Model merging

CS 5624: Natural Language Processing

Spring 2025

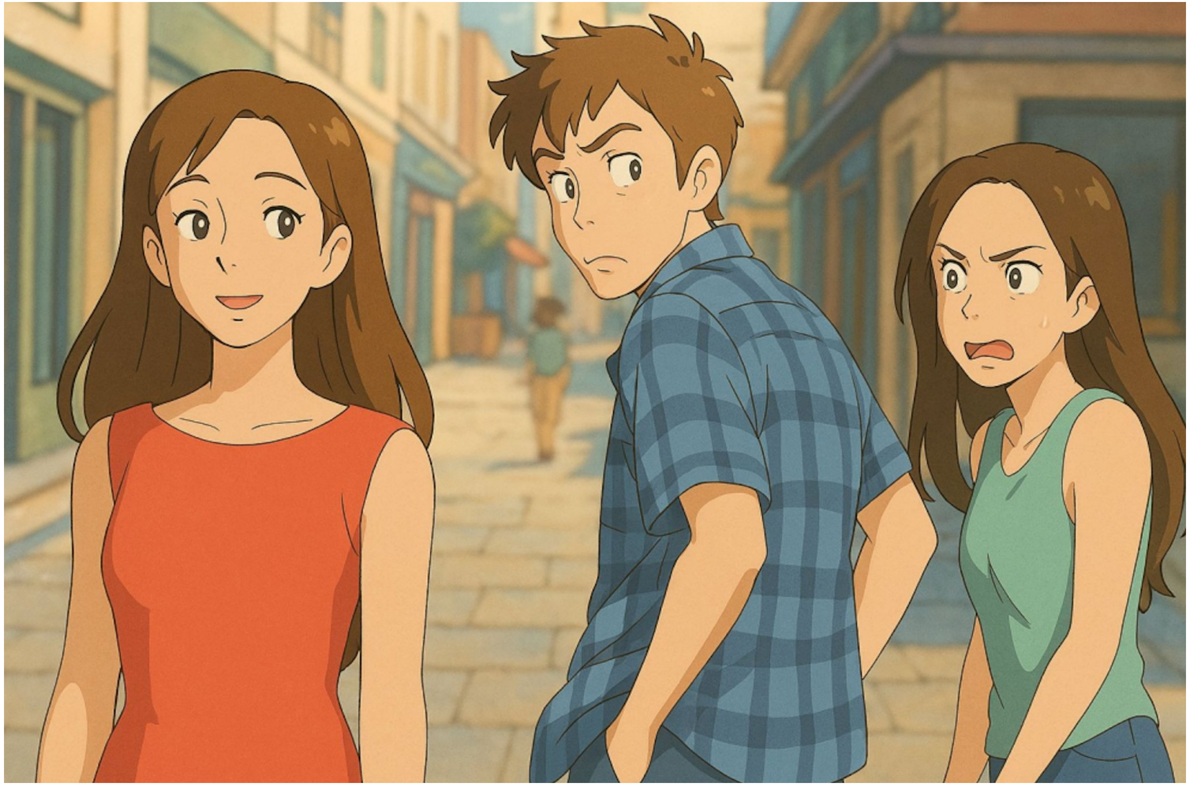
https://tuvllms.github.io/nlp-spring-2025

Tu Vu

VIRGINIA TECH.

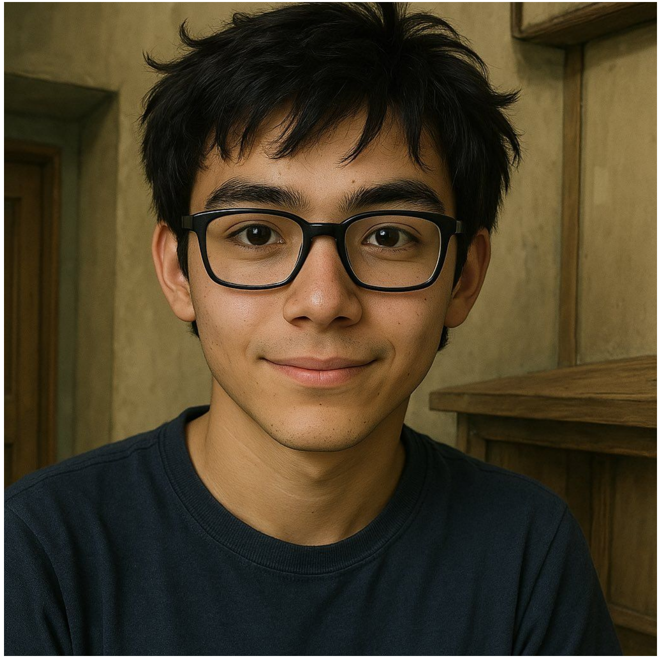
LLM News: GPT-4o's new image generation/editing tool

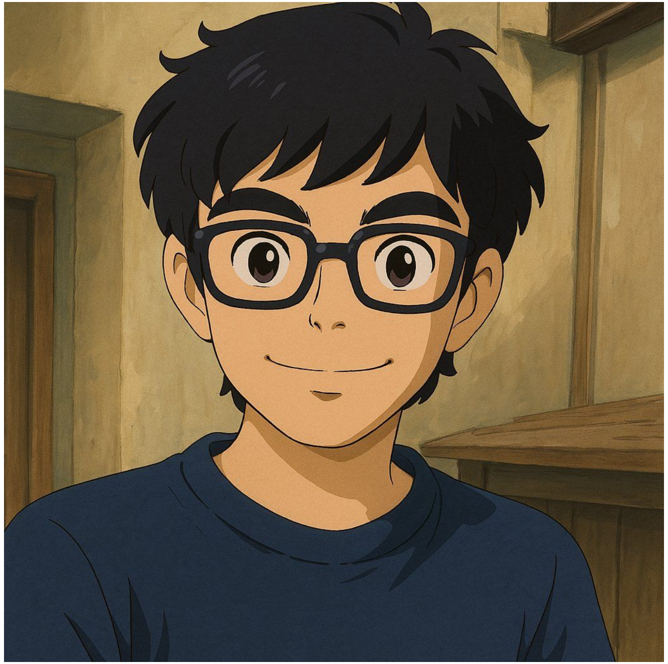
Ghibli images



@heyBarsee

LLM News: GPT-4o's new image generation/editing tool



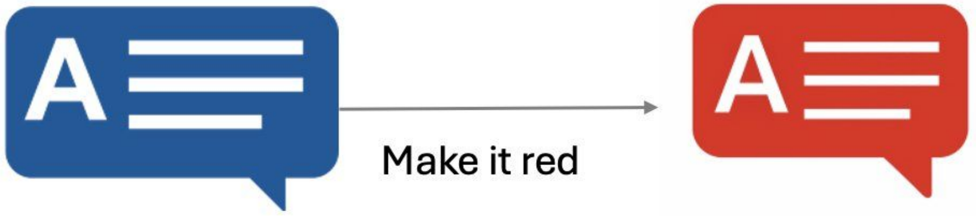


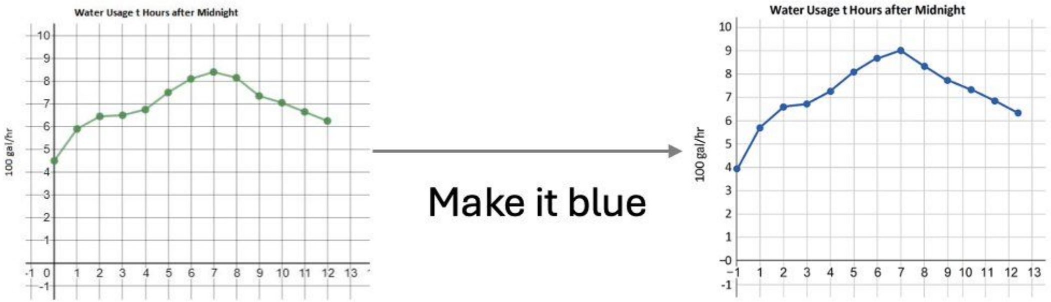
https://x.com/iScienceLuvr/status/1904842046244024540

LLM News: GPT-4o's new image generation/editing tool

https://x.com/natolambert/status/1905091174857085166

LLM News: GPT-4o's new image generation/editing tool

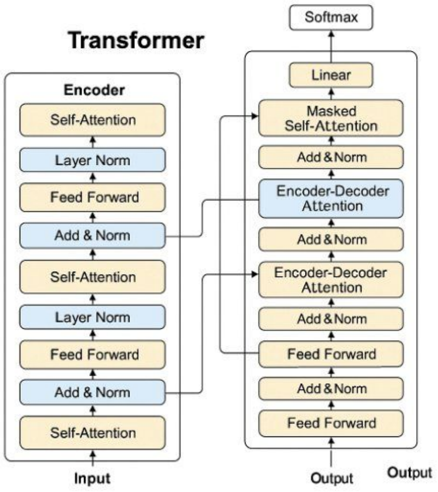




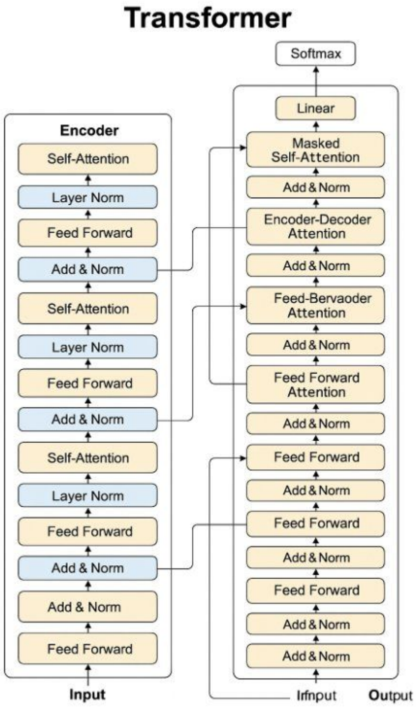
https://x.com/WenhuChen/status/1905042835650330815

LLM News: GPT-4o's new image generation/editing tool

Can you generate a figure about Transformer?



Make it deeper



https://x.com/WenhuChen/status/1905042835650330815

Ghibli podcast

- https://x.com/omooretweets/status/19050600511251621l1

EDITING MODELS WITH TASK ARITHMETIC

Gabriel Ilharco[*1] **Marco Tulio Ribeiro**[2] **Mitchell Wortsman**[1] **Suchin Gururangan**[1]
Ludwig Schmidt[1,3] **Hannaneh Hajishirzi**[1,3] **Ali Farhadi**[1]
[1]University of Washington [2]Microsoft Research [3]Allen Institute for AI

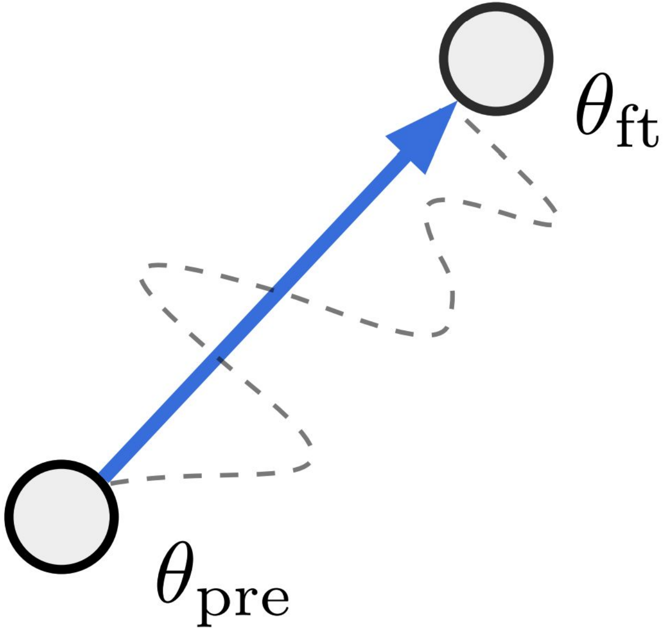
Why do we want to edit LLMs?

-

Why do we want to edit LLMs?

- improve performance on downstream tasks
- mitigate biases or unwanted behavior
- align models with human preferences
- update models with new information

The notion of task vectors



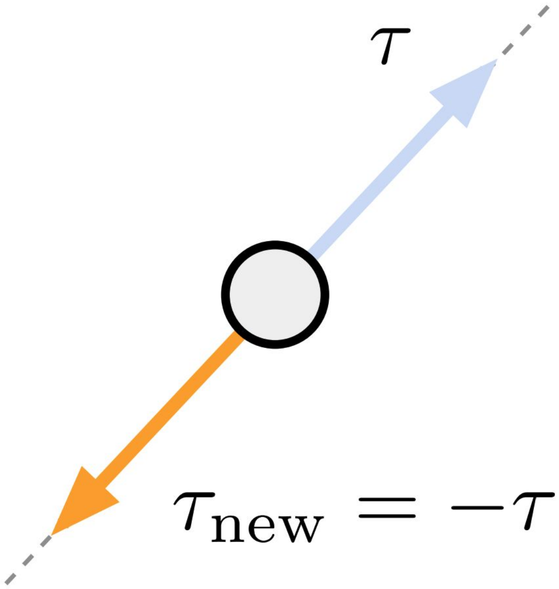
$$\tau = \theta_{\text{ft}} - \theta_{\text{pre}}$$

$$\theta_{\text{new}} = \theta + \tau$$

In practice, we have an optional scaling term λ

$$\theta_{\text{new}} = \theta + \lambda\tau$$

Forgetting via negation



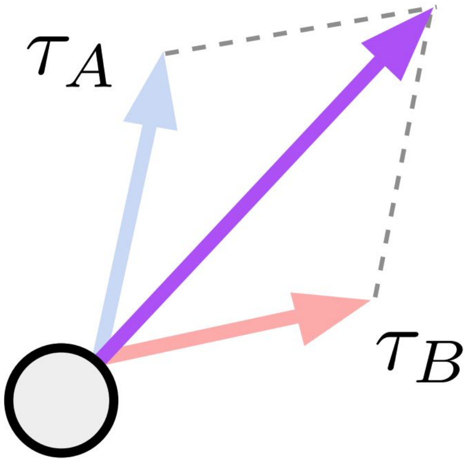
Example: making a language model produce less toxic content

$$\theta_{\text{new}} = \theta - \tau = \theta - (\theta_{ft} - \theta)$$

In practice, we have an optional scaling term λ

Learning via addition

$$\tau_{\text{new}} = \tau_A + \tau_B$$



$$\theta_{\text{new}} = \theta + \tau = \theta + (\tau_A + \tau_B)$$

$$= \theta + (\theta_A - \theta) + (\theta_B - \theta)$$

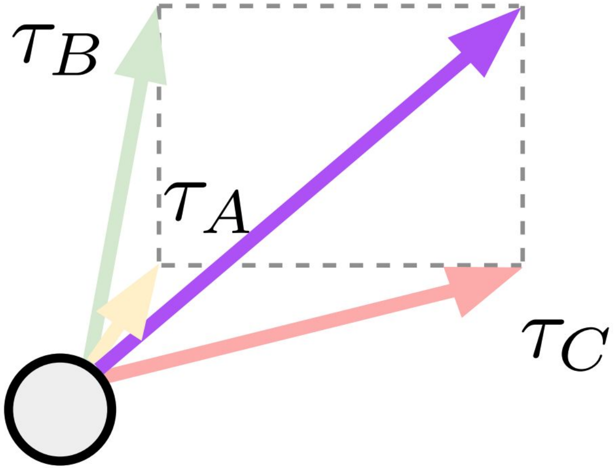
Example: building a multi-task model

In practice, we have optional scaling terms λ_A, λ_B

Task analogies

"A is to B as C is to D"

$$\tau_{\text{new}} = \tau_C + (\tau_B - \tau_A)$$



$$\tau_B - \tau_A = \tau_D - \tau_C$$

$$\tau_D = \tau_C + (\tau_B - \tau_A)$$

$$\theta_{\text{new}} = \theta + \tau_C + (\tau_B - \tau_A)$$

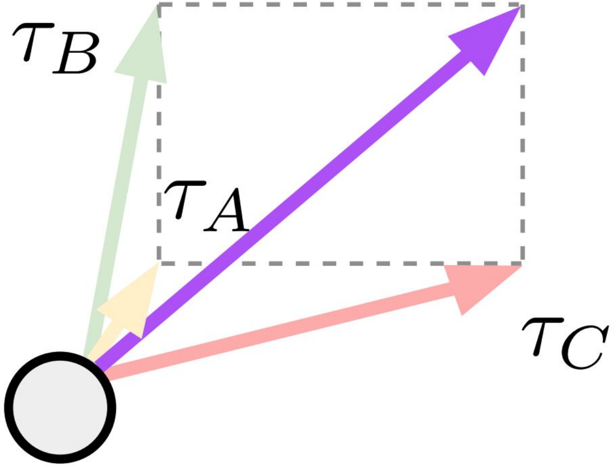
$$= \theta + (\theta_C - \theta) + (\theta_B - \theta) - (\theta_A - \theta)$$

Example: improving domain generalization

In practice, we have optional scaling terms λ_A, λ_B, λ_C

Task analogies

$$\tau_{\text{new}} = \tau_C + (\tau_B - \tau_A)$$



Example: improving domain generalization

"A is to B as C is to D"

$$\tau_B - \tau_A = \tau_D - \tau_C$$

$$\tau_D = \tau_C + (\tau_B - \tau_A)$$

$$\theta_{\text{new}} = \theta + \tau_C + (\tau_B - \tau_A)$$

$$= \theta + (\theta_C - \theta) + (\theta_B - \theta) - (\theta_A - \theta)$$

In practice, we have optional scaling terms λ_A, λ_B, λ_C

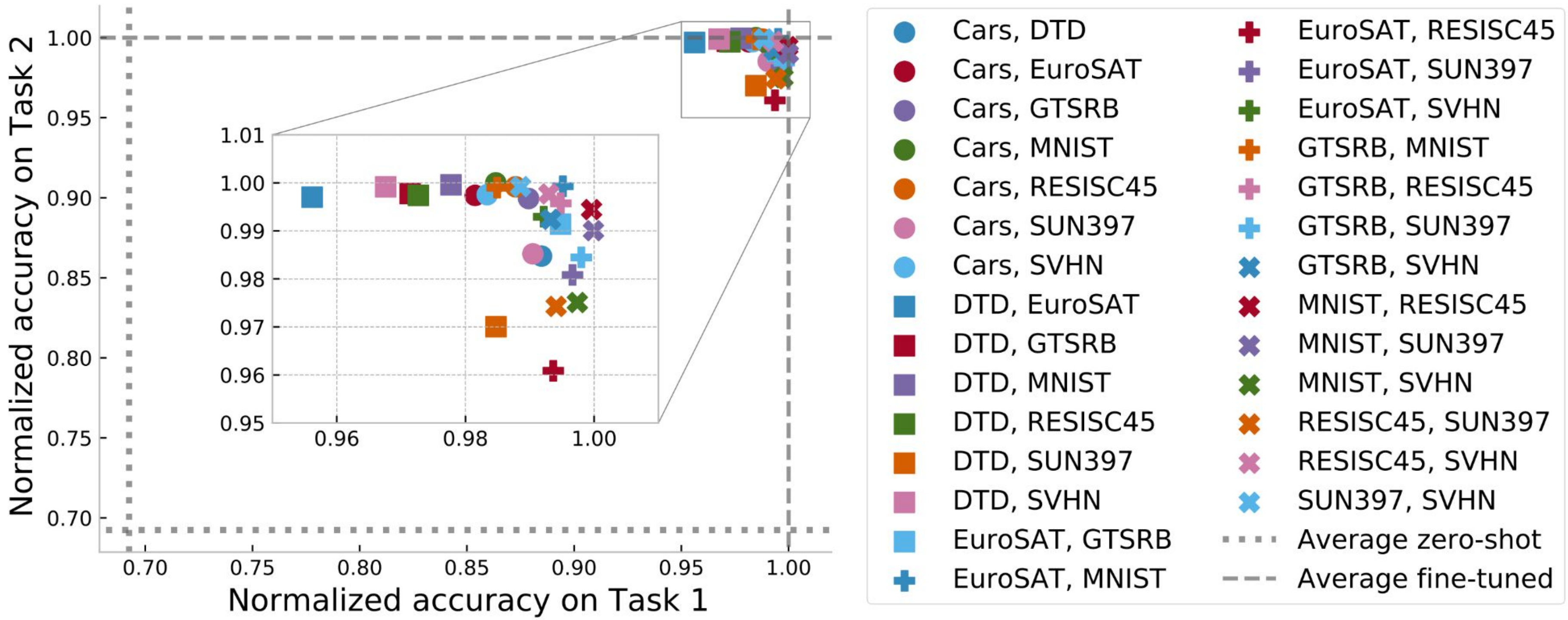
Forgetting image classification tasks via negation

Method	ViT-B/32		ViT-B/16		ViT-L/14	
	Target (↓)	Control (↑)	Target (↓)	Control (↑)	Target (↓)	Control (↑)
Pre-trained	48.3	63.4	55.2	68.3	64.8	75.5
Fine-tuned	90.2	48.2	92.5	58.3	94.0	72.6
Gradient ascent	2.73	0.25	1.93	0.68	3.93	16.3
Random vector	45.7	61.5	53.1	66.0	60.9	72.9
Negative task vector	24.0	60.9	21.3	65.4	19.0	72.9

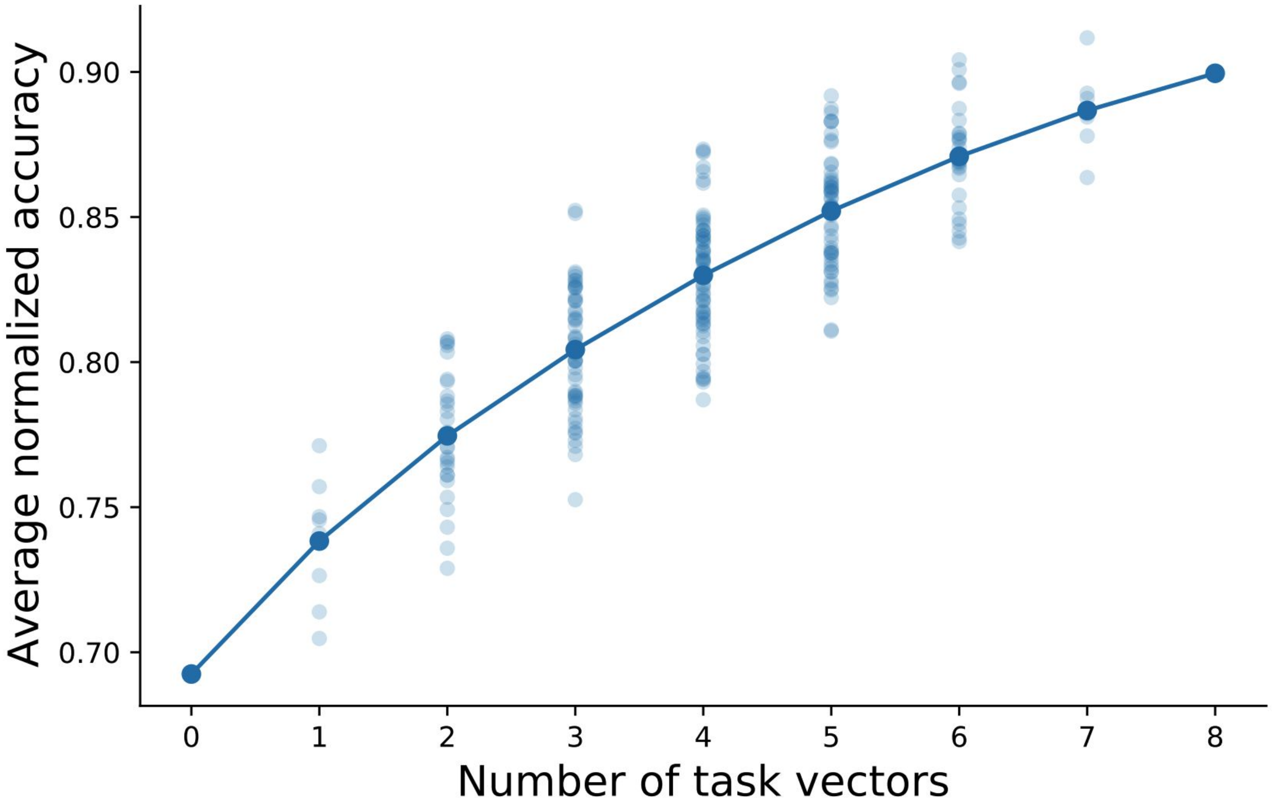
Making language models less toxic with negative task vectors

Method	% toxic generations (↓)	Avg. toxicity score (↓)	WikiText-103 perplexity (↓)
Pre-trained	4.8	0.06	16.4
Fine-tuned	57	0.56	16.6
Gradient ascent	0.0	0.45	$>10^{10}$
Fine-tuned on non-toxic	1.8	0.03	17.2
Random vector	4.8	0.06	16.4
Negative task vector	0.8	0.01	16.9

Adding pairs of task vectors



Adding task vectors builds multi-task models



Improving performance on target tasks with external task vectors

Method	MRPC	RTE	CoLA	SST-2	Average
Zero-shot	74.8	52.7	8.29	92.7	57.1
Fine-tuned	88.5	77.3	52.3	94.5	78.1
Fine-tuned + task vectors	89.3 (+0.8)	77.5 (+0.2)	53.0 (+0.7)	94.7 (+0.2)	78.6 (+0.5)

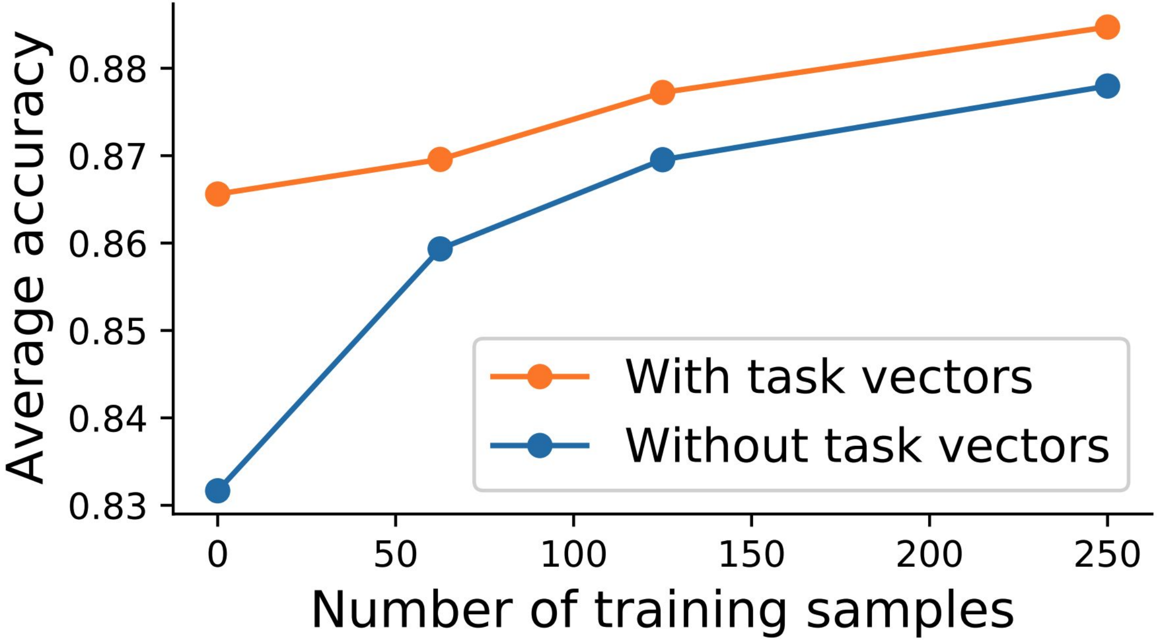
Improving domain generalization with task analogies

$$\hat{\tau}_{\text{yelp; sent}} = \tau_{\text{amazon; sent}} + (\tau_{\text{yelp; lm}} - \tau_{\text{amazon; lm}})$$

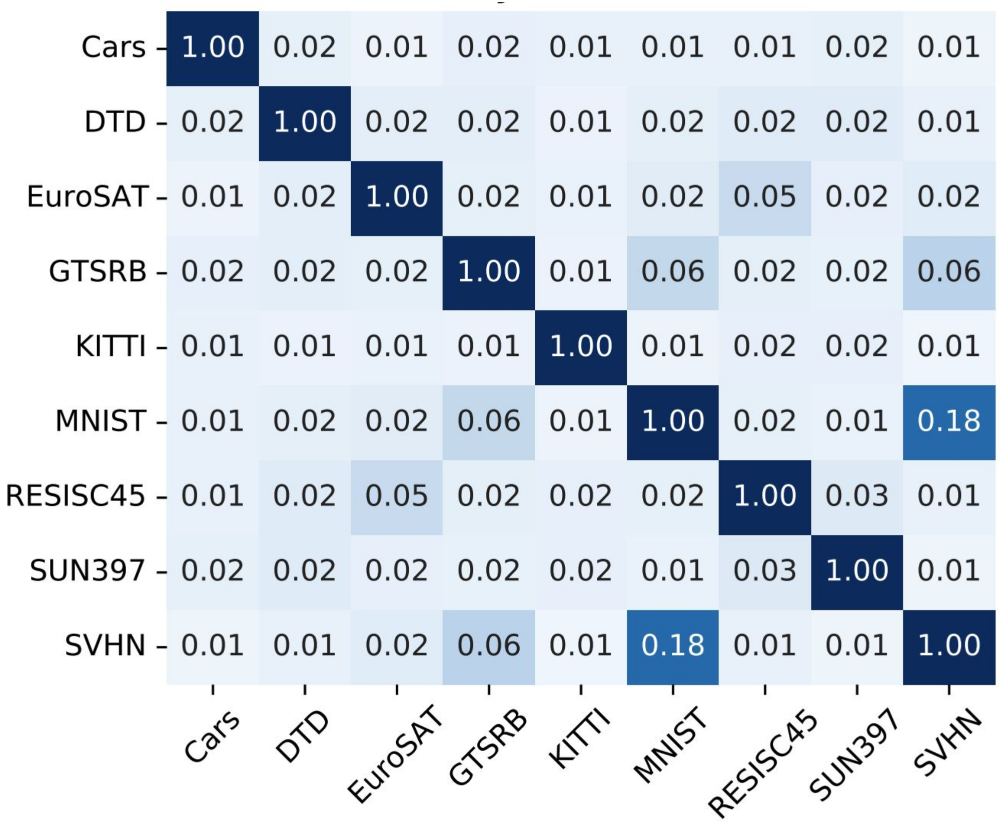
	target = Yelp			target = Amazon		
Method	T5-small	T5-base	T5-large	T5-small	T5-base	T5-large
Fine-tuned on auxiliary	88.6	92.3	95.0	87.9	90.8	94.8
Task analogies	89.9	93.0	95.1	89.0	92.7	95.2
Fine-tuned on target	91.1	93.4	95.5	90.2	93.2	95.5

Learning about subpopulations via analogy

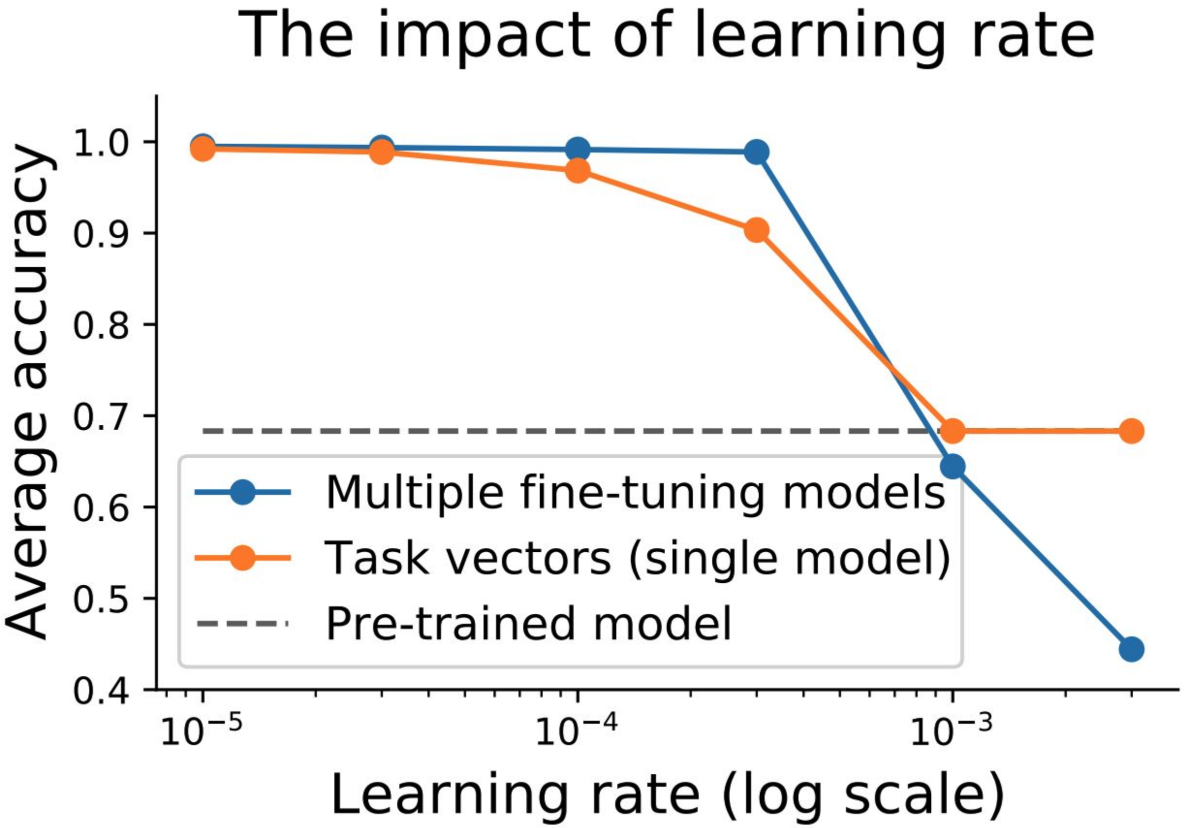
$$\hat{\tau}_{\text{lion indoors}} = \tau_{\text{lion outdoors}} + (\tau_{\text{dog indoors}} - \tau_{\text{dog outdoor}})$$



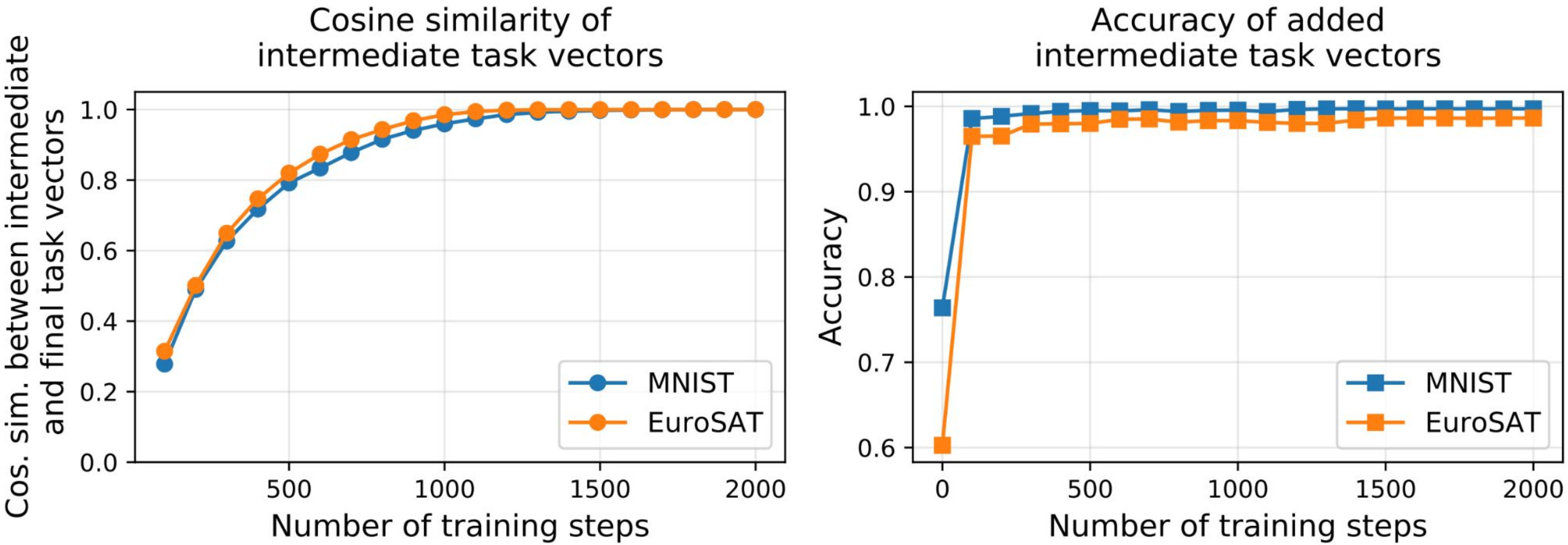
Cosine similarity between task vectors



The impact of learning rate when fine-tuning



How task vectors evolve throughout fine-tuning



Linear mode connectivity

- models fine-tuned from the same pre-trained initialization

Efficient Model Development through Fine-tuning Transfer

Pin-Jie Lin[1] *pinjie@vt.edu*

Rishab Balasubramanian[1] *rishbb@vt.edu*

Fengyuan Liu[2] *fy.liu@mail.utoronto.ca*

Nikhil Kandpal[2] *nkandpa2@cs.toronto.edu*

Tu Vu[1] *tuvu@vt.edu*

[1] *Virginia Tech* [2] *University of Toronto & Vector Institute*

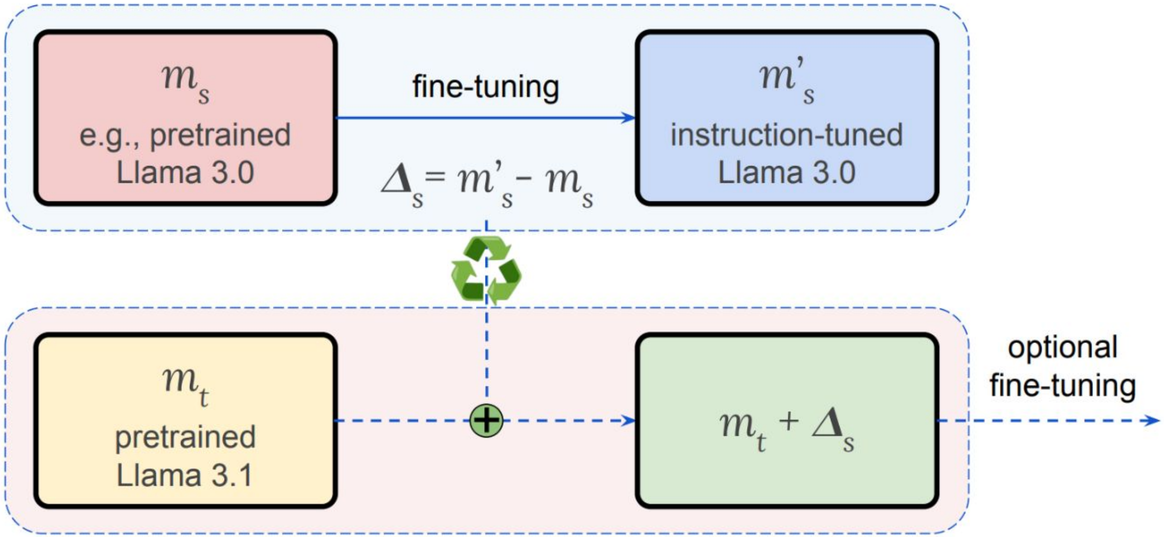


Figure 1: To transfer fine-tuning (e.g., instruction tuning) from a *source* model version s (e.g., Llama 3.0) to a *target* version t (Llama 3.1), we first compute the diff vector $\Delta_s = m'_s - m_s$ from version s, where m'_s is the fine-tuned model (instruction-tuned Llama 3.0) and m_s is the base model (pretrained Llama 3.0). Then, we add Δ_s to the target base model (pretrained Llama 3.1) to approximate the fine-tuned model in version t (instruction-tuned Llama 3.1). We explore two scenarios: (1) *recycling*—transferring from an older model version to a newer one to reduce retraining, and (2) *backporting*—transferring from a newer version to an older one to take advantage of the newer fine-tuning while maintaining optimization for specific use cases.

Transferring fine-tuning updates

Model	**GSM8K**	**MATH**	**$\mathrm{ARC_C}$**	**GPQA**	**MMLU**	**IFEval**
Llama 3.0 8B Instruct	81.1	28.8	82.4	**31.5**	64.9	**76.6**
Llama 3.0 8B	55.6	17.3	79.7	22.3	66.7	34.5
+ $\Delta_{3.1}$	**82.8**	**44.7**	**83.0**	25.9	**70.0**	**76.6**
Llama 3.1 8B Instruct	**86.5**	**50.3**	**83.8**	31.3	**72.9**	80.5
Llama 3.1 8B	56.6	19.3	79.2	21.9	66.8	36.4
+ $\Delta_{3.0}$	79.8	29.9	82.9	**32.6**	65.1	**83.3**

Table 1: Fine-tuning transfer significantly improves the performance of the target base model across various tasks, achieving results comparable to its fine-tuned counterpart in many cases. Here, $\Delta_{3.0}$ and $\Delta_{3.1}$ represent the diff vectors between Llama Instruct and Llama for versions 3.0 and 3.1, respectively. Notably, adding the diff vector Δ_s from a different model version can effectively transform a non-instruction-tuned model (e.g., Llama 3.0 or Llama 3.1) into one that follows instructions well (Llama 3.0 + $\Delta_{3.1}$ or Llama 3.1 + $\Delta_{3.0}$) without further training. Additional results for OLMo and Tülu can be found in Appendix A, where we additionally find that advanced LLM capabilities, attained through alignment tuning stages such as Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO), or Group Relative Policy Optimization (GRPO), can be successfully transferred across different model versions.

Linear mode connectivity

	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$
	13.2	19.4	24.4	64.5	65.5
$+ \Delta_1$		**26.6**	32.0	27.5	19.6
$+ \Delta_2$	**19.0**		**39.8**	25.9	17.3
$+ \Delta_3$	14.3	25.0		68.6	70.3
$+ \Delta_4$	11.8	18.0	22.6		**77.1**
$+ \Delta_5$	11.9	16.0	24.0	**72.9**	
FT($\mathcal{M}_i$)	45.1	50.7	60.4	75.7	75.5

Table 3: GSM8K accuracies indicating that more powerful models are better at leveraging transferred fine-tuning. Effective use of transferred fine-tuning only emerges once the target base model reaches a certain level of capability. Furthermore, fine-tuning transfer works best when the source and target models are close within a linearly connected region of the parameter space. Here, $\mathcal{M}_i$ represents different intermediate pretrained checkpoints of OLMo 2 7B (with smaller values of i indicating earlier checkpoints), and Δ_i refers to the diff vector resulting from the fine-tuning of version i. FT($\mathcal{M}_i$) denotes applying fine-tuning directly to $\mathcal{M}_i$. See Table 11 in Appendix C for MATH500 results.

Multilingual model development

Model	Malagasy	Sinhala	Turkish
Llama 3.0 8B Instruct	23.1	23.3	30.8
+ FT	30.8	29.0	43.2
Llama 3.1 8B Instruct	27.6	**33.0**	27.7
+ $\Delta_{3.0}$	**32.3**	32.3	**43.2**

Table 2: Recycling fine-tuning updates improves multilingual performance on Global MMLU without re-training, yielding a 4.7% and 15.5% absolute improvement for Malagasy and Turkish, respectively, compared to Llama 3.1 8B Instruct. $\Delta_{3.0}$ represents the diff vector between Llama 3.0 Instruct and its monolingual fine-tuned (FT) version.

What Matters for Model Merging at Scale?

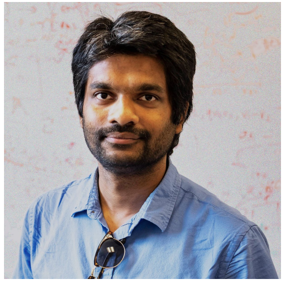
Prateek Yadav

Tu Vu

Jonathan Lai

Alexandra Chronopoulou

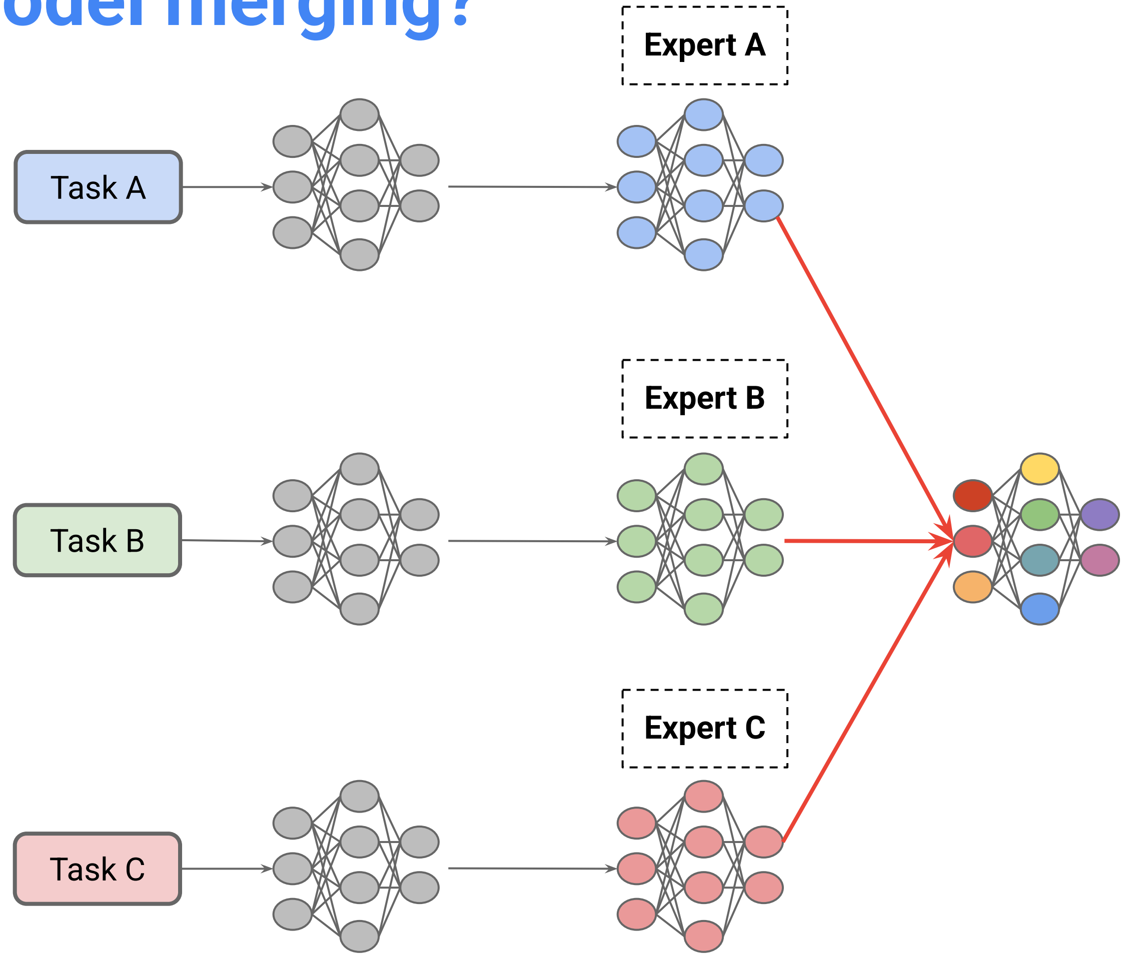
Manaal Faruqui



Mohit Bansal

Tsendsuren Munkhdalai

What is model merging?



Why model merging?

- dramatically reduces storage and serving costs by reusing a single model across tasks
- enables compositional combination of capabilities from expert models, which can improve generalization to novel tasks
- supports decentralized and modular model development by allowing multiple contributors to independently build models and later combine them together

Limitations of prior work

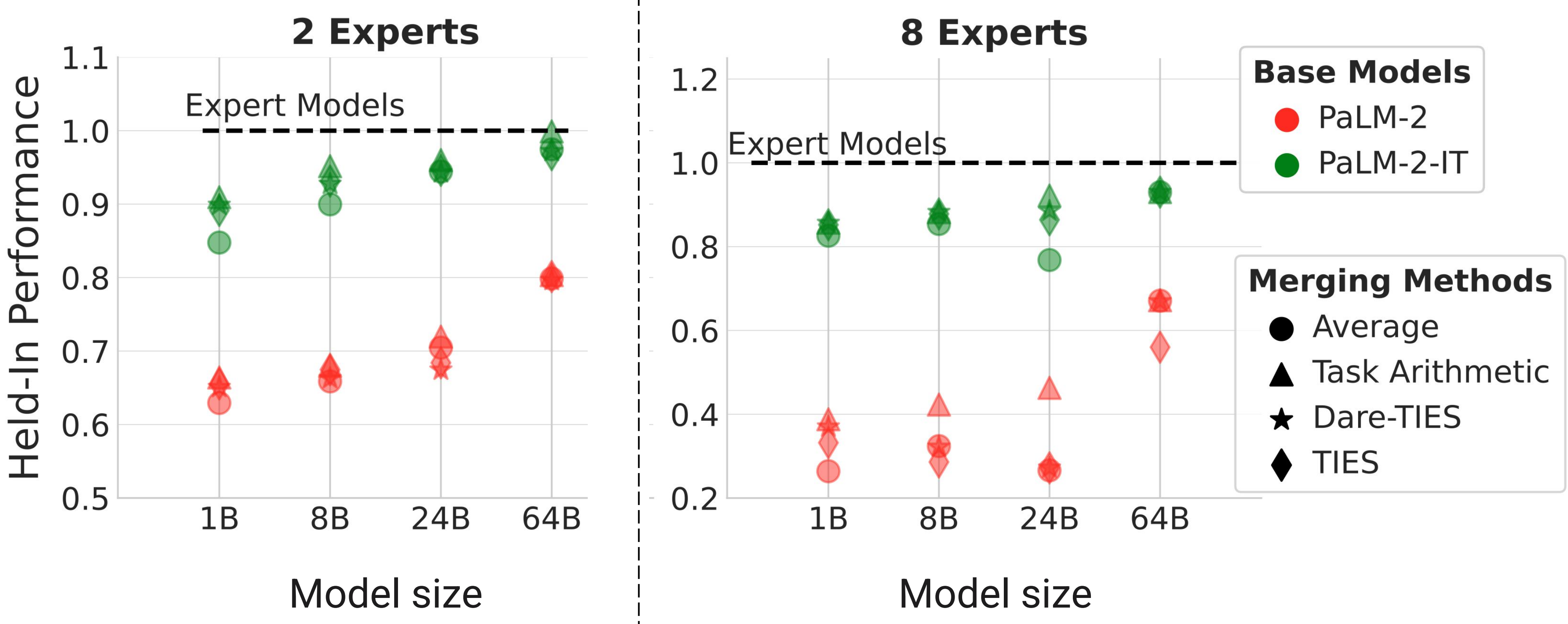
- Typically merges small or moderately-sized models (up to 7B parameters)
- Typically merges only 2-3 models
- Largely focuses on improving **“held-in”** performance on tasks the expert models were trained for

A large-scale empirical study

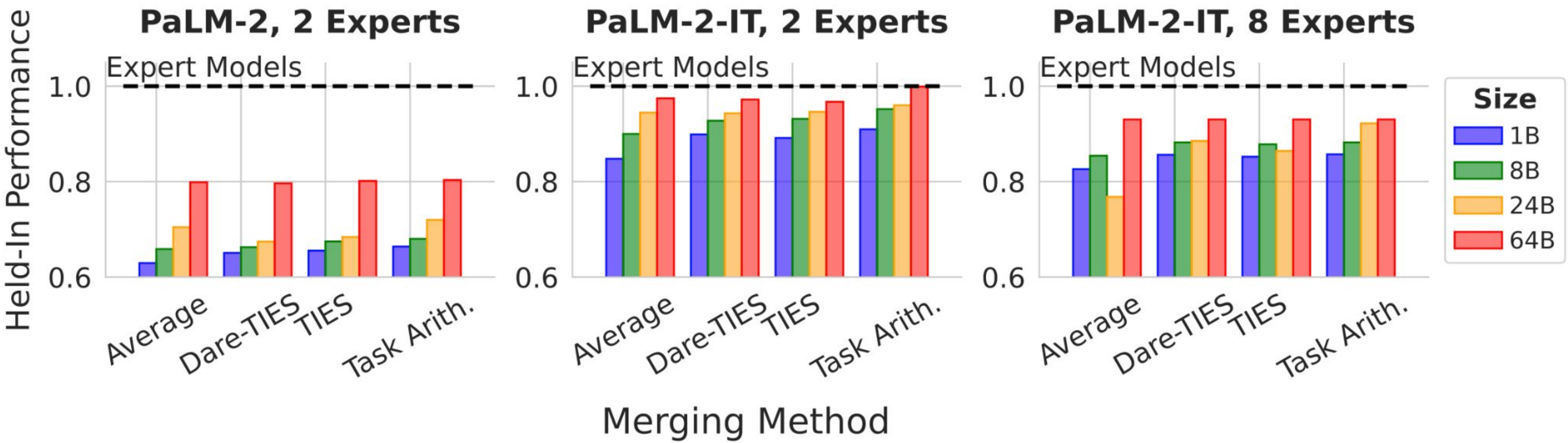
- 4 important factors
  - **model size**
    - 1B, 8B, 24B, 64B
  - **base model quality**
    - Pre-trained model (**PaLM**) vs. Instruction-tuned (**PaLM-IT**)
  - **merging method**
    - Average, Task Arithmetic / Task Vectors, TIES, DARE-TIES
  - **number of experts**
    - 2, 4, 6, 8

- Their impact on
  - **Held-In** performance
  - Zero-shot (**Held-Out**) generalization

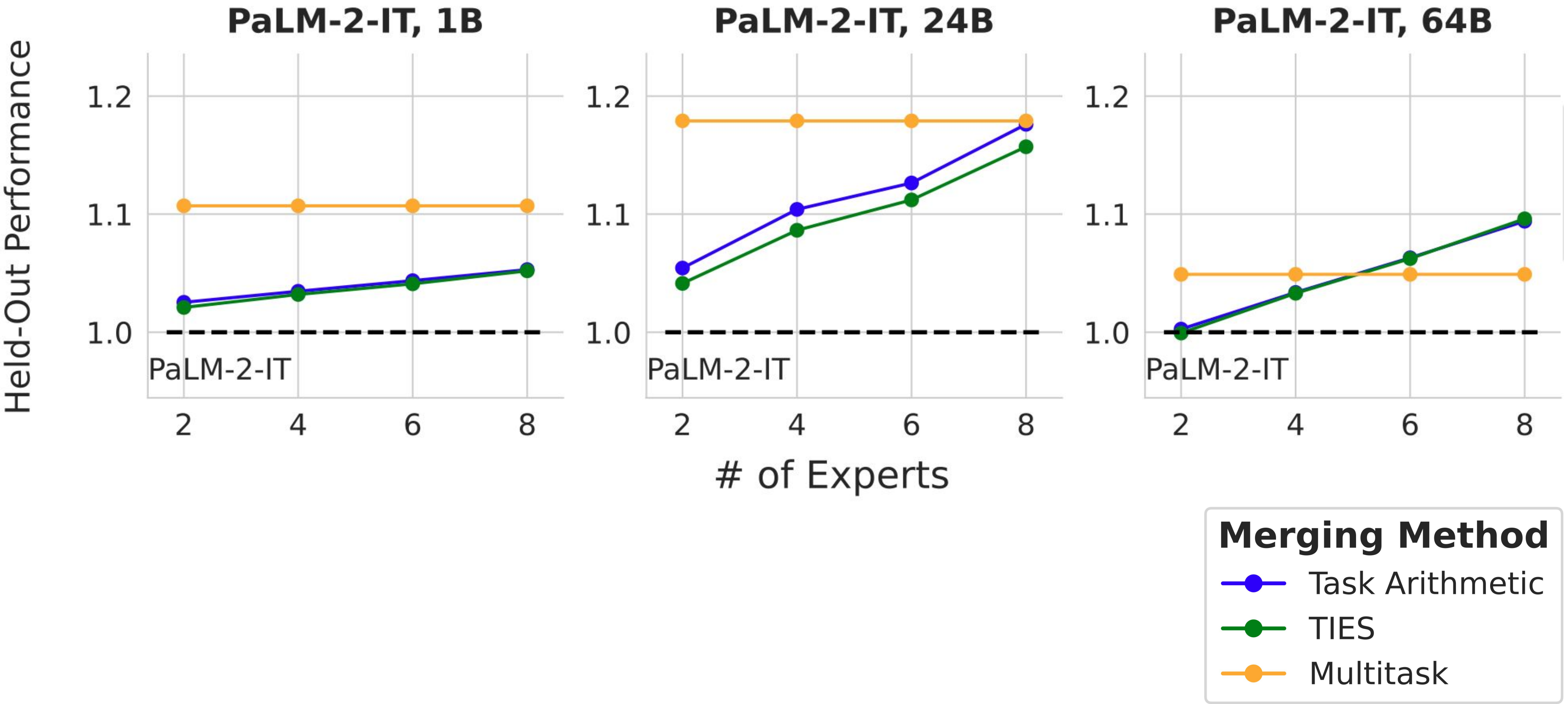
Instruction-tuned models facilitate easier merging



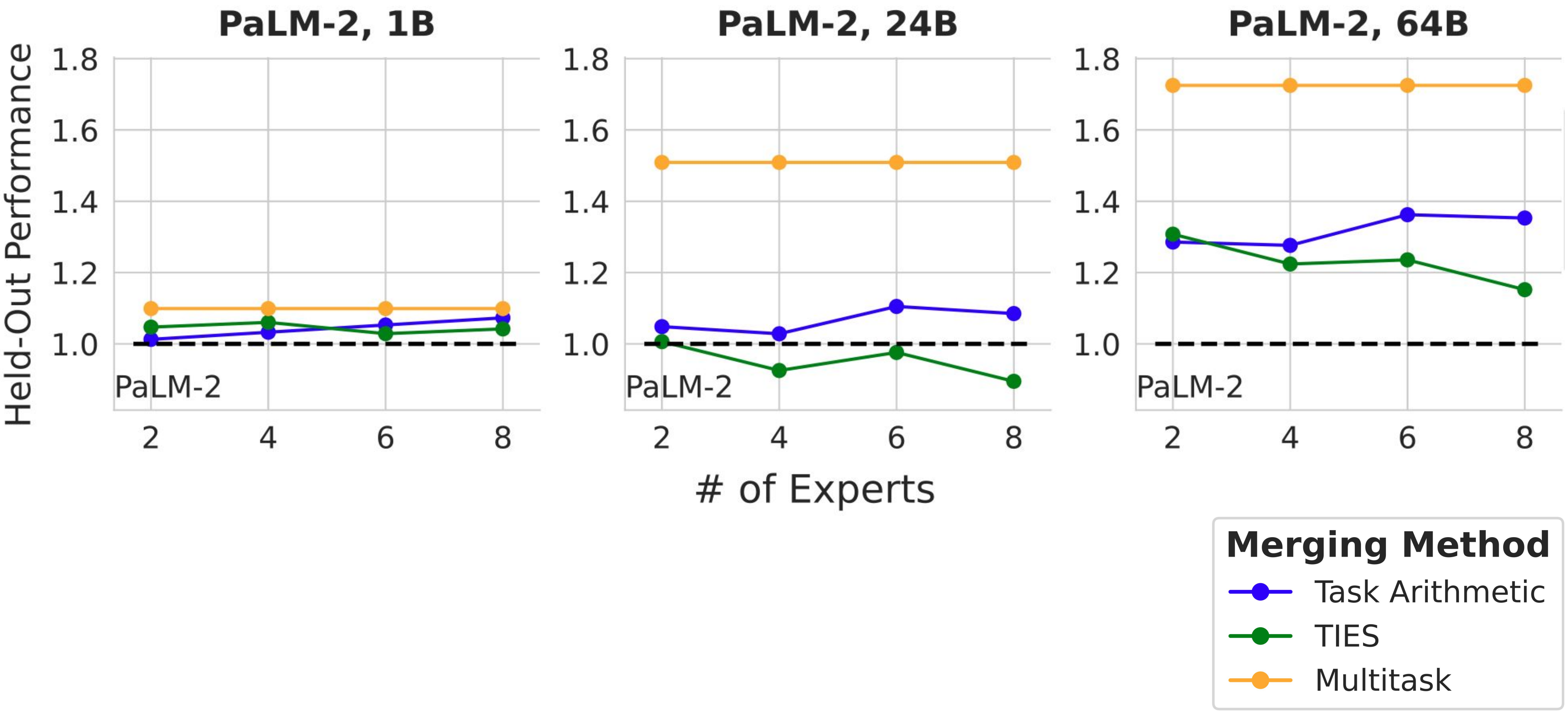
Bigger models are easier to merge



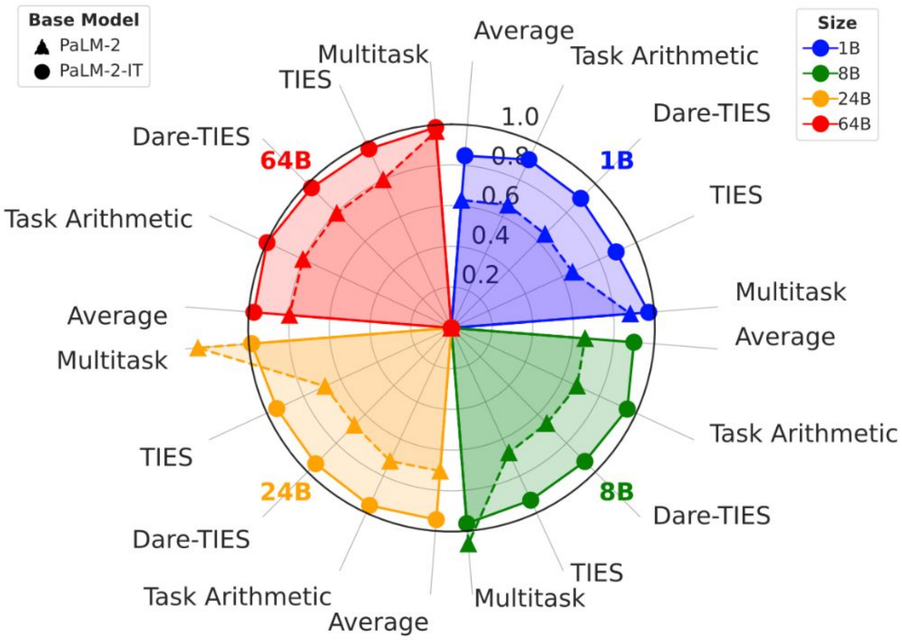
Merging boosts zero-shot generalization



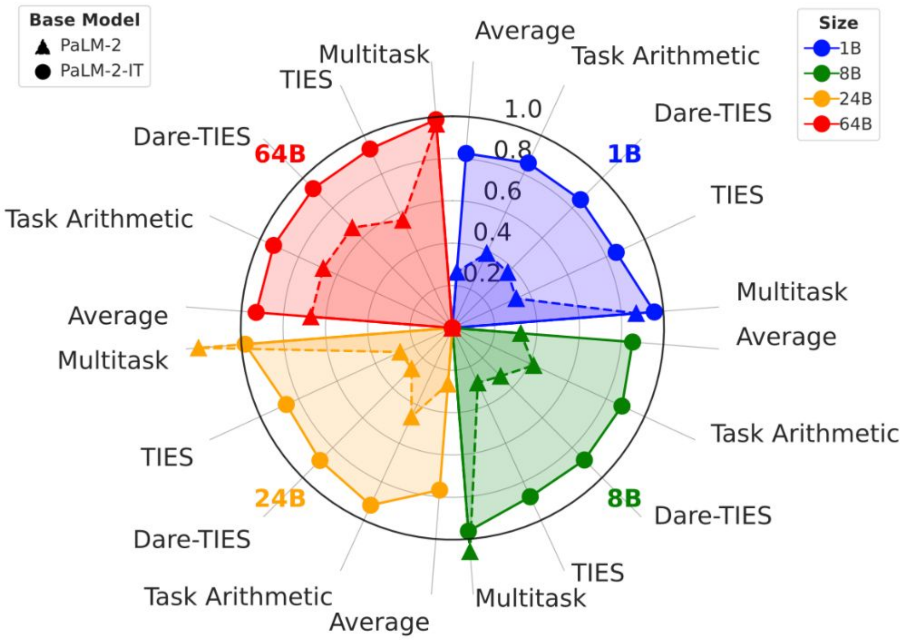
Merging boosts zero-shot generalization (cont.)



Bigger model sizes can merge more experts

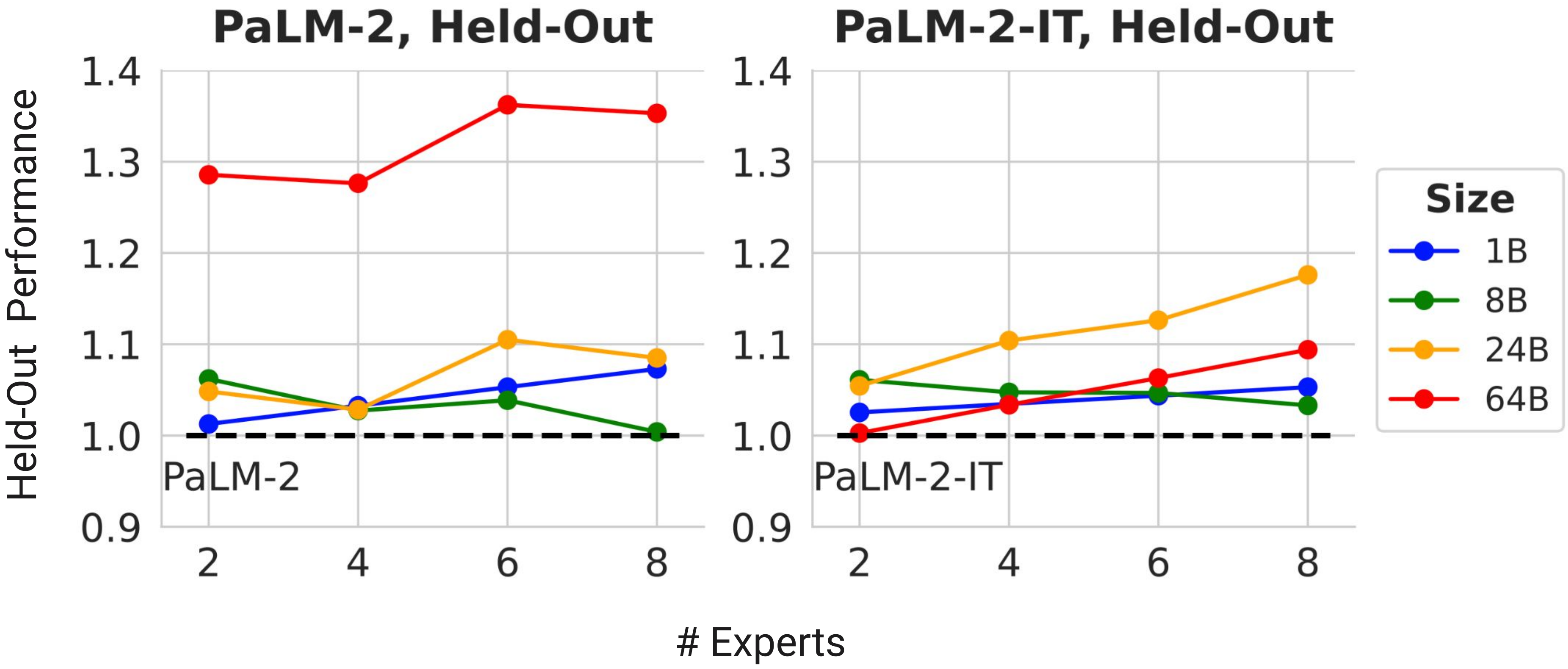


(a) Merging 2 experts, Held-In.

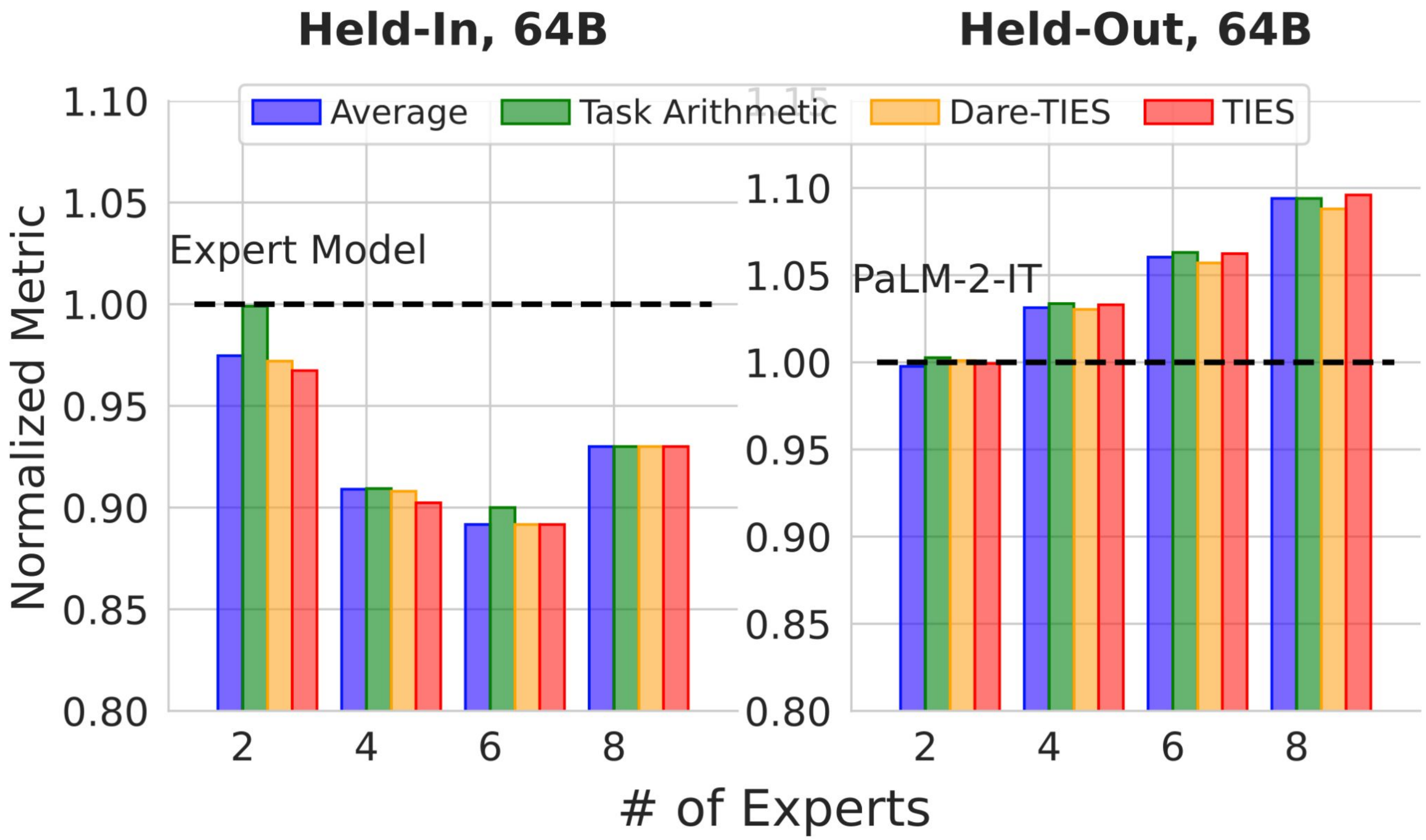


(b) Merging 8 experts, Held-In.

Bigger model sizes can merge more experts (cont.)



At large scales, merging methods converge



Thank you!