fine-tuning is hindered by two key barriers: (i) the scarcity of publicly available pre-trained SMoE checkpoints (Muennighoff et al., 2025) and (ii) the immense computational cost of (pre-)training them from scratch. To overcome these challenges, researchers and practitioners are increasingly exploring a cost-effective alternative known as *upcycling* (Komatsuzaki et al., 2023), which expands pre-trained dense LLMs into SMoE architectures by replacing subsets of their feed-forward networks (FFN) with SMoE modules. While cost-effective, existing upcycling methods remain fraught with limitations that undermine their efficacy.

First, manual selection of pre-trained parameters for upcycling: Existing methods upcycle FFN or attention blocks in pre-trained LLMs, assuming uniform utility across all upcycled layers. However, this fixed upcycling strategy fails to account for critical factors: (1) model-specific dynamics – layers and parameters within the same pre-trained LLM can exhibit diverse properties and varying importance to model functionality, and (2) the highly domain-specific nature of instruction-tuning data, which may require different parts of pre-trained models to be upcycled and fine-tuned optimally. This disconnect between the algorithm, model, and data renders existing upcycling heuristics highly inflexible, preventing them from adaptively meeting the needs of specific instruction-tuning scenarios. As a result, existing upcycled approaches lead to

*Corresponding authors; Equal contribution.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 16703–16717

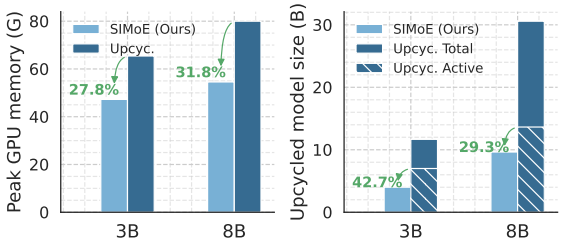

Figure 1: Compute cost in terms of (*left*) peak memory per GPU during upcycling instruction-tuning of 3B and 8B pre-trained LLMs, and (*right*) number of model parameters at inference.

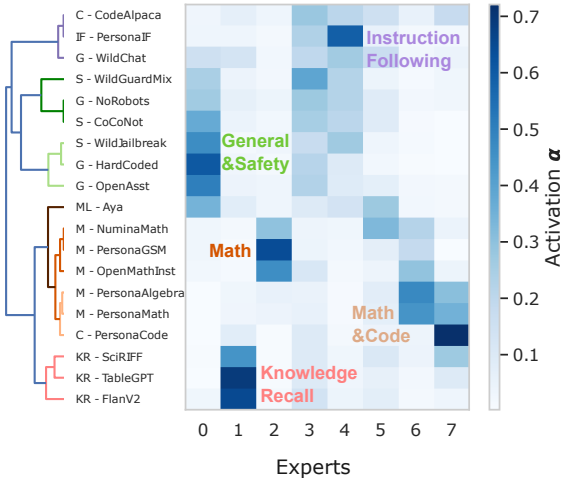

Figure 2: Dendrogram illustrating task similarity derived from learned expert activations, with prefixes indicating domain categories.

suboptimal performance and diminish the generalization capabilities of upcycled LLMs. The benefits of dynamically identifying rather than manually specifying components are well documented (Von Oswald et al., 2021; Schwarz et al., 2021; Chen et al., 2024, e.g.)

Second, a lack of systematic mechanisms to encourage expert specialization and cooperation. While Dai et al. (2024) propose an architectural approach to promote expert specialization by using a fixed shared expert that is always active in addition to the routed experts, recent work in Muennighoff et al. (2025) report performance degradation with this approach, possibly due to limited adaptability with rigid expert partitions. Alternatively, some upcycling approaches explicitly promote specialization by fine-tuning domain-specific experts independently and then merging them into a unified SMoE architecture. However, these methods critically depend on high-quality domain labels and optimal domain partition – requirements that are not readily met in practice. Consequently, upcycled experts often exhibit redundancy and fragmented specialization, undermining their efficacy.

These shortcomings point to an urgent need for an adaptive, automated framework that optimizes *where-to-upcycle* and fosters expert specialization and synergy. To this end, we introduce **S**parse **I**nterpolated **M**ixture-**o**f-**E**xperts (SIMoE), a novel algorithm and MoE architecture designed to address both challenges in upcycling instruction-tuning (Fig. 3). During training, SIMoE automatically identifies multiple experts for upcycling through sparsity-constrained optimization. Each expert represents a structurally sparse subset of the base LLM's parameters, corresponding to specialized knowledge within the training data (Fig. 2). Crucially, SIMoE enables learnable, soft parameter sharing between experts while imposing an orthogonal penalty to encourage specialization, thereby dynamically discovering a nuanced balance between synergy and specialization in the upcycled experts through optimization (Fig. 5). Our complementary innovations lead to empirical superiority in our method over strong baselines, delivering significant improvement over 1.6% in cross-task generalization (Tab. 1) with approximately 30% savings in training and inference memory costs (Fig. 1).

We summarize our main contributions as follows:

- We propose SIMoE, an effective and flexible method that systematically determines *where-to-upcycle* and fosters expert specialization and synergy during instruction-tuning.
- We offer an effective solution that maintains scalability, enabling its application to pre-trained LLMs with billion-scale parameters.
- We empirically validate the efficacy of our method, demonstrating superior cross-task generalization performance on the Super-NaturalInstruction benchmark, as well as outperforming the recently open-sourced state-of-the-art Tülu-v3-8B-SFT (Lambert et al., 2025) on common LLM benchmarks.

2 Related Work

2.1 Sparse Mixture-of-Experts

Sparse Mixture-of-Experts (SMoE) architectures (Jiang et al., 2024; Dai et al., 2024; Muennighoff et al., 2025) have gained prominence in

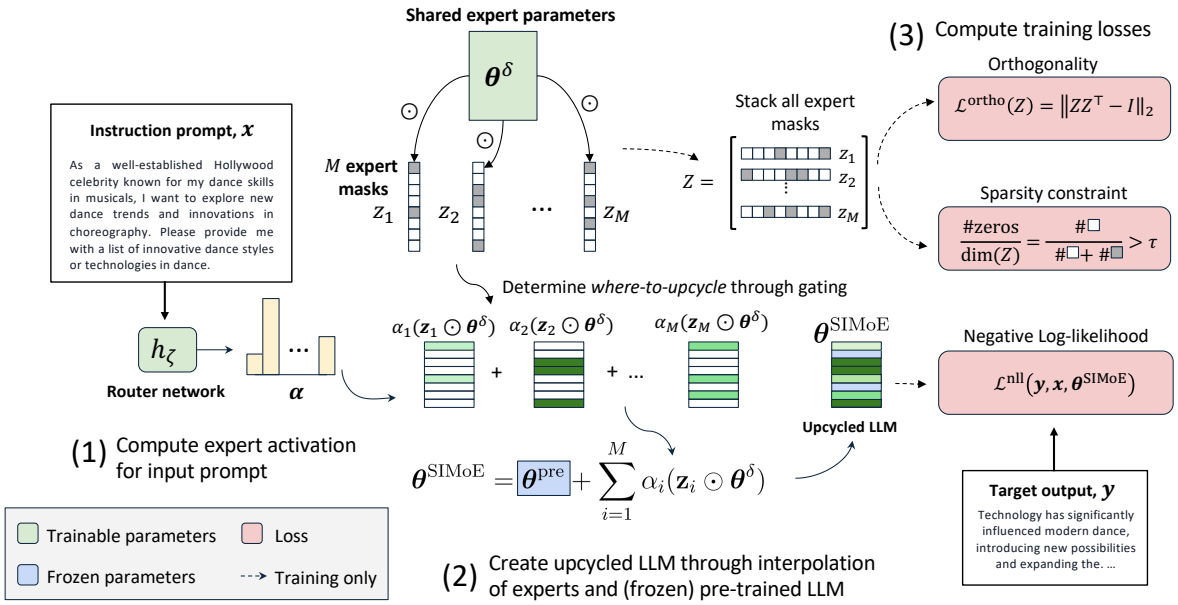

Figure 3: Overview of the proposed **S**parse **I**nterpolated **M**ixture-**o**f-**E**xperts (SIMoE) instruction-tuning approach. SIMoE conceptually resembles the MoE principle in routing and combining specialized parameter components through soft merging, while it differs in implementation from conventional MoE architectures by defining each expert as a specific subset of sparse parameters within a shared network. Specifically, SIMoE upcycles a pre-trained LLM into a MoE-style model characterized by M experts, consisting of a **shared**, trainable set of expert parameters $\boldsymbol{\theta}^{\delta}$ and M **distinct**, trainable sets of expert masks $\{\boldsymbol{z}_m\}_{m=1}^{M}$. In forward computation, **(1-2)** SIMoE merges experts via a weighted-sum with coefficients $\boldsymbol{\alpha}_i$ generated via a router network $h_{\boldsymbol{\zeta}}$ based on the input prompt $\boldsymbol{x}$, before combining with the frozen, pre-trained LLM. **(3)** During instruction-tuning, we enforce structured sparsity and orthogonality on the trainable masks in addition to the usual NLL loss, determining *where-to-upcycle* and encouraging expert specialization in a data-driven, fully automatic manner.

both LLM pre-training and post-training due to their superior scalability compared to dense counterparts. A standard SMoE architecture replaces the FFN $f : \mathcal{X} \to \mathcal{Y}$, which maps input to output, in the Transformer block (Vaswani et al., 2017) with a MoE module consisting of two components: (1) a set of M experts $\{f_1, f_2, \ldots, f_M\}$, and (2) a router function $h : \mathcal{X} \to \mathbb{R}^M$ that outputs expert activation $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_M]$ for input $\boldsymbol{x}$. The SMoE output $\boldsymbol{y}$ is then given by a weighted combination of expert outputs:

$$\boldsymbol{y} = \sum_{i=1}^{M} \alpha_i \cdot f_i(\boldsymbol{x}), \quad \text{where } \boldsymbol{\alpha} = h(\boldsymbol{x}). \qquad (1)$$

A defining feature of SMoE is its sparsity in expert activation: typically, only a subset of the most relevant experts is activated per input using the Top-K (Fedus et al., 2021) or Top-P (Huang et al., 2024) routing schemes. Early work have explored discrete expert-to-token routing – selecting only the most activated subsets of experts for each token in $\boldsymbol{x}$ (Shazeer et al., 2017; Riquelme et al., 2021; Lepikhin et al., 2021; Fedus et al., 2021), while later work introduced alternatives, such as token-to-expert routing – choosing the top scored subsets of tokens for each expert (Zhou et al., 2022), random and hash-based routing (Roller et al., 2021; Zuo et al., 2022). Advanced approaches further optimize routing stability and improve routing load-balancing (Lewis et al., 2021; Liu et al., 2023).

However, discrete and sparse expert activations often lead to training instability for gradient-based optimization (Mustafa et al., 2022; Dai et al., 2022). To address this, recent innovations propose soft, continuos routing alternatives. For example, Soft-MoE (Puigcerver et al., 2023) relaxes discrete assignments via continuous approximations, while SoftMerging (Muqeeth et al., 2023) explore combining experts through a weighted sum in the parameter space.

2.2 Upcycling Dense LLMs to SMoEs

Upcycling refers to the process of continual training pre-trained dense LLMs into SMoE architectures. This approach allows for efficient scaling

of model capacity without incurring the substantial computational costs associated with training an SMoE from scratch. The vanilla upcycling (Komatsuzaki et al., 2023) approach involves two main steps: 1) replacing specific modules within the dense LLM with SMoE modules by replicating the weights of pre-trained modules as experts and randomly initializing the router networks, and 2) continually training the resulting SMoE model until convergence or the exhaustion of a target compute budget.

Initially explored in pre-training to mitigate the training instability and high costs of training SMoE models from scratch (Komatsuzaki et al., 2023; He et al., 2024), vanilla upcycling has also adopted effectively in post-training scenarios, resulting in SMoE models with improved generalization performance (Jiang et al., 2025). Recent research (Sukhbaatar et al., 2024; Zhang et al., 2024) extends upcycling to multi-domain post-training, where the post-training data are divided into well-defined domain subsets. This enables parallel fine-tuning of multiple independent and specialized domain- (or task-) experts. These experts are then merged into a unified SMoE model through uniform weight averaging at non-expert layers, followed by a secondary fine-tuning stage to ensure performance.

Despite the simplicity and effectiveness of these approaches, the decision of *where-to-upcycle* throughout upcycling remains heuristically designed. Typically, this involves retaining the FNN (Sukhbaatar et al., 2024) and/or attention layers (Zhang et al., 2024) as SMoE modules while merging the remaining non-expert parameters through simple unweighted averaging. Such hand-picked strategy risks compromising performance due to architectural biases and uniform averaging divergent specializations (Li et al., 2022).

3 Sparse Interpolated MoE Instruction-Tuning

As outlined in the Introduction, the primary obstacles hindering effective upcycling instruction-tuning of SMoE models revolve around two unresolved challenges: 1) *where-to-upcycle*: determining optimal layer positions in pre-trained models to integrate experts, and 2) *specialization-cooperation trade-off*: balancing expert specialization with inter-expert synergy. Existing methods lack a systematic, data-driven framework to jointly address these challenges. We propose a unified solution grounded in *learnable structured sparsity*, enabling the automatic discovery of upcycling locations in the pre-trained seed LLM while intrinsically managing the specialization-cooperation equilibrium through end-to-end optimization. An overview of our method is shown in Fig. 3.

3.1 Sparse Interpolated MoE Module

3.1.1 Learning *where-to-upcycle* with Structured Sparsity

Current approaches to selecting layers for integrating MoE modules – such as targeting FNN layers or prioritizing attention components – rely on architectural heuristics or trial-and-error fine-tuning (Fedus et al., 2021; Sukhbaatar et al., 2024). However, emerging evidence (Zhang et al., 2024), along with our own experiments (Tab. 4 (a)), indicates that these fixed architectural biases can lead to suboptimal performance, potentially due to model-specific layer dynamics and data dependencies (Pan et al., 2024). This underscores the need for an adaptive, data-driven method.

To address this challenge, we formulate determining *where-to-upcycle* as a sparse optimization problem. Our key idea is to consider every pre-trained linear layer in the LLM as a potential candidate for upcycling by integrating trainable *per-expert* binary masks $\boldsymbol{z} \in \{0,1\}$ and expert parameters $\boldsymbol{\theta}_i^\delta$ into each layer. The linear layer now computes[1] $\boldsymbol{y} = f_\theta(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x}$, $\boldsymbol{\theta} = \sum_{i=1}^{M} \alpha_i \cdot \boldsymbol{z}_i \odot \boldsymbol{\theta}_i^\delta$, and M is the maximum number of experts allowed to upcycle per mask position. During instruction-tuning, we enforce L_0-like sparsity on $\boldsymbol{z}$, allowing the optimization to automatically prune non-essential experts, revealing the learned upcycling strategy.

The design of trainable masks z involves a critical trade-off between granularity and computational overhead. While parameter-wise masks enable maximal expressiveness, they are prohibitively compute-expensive for LLMs with billion-scale parameters[2]. To resolve this, we adopt *structured* sparsity on the masks, where masks $\boldsymbol{z} \in \{0,1\}^{\mathcal{X}}$ gates input neurons in weight matrix $\boldsymbol{\theta} \in \mathbb{R}^{\mathcal{Y} \times \mathcal{X}}$. This improve scalability while ensuring two key advantages: **(a)** Fine-grained upcycle control. Masks modulate expert contributions at the granularity of individual neurons, enabling more precise adapta-

[1] The bias term is omitted for neat presentation.

[2] For more details, see the ablation study in Appendix C.1

tion than layer or FNN upcycling. **(b)** Hardware-friendly sparsity: structured sparsity patterns align with modern accelerator architectures, avoiding irregular memory access penalties from parameter-wise sparsity.

The implementation still risks overwriting pre-trained knowledge and could incur training instability if all $\boldsymbol{z}_i$, at the same MoE layer, collapse to zeros. As a solution, we anchor the MoEs to the *frozen* pre-trained initialization, $\boldsymbol{\theta}^{\texttt{pre}}$, reparameterizing the output as

$$\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}),\ \boldsymbol{\theta} = \boldsymbol{\theta}^{\texttt{pre}} + \sum_{i=1}^{M} \alpha_i \cdot \boldsymbol{z}_i \odot \boldsymbol{\theta}_i^{\delta} \quad (2)$$

This ensures updates remain sparse additions to– rather than replacements of – pre-trained functionality, mitigating catastrophic forgetting while maintaining the same model expressiveness.

One remaining challenge is ensuring differentiability for gradient-based masks optimization. We address this by drawing on the sparse reparameterization method utilized in Schwarz and Teh (2022). At a high-level, each scaler mask z is derived from a latent variable s of a hard concrete distribution (Louizos et al., 2018) following reparameterization sampling and a transformation, i.e., $z = \min(1, \max(0, s)), s \sim q_\phi(s)$. Sparsity in z is then enforced via limiting its expected L_0-norm in probability, $\bar{L}_0(z) = P(z \neq 0)$, which translates to a penalty on the CDF of q_ϕ as $1 - Q_\phi(s \leq 0)$ that is both differentiable and analytical to compute. This approach avoids non-differentiable thresholding while allowing *exact* zeros in forward passes – a critical advantage over soft masking. More details are provided in Appendix A.1.

We enforce sparsity via solving a constrained optimization problem in instruction-tuning, gaining a precise control of the final sparsity in z hence the upcycled experts. More details in Section 3.2.

3.1.2 Parameter-Shared Experts with Mask-Driven Specialization

A fundamental challenge in SMoE upcycling lies in achieving an optimal balance between expert specialization and cooperation. While specialization is crucial for preventing model collapse, effective cooperation among experts is essential for maintaining stable training dynamics. We address this inherent tension through two complementary innovations, leading to our final SIMoE architecture, as illustrated in Fig. 3.

First, to promote collaboration among experts, we couple experts by sharing their trainable parameters in the MoE layer in Eqn. (2), i.e., $\boldsymbol{\theta}_i^{\delta} = \boldsymbol{\theta}^{\delta}$ for all $i \in [M]$, while maintaining distinct sparsity masks. This design serves dual purposes: it facilitates knowledge transfer through shared parameters and (training) gradients across experts, while substantially reducing memory requirements compared to traditional upcycling approaches that necessitate M separate copies of the original weights.

Second, to ensure expert specialization and prevent model collapse due to potential excessive parameter sharing, we impose an orthogonal penalty on the distinct masks, i.e., $\mathcal{L}^{\texttt{ortho}}(\boldsymbol{Z}) = \|\boldsymbol{Z}\boldsymbol{Z}^\top - \boldsymbol{I}\|_2$ enforcing complementary mask activation patterns while allowing overlap for cooperation. The expression softly penalizes deviation of the dot products between different expert masks – the off-diagonal terms in $\boldsymbol{Z}\boldsymbol{Z}^\top$ – from zero, thereby promoting orthogonality.

The interplay between parameter sharing and mask orthogonality enables our approach to dynamically discover the optimal balance between expert cooperation and specialization during instruction-tuning.

3.1.3 Instance-level Expert Routing

Common practices for implementing the router network h_ζ in SMoE systems encompass three primary strategies: token-level, instance-level, and task-level routing. Token-level routing allows different experts to process individual tokens within an input, while instance- and task-level routing maintain consistent expert activation patterns across all tokens in an input or across all inputs within a dataset (or task), respectively. SIMoE remains compatible with all three strategies, as its implementation modifies the configuration of experts only.

While task-level routing may incentivize task-specific expert specialization, but its applicability is limited by the frequent absence of task meta-information during training and inference. Between token- and instance-level routing, we empirically identify instance-level routing as the optimal choice (Tab. 4 (d)). For the router implementation, each input instruction prompt is processed through the pre-trained LLM to generate token embeddings. We extract the final token embedding as the instance representation, which is then processed by a MLP layer parameterized by ζ to produce routing logits. These logits are directly transformed

through a Softmax function to obtain the expert activations $\boldsymbol{\alpha}$ – similar to the soft merge mechanism proposed in Muqeeth et al. (2023), which enhances training stability and eliminates the need for auxiliary router losses (Dai et al., 2022).

3.2 Controlled Compute Cost via Sparsity-constrained Optimization

While it is straightforward to solve *where-to-upcycle* via a sparsity-regularized training objective, i.e., adding a sparsity regularization term with a fixed coefficient, we instead resort to a more practically compelling approach: enforcing sparsity via solving a sparsity-constrained optimization problem. The latter ensures precise control over the final sparsity in z hence upcycled experts, consequently governs the computational cost of the post-trained LLM at inference – as expert parameters with zero-valued masks can be safely pruned, eliminated from forward computations. To this end, we consider optimizing the Lagrangian:

$$\min_{\boldsymbol{\theta}^{\boldsymbol{\delta}},\boldsymbol{\zeta},\boldsymbol{\phi}} \max_{\lambda \geq \mathbf{0}} \mathbb{E}_{\mathbf{y},\mathbf{x}}\left[\mathcal{L}^{\texttt{nll}}(\mathbf{y},\mathbf{x},\boldsymbol{\theta}^{\texttt{SIMoE}})\right] + \beta\mathcal{L}^{\texttt{ortho}}(\boldsymbol{Z}) + \lambda(\tau - (1 - L_0(\boldsymbol{Z}))), \quad (3)$$

where the last term corresponds to the constraint $1 - L_0(\boldsymbol{Z}) \geq \tau$ that lower-bounds the sparsity (number of zeros) in masks, hence upcycled experts, by a hyperparameter $\tau \in [0, 1)$. We adopt simultaneous gradient descent and projected gradient ascent for optimizing the model parameters and the Lagrangian multiplier λ, respectively. To avoid over-penalizing the model capacity from surpassing the sparsity constraint, we reset λ to zero once the constraint is satisfied (Gallego-Posada et al., 2022). More details are deferred to Appendix A.2.

4 Experiment

4.1 Experimental Setup

We validate SIMoE on two sets of main experiments that vary in scales of instruction-tuning dataset and model size. For our implementation, we set the maximum number of experts to $M = 8$ and the target sparsity constraint to $\tau = 75\%$ by default. Thanks to our unique method design (Section 3), we were able to consider upcycling (attaching SIMoE layer) at every linear layer in the pre-trained LLM, enabling us to optimize for *where-to-upcycle* globally with minimum manual intervention while still maintain compute feasibility.

We compare our results against two types of baselines including: **(a)** full fine-tuning (Full FT), and **(b)** sparse upcycling approaches, including sparse upcycling (Komatsuzaki et al., 2023), and the more empirically competitive BTX (Sukhbaatar et al., 2024) which we use whenever domain labels are available for training. Additional implementation for baselines can be found in Appendix B.1. We strictly adhere to the official training and evaluation setups of the benchmarks.

Super-NaturalInstructions **SNI** (Wang et al., 2022) includes 1,616 instructed NLP tasks over 76 distinct task categories. We strictly follow the official recommended recipe for benchmarking cross-task generalization of instruction-tuned LLM on SNI: training on 64 task categories while leaving out 12 *unseen* categories covering 154 tasks for evaluation only. We adopt ROUGE-L (Lin, 2004) for reporting aggregated performance results.

Tülu-v3 The Tülu-v3 post-training recipe (Lambert et al., 2025) provides a large scale instruction-tuning dataset. We use the publicly available SFT data mixture for training, which consists of a total of 939,343 unique training instances from multiple natural and synthetic sub-datasets, spanning a wide range of domains; and the official Tülu-v3 evaluation suite for testing and reporting performance.

4.2 Results

Cross-task generalization SIMoE consistently achieves the strongest performance across all of our experiments. As shown in Tab. 1, SIMoE excels in cross-task generalization on the SNI benchmark, outperforming baselines in at least 7 out of 12 unseen task categories. This results in overall average gains of 2.5% and 1.6% over Full FT for the 3B and 8B pre-trained models, respectively.

Scalability and flexibility In Tab. 2, SIMoE demonstrates strong generalization performance when transferring to a larger pre-trained model and a relatively larger instruction fine-tuning dataset,i.e., the Tülu-v3. SIMoE maintains its competitive edge, surpassing all baseline methods on average over 12 common LLM evaluation benchmarks, with a noticeable improvement of 0.6% over the official Tülu-v3-8B-SFT model – the recent open-source state-of-the-art – while using only 1/3 of the model capacity (number of params) of the BTX upcycled LLM (Tab. 3), demonstrating high efficiency in parameter utilization of our SIMoE

Seed LLM	Method	TitleGen.	Coref. Res.	Text. Entail.	Quest. Rewrit.	Cause Eff. Class.	Dialog Act Recog.	Ans. Class.	Keyword Tag.	Data to Text	Word Analogy	Overlap Extr.	Grammar Corr.	Avg. (↑)
Llama3.2 3B	Full FT	40.20	55.33	58.80	67.60	**70.52**	62.38	**68.13**	59.60	**52.08**	39.50	66.35	**88.68**	60.76
	Upcyc.	**41.25**	57.54	62.28	67.97	68.82	66.14	67.23	63.61	51.21	46.17	62.05	87.86	61.84
	SIMoE	41.14	**57.67**	**63.17**	**68.08**	69.54	**68.31**	67.59	**67.87**	51.40	**48.50**	**68.27**	87.64	**63.26**
Llama3 8B	Full FT	41.35	57.20	64.46	67.35	71.05	**73.17**	67.14	66.58	**53.04**	52.71	66.93	**87.82**	64.07
	Upcyc.	41.95	62.24	64.45	68.49	73.40	69.06	66.93	66.61	52.30	**55.55**	72.05	87.59	65.05
	SIMoE	**43.04**	**64.37**	**66.49**	**68.86**	**76.40**	70.70	**68.92**	**67.79**	51.73	52.79	**72.06**	85.38	**65.71**

Table 1: Performance on the SNI benchmark evaluated across 12 unseen task categories, measured in ROUGE-L.

Method	MMLU	PopQA	Truthful QA	BBH	DROP	MATH	GSM8K	Human Eval	Human Eval+	IFEval	Alpaca Eval 2	Safety	Avg. (↑)
Tülu v2 8B SFT	61.8	23.3	49.4	57.1	**61.7**	14.0	60.4	66.9	63.1	42.3	8.9	70.7	48.3
RLHFlow v2 SFT	65.8	29.7	**56.0**	69.3	57.2	**35.7**	**81.6**	86.2	80.9	52.7	**13.6**	43.5	56.0
MAmmoTH2 8B	63.6	20.8	42.7	63.4	43.8	30.5	63.7	72.8	66.4	34.9	6.5	47.8	46.4
Tülu v3 8B SFT	65.9	29.3	46.8	67.9	61.3	31.5	76.2	86.2	**81.4**	72.8	12.4	93.1	60.4
BTX	64.5	**30.9**	48.9	69.0	58.9	33.0	80.9	85.2	80.9	73.1	11.7	93.4	60.9
SIMoE (Ours)	**66.5**	28.7	51.6	**69.5**	57.5	30.1	81.3	**86.5**	81.3	**74.1**	12.4	**94.8**	**61.1**

Table 2: Comparison of detailed evaluation results on the Tülu 3 eval suite (Lambert et al., 2025) between SOTA instruction-tuned modes and ours. All models are instruction-tuned from the pre-trained Llama3.1 8B model.

module design.

Additionally, SIMoE exhibits strong compatibility with different pre-trained LLM architectures: As shown by our earlier experimental results in Tab. A10, SIMoE continues to outperform Full FT by a significant margin of 2.3% when switching the pre-trained backbone from Llama3 (Meta, 2024) to T5 (Raffel et al., 2020), confirming the high flexibility and versatility of our proposed method.

Safety and reliability Notably, SIMoE obtains excellent safety evaluation metrics, outperforming Tülu-v3-8B-SFT by 1.7% on Safety (Tab. 2), and a remarkable 10% on the *DoAnythingNow* (Shen et al., 2024) benchmark in particular (Tab. A9, detailed safety results across sub-datasets). The results demonstrate SIMoE's strong resilience against potentially malicious instructions and jailbreak attacks. The dual capability of combining performance improvements with enhanced safety underscores SIMoE's unique potential to mitigate the performance-safety trade-off observed in fine-tuned LLMs (Qi et al., 2024).

Training and inference cost In Fig. 1 and Tab. A8, we compare the training and inference compute cost between: (1) SIMoE with a maximum of $M = 8$ upcycled sparse interpolated experts at each linear layer, and (2) Sparse upcycling with 4 experts at each FNN block and Top-2 expert routing. Thanks to the proposed learnable, structured sparsity masks in combination with expert parameter sharing, our method significantly reduces model size during training, immediately providing a substantial reduction in peak GPU memory usage. Furthermore, by targeting a final sparsity of $\tau = 75\%$ in upcycled experts, our model achieves an smaller inference size, with around $\sim$30% fewer parameters compared to only the number of active parameters per forward-pass in a upcycled SMoE model.

Method	SNI (Llama3.2 3B) Params. (↓)	SNI (Llama3.2 3B) Avg. (↑)	SNI (Llama3 8B) Params. (↓)	SNI (Llama3 8B) Avg. (↑)	Tülu v3 (Llama3.1 8B) Params. (↓)	Tülu v3 (Llama3.1 8B) Avg. (↑)
Upcyc. / BTX	11.67	61.84	30.58	65.05	30.58	60.90
SIMoE (Ours)	**4.01**	**63.26**	**10.04**	**65.71**	**10.04**	**61.10**

Table 3: Overview of average performance and upcycled model capacity on the SNI and Tülu v3 benchmarks.

4.3 Additional Analysis

Importance of SIMoE components To assess the effectiveness of each component in SIMoE, we conducted an ablation study by systematically removing individual components, resulting in four distinct ablated variants: (a) Instead of **L**earning *where-to-**U**pcycle*, we adopt the common practice and upcycle only the FNN layers in the pre-trained LLM for instruction-tuning. (b) We exclude the **O**rthogonal **P**enalty on expert masks from our opti-

Model	L.U.	O.P.	S.C.	I.R.	ROUGE-L(↑)	Params.(↓) (B)
(a)		✓	✓	✓	61.52	4.01
(b)	✓		✓	✓	62.67	4.01
(c)		✓		✓	62.54	6.08
(d)	✓	✓	✓		62.51	4.01
SIMoE	✓	✓	✓	✓	**63.26**	**4.01**

Table 4: Ablation results on the SNI benchmark.

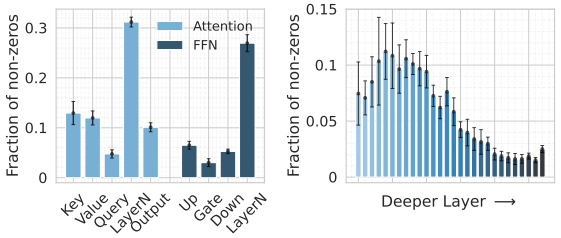

Figure 4: Upcycled model capacity after instruction-tuning (i.e., fraction of non-zero expert parameters) grouped by (*left*): layer types, and (*right*): layer depth. The bars represent the average over all experts, and the error bars reflect variation among different experts.

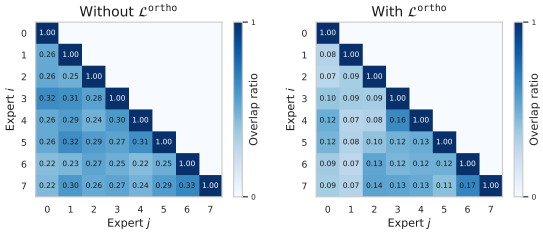

Figure 5: Expert overlap ratios of upcycled LLMs post-trained (*left*) without orthogonal penalty, and (*right*) with orthogonal penalty.

mization objective. (c) We do not impose **S**parsity **C**onstraints on the masks z, allowing them to be freely optimized as standard trainable parameters with values on the full real axis. Consequently, upcycling is almost always performed at all linear layers without pruning, hence also no **L.U**. (d) We replace **I**nstance-level **R**outing with token-level routing in the SIMoE module. Other training procedure and non-ablated components remain unchanged.

The variants are compared to the full SIMoE in Table 4. The results clearly demonstrate that the full SIMoE achieves the highest evaluation performance, quantitatively confirming the effectiveness of each proposed component. Notably, enforcing structured sparsity in upcycling not only enhances generalization performance compared to the non-sparse variant (c) – a result we attribute to its regularization effect – but also leads to a significant reduction in final model size – 4.01B vs 6.08B – which is practically advantageous.

Learned sparse upcycling patterns In Fig. 4, we visualize the distribution of non-zero experts in the upcycled LLM learned by SIMoE from instruction-tuning. Several key observations emerge. *First*, as shown in panel (*right*), upcycling primarily occurs in the shallow and intermediate Transformer layers, with significantly reduced activity in deeper layers. *Second*, panel (*left*) reveals that non-negligible upcycling manifests across all layer types, though with distinct intensity: layer normalization parameters exhibit the highest proportion of upcycled (non-zero) expert parameters, while the gate layer in the FNN demonstrates the lowest. Key, value, and output matrices in the attention block maintain a noticeably higher fraction of non-zero parameters than query weights, aligning with prior work that identified these matrices as crucial for knowledge injection and model editing (Meng et al., 2022; Gandikota et al., 2024).

Notably, the learned upcycling pattern by SIMoE, which achieves the best empirical performance, diverges substantially from manually prescribed strategies (e.g., upcycle FFN only), underscoring the critical advantage of data-driven approaches for determining *where-to-upcycle*.

Interpretable expert routing In Fig. 2, we visualize the average expert activation for different tasks. We notice that all experts exhibit some utilization across datasets, and hierarchical clustering of activation similarities reveals a clear dendrogram structure aligned with task and domain relationships. This demonstrates that SIMoE upcycling successfully induces both specialized experts and semantically meaningful routing behaviour.

Specialized experts and orthogonality In Fig. 5, we assess expert specialization through pairwise mask overlap ratios, defined as $\frac{|(\boldsymbol{z}_i \cap \boldsymbol{z}_j) \neq 0|}{|(\boldsymbol{z}_i \cup \boldsymbol{z}_j) \neq 0|}$. We observe that experts exhibit higher overlap ratios without the orthogonal penalty; Quantitively, having the penalty can lead to a noticeable 0.5% improvement on performance (Tab. 4) – Both observations validate its effectiveness.

In Fig. 5 (*right*), we observe that experts generally have low overlaps – sharing a small amount of parameters, though domain-similar experts (according to grouping in Fig. 2) exhibit marginally higher overlaps – for instances, maths- and code-domain experts {2,6,7}; general- and safety-domain experts {3,4}. The results demonstrate that SIMoE is capable of identifying a balanced shared and expert-specific parameter partitions, enabling nuanced specialization while maintaining strong synergies between distinct experts.

Hyperparameter sensitivity We conduct a comprehensive analysis of the sensitivity of SIMoE to

Orthogonality β	0	$5e^{-6}$	$\mathbf{5e^{-5}}$	$5e^{-4}$
Avg. ROUGE-L (↑)	65.56	65.71	**65.77**	65.31
Approx. Expert Overlap %	25	11	7	2

Sparsity τ	0	0.5	**0.75**	0.9
Avg. ROUGE-L (↑)	65.37	65.31	**65.71**	65.52

Table 5: Impact of key hyperparameters, τ and β, on the performance of SIMoE.

key hyperparameters, specifically the sparsity constraint τ and the orthogonality penalty coefficient β. During our experiments, we vary β through the values $\{0, 5e^{-6}, 5e^{-5}, 5e^{-4}\}$, and τ through $\{0, 0.5, 0.75, 0.9\}$, while keeping other hyperparameters constant at their optimal values. We evaluate the trained models on the SNI benchmark using the Llama3 8B model. The results are compared in Tab. 5.

We observe that SIMoE consistently outperforms the best baseline score of 65.05 across all evaluated hyperparameter settings, highlighting its robustness and reliability. Optimal performance is achieved with $\beta \in [5e^{-6}, 5e^{-5}]$ and a τ of 0.75. These settings yield the best empirical results, suggesting that they strike a balance between expert specialization and knowledge transfer. We hypothesize that extreme values of β lead to suboptimal outcomes due to excessive or minimal overlap among experts, which either impedes specialization or limits combinatorial generalization capabilities. Similarly, both low and high extremes of τ result in performance degradation, either through parameter redundancy or excessive sparsity, which constrains model capacity.

5 Conclusion

In this paper, we introduced Sparse Interpolated Mixture-of-Experts (SIMoE) for upcycling dense pre-trained LLMs into SMoEs within a single instruction-tuning stage. By addressing the dual challenges of *where-to-upcycle* and expert specialization-cooperation trade-offs, SIMoE automates the discovery of structurally sparse experts through sparsity-constrained optimization while promoting synergistic yet specialized experts parameter partitions via unique architectural design combined with a orthogonal penalty. Our methods demonstrate empirical superiority and enhanced memory savings compared to existing upcycling instruction-tuning methods, showcasing the efficacy of algorithm-model co-design in unlocking the full potential of upcycling instruction-tuning.

Limitations

While SIMoE shows promising results, there are two key limitations that warrant attention: (1) Our study is focused solely on NLP tasks, leaving its applicability to multimodal settings (e.g., vision-language models) untested, which is an important area for expanding the framework's impact; (2) We observe task interference in upcycling baselines and SIMoE, which can negatively affect generalization performance and sometimes cause the MoE models to slightly underperform compared to dense baselines. Future improvements could focus on addressing both challenges.

References

Shengzhuang Chen, Jihoon Tack, Yunqiao Yang, Yee Whye Teh, Jonathan Richard Schwarz, and Ying Wei. 2024. Unleashing the power of meta-tuning for few-shot generalization through sparse interpolated experts. *arXiv preprint arXiv:2403.08477*.

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. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *Preprint*, arXiv:2401.06066.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. 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)*, pages 7085–7095, Dublin, Ireland. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam M. Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23:120:1–120:39.

Jose Gallego-Posada, Juan Ramirez, Akram Erraqabi, Yoshua Bengio, and Simon Lacoste-Julien. 2022. Controlled sparsity via constrained optimization or: How i learned to stop tuning penalties and love constraints. In *Thirty-Sixth Conference on Neural Information Processing Systems*.

Rohit Gandikota, Hadas Orgad, Yonatan Belinkov, Joanna Materzyńska, and David Bau. 2024. Unified concept editing in diffusion models. *Preprint*, arXiv:2308.14761.

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

Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large language models for software engineering: A systematic literature review. *ACM Trans. Softw. Eng. Methodol.*, 33(8).

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. 2024. Harder task needs more experts: Dynamic routing in MoE models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12883–12895, Bangkok, Thailand. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Wangyi Jiang, Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. 2025. Improved sparse upcycling for instruction tuning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9485–9498, Abu Dhabi, UAE. Association for Computational Linguistics.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2023. Sparse upcycling: Training mixture-of-experts from dense checkpoints. In *The Eleventh International Conference on Learning Representations*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. 2025. Tulu 3: Pushing frontiers in open language model post-training. *Preprint*, arXiv:2411.15124.

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

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. 2022. Branch-train-merge: Embarrassingly parallel training of expert language models. *Preprint*, arXiv:2208.03306.

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

Zeming Lin, Halil Akin, Roshan Rao, Brian L. Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. 2022. Evolutionary-scale prediction of atomic level protein structure with a language model. *bioRxiv*.

Tianlin Liu, Joan Puigcerver, and Mathieu Blondel. 2023. Sparsity-constrained optimal transport. In *The Eleventh International Conference on Learning Representations*.

S. Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through $l_0 regularization$. In *International Conference on Learning Representations*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*, volume 35, pages 17359–17372. Curran Associates, Inc.

Llama Team Meta. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Evan Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. 2025. OLMoe: Open mixture-of-experts language models. In *The Thirteenth International Conference on Learning Representations*.

Mohammed Muqeeth, Haokun Liu, and Colin Raffel. 2023. Soft merging of experts with adaptive routing. *ArXiv*, abs/2306.03745.

Basil Mustafa, Carlos Riquelme Ruiz, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. 2022. Multimodal contrastive learning with LIMoe: the language-image mixture of experts. In *Advances in Neural Information Processing Systems*.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. 2024. LISA: Layerwise importance sampling for memory-efficient large language model fine-tuning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. 2023. From sparse to soft mixtures of experts. *ArXiv*, abs/2308.00951.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2024. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *The Twelfth International Conference on Learning Representations*.

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

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. 2021. Scaling vision with sparse mixture of experts. In *Neural Information Processing Systems*.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. 2021. Hash layers for large sparse models. In *Neural Information Processing Systems*.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, Alhussein Fawzi, Josh Grochow, Andrea Lodi, Jean-Baptiste Mouret, Talia Ringer, and Tao Yu. 2023. Mathematical discoveries from program search with large language models. *Nature*, 625:468 – 475.

Jonathan Schwarz, Siddhant Jayakumar, Razvan Pascanu, Peter E Latham, and Yee Teh. 2021. Powerpropagation: A sparsity inducing weight reparameterisation. *Advances in neural information processing systems*, 34:28889–28903.

Jonathan Schwarz and Yee Whye Teh. 2022. Meta-learning sparse compression networks. *Transactions on Machine Learning Research*.

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

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2024. "do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 1671–1685, New York, NY, USA. Association for Computing Machinery.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Roziere, Jacob Kahn, Shang-Wen Li, Wen tau Yih, Jason E Weston, and Xian Li. 2024. Branch-train-mix: Mixing expert LLMs into a mixture-of-experts LLM. In *First Conference on Language Modeling*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Johannes Von Oswald, Dominic Zhao, Seijin Kobayashi, Simon Schug, Massimo Caccia, Nicolas Zucchet, and João Sacramento. 2021. Learning where to learn: Gradient sparsity in meta and continual learning. *Advances in Neural Information Processing Systems*, 34:5250–5263.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Qizhen Zhang, Nikolas Gritsch, Dwaraknath Gnaneshwar, Simon Guo, David Cairuz, Bharat Venkitesh, Jakob Nicolaus Foerster, Phil Blunsom, Sebastian Ruder, Ahmet Üstün, and Acyr Locatelli. 2024. BAM! just like that: Simple and efficient parameter upcycling for mixture of experts. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2023. Instruction tuning for large language models: A survey. *ArXiv*, abs/2308.10792.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Y Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing.

Caleb Ziems, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. 2024. Can large language models transform computational social science? *Computational Linguistics*, 50(1):237–291.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Jianfeng Gao, and Tuo Zhao. 2022. Taming sparsely activated transformer with stochastic experts. In *International Conference on Learning Representations*.

A Method Details

A.1 Differentiable Sparsity

Louizos et al. (2018) introduce the hard concrete distribution for modeling sparse gates $z \in [0, 1]$. Using the default hyper-parameters from (Louizos et al., 2018), $\gamma = -0.1, \zeta = 1.1, \beta = 2/3$, and a random variable $U \sim \text{Unif}(0, 1)$, the hard-concrete distribution models the gate z through:

$$s \sim q_\phi(s) = \text{Sigmoid}\left(\frac{1}{\beta}\log\left(\frac{\phi U}{1-U}\right)\right), \quad (4)$$

$$z = \min(1, \max(0, s)), \quad (5)$$

where $q_\phi(s)$ is known as the concrete distribution, and ϕ is the underlying parameter being optimized.

The stochastic nature of $\mathbf{z}$ results in a model that is itself stochastic. Therefore, both its L_0-norm and predictions are random quantities. However, as shown in (Gallego-Posada et al., 2022), z can be replaced with its median $\bar{z}$:

$$\bar{s} = \text{Sigmoid}\left(\frac{\log\phi}{\beta}\right)(\zeta - \gamma) + \gamma), \quad (6)$$

$$\bar{z} = \min(1, \max(0, \bar{s})), \quad (7)$$

which is a deterministic function of ϕ, thus removing the stochastic nature in the gates for training and inference. The stretching and clamping allow the medians to attain values of exactly *0*, producing sparsity in our SIMoE experts when multiplied with expert parameters $\boldsymbol{\theta}^\delta$.

Furthermore, Louizos et al. (2018) show that the expected L_0-norm of mask z can be expressed in closed-form as:

$$\bar{L}_0(z) = P(z \neq 0) = 1 - Q_\phi(s \leq 0)$$
$$= \text{Sigmoid}\left(\log\phi - \beta\log\frac{-\gamma}{\zeta}\right), \quad (8)$$

where Q_ϕ is the CDF of q_ϕ. To this end, sparsity in z is enforced by optimizing ϕ to limit $\bar{L}_0(z)$, enabling end-to-end gradient-based optimization under the sparsity constraint in our framework.

A.2 SIMoE Training Objective Eqn. (3)

Recall our SIMoE module computes its parameters for inference at each linear layer as:

$$\boldsymbol{\theta}^{\texttt{SIMoE}} = \boldsymbol{\theta}^{\texttt{pre}} + \sum_{i=1}^{M} \alpha_i \cdot \boldsymbol{z}_i \odot \boldsymbol{\theta}^\delta, \quad (9)$$

where $\boldsymbol{\theta}^{\texttt{pre}}$ is the frozen pre-trained parameters; $\boldsymbol{\alpha} = h_\zeta(\boldsymbol{x})$ is the expert activation for input prompt $\boldsymbol{x}$ computed by the router network h_ζ; $\{\boldsymbol{z}_i\}_{i=1}^M$ and $\boldsymbol{\theta}^\delta$ are respectively the sets of distinct masks and shared parameters that jointly define the M experts.

During instruction-tuning, SIMoE aims to uncover *where-to-upcycle* through solving the following sparsity-constrained optimization problem:

$$\min_{\boldsymbol{\theta}^\delta, \boldsymbol{\zeta}, \boldsymbol{\phi}} \mathbb{E}_{\mathbf{y},\mathbf{x}}\left[\mathcal{L}^{\texttt{nll}}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta}^{\texttt{SIMoE}})\right] + \beta\mathcal{L}^{\texttt{ortho}}(\boldsymbol{Z}),$$
$$\text{s.t.} \quad 1 - L_0(\boldsymbol{Z}) \geq \tau, \quad (10)$$

where the goal is to find model parameters $\{\boldsymbol{\theta}^\delta, \zeta, \phi\}$ that minimize a weighted objective between the standard negative log-likelihood loss on the target output $\boldsymbol{y}$ and an orthogonal penalty on the expert masks, while satisfying a constraint that requires mask sparsity, $1 - L_0(\boldsymbol{Z})$, to be at least τ.

In practice, to enable gradient-based optimization, we solve the Lagrangian associated with the sparsity-constrained optimization problem above. Let $\lambda \geq 0$ be the Lagrangian multiplier associated with the sparsity constraint, the min-max Lagrangian problem is then:

$$\min_{\boldsymbol{\theta}^\delta, \boldsymbol{\zeta}, \boldsymbol{\phi}} \max_{\lambda \geq \mathbf{0}} \mathbb{E}_{\mathbf{y},\mathbf{x}}\left[\mathcal{L}^{\texttt{nll}}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta}^{\texttt{SIMoE}})\right] + \beta\mathcal{L}^{\texttt{ortho}}(\boldsymbol{Z})$$
$$+ \lambda(\tau - (1 - L_0(\boldsymbol{Z}))), \quad (11)$$

which mirrors our training objective in Eqn. (3) and we restate it here for completeness.

We employ simultaneous gradient descent on model parameters $\{\boldsymbol{\theta}^\delta, \zeta, \phi\}$ and projected (to $\mathbb{R}^+$) gradient ascent on λ, adjusting the strength of the sparse penalty in Eqn. (3) dynamically throughout

the optimization (Gallego-Posada et al., 2022). The gradient updates in each optimization step are [3]:

$$[\boldsymbol{\theta}^{\boldsymbol{\delta}}, \boldsymbol{\zeta}, \boldsymbol{\phi}] \leftarrow$$
$$[\boldsymbol{\theta}^{\boldsymbol{\delta}}, \boldsymbol{\zeta}, \boldsymbol{\phi}] - \nabla_{[\boldsymbol{\theta}^{\boldsymbol{\delta}}, \boldsymbol{\zeta}, \boldsymbol{\phi}]} \Big[\mathbb{E}_{\mathbf{y},\mathbf{x}} \big[\mathcal{L}^{\texttt{nll}}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta}^{\texttt{SIMoE}}) + \beta \mathcal{L}^{\texttt{ortho}}(\boldsymbol{Z}) - \lambda(1 - L_0(\boldsymbol{Z})) \big] \Big], \quad (12)$$
$$[\lambda] \leftarrow \max \Big(0, [\lambda] + (\tau - (1 - L_0(\boldsymbol{Z}))) \Big). \quad (13)$$

The sparsity $(1 - L_0(\boldsymbol{Z}))$ in Eqn. (12) remains non-differentiable w.r.t. ϕ, the parameters of the hard-concrete distribution producing masks $\boldsymbol{Z}$. Thus, we compute the expected sparsity $(1 - \bar{L}_0(\boldsymbol{Z}))$, which is differentiable w.r.t. ϕ (see Appendix A.1).

Overall, λ continues to increase, progressively emphasizing the sparsity penalty $(1 - \bar{L}_0(\boldsymbol{Z}))$ until the constraint is satisfied during optimization.

B Experiment Details

B.1 Implementation

For our method, we set the maximum number of experts to $M = 8$ and the target sparsity constraint to $\tau = 75\%$ by default, unless stated otherwise. We initialize the trainable parameters so that the SIMoE model $\theta^{\texttt{SIMoE}}$ in Eqn. (9) behaves identically the pre-trained LLM $\theta^{\texttt{pre}}$ at the start of instruction-tuning. Specifically, we initialize the shared expert parameters, θ^{δ}, to zero, and the mask parameters ϕ from a Gaussian distribution with a mean that results in an initial expected sparsity of 0.05 in each mask, i.e., $P(z = 0) = Q_{\phi}(s \leq 0) = 0.05, \forall z \in \boldsymbol{Z}$. Thanks to our unique method design (Section 3), we are able to consider upcycling (attaching SIMoE layer) at every linear layer in the pre-trained LLM. This enables us to optimize *where-to-upcycle* globally with minimal manual intervention while still maintaining compute feasibility. The detailed training hyperparameters for our method can be found in Tab. 6 below.

For the baseline sparse upcycling and BTX, following the original implementation described in (Sukhbaatar et al., 2024), we upcycle FNN experts in the pre-trained dense LLM, and use a default Top-2 routing function. We use a load-balancing loss with coefficient $1e-2$ (Shazeer et al., 2017; Fedus et al., 2021) and a router-z loss with strength $1e-3$ to stabilize training (Dai et al., 2022). We note that both methods can eventually produce an SMoE model with the same architecture and model size, given the same number of upcycled experts and the same seed LLM as fixed hyperparameters.

On the Super-NaturalInstruction benchmark, we set the number of upcycling experts to 4, which already yields a model size approximately equals to 4 times that of the original LLM. For Tulu-v3 experiment with BTX (Sukhbaatar et al., 2024), we initialize experts with FNN from independently trained domain experts on pre-defined domains in the Tulu-v3 training data mixture, including: math, code, safety, instruction following, multilingual, knowledge recall and general. We also include the pre-trained checkpoint as an additional expert as done in Sukhbaatar et al. (2024).

[3] We use vanilla SGD for illustration and omit the learning rates for clarity

Hyperparameter	SNI	Tülu-v3
Learning rate	2e-5	2e-5
Learning rate scheduler	constant	linear
Batch size	16	128
Optimizer	Adam	Adam
Sparsity (τ)	0.75	0.75
Orthogonality (β)	5e-6	5e-6
Experts (M)	8	8

Table 6: Training hyperparameters used for SIMoE in SNI and Tülu-v3 experiments.

C Additional Results

C.1 Full Granular vs. Structured Sparsity Masks

We explore the impact of mask granularity on the trade-off between model performance and computational efficiency, focusing on SIMoE models.

Full granular masks, while potentially enhancing model expressiveness by applying distinct masks to each parameter, present significant challenges. For models scaled to a billion parameters with M experts, these masks can demand up to M times the original model size, making them **less scalable** with larger models. Additionally, training with parameter-wise sparse, *stochastic* masks can lead to **optimization difficulties**, such as training instability and an increased risk of overfitting.

To assess these effects, we conducted experiments with the Llama3.2 1B model on the SNI benchmark. As shown in Table 7, the results demonstrate that structured sparsity masks not only achieve superior performance but also offer better computational efficiency. This highlights the

advantages of adopting structured sparsity in large-scale models.

Mask Type	Rouge-L (↑)	Param.(↓)
Full granular	59.36	3.42B
Structured	**59.94**	**1.45B**

Table 7: Comparison of SIMoE with full granular masks and structured sparsity masks.

C.2 Activated Parameter Counts in Masked Models

Tab. 8 presents a detailed comparison of activated versus total parameter counts for SMoE (Upcyc/BTX) and our proposed SIMoE model. Please refer to Fig. 1 for a qualitative visualization of these results.

Method	3B Seed	8B Seed
Upcyc/BTX	7 / 11.67B	13.64 / 30.58B
SIMoE (Ours)	4.01 / 4.01B	10.4 / 10.4B

Table 8: Comparison of activated versus total parameter counts for SMoE and SIMoE models.

C.3 Detailed Tülu-v3 Safety Evaluation Results

Benchmark	Tulu 3 8B SFT	Ours
HarmBench	**98.4**	97.5
XSTest	90.4	90.4
WildGuardTest	99.2	**99.5**
Jailbreaktrigger	95.8	95.8
DoAnythingNow	88.3	**98.0**
WildjailbreakTest	86.7	**87.7**
Average (↑)	93.1	**94.8**

Table 9: Breakdown of safety scores by benchmark of ours compared with the open-source state-of-the-art, Tulu 3 8B SFT (Lambert et al., 2025).

Tab. 9 presents a comparison of safety scores between SIMoE and the Tulu 3 8B SFT (Lambert et al., 2025), across various benchmarks. While Tulu 3 8B SFT outperforms in HarmBench, our model maintains a competitive edge overall, with an average score of 94.8 compared to Tulu's 93.1. This highlights the potential of our approach in safety-critical evaluations.

Tasks	Tk-Instruct 3B	Ours
Average (↑)	56.72	59.21

Table 10: Performance of Tk-instruct (Wang et al., 2022) and our method on the SNI benchmark. Both models start from pre-trained T5-XL (Raffel et al., 2020).

C.4 Instruction-tuned T5-XL on Super-NaturalInstruction

C.5 Detailed Tülu-v3 and SNI Evaluation Results

We present detailed evaluation results, including parameter counts, comparing SIMoE with previous sparse upcycling methods in Tables 11 and 12 below.

Method	Params. (B) (↓)	Title Gen.	Coref. Res.	Text. Entail.	Quest. Rewrit.	Cause Eff. Class.	Dialog Act Recog.	Ans. Class.	Keyword Tag.	Data to Text	Word Analogy	Overlap Extr.	Grammar Corr.	Avg. (↑)
Upcycling (3B)	11.67	**41.25**	57.54	62.28	67.97	68.82	66.14	67.23	63.61	51.21	46.17	62.05	**87.86**	61.84
SIMoE (3B)	**4.01**	41.14	**57.67**	**63.17**	**68.08**	**69.54**	**68.31**	**67.59**	**67.87**	**51.40**	**48.50**	**68.27**	87.64	**63.26**
Upcycling (8B)	30.58	41.95	62.24	64.45	68.49	73.40	69.06	66.93	66.61	**52.30**	**55.55**	72.05	**87.59**	65.05
SIMoE (8B)	**10.04**	**43.04**	**64.37**	**66.49**	**68.86**	**76.40**	**70.70**	**68.92**	**67.79**	51.73	52.79	**72.06**	85.38	**65.71**

Table 11: Performance on the SNI benchmark evaluated across 12 unseen task categories, measured in ROUGE-L. The top and bottom sections respectively show results for the Llama3.2-3B and the Llama3-8B pre-trained models.

Method	Params. (B) (↓)	MMLU	PopQA	Truthful QA	BBH	DROP	MATH	GSM8K	Human Eval	Human Eval+	IFEval	Alpaca Eval 2	Safety	Avg. (↑)
BTX	30.58	64.5	**30.9**	48.9	69.0	**58.9**	**33.0**	80.9	85.2	80.9	73.1	11.7	93.4	60.9
SIMoE	**10.04**	**66.5**	28.7	**51.6**	**69.5**	57.5	30.1	**81.3**	**86.5**	**81.3**	**74.1**	**12.4**	**94.8**	**61.1**

Table 12: Overview of results on the Tülu 3 eval suite (Lambert et al., 2025). All models are instruction-tuned from the pre-trained Llama3.1 8B model.