TinyR1-32B-Preview: Boosting Accuracy with Branch-Merge Distillation

Anonymous ACL submission

Abstract

It is beneficial but challenging to reduce the size of Large Language Models (LLMs) while maintaining their performance. Existing methods, such as ordinary model distillation, often fail to achieve high accuracy. To address this limitation, we introduce our Branch-Merge distillation approach: First, a student model is *selectively distilled* to yield different experts with respective domain-specific knowledge from a large teacher model; then, we merge these distilled experts in order to build a generalized model with cross-domain knowledge. With our distillation approach, we create TinyR1-32B-Preview, which outperforms the original student across multiple benchmarks, including Mathematics (+5.5), Coding (+4.4) and Science (+2.9), and achieves comparable performance to DeepSeek-R1 on AIME 2024. Our Branch-Merge distillation provides a novel solution for creating smaller, high-performing LLMs with reduced computational cost and time.

1 Introduction

1.1 Motivation

Recently, DeepSeek-R1 has achieved strong results, and its R1-Distill models (DeepSeek-AI, 2025) show that small models distilled from large teachers can excel at reasoning. Smaller models are also easier to deploy and cheaper at inference. A key challenge is therefore to *shrink model size while maintaining performance*.

In practice, building a small yet capable model typically starts from a pretrained base model and goes through multiple stages, among which Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) are critical. While RL is widely used to improve reasoning, a strong SFT cold start is often essential to unlock initial capabilities and support further RL gains (Qian et al., 2025). To our knowledge, the most effective SFT strategy is multi-domain distillation from a larger teacher (Jiao et al., 2020; DeepSeek-AI, 2025; Team, 2025a; Muennighoff et al., 2025). However, naive mixed-data distillation requires careful domain selection and proportion tuning, which is time-consuming and error-prone (Guo et al., 2019; Ji et al., 2024). Moreover, optimizing many domains jointly can cause task interference and conflicting gradients, slowing or degrading learning (Yu et al., 2020; Jiang et al., 2024). As a result, distilled models may underperform on specialized tasks, even when those tasks appear in training.

To address these issues, we propose a **Branch-Merge** approach that integrates model merging into distillation. As shown in Figure 1 (A), it has two phases: (1) **Branch Phase**, where knowledge is selectively distilled from a unified large teacher model (*e.g.*, DeepSeek-R1 671B) into several specialized student models (*e.g.*, math, coding, science) via domain-specific SFT. (2) **Merge Phase**, where specialized models are combined into a single unified model, enabling cross-domain transfer while preserving specialized capabilities.

1.2 Contribution

- **Accuracy**: We can see from Figure 1 (B) that the Branch-Merge distillation approach significantly improves model accuracy. Our distilled Qwen-32B model Tiny-R1-32B-Preview surpasses DeepSeek-R1-Distill-Qwen-32B with about 5% more accuracy, and its math accuracy approaches that of the original R1 teacher, which traditional distillation methods have not yet achieved. We demonstrate the significant potential of SFT in model tuning, thereby providing a robust starting point for future optimization methods such as RL.

- **Simplicity & Low Cost**: Branch-Merge distillation approach significantly reduces the time and computational costs of the merging stage. Compared to traditional methods, we save 90% of

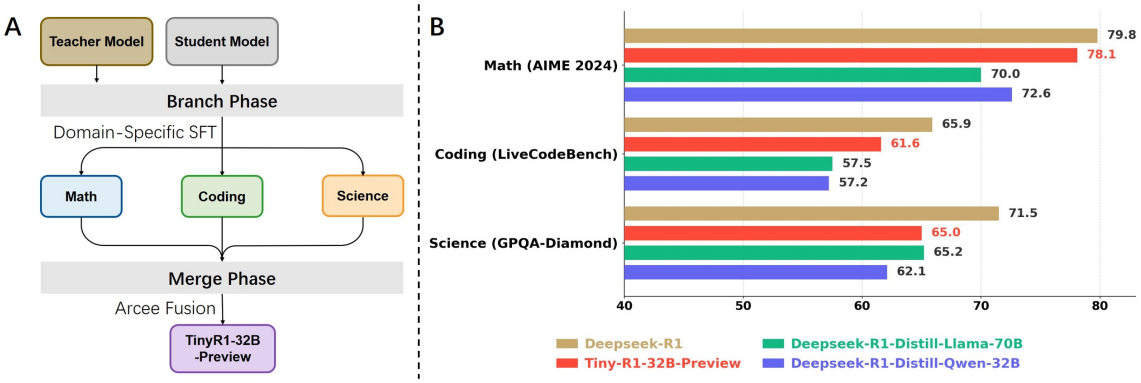

Figure 1: (A) A simplified diagram of our Branch-Merge distillation approach. (1) In the Branch phase, each copy of the Initial Model (backbone) is trained on knowledge from a different domain; (2) In the Merge phase, models are merged based on Arcee Fusion rules. (B) Performance Comparison of different LLM models (Mustar, 2025). TinyR1-32B-Preview outperforms distilled models of the same size in science, math, and coding and achieves comparable results to Deepseek R1. LiveCodeBench here refers to the 24.08-25.02 subset of full LiveCodeBench.

the time in the merging phase (0.5 hours with 4 H800 GPUs vs. 23 hours with 32 H800 GPUs for merged data retraining). The ideal reproduction cost for TinyR1-32B-Preview is 744 H800 GPU hours, approximately $1500 (excluding ablation experiments and parameter search).

- **Openness**: We stand on the shoulders of giants in the open-source community and aim to give back. We have released our model, and will release all data, training code, evaluation code, and logs so anyone can reproduce our results.

2 The Branch-Merge Distillation Approach

This section describes our branch-merge distillation approach (as shown in Figure 1 (A)), which consists of two phases: Branch and Merge. This two-phase distillation strategy directly addresses the issues of data selection and gradient conflict by decoupling training domains (Branch) and then reconciling them (Merge). Each phase is described in detail below.

2.1 The Branch Phase

In the Branch phase, we first constructed separate datasets for math, science, and coding. Then, we fine-tuned DeepSeek-R1-Distill-Qwen-32B on each dataset using SFT, resulting in three specialized expert models.

- **Math**: We sift 58k samples from 94k questions in NuminaMath1.5 (LI et al., 2024) with corresponding solutions from OpenR1 (Team, 2025a) trajectories. The selection is based on three aspects: *question_type*, *source*, and *correctness_math_verify*. Ultimately, we adopted a minimal dataset while maintaining comparable performance.

- **Coding**: The OpenThoughts (Team, 2025b) dataset is filtered to form 20k trajectories of coding solutions.

- **Science**: DeepSeek-R1 generates 1 CoT trajectory for each of the 8.6k seed examples (2.7k from the science and health science subsets of *data_ablation_full59k* in S1 (Muennighoff et al., 2025), 1.0k from S1k (Muennighoff et al., 2025), 4.9k from the science subset of OpenThoughts (Team, 2025b)), resulting in 8.6k CoT trajectories.

We apply SFT on DeepSeek-R1-Distill-Qwen-32B with the three datasets to obtain three specialized models.

2.2 The Merge Phase

In the Merge phase, we use Arcee Fusion (Goddard et al., 2024) to merge models from different domains.

The core idea of Arcee Fusion is as follows: when merging a new model of identical size and structure into an existing one, parameters pairs at corresponding positions are processed sequentially. For each pair, the decision of which parameter to keep is based on the magnitude of their difference. The greater the difference, the stronger the preference for adopting the parameter from the new model. In this merging process, we refer to the existing model as the **Left Model** and the new model

as the **Right Model**, with their parameters denoted by $\theta_L, \theta_R \in \mathbb{R}^M$, where M is the number of model parameters, respectively.

The merge process is divided into three steps:

1) **Computing Importance Score.**

We define an importance score vector $S_{IS} \in \mathbb{R}^M$ to quantify how salient each coordinate-wise update from the Right Model is, relative to the Left Model. Concretely, for each parameter tensor (e.g., a weight matrix) we flatten its entries into a length-N vector and treat it as a sample $X = \{x_1, \ldots, x_N\}$ from the Left Model, and similarly $Y = \{y_1, \ldots, y_N\}$ from the Right Model. We then normalize them into discrete distributions via a softmax transform:

$$\tilde{x}_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} + \epsilon, \quad \tilde{y}_i = \frac{e^{y_i}}{\sum_{j=1}^{N} e^{y_j}} + \epsilon,$$

where ϵ is a small constant with empirical default value 10^{-8}. We compute a tensor-level divergence $D_{KL}(\tilde{X} \parallel \tilde{Y})$ and broadcast it to obtain coordinate-wise saliency by scaling the parameter delta:

$$S_{IS} = D_{KL}(\tilde{X} \parallel \tilde{Y}) \cdot (\theta_L - \theta_R). \quad (1)$$

Intuitively, $D_{KL}(\tilde{X} \parallel \tilde{Y})$ captures how strongly the Right Model departs from the Left Model within the tensor, while $(\theta_L - \theta_R)$ localizes this departure at each coordinate.

2) **Calculating Dynamic Selection:**

We sorted S_{IS} and define these variables: $\mathcal{Q}_{\frac{1}{4}}, \mathcal{Q}_{Med}, \mathcal{Q}_{\frac{3}{4}}$, which denotes the 25%, 50%, 75% quantile value of the sorted S_{IS}, respectively. Additionally, we calculate the interquartile range $\mathcal{Q}_{IR}$ as $\mathcal{Q}_{\frac{3}{4}} - \mathcal{Q}_{\frac{1}{4}}$. Note that $\mathcal{Q}_{\frac{3}{4}} \geq \mathcal{Q}_{\frac{1}{4}}$ holds, ensuring $\mathcal{Q}_{IR} \geq 0$.

$\mathcal{S}_{THR} \in \mathbb{R}$ is then defined via the median $\mathcal{Q}_{Med}$ and interquartile range $\mathcal{Q}_{IR}$:

$$\mathcal{S}_{THR} = \mathcal{Q}_{Med} + \lambda \cdot \mathcal{Q}_{IR}, \quad (2)$$

where λ is a hyperparameter defined to balance coefficient.

Since $\mathcal{S}_{THR}$ is derived from IS statistics, it dynamically adapts to model parameters, ensuring an optimal update ratio.

3) **Selective Integration**:

The Merged Model is defined as $\theta_M \in \mathbb{R}^M$. We applied the following rule to merge θ_L and θ_R:

$$\theta_M^i = \theta_L^i + (\theta_R^i - \theta_L^i) \cdot \lceil \max(0, \mathcal{S}_{IS}^i - \mathcal{S}_{THR}) \rceil. \quad (3)$$

Only Right Model's parameters with an importance score above the threshold are retained; otherwise, Left Model's parameters are kept.

By focusing on the most significant changes, Arcee Fusion avoids over-updating and maintains model stability. The details of our merge sequence is in Section 3.2.

Why branch-merge helps. A data-mixture baseline jointly optimizes a single parameter vector over multiple domains and can suffer from cross-domain gradient interference, which empirically appears as a seesaw effect across domain scores. Our branch-merge strategy first decouples these conflicts by training domain experts independently, and then selectively integrates only high-importance expert updates via Arcee Fusion, approximating the union of expert capabilities rather than a compromise solution. We provide a more detailed theoretical discussion in Appendix C.

3 Experiment

We choose Deepseek-R1 (DeepSeek-AI, 2025) and its distilled DeepSeek-R1-Distill-Qwen-32B and DeepSeek-R1-Distill-Llama-70B as baselines. Additionally, we conducted a comprehensive ablation study. We compared TinyR1-32B-Preview with: (a) three domain expert models (Math Expert, Coding Expert, Science Expert) before merging; (b) a 'Data Mixture' model trained on a combined Math/Coding/Science dataset; and (c) variants of our model obtained via different merging sequences. Detailed experiment setup is discussed in Appendix A.

3.1 Main Results

We compare our TinyR1-32B-Preview model and other models in Table 1.

- In terms of accuracy, we outperform our backbone model, DeepSeek-R1-Distill-Qwen-32B (Math +5.5, Coding +4.4, Science +2.9), and generally surpass DeepSeek-R1-Distill-Llama-70B (Math +8.1, Coding +4.1, Science -0.2),

Model	Math (AIME 2024)	Coding (LiveCodeBench 24.08-25.02)	Science (GPQA-Diamond)
DeepSeek-R1-Distill-Qwen-32B†	72.6 (9.6k Tokens)	57.2 (10.1k Tokens)	62.1 (5.3k Tokens)
DeepSeek-R1-Distill-Llama-70B†	70.0	57.5	65.2
DeepSeek-R1†	79.8 (9.6k Tokens)	65.9 (10.4k Tokens)	71.5 (5.3k Tokens)
TinyR1-32B-Preview (Ours)	78.1 (11.8k Tokens)	61.6 (12.4k Tokens)	65.0 (8.6k Tokens)

Table 1: Performance comparison on benchmark datasets. All scores are reported as pass@1. Scores reported from DeepSeek-R1 paper (DeepSeek-AI, 2025) are noted with †. The number in parentheses represents the average output token length (including the chain of thought), obtained from our testing.

Model	Math (AIME 2024)	Coding (LiveCodeBench)	Science (GPQA-Diamond)	Merging Time (GPU Hours)
Math Expert	73.1	-	-	-
Coding Expert	-	63.4	-	-
Science Expert	-	-	64.5	-
Data Mixture	75.3	61.0	**65.7**	740 h
Merging: (Math & Coding) & Science	77.3	**63.8**	64.0	**4h**
Merging: (Math & Science) & Coding	**78.1**	61.6	65.0	**4h**
TinyR1-32B-Preview	**78.1**	61.6	65.0	**4h**

Table 2: Performance comparison between backbone experts, the data-mixture model, and merged model. All scores are reported as pass@1. LiveCodeBench here refers to the 24.08-25.02 subset of full LiveCodeBench.

approaching the performance of DeepSeek-R1 (Math -1.7, Coding -4.3, Science -6.5).

- In terms of inference cost, our model generates slightly more output tokens than DeepSeek-R1 (Math +23%, Coding +19%, Science +62%). Note that our model is smaller compared to DeepSeek-R1, making it more suitable for local deployment by users and small groups.

3.2 Ablation Study

As shown in Table 2, we made a comprehensive ablation study to explore if our merging distillation approach works.

Compared to the domain-specific experts, the Data Mixture model surpasses them in math and science but shows decreased performance in coding. This is a seesaw problem that traditional data mixture is difficult to solve (Zamir and Arbeláez, 2018; Liu and Yao, 2019; Radford and W., 2021). In comparison, our merged model improves performance in math and science domains while largely retaining the coding capability. We also compared two different model-merging sequences: (1) first merging Math with Coding, then merging with Science; (2) first merging Math with Science, then with Coding. The results show only minor performance differences between these sequences.

In addition, the average Merging time (GPU hours) of the current Data Mixture is 740 hours[1]. On the contrary, the average merging time of TinyR1-32B-preview is 4 hours. In summary, we only used 0.5% of the Data-Mixture computational overhead on merging models to surpass the effect of traditional data mixture methods. In addition to reducing computational overhead, model merging significantly accelerates our model release process by avoiding the delays introduced by mixed-data re-SFT on the development model. The model merging is a "free-lunch" approach, as it reduces costs and increases efficiency at the same time. The further studies are in Appendix D.

4 Conclusion

We introduce TinyR1-32B-Preview, a model using the Branch-Merge distillation approach to boost reasoning accuracy while preserving efficiency. We achieve significantly higher accuracy than our backbone model, DeepSeek-R1-Distill-Qwen-32B, and generally outperform DeepSeek-R1-Distill-Llama-70B while approaching the performance of DeepSeek-R1. Moreover, its smaller parameter size makes it more efficient and better suited for local deployment by users and small groups. For future optimization methods like RL, our model provides a robust starting point.

[1]Note that the merging time is the SFT experiment time, excluding the SFT time for single experts and the downstream evaluation time of the Data Mixture model

Limitations

As a small-scale research project, we have the following limitations:

- **Incomplete Benchmark Evaluations** – Our report does not include many reasoning benchmarks such as Codeforces rating and SWE-bench.
- **Unfinished Future Works** - There remains several promising directions for extending the training framework, such as enhancing its instruction-following (IF) and safety capabilities. Additionally, our approach can be integrated with reinforcement learning (RL) to further improve training outcomes, and we look forward to validating this potential in future studies.

References

Gregory Benton, Wesley Maddox, Sanae Lotfi, and Andrew Gordon Gordon Wilson. 2021. Loss surface simplexes for mode connecting volumes and fast ensembling. In *International Conference on Machine Learning*, pages 769–779. PMLR.

Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *Preprint*, arXiv:2401.01335.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. 2022. Fusing finetuned models for better pretraining. *arXiv preprint arXiv:2204.03044*.

Wojciech Marian Czarnecki, Simon Osindero, Razvan Pascanu, and Max Jaderberg. 2019. A deep neural network's loss surface contains every low-dimensional pattern. *arXiv preprint arXiv:1912.07559*.

Nico Daheim, Thomas Möllenhoff, Edoardo Maria Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. 2023. Model merging by uncertainty-based gradient matching. *arXiv preprint arXiv:2310.12808*.

MohammadReza Davari and Eugene Belilovsky. 2024. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision*, pages 270–287. Springer.

J. Dean and et al. 2018. Scaling neural networks with efficient memory and batching. In *ICML*.

Pala Tej Deep, Rishabh Bhardwaj, and Soujanya Poria. 2024. Della-merging: Reducing interference in model merging through magnitude-based sampling. *arXiv preprint arXiv:2406.11617*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. 2018. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR.

Gintare Karolina Dziugaite and Daniel M Roy. 2017. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. 2021. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online. Association for Computational Linguistics.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. 2020. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. 2019. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*.

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

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's MergeKit: A toolkit for merging large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language*

Processing: Industry Track, pages 477–485, Miami, Florida, US. Association for Computational Linguistics.

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2019. AutoSeM: Automatic task selection and mixing in multi-task learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3520–3531, Minneapolis, Minnesota. Association for Computational Linguistics.

M. R. Henry, A. Y. Shi, and et al. 2019. Memory-efficient batching for neural networks. In *NeurIPS*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

Sepp Hochreiter and Jürgen Schmidhuber. 1994. Simplifying neural nets by discovering flat minima. *Advances in neural information processing systems*, 7.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Flat minima. *Neural computation*, 9(1):1–42.

Chengming Hu, Xuan Li, Dan Liu, Haolun Wu, Xi Chen, Ju Wang, and Xue Liu. 2023. Teacher-student architecture for knowledge distillation: A survey. *Preprint*, arXiv:2308.04268.

Yukun Huang, Yanda Chen, Zhou Yu, and Kathleen McKeown. 2022. In-context learning distillation: Transferring few-shot learning ability of pre-trained language models. *Preprint*, arXiv:2212.10670.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022a. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

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

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Qiu, Boxun Li, and Yaodong Yang. 2024. Pku-saferlhf: Towards multi-level safety alignment for llms with human preference. *Preprint*, arXiv:2406.15513.

Li Jiang, Yusen Wu, Junwu Xiong, Jingqing Ruan, Yichuan Ding, Qingpei Guo, Zujie Wen, Jun Zhou, and Xiaotie Deng. 2024. Hummer: Towards limited competitive preference dataset. *Preprint*, arXiv:2405.11647.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. *Preprint*, arXiv:1909.10351.

Jean Kaddour, Linqing Liu, Ricardo Silva, and Matt J Kusner. 2022. When do flat minima optimizers work? *Advances in Neural Information Processing Systems*, 35:16577–16595.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Rohith Kuditipudi, Xiang Wang, Holden Lee, Yi Zhang, Zhiyuan Li, Wei Hu, Rong Ge, and Sanjeev Arora. 2019. Explaining landscape connectivity of low-cost solutions for multilayer nets. *Advances in neural information processing systems*, 32.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-1.5](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).

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

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, Wenhu Chen, and Xifeng Yan. 2022b. Explanations from large language models make small reasoners better. *Preprint*, arXiv:2210.06726.

L. Liu and A. Yao. 2019. Meta-learning for low-shot neural architecture search. In *ICLR*.

Ekdeep Singh Lubana, Eric J Bigelow, Robert P Dick, David Krueger, and Hidenori Tanaka. 2023. Mechanistic mode connectivity. In *International Conference on Machine Learning*, pages 22965–23004. PMLR.

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

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

Victor Mustar. 2025. Tinyr1-32b-preview, the new local king? Accessed: 2025-03-05.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2020. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33:512–523.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Jielin Qiu, Zhiwei Liu, Haolin Chen, Shirley Kokane, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. 2025. Userrl: Training interactive user-centric agent via reinforcement learning. *Preprint*, arXiv:2509.19736.

A. Radford and J. W. 2021. Improving generalization via scalable learning and domain adaptation. In *NeurIPS*.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

K. Tay, H. Dehghani, L. R. Jones, S. J. Koyejo, and A. S. S. Ahmed. 2020. Efficient transformers: A survey. *ACM Computing Surveys (CSUR)*, 53(4):1–38.

Open R1 Team. 2025a. Open r1. `https://huggingface.co/open-r1`.

OpenThoughts Team. 2025b. Open Thoughts. https://open-thoughts.ai.

Fanqi Wan, Longguang Zhong, Ziyi Yang, Ruijun Chen, and Xiaojun Quan. 2024. Fusechat: Knowledge fusion of chat models. *arXiv preprint arXiv:2408.07990*.

Chenyang Wang, Liang Wen, Shousheng Jia, Xiangzheng Zhang, and Liang Xu. 2025. Light-if: Endowing llms with generalizable reasoning via preview and self-checking for complex instruction following. *arXiv preprint arXiv:2508.03178*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. *Preprint*, arXiv:2212.10560.

Mitchell Wortsman, Maxwell C Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. 2021. Learning neural network subspaces. In *International Conference on Machine Learning*, pages 11217–11227. PMLR.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and 1 others. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *Preprint*, arXiv:2304.12244.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *Preprint*, arXiv:2402.13116.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115.

Chuanpeng Yang, Wang Lu, Yao Zhu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. 2024. Survey on knowledge distillation for large language models: Methods, evaluation, and application. *Preprint*, arXiv:2407.01885.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Preprint*, arXiv:2001.06782.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *Preprint*, arXiv:2401.10020.

L. A. Zamir and E. Arbeláez. 2018. Taskonomy: Disentangling task transfer learning. In *CVPR*, pages 2647–2656.

Dan Zhang, Ziniu Hu, Sining Zhoubian, Zhengxiao Du, Kaiyu Yang, Zihan Wang, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Sciinstruct: a self-reflective instruction annotated dataset for training scientific language models. *Preprint*, arXiv:2401.07950.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Haosheng Zou, Xiaowei Lv, Shousheng Jia, and Xiangzheng Zhang. 2024. 360-llama-factory.

A Experiment Setup Details

A.1 Training Details

We employ DeepSeek-R1-Distill-Qwen-32B as our backbone model. Leveraging the 360-Llama-Factory (Zou et al., 2024) training framework, we develop three domain-specific expert models applying 16384 sequence length with constructed Math, Coding, and Science datasets.

- Math Expert: The math expert model is trained with 5 epochs, batch size 96, and the learning rate is set to constant 1e-5.
- Science Expert: The science expert model is trained with 5 epochs, batch size 32 with the neat packing mechanism (Tay et al., 2020; Henry et al., 2019; Dean and et al., 2018), and the learning rate is set to cosine 1e-5.
- Coding Expert: The coding expert model is trained with 15 epochs, batch size 96 with the neat packing mechanism, and the learning rate is set to constant 1e-5.

We merged the models trained separately in three fields into a single model. We use the Arcee merging (Goddard et al., 2024) method with the θ=1.5 and threshold THR=0.5. We will compare different model merging methods in Section 3.2.

A.2 Evaluation Details

For evaluation, we compare the performance on three benchmark datasets: AIME 2024 for Math, LiveCodeBench (24.08-25.02) for Coding, and GPQA-Diamond for Science. The accuracy is calculated as an average pass@1 across 16, 4, and 4 independent trials for these benchmarks, respectively. Meanwhile, we did not use a greedy way to evaluate the model due to its long-COT output, we set the max tokens to 32768 and evaluated the models with Temperature=0.6 and Top-p=0.95 as recommended in DeepSeek-R1 (DeepSeek-AI, 2025). We tried various open-source frameworks for the evaluation on livecodebench and ultimately selected the evaluation code from FuseAI (Wan et al., 2024) utilizing the vLLM implementation, as it can reproduce the effects of the DeepSeek-R1 and its distilled models.

B Related Work

B.1 Model Distillation

Knowledge Distillation (KD) (Romero et al., 2015; Hinton et al., 2015) has been proposed to create cheaper strong models (Gou et al., 2021; Hu et al., 2023; Yang et al., 2024; Xu et al., 2024). Primarily, recognizing the disparities between proprietary and open-source LLMs, KD techniques have surged to bridge the performance gap between these models. Distillation methods can be categorized into two types: (1) the logits-based methods (Hinton et al., 2015), which transfer knowledge at the logits level, and (2) the feature-based methods (Romero et al., 2015), which transmit knowledge through intermediate features.

Compared to traditional knowledge distillation techniques (Gou et al., 2021), data augmentation (DA) (Feng et al., 2021) has emerged as an effective method for distilling knowledge in large language models (LLMs). In this approach, a small seed of knowledge is used to prompt LLMs, enabling them to generate additional data tailored to specific domains or skills (Taori et al., 2023). More recently, an API-based strategy has gained attention, where open-source LLMs serve as teachers to self-improve through self-distillation (Yuan et al., 2024; Chen et al., 2024). By applying a range of distillation techniques, this strategy effectively narrows the performance gap between closed-source and open-source models (Chiang et al., 2023; Xu et al., 2023). In this context, the method involves using only the outputs of the teacher model via an API. This strategy includes approaches such as In-Context Learning (Huang et al., 2022), Chain-of-Thought (Li et al., 2022b), and Instruction Following (Wang et al., 2023). In specialized fields, like science (Zhang et al., 2024), where domain-specific knowledge and accuracy are essential, distillation allows open-source models to significantly improve their performance by learning from proprietary models that are extensively trained and fine-tuned in these domains.

B.2 Model Merging

Recent advances in model merging have explored diverse strategies (Ilharco et al., 2022a; Yadav et al., 2023; Davari and Belilovsky, 2024; Deep et al., 2024) to combine neural network parameters while preserving or enhancing performance. Early approaches focused on linear interpolation techniques, such as weight averaging (Wortsman et al., 2022), where models finetuned from shared pretrained checkpoints are merged via arithmetic mean. While computationally efficient, these methods assume approximate parameter space alignment and often degrade when models exhibit divergent optimization trajectories (Frankle et al., 2020; Izmailov et al., 2018; Neyshabur et al., 2020; Fort et al., 2020; Wortsman et al., 2022; Choshen et al., 2022; Ilharco et al., 2022b).

Theoretical underpinnings for these methods derive from studies on loss landscape geometry (Ilharco et al., 2022a; Li et al., 2018; Garipov et al., 2018; Draxler et al., 2018; Kuditipudi et al., 2019; Fort et al., 2019; Czarnecki et al., 2019; Wortsman et al., 2021; Benton et al., 2021; Entezari et al., 2021; Li et al., 2022a; Lubana et al., 2023). Research on flat local minima (Kaddour et al., 2022; Wortsman et al., 2022; Keskar et al., 2016; Dziugaite and Roy, 2017) dating back from the 1990s (Hochreiter and Schmidhuber, 1994, 1997) suggests that averaged weights reside in flatter regions of the loss surface, correlating with improved out-of-distribution generalization. Further analyses (Daheim et al., 2023; Matena and Raffel, 2022) formalize model merging as identifying connected basins in parameter space, where interpolated solutions maintain low loss. Empirical validations, such as model soups (Wortsman et al., 2022), corroborate that aggregated weights often outperform individual models, particularly under distribution shifts.

C Theoretical Discussion

C.1 Why data mixture can underperform: gradient interference

Consider multi-domain training over $\mathcal{D} = \{\text{math}, \text{code}, \text{sci}\}$. A data-mixture baseline optimizes a single parameter vector θ to minimize the summed objective

$$\min_{\theta} \sum_{d \in \mathcal{D}} \mathbb{E}_{x \sim d}\big[\mathcal{L}_d(\theta; x)\big]. \tag{4}$$

Let $g_d(\theta) = \nabla_\theta \mathbb{E}_{x \sim d}[\mathcal{L}_d(\theta; x)]$ denote the gradient contributed by domain d. A well-studied issue in multi-task learning is *gradient interference*, where gradients from different domains can conflict, e.g.,

$$\langle g_i(\theta), g_j(\theta) \rangle < 0, \quad i \neq j, \tag{5}$$

which implies that a descent step that improves one domain may increase the loss of another (see, e.g., discussions around conflicting gradients in multi-task optimization). Intuitively, the optimizer is pushed toward a compromise region where gradients partially cancel, limiting progress toward any single domain-preferred solution. This provides a principled explanation for the empirically observed seesaw effect in mixture baselines, where gains on one benchmark can coincide with regressions on another.

C.2 Branch: decoupling optimization landscapes via domain experts

Our Branch phase trains domain-specific experts independently, producing expert parameters $\theta^{(d)}$ for each domain d from a shared initialization (e.g., the base model). This decouples optimization landscapes: each expert can move toward a domain-preferred region without being constrained by conflicting gradients from other domains.

Let $\Delta\theta^{(d)} = \theta^{(d)} - \theta_{\text{base}}$ denote the expert update relative to the base model. In the over-parameterized regime typical for LLMs, it is empirically common that effective task updates are sparse and/or concentrated on subsets of coordinates. A stylized assumption capturing this behavior is that for distinct domains $i \neq j$, the overlap of their dominant update supports is limited, or equivalently the normalized alignment is small:

$$\frac{\langle \Delta\theta^{(i)}, \Delta\theta^{(j)} \rangle}{\|\Delta\theta^{(i)}\| \, \|\Delta\theta^{(j)}\|} \approx 0. \tag{6}$$

Under such a regime, independent experts can encode domain-specific capabilities in partially distinct functional subspaces of the parameter space.

C.3 Merge: saliency-guided selective integration as a high-pass gate

The Merge phase aims to combine complementary expert capabilities while avoiding dilution from naïve weight averaging. We merge an existing model (Left) with a new expert (Right), denoting parameters by $\theta_L, \theta_R \in \mathbb{R}^M$, and the merged

model by $\theta_M \in \mathbb{R}^M$. Arcee Fusion produces a coordinate-wise importance score $S_{IS} \in \mathbb{R}^M$ and a data-adaptive threshold $S_{THR} \in \mathbb{R}$ (computed from robust statistics of S_{IS}, such as the median and interquartile range). The merge rule can be written in a gate form:

$$\theta_M^i = \theta_L^i + (\theta_R^i - \theta_L^i) \cdot \mathbb{I}\big(S_{IS}^i > S_{THR}\big) \quad (7)$$

Equation (7) makes explicit that Arcee Fusion performs *saliency-guided selective integration*: only coordinates with sufficiently high importance are copied from the Right Model; otherwise the Left coordinate is preserved.

This can be interpreted as a data-driven *high-pass filter* on expert updates. Large, salient parameter shifts—which more likely encode crucial domain-specific logic—are retained, while small or noisy deviations are suppressed. When combined with the Branch phase (which reduces interference and encourages partially distinct expert updates as in Eq. (6)), the gate in Eq. (7) approximates a union of expert capabilities across domains, rather than the compromise solution encouraged by mixture training in Eq. (4). This perspective aligns with our empirical results, where the merged model consistently improves over the data-mixture baseline on domain benchmarks.

D Further Study

D.1 Comparison of Different Merging Methods

A comparison with other model merging methods appears in Figure 2. We compare various merging methods on merging models trained from the math and science domains, and we find that Arcee achieves the highest scores on GPQA-Diamond. We found a similar method ranking on the AIME 2024 benchmark, and we omit the graph.

D.2 Generalization Beyond Target Benchmarks

Beyond our primary evaluation suite (AIME, GPQA, and LiveCodeBench), we further assess generalization on three auxiliary benchmarks: IFEval (instruction following), BBH (Suzgun et al., 2023), and HumanEval (Chen, 2021). Among them, IFEval is the clearest out-of-domain (OOD) probe for our setting, since our distillation data does not contain instruction-following supervision; thus any changes on IFEval reflect OOD generalization. In contrast, BBH and HumanEval overlap more with the broad reasoning/coding skills already exercised by our main benchmarks, so we view them as complementary generalization checks rather than strictly OOD.

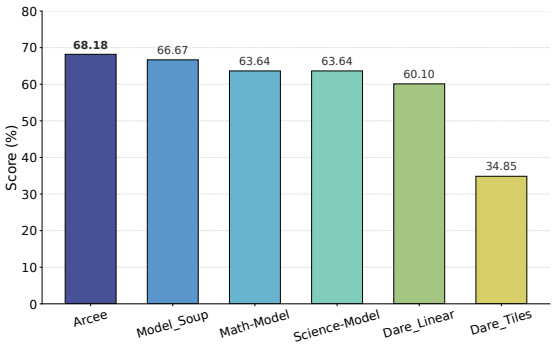

Figure 2: Performance Comparison of merged models on the GPQA-Diamond benchmark.

Model	IFEval	HumanEval	BBH
Base Model	75.5	83.5	47.0
Math Expert	63.1	85.5	48.2
Science Expert	60.4	85.5	61.1
Code Expert	62.2	85.5	63.5
TinyR1-32B-Preview	64.0	86.4	66.1

Table 3: Generalization results beyond our primary benchmarks (AIME, GPQA, LiveCodeBench). IFEval probes instruction-following and serves as an OOD check in our setting; BBH and HumanEval are complementary evaluations of general reasoning and coding.

As shown in Table 3, we observe two consistent phenomena on IFEval: (i) models after SFT (and subsequent merging) score lower than the base model (75.5), which is expected and consistent with prior observations (Zhou et al., 2023; Wang et al., 2025); and (ii) our merged model (TinyR1-32B-Preview) outperforms all individual SFT experts evaluated on IFEval, suggesting that our branch-merge distillation does not further erode—and may partially preserve—instruction-following behavior relative to the pre-merged experts. On HumanEval and BBH, expert models generally improve over the base model, and the merged model achieves the best HumanEval score (86.4) among the evaluated models, indicating that the branch-merge procedure can retain (or enhance) general coding performance.