Merge to Learn: Efficiently Adding Skills to Language Models with Model Merging

Jacob Morrison[1] Noah A. Smith[1,2] Hannaneh Hajishirzi[1,2]

Pang Wei Koh[1,2] Jesse Dodge[1] Pradeep Dasigi[1]

[1]Allen Institute for AI [2]University of Washington

Correspondence: jacobm@allenai.org

Abstract

Adapting general-purpose language models to new skills is currently an expensive process that must be repeated as new instruction datasets targeting new skills are created, or can cause the models to forget older skills. In this work, we investigate the effectiveness of adding new skills to preexisting models by training on the new skills in isolation and later merging with the general model (e.g. using task vectors). In experiments focusing on scientific literature understanding, safety, and coding, we find that the *parallel-train-then-merge* procedure, which is significantly cheaper than retraining the models on updated data mixtures, is often comparably effective. Our experiments also show that parallel training is especially well-suited for enabling safety features in LMs relative to continued finetuning and retraining, as it dramatically improves model compliance with safe prompts while preserving its ability to refuse dangerous or harmful prompts.

1 Introduction

Recent work has shown that instruction tuning pretrained language models (LMs) can result in strong generalist models that can perform a variety of comprehension and generation tasks, owing largely to the availability of high quality datasets.

As training datasets targeting new skills are constructed, it is an open question how best to *patch* preexisting models to incorporate the new skills represented by those datasets. Commonly used approaches include continued finetuning (CFT) of the existing models on the new datasets and retraining (RT) the models on a combination of old and new instruction tuning datasets, both of which have clear pros and cons associated with them.

CFT, while computationally cheaper, may cause the model to *forget* the skills from earlier rounds of training. On the other hand, in addition to being more expensive, RT on merged datasets is possible only when the practitioner has access to the the training datasets from earlier rounds, which is not the case for many publicly available instruction-tuned LMs such as Mistral 7B (Jiang et al., 2023) and Llama 3 (AI@Meta, 2024). While we focus on publicly available mixes, this is an important consideration as new models and datasets are released.

As an alternative, we explore merging the parameters of models separately trained for individual skills—merging to learn—a family of approaches we refer to as "parallel train then merge" (PTM). This general idea is shared by several well-known methods (e.g., model patching, Ilharco et al., 2022; WiSE-FT, Wortsman et al., 2022b; task arithmetic, Ilharco et al., 2023; TIES Yadav et al., 2023; DARE, Yu et al., 2024; time vectors, Nylund et al., 2024). Unlike RT, PTM does not require access to the original training data, as one can separately train only on the new data, also making PTM a computationally cheaper option. Since PTM does not directly change the weights associated with previously learned tasks, it should allow the model to retain more of its original skills compared to other methods. PTM also enables the efficient addition of *multiple* new skills to a single model. This work is the first to systematically explore PTM for instruction tuning.

We compare the three methods for model patching (CFT, RT, and PTM) in terms of model performance as well as computational cost. We experiment with adding three new skills: scientific literature understanding, coding, and refusing unsafe requests, to Tülu, a general-purpose instruction-tuned model, and evaluate the performance of the models resulting from each method on a suite of evaluation datasets representing general instruction-following skills as well as those that target the new skills being added. We find that:

- When optimizing performance on the new skills, PTM achieves competitive performance with the best RT models, with a 50–95% improvement in training efficiency.

- PTM preserves nearly all of the original model's general skills versus a 10–40% drop for CFT, while achieving similar skill-specific performance.
- Setting the mixture weight proportional to the ratio of the number of training steps dedicated to the new skill being added is a good heuristic when held-out data is not available, resulting in good model performance on new skills while preserving general performance. We find that this heuristic can also be effective for adding *multiple* new skills to a single model.
- For enabling safety-related refusals, a skill that can be at odds with general skills, PTM proves to be particularly effective compared to RT and CFT by improving unsafe refusal rates, preserving general skills, and reducing exaggerated refusals by 30–80%.

Overall, we find that PTM is consistently a better choice than CFT for teaching instruction-tuned models new skills without compromising on general skills. We note that RT is not always a feasible option because the training datasets for many publicly available instruction-tuned models are not available, and even when it is, PTM generally offers comparable performance tradeoffs while being significantly cheaper.

2 Problem Setup

We aim to add new, diverse behaviors to a general-purpose instruction-tuned model, while preserving the original model's overall performance. We want to do so in a computationally efficient manner, without increasing inference cost. Defined concretely, we want to use a new skill-specific dataset D to improve the performance of a general model θ_G, trained on general data G, on an evaluation set for new skills E_D, without losing performance on a general set E_G, while minimizing the computational requirements.

We describe the methodology and training complexity of three methods for adding new skills to preexisting models: continued finetuning, retraining from scratch, and model merging. We measure training complexity in terms of how many training steps are required to create the pool of models we select from.

2.1 Continued Finetuning on Target Skills

One straightforward method is to continue finetuning the instruction-tuned model on instruction data targeting the new skills, which is much cheaper than retraining from scratch. To determine the best amount of D to train on, we try n different subsamples D_i, each requiring $|D_i|$ training steps. In total, this requires $\sum_{i=1}^{n} |D_i|$ training steps.

We will see in Section 4.2 that continued finetuning substantially degrades general skills.

2.2 Retraining From Scratch

Another method is to retrain the instruction-tuned model starting from the pretrained model, with the skill-specific data added into the original data mix. Since we care about retaining the general performance as well, the ratio of the amount of the skill-specific data to that of the original data needs to be determined carefully. The ideal way to do this is to retrain with different data mixing ratios and then perform *model selection* based on these models' performance on a held-out validation set.

While retraining should lead to competitive performance on both general and skill-specific evaluations, it is inefficient compared to other methods due to the need to retrain on the entire general mix for every training run, especially considering the model selection described above.

For a given subsample of the skill-specific data D_i, a single retraining run requires $|G| + |D_i|$ training steps. For n data mix variations, the total is $n \cdot |G| + \sum_{i=1}^{n} |D_i|$. General instruction datasets tend to be larger than skill-specific datasets, containing hundreds of thousands to millions of instances (Ivison et al., 2023; Lian et al., 2023; Singh et al., 2024), while many skill-specific datasets contain on the order of tens to a hundred thousand instances (Chaudhary, 2023; Wadden et al., 2024; Zheng et al., 2024), highlighting the overall expense of retraining.

Additionally, we note that retraining is *not possible* in cases where the pretrained and instruction-tuned models have been released but the general instruction mix has *not*, such as Llama 3 (AI@Meta, 2024), Mistral 7B (Jiang et al., 2023) and Gemma (Gemma-Team et al., 2024). While we experiment with publicly available data, this is important for future work with the latest models and datasets.

2.3 Parallel Training & Merging

We next describe the three model merging methods that we explore: **task arithmetic** (Ilharco et al., 2023), **linear interpolation** (Rofin et al., 2022), and **WiSE-FT** (Wortsman et al., 2022b). We consider these as instantiations of a general *parallel-train-then-merge* (PTM) framework:[1] training a base model on *only* the skill-specific data D to create the skill-specific model θ_D, and then weight-space merging θ_D with a generalist model θ_G with weight ω.

While retraining and continued finetuning require training multiple models to determine how much the skill-specific data should influence the general model, in PMT, this is accomplished through the mixture weighting parameter ω. This means that the total training cost for PMT is $|D|$, dramatically lower than other methods.

The three PTM methods we consider correspond to different ways of training separately on the skill-specific data and incorporating it into the final model. While we describe all three methods below, we primarily focus on task arithmetic as we find that it is particularly adept at improving performance on new skills while preserving general skills. We compare all three directly in Section 4.4.

Task Arithmetic For task arithmetic, we finetune the pretrained model θ_{pre} on all of the available skill-specific data D to create θ_D, and then subtract θ_{pre} to get the new task vector τ_D:

$$\tau_D = \theta_D - \theta_{pre} \tag{1}$$

We then merge the task vector into the general-purpose instruction-tuned model θ_G:

$$\theta_{final} = \theta_G + \omega \cdot \tau_D, \tag{2}$$

where ω is selected using held out data when available, or with a heuristic. Experimentally, we find that setting $\omega < 1.0$ is better than naively setting $\omega = 1$.

Linear Interpolation For linear interpolation, we create a general skill task vector τ_G by subtracting θ_{pre} from the instruction-tuned model:

$$\tau_G = \theta_G - \theta_{pre} \tag{3}$$

We then interpolate between the task vector τ_D and the general skill vector:

$$\theta_{final} = \theta_{pre} + \omega \cdot \tau_D + (1 - \omega) \cdot \tau_G \tag{4}$$

[1]This name derives from "branch-train-merge," a similar technique designed for pretraining (Li et al., 2022).

WiSE-FT For WiSE-FT, we first continue to finetune θ_G on the new skill-specific dataset to create θ_{CFT}. We then subtract θ_G to create τ_{CFT}:

$$\tau_{CFT} = \theta_{CFT} - \theta_G \tag{5}$$

We then add τ_{CFT} to the general model with weight ω, downweighting the impact of CFT:

$$\theta_{final} = \theta_G + \omega \cdot \tau_{CFT} \tag{6}$$

Notably, WiSE-FT is especially suitable for models without publicly available pretrained checkpoints, as it does not require access to θ_{pre}.

3 Experimental Setup

3.1 Datasets & Evaluations

We explore the trade-off between general and skill-specific performance across three sets of skills: **Science**, **Safety**, and **Coding**. We train general purpose models using a modified version of the Tülu V2 mix (Ivison et al., 2023), and train skill-specific models with (1) SciRIFF (Wadden et al., 2024), (2) a novel refusals dataset, and (3) CodeFeedback (Zheng et al., 2024). We additionally choose consistent sets of evaluations designed to capture either general or specialized skills. Datasets and relevant evaluations are described in greater depth below and in Table 1.

General-purpose To train our general-purpose models, we use a modified version of the Tülu V2 mix. We removed the science subset (7.5k examples), CodeAlpaca (20k examples), and refusals identified by heuristics (23k examples), resulting in 275k total instances, to simulate a setting where the base model has plenty of room to improve in our target skills. We refer readers to Wang et al., 2023 and Ivison et al., 2023 for more details on the original mix.

We evaluate general skills on a subset of the Tülu 2 evaluation suite to cover a broad range of skills: world knowledge (MMLU, Hendrycks et al., 2021), mathematics (GSM8K, Cobbe et al., 2021), open-ended generation (AlpacaEval, Li et al., 2023), reasoning (Big Bench Hard, Suzgun et al., 2022), and truthfulness (TruthfulQA, Lin et al., 2022). More details are available in Ivison et al., 2023.

Science To train our skill-specific models on science, we use SciRIFF (Wadden et al., 2024), an instruction dataset covering tasks like information extraction, question answering, and more for scientific literature understanding. The dataset covers

Dataset	Training Set Size	Number of Evals	Domain
Tülu V2 (modified) (Ivison et al., 2023)	275,464	5	Collection of general instructions
SciRIFF (Wadden et al., 2024)	61,349	9	Scientific literature understanding
Safety (internal)	66,161	4	Prompt/refusal pairs
CodeFeedback (Zheng et al., 2024)	156,526	2	Single-turn coding examples

Table 1: Summary of datasets used in this work.

scientific disciplines like biomedicine, artificial intelligence, and others. We refer readers to Wadden et al., 2024 for full details on the dataset.

We evaluate our models on Science using SciRIFF's validation and test sets, measuring performance across nine held-out tasks not seen during training. Unless otherwise noted below, we report average *validation* set scores during our analyses. We refer readers to Wadden et al., 2024 for more details on evaluations.

Safety To train our skill-specific models on safety, we use an internally developed safety dataset covering broad categories like harmful language, malicious uses, misinformation, and more. Each example is a potentially dangerous or harmful prompt paired with a refusal generated from GPT-4 (OpenAI et al., 2024). A seed set of prompts were written by humans, and more prompts were generated based on this seed set by GPT-4. Refusals were collected by prompting GPT-4, and keeping responses that were classified as a refusal. We will release this data upon publication.

To evaluate our models on safety, we use four categories of safety evaluations in our experiments: toxicity (ToxiGen, Hartvigsen et al., 2022), automated red teaming (HarmBench, Mazeika et al., 2024), refusing unsafe prompts (XSTest Unsafe, Röttger et al., 2024), and exaggerated refusals (XSTest Safe). We normalize scores for these evaluations so 0.0 is the worst possible score and 100.0 is the best. For scores reported below, we report the average of the first three metrics and consider exaggerated refusals separately.

Coding To train our skill-specific models on coding, we use the single-turn subset of CodeFeedback (Zheng et al., 2024) of instruction/code pairs. We refer readers to Zheng et al., 2024 for more details.

We evaluate our models on coding by taking the average of scores on HumanEval+ (Chen et al., 2021; Liu et al., 2023) and Mostly Basic Python Programs+ (MBPP+) (Austin et al., 2021; Liu et al., 2023). We sample with a temperature of 0.8 and report pass@10 metrics for both.

3.2 Training Setup

Settings We run all of our experiments on top of Llama 2 7B (Touvron et al., 2023). We fully finetune all of our models for two epochs with a context length of 4,096 and a batch size of 128, and we follow the other hyperparameters used in Ivison et al., 2023. Our general purpose model was created by training Llama 2 7B on our modified Tülu 2 mix, described in Section 3.1. Full training details are available in Appendix A.

Methods For Adding New Tasks We compare three methods for add new skills to an instruction-tuned model, described in Section 2: **continued finetuning** (CFT) on skill-specific data, **retraining** (RT) from scratch on a mix of all of the general data and some amount of skill-specific data, and task arithmetic, a form of **parallel-train-then-merge** (PTM), by training the base pretrained model on the skill-specific data and adding the task vector directly to the instruction-tuned model.

We evaluate five checkpoints for each setting, varying the influence of the skill-specific data on the general model. For CFT and RT, we train on five different amounts of the skill-specific data. For PTM, we choose five different values for the mixture weight. We also compare PMT with linear interpolation and WiSE-FT in Section 4.4.

Many instruction datasets do not have validation sets (Ivison et al., 2023; Zheng et al., 2024; Lian et al., 2023; Singh et al., 2024), and thus how to select models is an open question. We take advantage of SciRIFF's validation set to select models in Section 4.1. Otherwise, we select using heuristics.

Model	General	Science	Training Steps
Tülu Only	49.9	27.9	-
Best CFT	33.7	**40.6**	1,005
Best RT	**50.6**	37.8	11,766
Best PTM	47.1	38.2	**479**

Table 2: Performance on general and science test evaluations for the best continued finetuning (CFT), retraining from scratch (RT), and parallel-train-then-merge (PTM) models based on science validation performance. PTM shows equivalent science performance to the best RT model with slightly lagging general skills, while taking about 4% as many training steps. PTM shows slightly lower science performance than CFT with an over 13 point gain on general skills, while taking less than half the total training steps.

4 Results

4.1 Trade-off Between Computation Cost and Model Performance

In this section, we explore the performance and cost trade-offs when optimizing for performance based on held-out data. We look at improving scientific literature understanding through three methods: CFT, RT, and PTM.

For both CFT and RT, we experiment with the five different amounts of science data reported in SciRIFF (Wadden et al., 2024), ranging from about 4k to about 61k training examples. For PTM, we train on all 61k training examples, and test five different values for the mixture weight ω: 0.2, 0.4, 0.6, 0.8, and 1.0. We additionally report the total number of training steps required (with our batch size of 128) to create the models in each group to estimate the cost of each method. We then select the best model from each of these three methods based on SciRIFF validation set performance.

As shown in Table 2, we find that PTM achieves strong performance on both general and science evaluations, with a fraction of the compute. PTM requires about **4%** of the compute compared to RT, and slightly edges out the best RT model on science while being within a few points on general evaluations. PTM is also within a few points on science compared to the best CFT model, with a more than 13 point improvement in general skills and less than 50% the compute.

Model	%Δ Gen.	%Δ Spec.	Training Steps
Best CFT (Science)	–32.5	46.0	1,005
Best RT (Science)	1.37	39.1	11,766
Best PTM (Science)	1.30	26.3	479
Best CFT (Safety)	–40.1	98.9	1,551
Best RT (Safety)	0.66	89.6	12,311
Best PTM (Safety)	–0.13	88.9	517
Best CFT (Coding)	–7.73	51.6	3,669
Best RT (Coding)	0.13	50.7	14,429
Best PTM (Coding)	1.43	33.3	1,223
Best CFT (Ex. Ref.)	–85.1	98.9	1,551
Best RT (Ex. Ref.)	–39.9	87.2	12,311
Best PTM (Ex. Ref.)	–6.45	72.6	517

Table 3: Absolute percentage change compared to the Tülu baseline on two evaluations: general and specialized for science, safety, and coding, and exaggerated refusals and safety for the exaggerated refusals rows. PTM preserves general performance better than CFT and comparably to RT in all four settings, while requiring a fraction of the compute compared to either method. PTM also improves skill-specific performance in every scenario, and improves safety as much as RT while taking 4% the compute.

4.2 Model Performance Across Skills

Overall Trends We now look at overall trends when comparing PTM to CFT and RT. In Table 3, we compare CFT, RT, and PTM across all three sets of skills, as well as for exaggerated refusals. We consider an equal number of models for each method to ensure a fair comparison: for CFT and RT, we perform five data mix trials for each dataset, and for PTM, we explore five evenly spaced values for ω. We select models in each category based on their average percentage improvement over the baseline model in two dimensions: general skills and performance for science, safety, and coding, and exaggerated refusal compliance rate and safety for the exaggerated refusals rows.

From this table, we see a few trends. First, PTM preserves much more of the underlying model's general skills than CFT, which consistently suffers a substantial drop in general performance across all four settings. Second, for safety, PTM achieves comparable or better general *and* safety-specific performance compared to RT. Finally, across all settings, PTM is substantially cheaper than both CFT and RT, requiring a fraction of the total train-

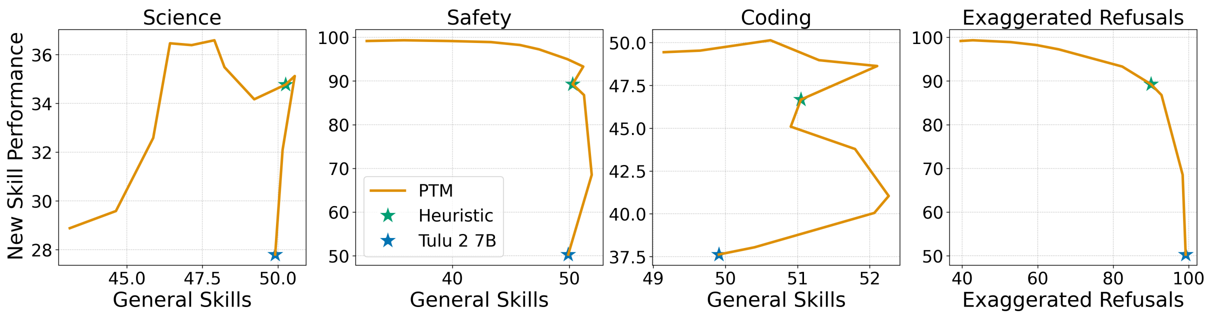

Figure 1: Trade-offs managed through ω. We highlight the point along each curve that corresponds to using our weighting heuristic, $\omega = \frac{|D|}{|G|}$. This point consistently achieves strong performance on all settings, without requiring held out data. We take advantage of PTM's negligible cost to test different mixture weights to plot 10 checkpoints from evenly spaced values of ω as well as the heuristic

Model	General	Science	Coding	Safety	Exaggerated Refusals	Additional Training Steps
Tülu Only	49.9	27.8	37.6	50.3	99.2	-
CFT (All 3)	40.3	37.9	**58.2**	**99.8**	16.0	2,219
RT (All 3)	50.1	**39.2**	57.9	95.0	37.2	4,732
PTM (All 3)	**51.1**	26.6	45.3	84.0	**93.2**	**0**

Table 4: PTM with all three skill-specific models improves general performance, while noticeably improving both coding and safety over the baseline. PTM also shows strong improvement on exaggerated refusals, with a **77 point gain** over CFT and a **56 point gain** over RT. However, science performance is adversely impacted, and CFT and RT achieve stronger coding and safety performance overall. If single-skill models have already been trained, three-skill PTM has **no** additional training cost, while both the CFT and RT models must be separately trained.

ing cost of either method.

PTM Mitigates Exaggerated Refusals As mentioned in Section 3.1, when evaluating safety, we consider two metrics: our safety average, measuring how often a model correctly refuses to comply with a prompt, and exaggerated refusals, measuring how often the model complies with a prompt, written to be superficially similar to offensive or harmful prompts, that it *should* comply with. In this section, we analyze the effect of using task vectors on exaggerated refusals.

We compare improvement over the baseline model for these metrics in Table 3. By training a separate "safety vector" and applying it to an SFT model, we are able to achieve comparable, if not better, general and safety performance to CFT and RT, while **dramatically** improving compliance on exaggerated refusals. In other words, PTM makes the final model substantially better at safe but misleading prompts, improving performance on XSTest Safe by 30–50 points versus the other methods, while achieving similar safety and general performance.

Heuristics For settings without held-out data, such as our safety and coding datasets, we find that a consistently strong heuristic for selecting a model checkpoint is to 1) train the task vector on all of the available skill-specific data, and then 2) weight the vector with

$$\omega = \frac{|D|}{|G|}. \tag{7}$$

In Figure 1, we plot the trade-off between general and skill-specific performance as we vary ω between 0.0 and 1.0. We highlight the point on each curve selected by our heuristic, and we see that it consistently selects a point on the curve that preserves most, if not all of the original model's general performance, while substantially improving skill-specific performance.

4.3 Multiple New Skills

We next investigate adding multiple new skills to a single model. In Table 4, we report the results of merging all three skills into a single model using the heuristic, and compare with both CFT and RT using all of the available skill-specific data. We also report scores for using our heuristic for each skill individually.

We see that when we merge all three skills into a single model, we achieve best overall general and exaggerated refusals performance, once again **substantially** improving the latter over both CFT and RT. Additionally, we get very similar performance on both coding and safety when compared with single skill PTM models. However, we also see a large drop in science performance. We take advantage of PTM's very low ablation cost to attempt to diagnose this issue. We investigate in Table 5 by merging each skill-specific model together pairwise, and find that the drop in science performance is most likely caused by interference between the coding and safety vectors, as demonstrated by the large difference in science performance between this merge, the baseline, and the two pairwise merges involving science. In Appendix B, we investigate if other model merging methods—designed to minimize interference—can help mitigate this issue.

4.4 Alternative PTM Methods

The first alternatives we explore are the two other relative weighting schemes described in Section 2: linear interpolation and WiSE-FT.

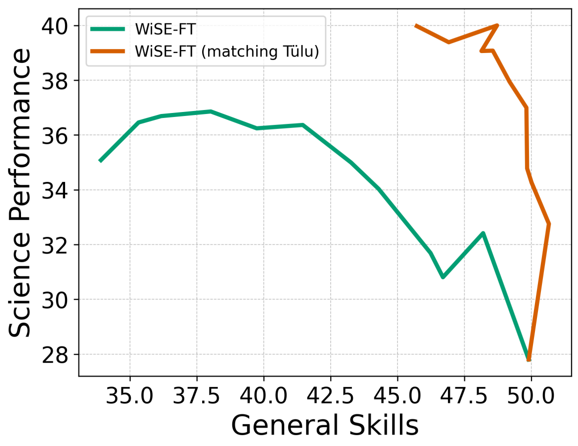

Figure 2: WiSE-FT performance on all of SciRIFF vs. all of SciRIFF mixed with a matching amount of Tülu data. A matching amount of general data in the mix leads to an improvement in skill-specific performance and a much smaller degradation in general skills.

We compare all three methods for all three settings in Figure 3, using all of the skill-specific data in every scenario. We see that both linear interpolation and WiSE-FT can achieve strong skill-specific performance, and occasionally best overall, but consistently experience dramatic degradation in general performance compared to PTM.

For linear interpolation, as one vector is more highly weighted the other is downweighted, showing clear trade-offs between both sets of evaluations as we vary ω.

For WiSE-FT, the findings seem strange at first glance. Why does continuing to finetune a strong generalist model on a specific skill degrade general performance *more* than adding a task vector, which was trained in parallel on the base model? We hypothesize that this is caused by the differences between the skill-specific and general training data distributions. In Figure 2, we compare WiSE-FT trained on only science and WiSE-FT train on a mixture of science and a matching subsample of Tülu data. We see that this new mixture, which by construction is more similar to the general instruction data distribution already seen by the base model, achieves stronger skill-specific *and* general performance compared to standard WiSE-FT. We leave it to future work to explore two follow up questions from this result:

1. How much general data is needed during CFT to preserve general performance?

2. When the base mix is not publicly available, is it possible to use data from a *different* general distribution to preserve general performance?

5 Related Work

Mitigating Forgetting During Finetuning The problem of reduced generality in language models during finetuning, and more generally during continual learning is a fundamental problem in gradient based learning and has been widely studied in prior work (McCloskey and Cohen, 1989; Goodfellow et al., 2013; Luo et al., 2023). Several techniques have been proposed to mitigate this issue, including regularization to minimize overfitting (Ahn et al., 2019; Lee et al., 2019) and recalling or replaying prior knowledge during training (Kirkpatrick et al., 2017; Röttger et al., 2024; Chen et al., 2020). More recently, it has been shown that parameter-efficient learning methods, particularly low-rank adaptation (Hu et al., 2021) forget less than conventional full-parameter finetuning methods. Unlike most of these methods, model merging does not require access to the original training data.

Model	General	Science	Coding	Safety	Exaggerated Refusals
Tülu Only	49.9	27.8	37.6	50.3	99.2
PTM (Science)	50.3	34.8	36.1	50.9	98.0
PTM (Safety)	50.3	25.6	36.9	89.3	90.0
PTM (Coding)	51.0	24.9	46.7	50.5	98.0
PTM (Science and Safety)	50.8	31.6	38.5	89.1	89.6
PTM (Science and Coding)	51.3	32.1	45.5	49.5	98.4
PTM (Safety and Coding)	52.1	18.8	45.2	85.0	92.4

Table 5: We take advantage of the efficiency of PTM to attempt to diagnose the degraded science performance in the three-skill PTM results shown in Table 4, with no additional training cost. By merging each specialized model pairwise, we see that safety and coding together show a large drop in science performance compared to the single skill and baseline models, suggesting that the drop in science performance in the three skill model is caused by interference between the other two skills.

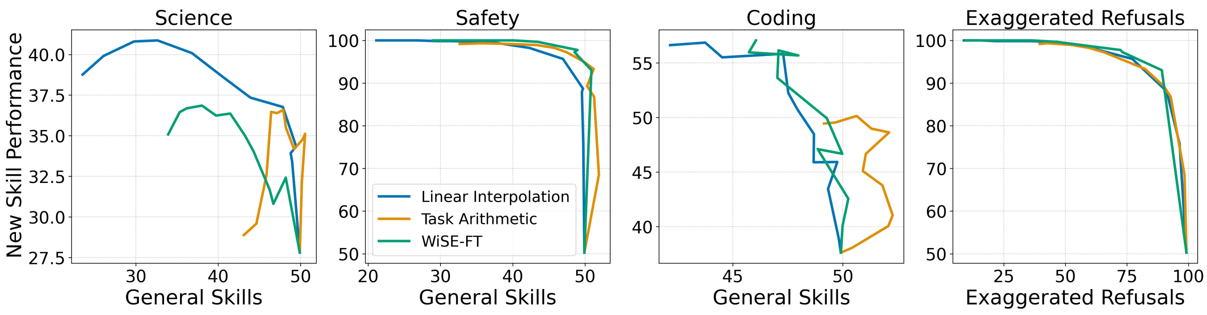

Figure 3: Plotting three PTM methods for each scenario. Both linear interpolation and WiSE-FT can achieve very strong domain-specific performance, at the cost of general performance and exaggerated refusals. While task arithmetic also improves in skill-specific performance, it preserves much more of the general skills.

Model Editing As discussed earlier, Ilharco et al. (2022) introduced an approach that used model interpolation between a pretrained model and a model fine-tuned on a downstream task. This is conceptually similar to our work here (though we focus on language), and one of the foundational papers that spawned a variety of work. Ilharco et al. (2023) introduced "task vectors", which are calculated as the difference (in weight space) between a pretrained model and that same model after finetuning. Wortsman et al. (2022b) introduced WiSE-FT, an approach of linearly interpolating between models, which they found to lead to increased distributional robustness. Wortsman et al. (2022a) introduced "model soups", made by weight-space averaging multiple models trained on the same dataset (using different hyperparameters, amounts of training data, etc.). Nylund et al. (2024) introduced "time vectors", showing that *temporal* information can also be created and applied to new models, similar to task vectors.

6 Conclusions

We explore the effectiveness of PTM for adding new skills to instruction-tuned models. We find that PTM is an efficient and effective method of augmenting preexisting models, enabling the addition of new skills with a fraction of the compute required compared to other common methods. In addition, we find that PTM achieves much better trade-offs between model safety and capability over other common methods tested. Finally, we report heuristics for selecting merging coefficients when held out data is not available, and find that these strategies together enable the addition of multiple skills into a single model with no additional training required.

Limitations

While we aim to be comprehensive in our experiments, focusing on specific skills is inherently limiting. Instruction-tuning is also one step in a larger, constantly evolving adaptation framework, and this work does not test models that have undergone processes such as reinforcement learning from human feedback after instruction-tuning. Additionally, our evaluations, while covering a broad set of capabilities, do not capture the full set of abilities models can exhibit, such as general reasoning, or more abstract concepts such as helpfulness.

References

Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. 2019. Uncertainty-based continual learning with adaptive regularization. *Advances in neural information processing systems*, 32.

AI@Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. *arXiv preprint arXiv:2004.12651*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Gemma-Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's mergekit: A toolkit for merging large language models. *arXiv preprint arXiv:2403.13257*.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *Preprint*, arXiv:2203.09509.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

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

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. *Preprint*, arXiv:2212.04089.

Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. 2022. Patching open-vocabulary models by interpolating weights. *ArXiv*, abs/2208.05592.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *Preprint*, arXiv:2311.10702.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

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

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2019. Mixout: Effective regularization to finetune large-scale pretrained language models. *arXiv preprint arXiv:1909.11299*.

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.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". 2023. Openorca: An open dataset of gpt augmented flan reasoning traces. `https://https://huggingface.co/Open-Orca/OpenOrca`.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. *Preprint*, arXiv:2109.07958.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Preprint*, arXiv:2305.01210.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *Preprint*, arXiv:2402.04249.

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

Kai Nylund, Suchin Gururangan, and Noah A. Smith. 2024. Time is encoded in the weights of finetuned language models.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Mark Rofin, Nikita Balagansky, and Daniil Gavrilov. 2022. Linear interpolation in parameter space is good enough for fine-tuned language models. *Preprint*, arXiv:2211.12092.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2024. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. *Preprint*, arXiv:2308.01263.

Shivalika Singh, Freddie Vargus, Daniel Dsouza, Börje F. Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura OMahony, Mike Zhang, Ramith Hettiarachchi, Joseph Wilson, Marina Machado, Luisa Souza Moura, Dominik Krzemiński, Hakimeh Fadaei, Irem Ergün, Ifeoma Okoh, Aisha Alaagib, Oshan Mudannayake, Zaid Alyafeai, Vu Minh Chien, Sebastian Ruder, Surya Guthikonda, Emad A. Alghamdi, Sebastian Gehrmann, Niklas Muennighoff, Max Bartolo, Julia Kreutzer, Ahmet Üstün, Marzieh Fadaee, and Sara Hooker. 2024. Aya dataset: An open-access collection for multilingual instruction tuning. *Preprint*, arXiv:2402.06619.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

David Wadden, Kejian Shi, Jacob Morrison, Aakanksha Naik, Shruti Singh, Nitzan Barzilay, Kyle Lo, Tom Hope, Luca Soldaini, Shannon Zejiang Shen, Doug Downey, Hannaneh Hajishirzi, and Arman Cohan. 2024. Sciriff: A resource to enhance language model instruction-following over scientific literature. *Preprint*, arXiv:2406.07835.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *Preprint*, arXiv:2306.04751.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt.

2022a. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *Preprint*, arXiv:2203.05482.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo-Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. 2022b. Robust fine-tuning of zero-shot models. *Preprint*, arXiv:2109.01903.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. *Preprint*, arXiv:2306.01708.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *Preprint*, arXiv:2311.03099.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. 2024. Opencodeinterpreter: Integrating code generation with execution and refinement. *Preprint*, arXiv:2402.14658.

A Training Details

A.1 Compute

All of our models were trained on v3-128 TPUs on the Google TPU Research Cloud, and models were merged with the publicly available mergekit (Goddard et al., 2024) toolkit.

A.2 Hyperparameters

We follow the hyperparameters used in Ivison et al., 2023:

- Precision: BFloat16
- Epochs: 2
- Weight decay: 0
- Warmup ratio: 0.03
- Learning rate: 2e-5
- Max. seq. length: 4,096
- Effective batch size: 128

B Other Results

Other Merging Algorithms We also briefly experiment with two merging algorithms that aim to minimize interference between models: TIES (Yadav et al., 2023) and DARE (Yu et al., 2024). We compare these in Table 6 and find that they do not improve performance over weighted averaging.

Exaggerated Refusals vs General Skills We directly compare the trade-offs between general skills and exaggerated refusals in Figure 4 and Figure 5. We see that modifying the mixture weight ω shows a clear relationship between the two skills, and that PTM shows large improvements in exaggerated refusals over both CFT and RT

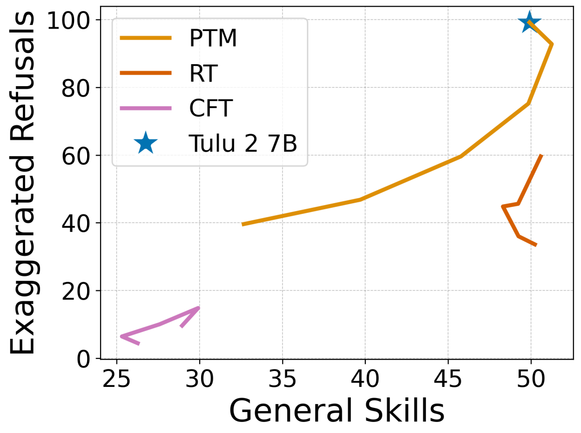

Figure 4: We show general skills versus exaggerated refusals, and show a clear relationship between the two skill sets. Additionally, for the same general performance, PTM achieves much higher exaggerated refusals compliance than RT and CFT.

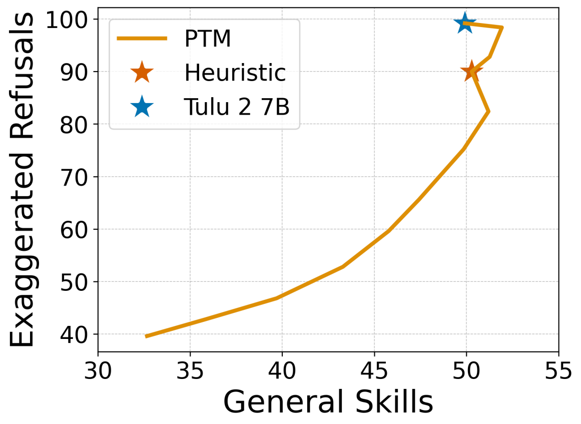

Figure 5: We show general skills versus exaggerated refusals, and highlight the point chosen by our heuristic, showing at most a small degradation in exaggerated refusal performance while preserving general skills.

Model	**General**	**Science**	**Coding**	**Safety**	**Exaggerated Refusals**
PTM (All 3)	51.1	26.6	45.3	84.0	93.2
TIES PTM (All 3)	51.2	28.0	44.5	82.7	92.8
DARE PTM (All 3)	49.9	24.7	45.4	84.7	92.4

Table 6: Results of using our heuristic to merge all three task vectors into a single model with standard weighted averaging PTM, TIES (Yadav et al., 2023), and DARE (Yu et al., 2024). TIES and DARE also do not mitigate interference relative to the base method on science, and three methods show similar performance across all skills, showing that other popular merging algorithms do not mitigate interference in this setting.

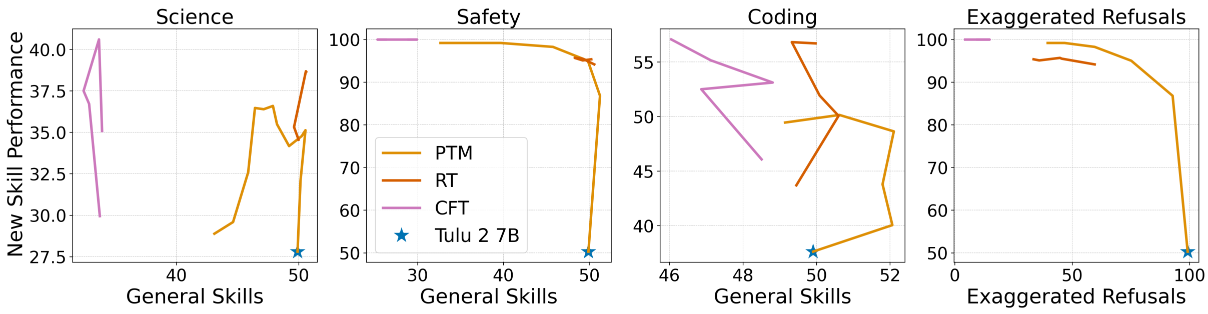

Figure 6: Curves showing general skills versus specialized skill performance for CFT, RT, and PTM across all three domains and exaggerated refusals. PTM consistently achieves strong specialized performance while preserving much more general performance compared to CFT. PTM also exhibits comparable performance to many RT checkpoints. In the case of exaggerated refusals, PTM shows a clear improvement over all other methods tested.