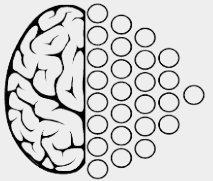
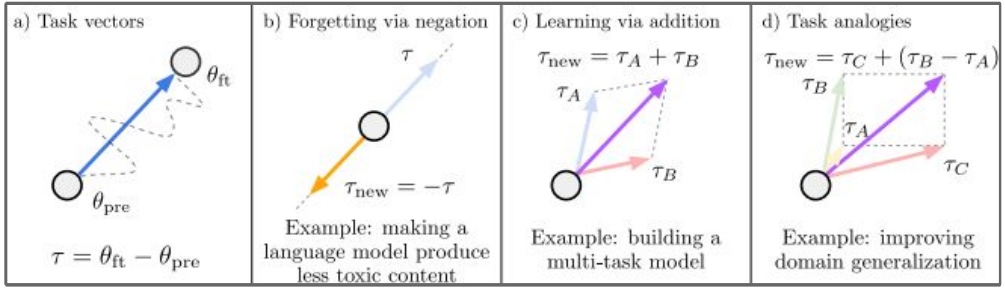


LLM Merging - Combined Paper discussion

Paper	Assigned to
Evolutionary Optimization of Model Merging Recipes	Sarang
Editing Models with Task Arithmetic	Tejas
WARM: On the Benefits of Weight Averaged Reward Models	Yash
Rewarded soups: towards Pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards	Shreyas V
Arcee's MergeKit: A Toolkit for Merging Large Language Models	Ankita
TIES-Merging: Resolving Interference When Merging Models	Karan
LoraHub: Efficient Cross-Task Generalization via Dynamic LoRA Composition	Aaron
ZipIt! Merging Models from Different Tasks without Training	Harsh
Model Ratatouille: Recycling Diverse Models for Out-of-Distribution Generalization	Sasmit

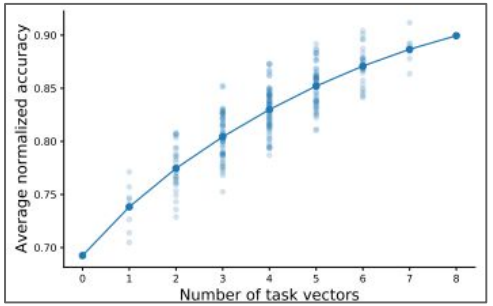
Drive link

ICLR 2023 - EDITING MODELS WITH TASK ARITHMETIC



Addition - Learning/Merging

* results in better multi-task models
* maintains 98.9% of the accuracy
* average perf. increases as more task Vectors are added

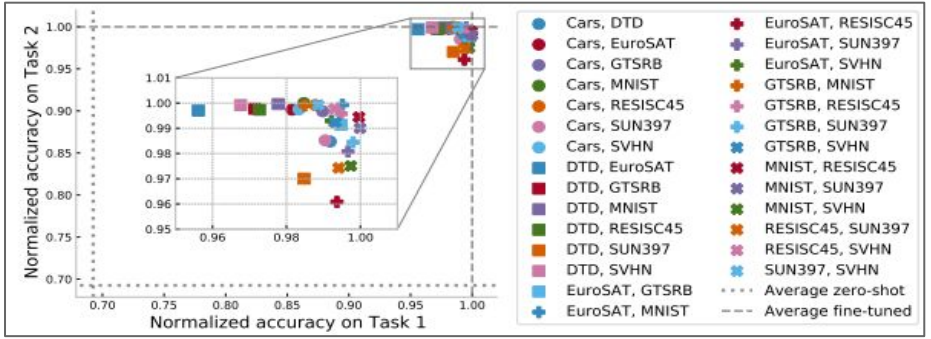


A **task** is defined by **a dataset and a loss function** used for fine-tuning.
A **task vector** specifies a direction in the weight space of a pre-trained model, such that movement in that direction improves performance on the task.

$$\theta_{new} = \theta + \lambda\tau_{new}$$

Negation - Forgetting or Unlearning
CV: CLIP - control task (ImageNet), Forget task (CARS, SUN397, DTD, MNIST)
NLP: Toxic generation in GPT-2, ~(finetune to produce toxic content)

Task Analogy - A:B :: C:D
Domain Generalise: Yelp - sentiment, unsup lm; Amazon - sentiment, unsup lm;
Subpopulation w/ little data: (Imagenet, human draw); (real dog:real lion :: sketch dog:sketch lion)
Kings & Queens: Kings:Queens :: Man:Woman (FT over pretrained CLIP classifier)

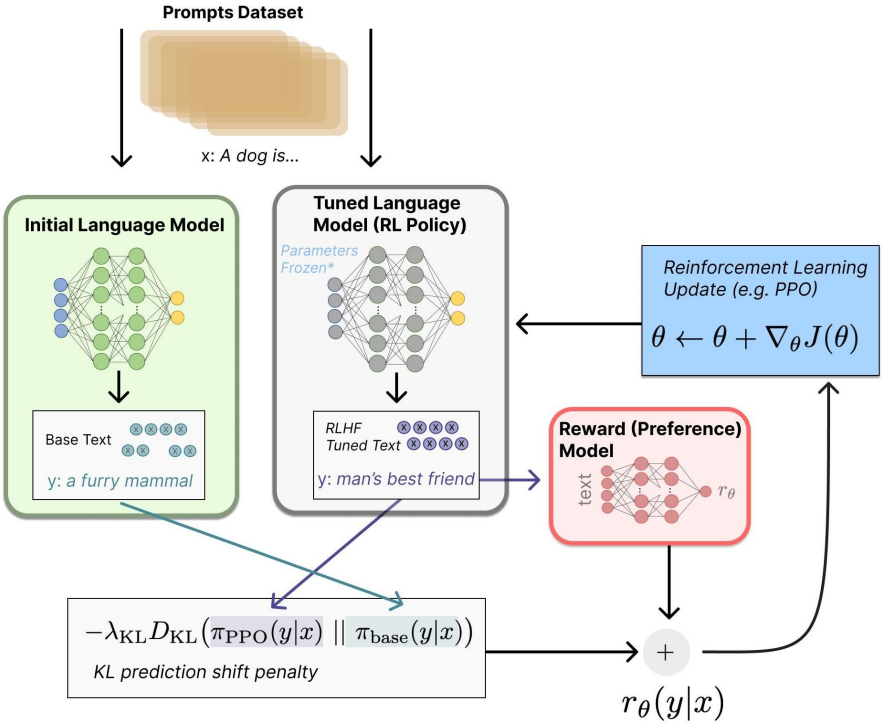


Adding pairs of task vectors from **image classification tasks**

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)

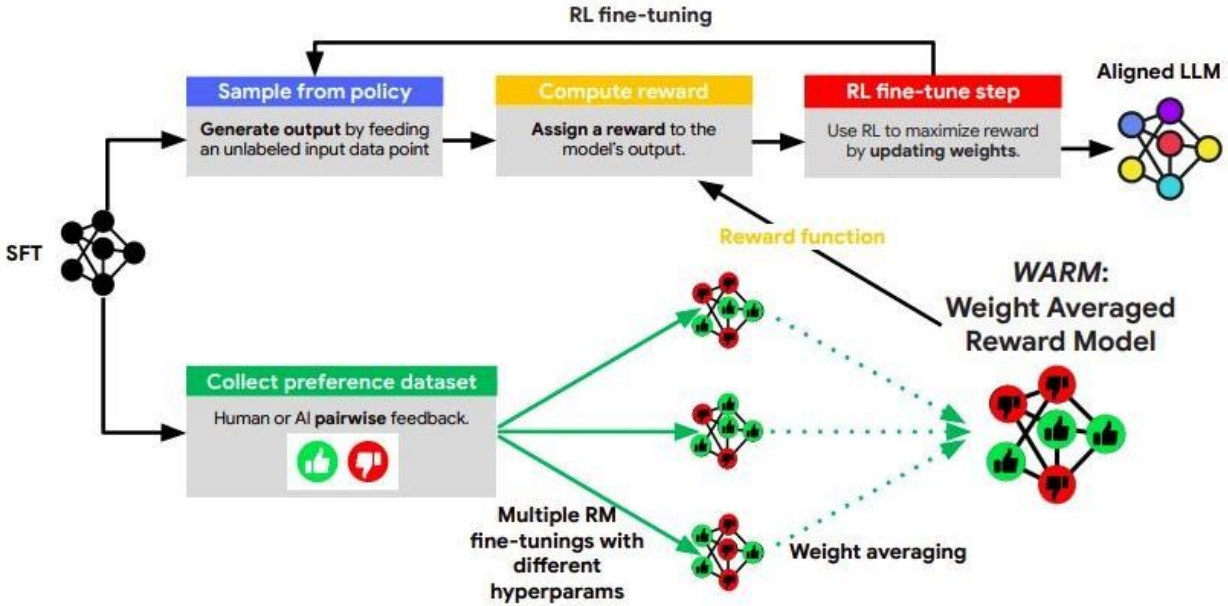
For four text classification tasks from the GLUE benchmark, adding task vectors downloaded from the Hugging Face Hub can improve accuracy of fine-tuned T5 models

RLHF



Source: https://huggingface.co/blog/rlhf

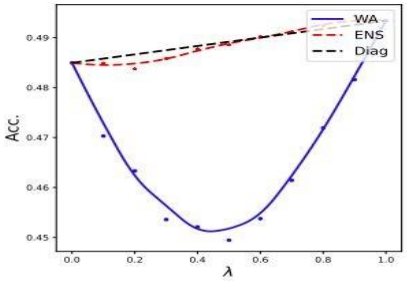
WARM



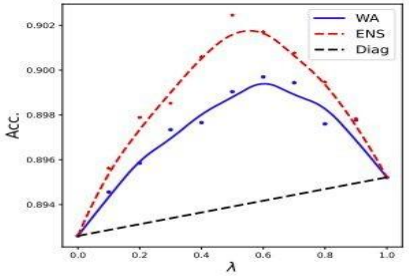
(a) *WARM* procedure with $M = 3$.

Source: https://arxiv.org/pdf/2401.12187

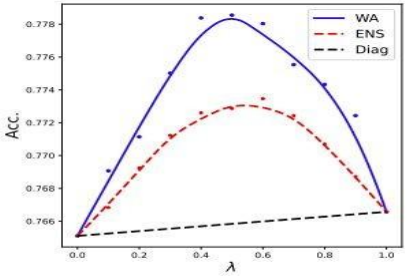
Experiment Results



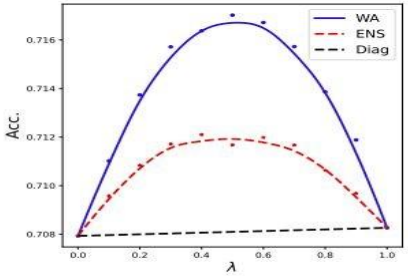
(a) Train (corrupt).



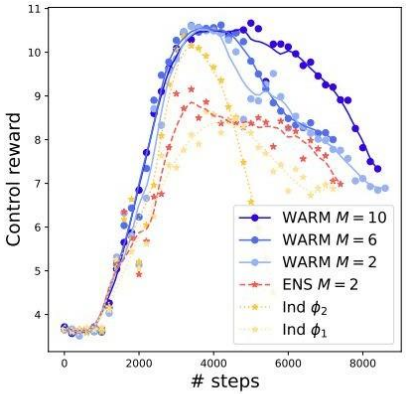
(b) Train (clean).

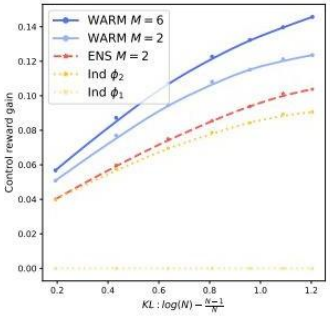


(c) Validation (ID).

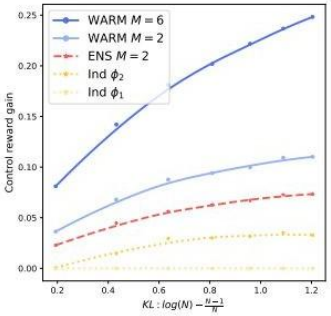


(d) Test (OOD).

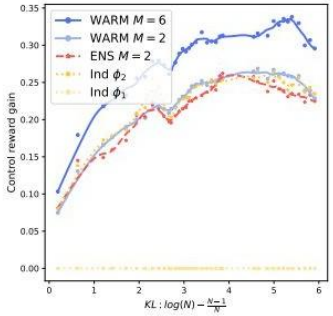




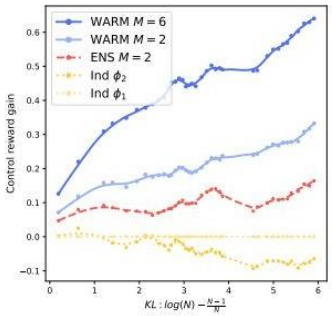
(a) PaLM (clean).



(b) PaLM (corrupt).



(c) T5 (clean).



(d) T5 (corrupt).

(b) *WARM* mitigates reward hacking.

LoRA Hub

Mashing together many LoRA modules

Problem Statement + Proposed Solution

Problem: LoRA Fine tuning is generally done on similar tasks and does not generalize well across multiple tasks.

Solution: Train many LoRA modules and use them together. This is done in 2 phases, Compose and Adapt.

Compose - Element wise composition of LoRA modules

$$\hat{m} = (w_1 A_1 + w_2 A_2 + \cdots + w_N A_N)(w_1 B_1 + w_2 B_2 + \cdots + w_N B_N).$$

Adapt - Weight optimization via Gradient Free methods

I couldn't understand what gradient free methods were used here. But the paper mentions Covariance Matrix Adaptive Evolution Strategies (CMA-ES). Authors did not use Gradient Descent.

Results

Authors used Flant-T5-Large and evaluated performance on Big-Bench Hard (BBH) Benchmark. Authors picked 20 random LoRA modules for the results.

Results on next slide.

QnA by authors:

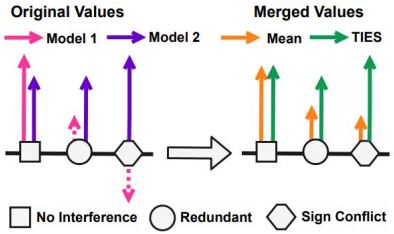
1. Does rank of LoRA module matter? Somewhat, in comparison of 4 to 64 ranks, 16 outperforms consistently.
2. Is more LoRA modules better? Not after a certain point, it may give better performance but variance in results increases drastically.

Table 1: Experimental results of zero-shot learning (Zero), few-shot in-context learning (ICL), IA3 fine-tuning (IA3), LoRA tuning (LoRA), full fine-tuning (FFT) and our proposed few-shot LoraHub learning (LoraHub) on the BBH benchmark with FLAN-T5-large as the base LLM. We denote algorithmic tasks with the superscript § following previous work (Wu et al., 2023b). Note that we employ three runs, each leveraging different 5-shot examples per task, as demonstrations for all few-shot methods. The average performance of all methods is reported below, and the best performance of each few-shot method can be found in the Appendix A.

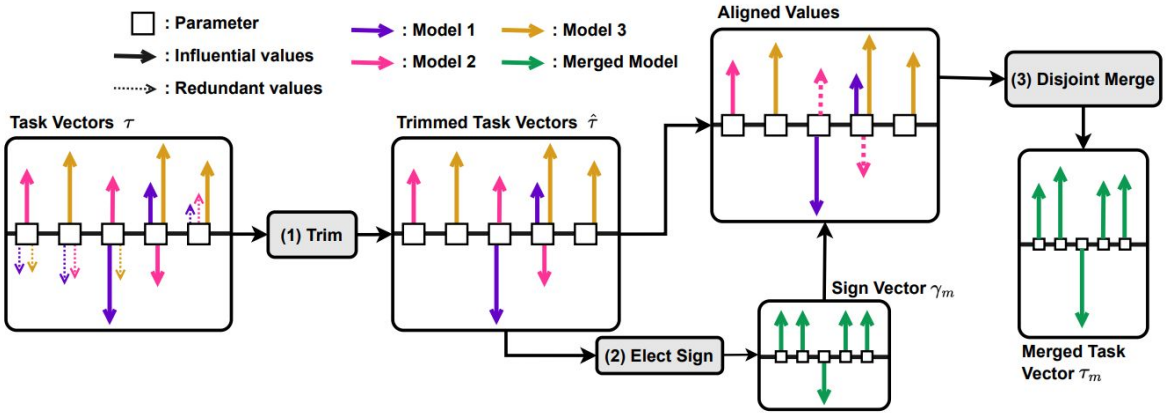
Task	Zero	ICL_{avg}	IA3_{avg}	LoRA_{avg}	FFT_{avg}	$\text{LoraHub}_{\text{avg}}$
Boolean Expressions	54.0	59.6	56.2	56.0	62.2	55.5
Causal Judgement	57.5	59.4	60.2	55.6	57.5	54.3
Date Understanding	15.3	20.4	20.0	35.8	59.3	32.9
Disambiguation	0.0	69.1	0.0	68.0	68.2	45.2
Dyck Languages	1.3	0.9	4.2	22.2	19.5	1.0
Formal Fallacies	51.3	55.3	51.5	53.6	54.0	52.8
Geometric Shapes	6.7	19.6	14.7	24	31.1	7.4
Hyperbaton	6.7	71.8	49.3	55.3	77.3	62.8
Logical Deduction§ (five objects)	21.3	39.1	32.7	40.0	42.2	36.1
Logical Deduction§ (seven objects)	12.7	40.7	33.8	37.3	44.9	36.8
Logical Deduction§ (three objects)	0.0	51.6	8.5	53.6	52.9	45.7
Movie Recommendation	62.7	55.8	61.8	51.5	66.0	55.3
Multistep Arithmetic	0.7	0.7	0.7	0.2	0.0	0.4
Navigate	47.3	45.3	46.2	48.0	48.0	47.1
Object Counting	34.7	32.4	35.1	38.7	35.6	33.7
Penguins in a Table	43.5	41.3	45.0	36.2	31.9	35.9
Reasoning about Colored Objects	32.0	40.2	40.7	39.6	37.6	40.0
Ruin Names	23.3	19.3	24.4	37.8	61.3	24.4
Salient Translation Error Detection	37.3	47.3	37.1	16.0	16.2	36.0
Snarks	50.0	54.2	53.9	55.6	66.7	56.9
Sports Understanding	56.0	54.7	55.1	56.5	54.0	56.7
Temporal Sequences	16.7	25.1	18.2	25.1	37.8	18.2
Tracking Shuffled Objects§ (five objects)	12.0	12.0	12.0	13.8	16.9	12.3
Tracking Shuffled Objects§ (seven objects)	6.7	6.7	6.7	10.0	9.8	7.7
Tracking Shuffled Objects§ (three objects)	24.7	31.1	30.7	30.9	32.0	29.2
Web of Lies	54.0	53.8	54.2	52.7	48.2	50.1
Word Sorting	1.3	0.5	1.3	4.9	4.9	1.1
Avg Performance Per Task	27.0	37.3	31.6	37.7	42.1	34.7
Avg Tokens Per Example	111.6	597.8	111.6	111.6	111.6	111.6
Gradient-based Training	No	No	Yes	Yes	Yes	No

TIES-MERGING: Resolving Interference When Merging Models

[arXiv][GitHub]



Prateek Yadav[1] **Derek Tam**[1]
Leshem Choshen[2,3] **Colin Raffel**[1] **Mohit Bansal**[1]
[1] University of North Carolina at Chapel Hill [2] IBM Research [3] MIT
leshem.choshen@ibm.com
{praty,dtredsox,craffel,mbansal}@cs.unc.edu



Algorithm 1 TIES-MERGING Procedure.

Input: Fine-tuned models $\{\theta_t\}_{t=1}^{n}$, Initialization θ_{init}, k, and λ.
Output: Merged Model θ_m
forall t ***in*** $1, ..., n$ **do**
  ▷ Create task vectors.
  $\tau_t = \theta_t - \theta_{\text{init}}$
  ▷ Step 1: Trim redundant parameters.
  $\hat{\tau}_t \leftarrow \text{keep\_topk\_reset\_rest\_to\_zero}(\tau_t, k)$
  $\hat{\gamma}_t \leftarrow sgn(\hat{\tau}_t)$
  $\hat{\mu}_t \leftarrow |\hat{\tau}_t|$
end
▷ Step 2: Elect Final Signs.
$\gamma_m = sgn(\sum_{t=1}^{n} \hat{\tau}_t)$
▷ Step 3: Disjoint Merge.
forall p ***in*** $1, ..., d$ **do**
  $\mathcal{A}^p = \{t \in [n] \mid \hat{\gamma}_t^p = \gamma_m^p\}$
  $\tau_m^p = \frac{1}{|\mathcal{A}^p|} \sum_{t \in \mathcal{A}^p} \hat{\tau}_t^p$
end
▷ Obtain merged checkpoint
$\theta_m \leftarrow \theta_{\text{init}} + \lambda * \tau_m$
return θ_m

Method	T5-base	$(IA)^3$
TIES-MERGING	**74.5**	**70.7**
– TRIM	73.0	70.6
– ELECT	73.1	69.6
– DISJOINT MEAN	72.6	67.5
– SCALE	72.0	65.5

Table 6: Ablation on all the steps of TIES-MERGING.

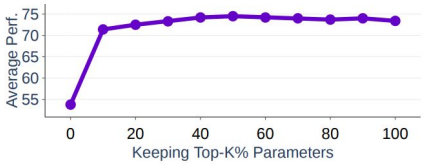


Figure 3: **Performance depends on a small fraction of high-magnitude parameters.** For each task vector, we keep only the largest - top-k% parameters and plot the average performance across eleven tasks. Keeping only the top-20% of the parameter does not degrade the performance.

Figure 4: **Sign conflicts occur even after trimming and increase with the number of models.** We plot the fraction of parameters that have a sign conflict after trimming versus the number of models being merged.

Method (↓)	Validation	PEFT	Full Finetuning			
Model (→)		$(IA)^3$	T5-Base	T5-Large	ViT-B/32	ViT-L/14
FINE-TUNED	-	71.4	82.8	88.8	90.5	94.2
MULTITASK	-	73.1	83.6	88.1	88.9	93.5
AVERAGING [82, 9]	✗	-	65.9	59.6	65.8	79.6
TASK ARITHMETIC [29]	✗	-	**73.2**	73.5	60.4	83.3
TIES-MERGING	✗	-	69.7 [-3.2]	**74.4** [+0.9]	**72.4** [+6.6]	**86.0** [+2.7]
FISHER MERGING [45]	✓	62.2	68.9	64.6	68.3	82.2
REGMEAN [31]	✓	58.0	71.2	73.2	71.8	83.7
TASK ARITHMETIC [29]	✓	63.9	73.2	73.3	70.1	84.5
TIES-MERGING	✓	**66.4** [+2.5]	**73.9** [+0.7]	**76.9** [+3.6]	**73.6** [+1.8]	**86.0** [+1.5]

Table 1: Comparing model merging methods across multiple fine-tuning settings and modalities (NLP and Vision) with and without the availability of a validation set.

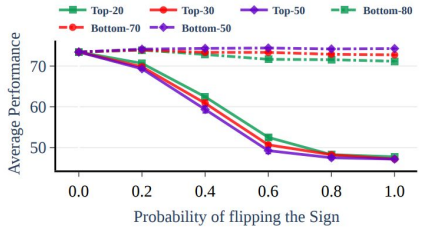


Figure 7: **Flipping the signs of high magnitude parameters leads to catastrophic performance drops.** Average Performance when flipping the directions of Top-k% and Bottom-k% parameters for each task. We report the results averaged over eleven $(IA)^3$ tasks.

Method	Average
Fine-Tuned	71.4
Multitask	73.1
Averaging [9, 82]	58.0
Task Vectors [29]	63.9
TIES-MERGING	**66.4**
TIES-MERGING (Oracle Sign)	**72.0** [+5.6]

Table 5: **TIES-MERGING can perform close to multitask models if the signs can be estimated correctly.** We use the signs from the multitask vector as the elected sign and perform merging and report the performance.

Model	T5-Base	T5-Large
Zeroshot	31.1	27.6
Simple Averaging [9, 82]	31.7	30.4
Fisher [45]	33.8	32.0
RegMean [31]	34.3	36.0
Task Arithmetic [29]	31.9	32.3
TIES-MERGING	**35.3** [+1.0]	**40.4** [+4.4]

Table 2: **TIES-MERGING generalizes better.** Out of Distribution Generalization for T5-Base and T5-Large on six held-out tasks.

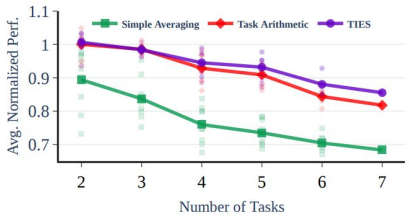


Figure 5: **TIES-MERGING scales better.** Average performance when merging a different number of tasks.

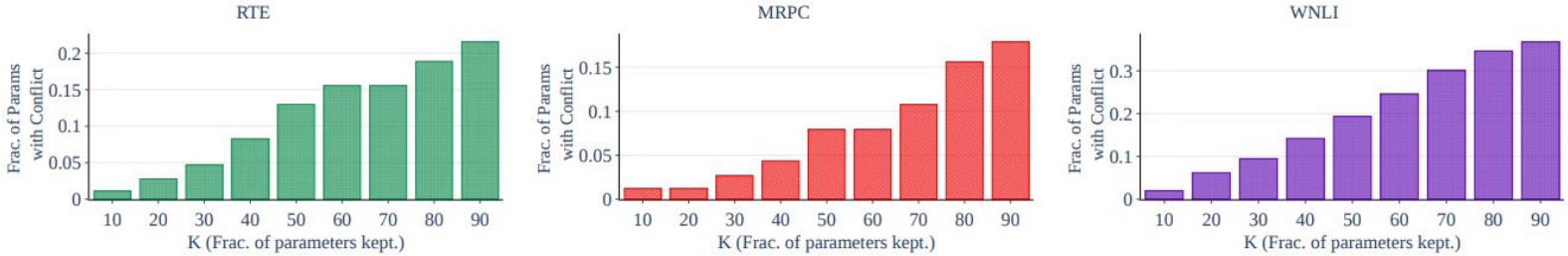


Figure 9: **Sign conflict increases as we trim less parameters.** For each task, we merge 10 different checkpoints from hunggingface hub and plot the sign conflict as a function of keeping only the top-k% parameters.

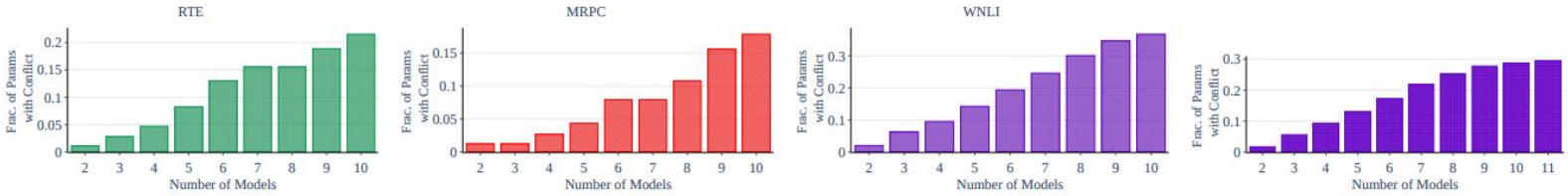


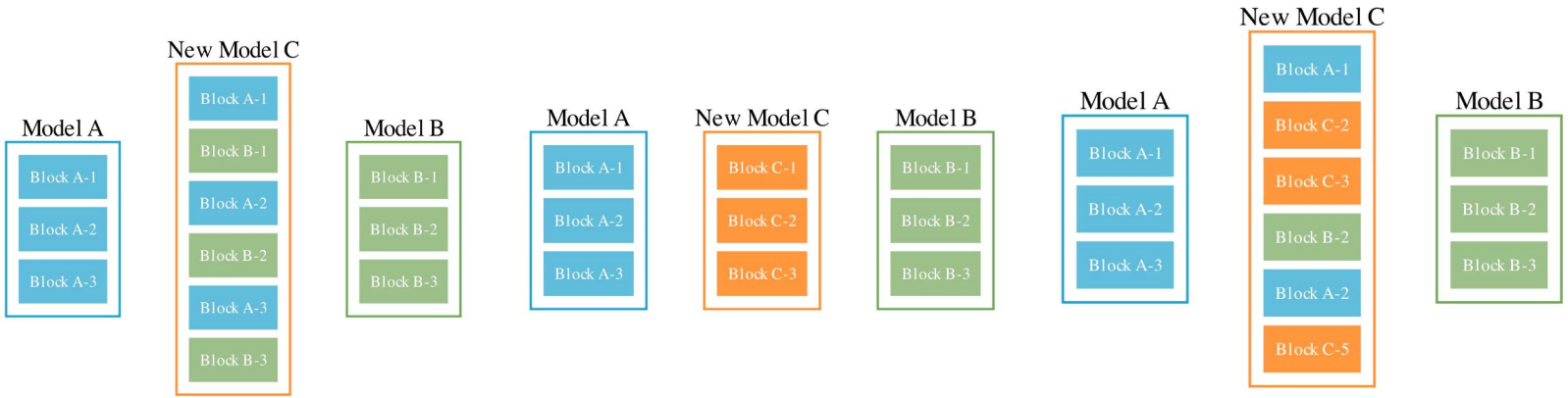
Figure 10: **Sign Conflict exists even when merging multiple checkpoints for the same task.** The first three plots are for RTE, MRPC, WNLI datasets when merging 10 Huggingface checkpoints, while the last one is when merging different tasks (Figure 4 from the main paper).

Evolutionary Optimization of Model Merging Recipes

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, David Ha
Sakana AI

- Uses evolutionary methods to discover non-trivial ways to merge models.
- Outperforms human heuristic and intuition based model merging.
- Think NAS and NEAT.
- Implements Covariance Matrix Adaptation Evolution Strategy
- **Parameter Space**: Uses TIES-Merging enhanced with DARE.
- **Data Flow Space**: Literature limits itself to sequential ordering of layers. (Unstable!)

Optimizes both Data Flow Space and Parameter Space



*Merging Models in the **Data Flow Space** (Layers)*

*Merging Models in the **Parameter Space** (Weights)*

*Merging Models in both **Data Flow Space** and **Parameter Space***

Experiments

- Japanese LLM + English Math LLM -> Japanese Math LLM
- Japanese LLM + English VLM -> Japanese VLM.

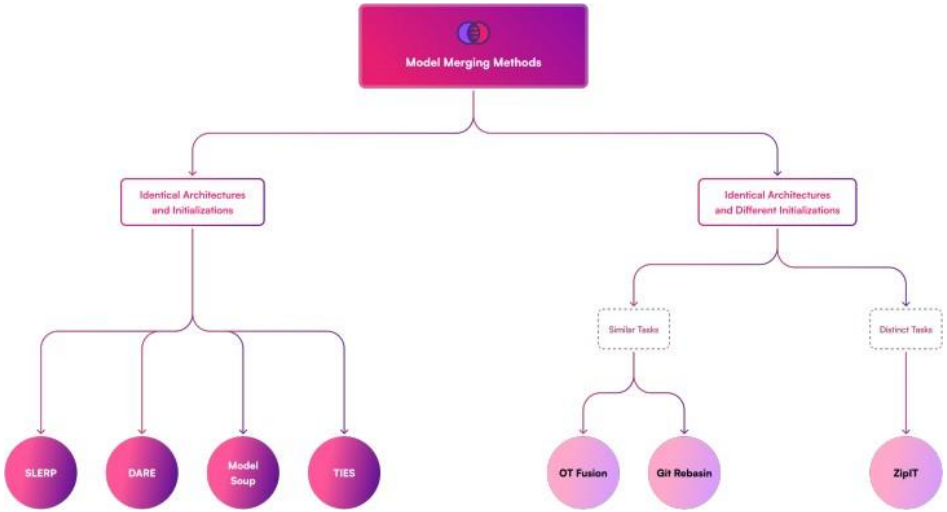
		JP Language Model Evaluation Harness									
Model	**Size**	JComQA	JNLI	MARC	JSQuAD	JAQKET	XLSum	XWino	MGSM	JCoLA	**Avg**
Shisa Gamma 7b v1	7B	91.2	72.1	94.6	73.9	68.0	25.9	80.5	29.6	58.7	66.1
WizardMath 7B V1.1	7B	74.7	42.7	90.4	84.6	68.5	22.3	69.8	38.8	48.9	60.1
Abel 7B 002	7B	70.3	51.8	62.3	83.8	69.0	22.5	68.2	28.0	52.7	56.5
Ours (PS)	7B	89.1	65.7	95.4	89.5	77.7	25.5	81.2	50.0	60.5	**70.5**
Ours (DFS)	10B	67.7	58.2	53.5	66.8	54.3	17.3	65.6	30.0	65.6	**53.2**
Ours (PS+DFS)	10B	88.2	50.3	91.5	78.6	77.8	23.2	73.0	40.0	73.0	**66.2**
Llama 2 70B	70B	80.2	53.4	94.4	91.6	80.1	21.8	73.6	30.4	54.6	64.5
Japanese Stable LM 70B	70B	91.2	50.4	92.9	87.1	88.4	24.3	82.0	37.2	61.7	68.3
Swallow 70B	70B	95.3	57.2	91.7	94.1	93.9	23.1	83.3	45.2	59.5	71.5

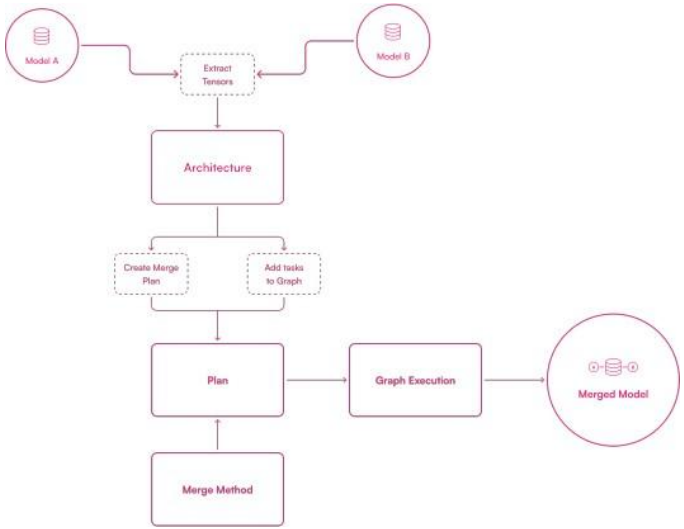
Type	**Size**	**MGSM-JA** (acc ↑)	**JP-LMEH** (avg ↑)
JA general	7B	9.6	66.1
EN math	7B	18.4	60.1
EN math	7B	30.0	56.5
1 + 2 + 3	7B	**52.0**	**70.5**
3 + 1	10B	**36.4**	**53.2**
4 + 1	10B	**55.2**	**66.2**
EN general	70B	18.0	64.5
JA general	70B	17.2	68.3
JA general	70B	13.6	71.5

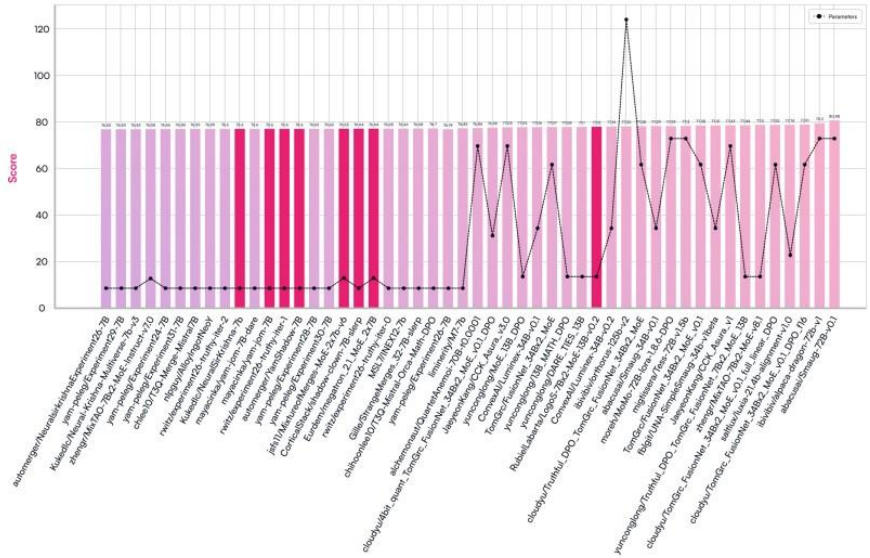
[Sakana.ai's Blog]
[Arcee's Blog]
[Github - mergekit-evolve]

Arcee's MergeKit: A Toolkit for Merging Large Language Models

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers,
Vlad Karpukhin, Brian Benedict, Mark McQuade, Jacob Solawetz
Arcee, Florida, USA







Model	Medical Benchmarks			General Benchmarks		
	USMLE	MedMCQA	PubMedQA	Arc Challenge	HellaSwag	MMLU
Llama2-7B-Chat (Touvron et al., 2023)	35.90	35.45	73.40	44.20	55.40	46.37
Meditron-7B (Chen et al., 2023)	38.40	24.07	71.40	40.20	54.50	33.06
MeditronLlama-7B-Lerp	39.10	36.65	**75.60**	46.76	58.66	**48.44**
MeditronLlama-7B-Slerp	**39.20**	**36.91**	**75.60**	**46.84**	**58.67**	47.97
MeditronLlama-7B-Dare-Ties	36.37	27.56	72.20	42.92	54.79	41.17
MeditronLlama-7B-Ties	38.73	32.27	**75.60**	45.05	58.23	45.03

ZipIt! Merging Models from Different Tasks without Training
ICLR 2024

Background and Motivation

- Previous work focuses on merging 2 models trained on same task and merge entire network, and involves permuting one model to the other
- Permuting creates a 1-1 mapping between the models, assuming a correlation in features - fine for models on same task, but not so much for different tasks
- Merging whole network forces merging models on similar tasks, not ideal for merging models trained on different tasks
- Models fine tuned from same ckpt / initalization - lie in same error/loss basin (This paper does not utilize this fact) $W_i^* = \gamma W_i^A + (1-\gamma) W_i^B$
- Models with different initialization - models permuted to same loss basin can be merged by averaging weight

$$W_i^* = \gamma W_i^A + (1-\gamma) P_i W_i^B P_{i-1}^T$$

- Git Re-basin - interpolates weights by using similarity in weights not good for merging models trained on different tasks as it may find a minima that performs worse for both. Permute relies on the assumption that likelihood of both models lie in same basin

Method

- **What does it do ? -** Merges models trained on different tasks, and allows for merging within or across a model, with also the ability to to merge models partially.
- **Approach**: Generalize merges and define unmerges, using the 'zip' and 'unzip' operation
- **Why within ? -** Reduces feature redundancy in a model, paper proves that loss increase has tighter bounds than previous work. Models merged within perform on par if not better than model not merged within.
- **Merging -** $f_i = L_i(x) = W_i x + b_i$

  ->Concatenate both feature vectors - $f_i^A \| f_i^B \in \mathbb{R}^{2n_i}$

  ->Find correlation between every element in this (prev work only did between fa and fb)

  ->if well correlated average them - find ni pairs per 2i features - $M_i \in \mathbb{R}^{n_i \times 2n_i}$

  -> equivalent to normal averaging, but more general $f_i^* = M_i \left(f_i^A \| f_i^B\right)$
- **Unmerging -** U is pseudoinverse of M

  Split U in 2 halves, merge is lossy $U_i f_i^* \approx f_i^A \| f_i^B \quad 2M_i^T$

  $U_i^A, U_i^B \in \mathbb{R}^{n_i \times n_i}$

$$f_{i+1}^A \approx L_{i+1}^A (U_i^A f_i^*) \qquad f_{i+1}^B \approx L_{i+1}^B (U_i^B f_i^*)$$

ZipIt! Merging Models from Different Tasks without Training
ICLR 2024

Zip Operation

$$W_i^* = M_i^A W_i^A U_{i-1}^A + M_i^B W_i^B U_{i-1}^B$$

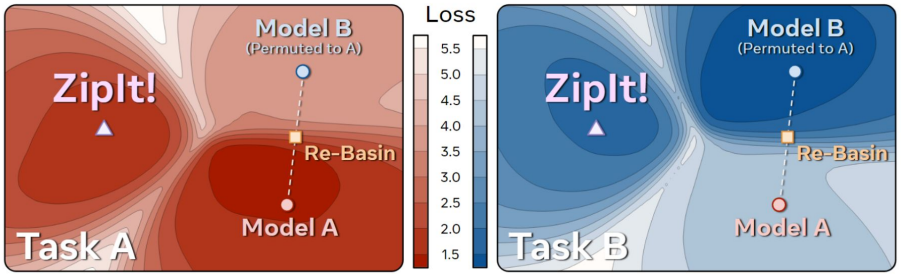
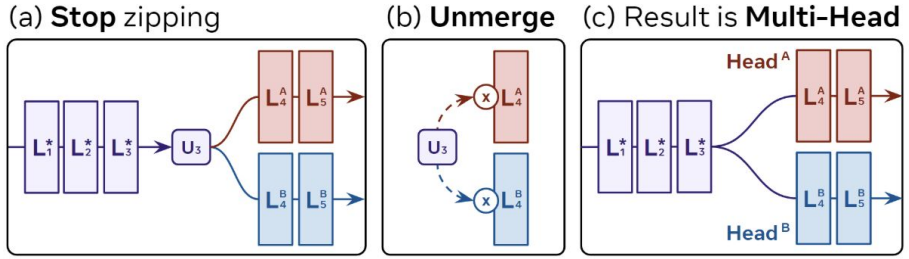
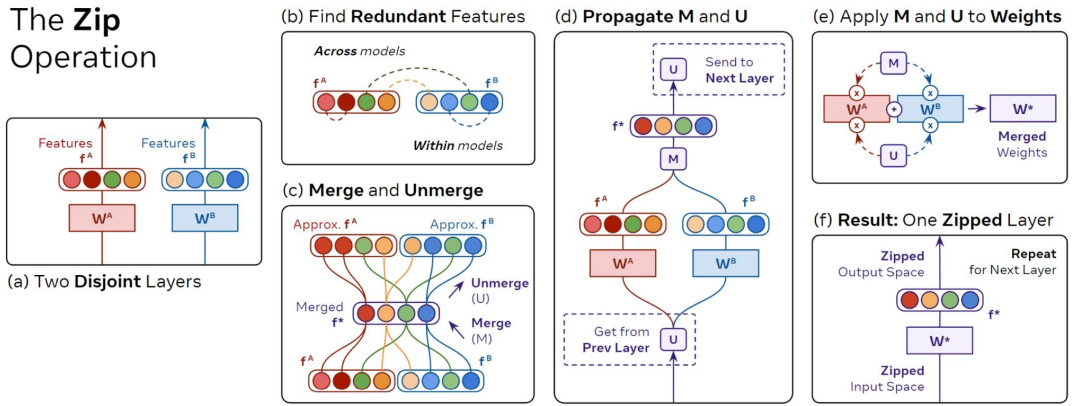


Figure 2: **Task Loss Landscapes** for models in Tab. 1b. Model A and Model B lie in low loss





ZipIt! Merging Models from Different Tasks without Training
ICLR 2024

Method	FLOPs (G)	Accuracies (%) Joint	Task A	Task B	Avg
Model A	0.68	48.2±1.0	97.0±0.6	45.1±8.6	71.0±4.4
Model B	0.68	48.4±3.8	49.1±9.3	96.1±1.1	72.6±4.9
W. Avg (Eq. 1)	0.68	43.0±1.6	54.1±1.4	67.5±1.2	60.8±4.5
Git Re-Basin‡	0.68	46.2±0.8	76.8±8.9	82.7±5.1	79.8±6.5
Permute (Eq. 2)	0.68	58.4±6.8	86.6±2.1	87.4±1.1	87.4±1.4
ZipIt!$_{20/20}$	0.68	**79.1**±1.1	**92.9**±1.1	**91.2**±1.4	**92.1**±1.0
Ensemble	1.37	87.4±2.6	97.0±0.6	96.1±1.1	96.6±0.4
ZipIt!$_{13/20}$	0.91	**83.8**±3.1	**95.1**±0.7	**94.1**±1.5	**94.6**±0.6

(a) **CIFAR-10 (5+5).** ResNet-20 (4× width).

Method	FLOPs (G)	Accuracies (%) Joint	Task A	Task B	Avg
Model A	2.72	41.6±0.3	82.9±0.7	24.8±0.4	53.9±0.5
Model B	2.72	41.6±0.2	25.1±1.2	82.8±0.2	54.0±0.6
W. Avg (Eq. 1)	2.72	17.0±1.7	23.8±6.9	24.8±5.9	24.3±1.9
Git Re-Basin‡	2.72	40.9±0.2	57.3±1.5	56.7±0.7	57.0±0.8
Permute (Eq. 2)	2.72	42.8±0.7	61.6±1.4	60.5±0.5	61.0±0.8
ZipIt!$_{20/20}$	2.72	**54.9**±0.8	**68.2**±0.8	**67.9**±0.6	**68.0**±0.4
Ensemble	5.45	73.5±0.4	82.9±0.7	82.8±0.2	82.8±0.4
ZipIt!$_{13/20}$	3.63	**70.2**±0.4	**80.3**±0.8	**80.1**±0.7	**80.2**±0.6

(b) **CIFAR-100 (50+50).** ResNet-20 (8× width).

Method	FLOPs (G)	Accuracies (%) Joint	Task A	Task B	Avg
Model A	4.11	37.2±2.0	74.3±4.0	0.5±0.1	37.4±2.0
Model B	4.11	35.3±1.6	0.5±0.1	70.5±3.2	35.5±1.6
W. Avg (Eq. 1)	4.11	0.3±0.1	0.6±0.1	0.7±0.1	0.6±0.1
Git Re-Basin‡	4.11	3.1±1.2	5.3±2.6	5.7±2.4	5.5±1.7
Permute (Eq. 2)	4.11	**8.6**±5.8	**10.1**±4.4	**15.3**±11.1	**12.7**±7.7
ZipIt!$_{50/50}$	4.11	**8.6**±4.7	**12.4**±5.9	**14.7**±7.8	**13.5**±6.6
Ensemble	8.22	63.3±4.9	74.3±4.0	70.5±3.2	72.4±2.5
ZipIt!$_{22/50}$	6.39	55.8±4.1	65.9±2.5	64.1±3.0	65.0±2.3
ZipIt!$_{10/50}$	7.43	**60.9**±4.1	**70.7**±3.0	**69.0**±2.9	**69.9**±1.9

Method	FLOPs (G)	Per-Task Accuracies (%) SD	OP	CUB	NAB	Avg
Merging Pairs						
W. Avg (Eq. 1)	4.11	12.9	18.2	13.9	0.2	11.3
Permute (Eq. 2)	4.11	46.2	47.6	35.6	**13.5**	35.7
ZipIt!$_{49/50}$	4.11	**46.9**	**50.7**	**38.0**	12.7	**37.1**
Ensemble	8.22	72.7	81.1	71.0	77.2	75.5
ZipIt!$_{22/50}$	6.39	62.6	71.2	62.8	53.0	62.4
ZipIt!$_{10/50}$	7.42	**66.5**	**75.8**	**65.6**	**66.8**	**68.7**
Merging All 4						
W. Avg (Eq. 1)	4.12	0.8	3.0	0.6	0.3	1.2
Permute (Eq. 2)	4.12	15.7	26.1	**14.0**	**5.3**	15.3
ZipIt!$_{49/50}$	4.12	**21.1**	**33.3**	8.6	3.9	**16.8**
Ensemble	16.4	72.7	81.2	71.0	77.2	75.5
ZipIt!$_{22/50}$	11.0	50.2	55.9	44.0	32.0	45.5
ZipIt!$_{10/50}$	14.1	**63.5**	**70.8**	**63.7**	**63.1**	**65.3**