# REALISTIC EVALUATION OF MODEL MERGING FOR COMPOSITIONAL GENERALIZATION

Anonymous authors

Paper under double-blind review

#### ABSTRACT

Merging has become a widespread way to cheaply combine individual models into a single model that inherits their capabilities and attains better performance. This popularity has spurred rapid development of many new merging methods, which are typically validated in disparate experimental settings and frequently differ in the assumptions made about model architecture, data availability, and computational budget. In this work, we characterize the relative merits of different merging methods by evaluating them in a shared experimental setting and precisely identifying the practical requirements of each method. Specifically, our setting focuses on using merging for *compositional generalization* of capabilities in image classification, image generation, and natural language processing. Additionally, we measure the computational costs of different merging methods as well as how they perform when scaling the number of models being merged. Taken together, our results clarify the state of the field of model merging and provide a comprehensive and rigorous experimental setup to test new methods.

025 026

027

004

010 011

012

013

014

015

016

017

018

019

021

#### 1 INTRODUCTION

028 The release of performant pretrained models like LLaMA (Touvron et al., 2023a;b), Stable Diffusion 029 (Rombach et al., 2022; Podell et al., 2023), and CLIP (Radford et al., 2021), alongside the development of efficient ways to fine-tune large models (Dettmers et al., 2024), has led to a widespread proliferation of fine-tuned models that are specialized to specific use cases. Hundreds of thousands of these 031 specialized models are shared in repositories like the HuggingFace Hub.<sup>1</sup> Model merging (Raffel, 2023) aims to recycle specialized models to create new improved models that generalize to new 033 settings. Merging methods vary in sophistication from simply averaging the model parameters 034 (Choshen et al., 2022) to solving a linear system of equations that captures the importance of each 035 parameter (Tam et al., 2023). Beyond retaining—or improving—performance on tasks the constituent models were trained on, merging aims to compositionally combine the capabilities of the constituent 037 models, thus enabling generalization to new tasks. Model merging has exploded in popularity as it 038 provides an effective and extremely low-cost way to create new models-for example, many of the 039 top models on the Open LLM Leaderboard<sup>2</sup> were created through model merging.

The growing popularity of merging has led to the recent development of many new merging methods (Tam et al., 2023; Matena & Raffel, 2022; Ilharco et al., 2022; Yadav et al., 2023; Jin et al., 2022; Yang et al., 2024a; Yu et al., 2023; Zhao et al., 2024b; Shah et al., 2023, *inter alia*). This proliferation naturally raises the question: Which method works best in a particular settings? However, different merging methods are rarely evaluated in the same experimental setting, which makes comparison challenging. Additionally, merging methods impose different requirements in terms of computational costs, data availability, and hyperparameter tuning.

Our goal in this work is to better characterize the performance and merits of different merging methods by providing a rigorous comparison in a shared experimental setting. Most past work evaluates merging methods using the merged model's multitask performance on held-in tasks (i.e., on the tasks the original constituent models were trained on). Instead, we primarily focus on measuring generalization to new tasks that require the compositional combination of capabilities. Apart from

<sup>&</sup>lt;sup>1</sup>https://huggingface.co/docs/hub/en/index

<sup>&</sup>lt;sup>2</sup>https://huggingface.co/spaces/HuggingFaceH4/open\_llm\_leaderboard

being more challenging, we argue compositional generalization provides a more realistic motivation
for merging: assuming that merging does not improve performance on the held-in tasks (which it
seldom does), the primary benefit of using merging to create a multitask model is storage savings
because one can always use the corresponding constituent model for a held-in task. On the other
hand, compositional generalization can unlock new capabilities and applications that the individual
models lack.

060 To ensure our results are not modality-specific, we benchmark across image classification, image 061 generation, and natural language processing. Since merging methods can be used to combine 062 an arbitrary number of models, we also measure how each method scales with the number of 063 merged models. In addition, we explicitly enumerate extra requirements of each method, including 064 hyperparameters that require tuning, the amount of compute required, and whether auxiliary data is required for merging. Finally, to better contextualize the performance of merging methods, we 065 compare to the often-neglected baselines of multitask training, single-model performance, and the 066 performance of the pretrained base model. 067

Our experimental results provide new insights that shed light on the promises and shortcomings of existing merging methods. Overall, we find that merging performance depends on the application for example, we find that held-in task performance and generalization task performance are correlated in image classification but are *anticorrelated* for natural language processing. Additionally, we find that increasing the number of models being merged tends to result in *worse* multitask performance on held-in tasks but *better* generalization performance on unseen tasks. As a whole, our results clarify the state of the field and highlight many paths forward for improving model merging. To encourage realistic and comprehensive evaluation of future merging methods, we release our code.<sup>3</sup>

076 077

078

#### 2 BACKGROUND

079 Model merging aims to cheaply combine models that share an architecture and an initialization (i.e., a pretrained model) in order to create an aggregate model that retains the capabilities of the 081 individual models. We refer to the models being merged as the "constituent" models, which typically 082 are fine-tuned on different datasets that cover different tasks and/or domains and therefore have 083 complementary capabilities. We refer to the datasets, tasks, and/or domains that the constituent 084 models are trained on as "held-in". The specific goal of merging can vary, but can include improving 085 performance on a target task, creating a multitask model, retaining the capabilities of a base model, or 086 generalizing to new tasks. A popular use case of merging involves combining fine-tuned variants of 087 Stable Diffusion that have been specialized to improve the quality of a particular style or object type 088 (e.g., merging a "lego style" model with a "cute cat" model to generate cats made out of legos) (Shah et al., 2023; Zhong et al., 2024; Gu et al., 2023; Yang et al., 2024b). Merging has also been widely 089 applied to combining fine-tuned variants of open-source language models to improve and broaden 090 the capabilities of the base language model (Yadav et al., 2023; Yu et al., 2023). 091

092

#### 2.1 MERGING METHODS

094

096

097

098

099

100

An exhaustive comparison of merging methods is beyond the scope of this work, so we focus on eight popular merging methods that represent the diversity of approaches and discuss additional methods in Section 5. We use  $\theta_m$  to denote the parameters of the merged model,  $\theta_i$  with  $i \in \{1, \ldots, M\}$  as the *M* constituent models being merged, and  $\theta_p$  as the base model which we assume the constituent models are fine-tuned from. In this work, we assume all models are "open vocabulary", i.e., they use natural language for classification or generation and do not require task-specific classification heads.

101 Simple Averaging (McMahan et al., 2017; Wortsman et al., 2022b; Choshen et al., 2022) uses an 102 element-wise average of constituent model parameters  $\theta_m = \frac{1}{M} \sum_i \theta_i$ .

SLERP (Shoemake, 1985) interpolates models along the curved path connecting them (instead of along the direct line used in simple averaging). To merge more than two models, we use MLERP (Kim et al., 2024), which computes a norm-preserving average.

<sup>107</sup> 

<sup>&</sup>lt;sup>3</sup>Redacted for anonymity.

**Task Arithmetic** (Ilharco et al., 2022) constructs a "task vector"  $\tau_i = \theta_i - \theta_p$  for each constituent model. The merged model is created by adding the sum of the task vectors, scaled by a hyperparameter  $\lambda$ , to the the pretrained model,  $\theta_m = \theta_p + \lambda \sum_i \tau_i$ .

**DARE** (Yu et al., 2023) extends Task Arithmetic by applying dropout to the task vectors. Parameters from each task vector are randomly zeroed out with probability p using mask  $M_i \sim \text{Bernoulli}(p)$ and rescaled such that the expected value of the task vector is maintained. The modified task vectors  $\tau'_i = \frac{(1-M_i)\tau_i}{1-p}$  are then used as in Task Arithmetic.

TIES (Yadav et al., 2023) improves Task Arithmetic by zeroing out values in each task vector with
 low magnitude. A aggregate sign for parameter is chosen based on if the positive or the negative
 parameters have higher total magnitude. Finally, the parameters from each model that match the
 aggregate sign are added as in Task Arithmetic.

**Fisher Merging** (Matena & Raffel, 2022) merges models by finding a set of parameters that maximizes the joint posterior distribution of the constituent models. Posteriors are estimated via the Laplace approximation—a normal distribution with mean  $\theta_i$  and the inverse of the Fisher information matrix (Amari, 1998) for covariance. Fisher merging uses the closed-form solution  $\theta_m = \sum_i F_i \odot \theta_i / \sum_i F_i$  where  $F_i$  is the diagonal Fisher of model *i*.

125 RegMean (Jin et al., 2022) merges each linear layer by finding a weight matrix that minimizes the L2 126 distance between the activations of constituent models and the merged model. This can be cast as least 127 squares regression between the input and output activations for each linear layer. Let  $Z_i \in \mathbb{R}^{L_i \times k}$  be 128 the collection of  $L_i$  activations, each of dimensionality k, computed over examples from the dataset 129 used to train constituent model i, and let  $W_i$  be the parameters of some particular linear layer in model *i*. The closed form solution for least squares regression yields the the merged weight matrix, 130  $W_m = \left(\sum_i \frac{1}{L_i} Z_i^\top Z_i\right)^{-1} \left(\sum_i \frac{1}{L_i} (Z_i^\top Z_i) W_i\right)$ . Other parameters are merged via simple averaging. Note that only the gram matrix of the input activations for each model  $Z_i^\top Z_i$  are required to compute 131 132 133 the merge.

MaTS (Tam et al., 2023) unifies Fisher Merging and RegMean by solving a linear system that implicitly upweights models along the most important directions in parameter subspace for performing the fine-tuning task. The linear system is solved using the conjugate gradient (Hestenes & Stiefel, 1952) method, which allows for better approximations of the Fisher Information Matrix. Parameters not in the linear layers are merged via simple averaging.

139 140 141

142

#### 2.2 CHALLENGES IN COMPARING MERGING METHODS

The rapid pace of development of merging methods has led to a lack of standardization around the
experimental setup used to validate each method as well as the practical assumptions each method
makes. To motivate our work, we begin by highlighting these differences.

146 **Different Goals** Papers presenting new merging methods often perform evaluation with different 147 goals. For example, Matena & Raffel (2022) explored an intermediate-task setup that aims to improve 148 performance on a "downstream" task by merging with a model trained on a "donor" task, whereas 149 Ilharco et al. (2022) study the problem of creating a multitask model by merging models fine-tuned 150 on different tasks. In our work, we primarily focus on whether merging can enable compositional 151 generalization of the capabilities of the constituent models. We argue that compositional generalization realistic goal because it reflects typical use cases of merging (e.g., combining styles or objects 152 in image generation models or enabling zero-shot generalization to new tasks for language models 153 (Sanh et al., 2021)). In addition, as we will demonstrate, current merging methods often struggle to 154 provide compositional generalization, making it a challenging and meaningful evaluation setting. 155

**Different Experimental Setups:** Past works on merging rarely use a common experimental setup i.e., they differ in terms of the models and datasets they consider. For example, Jin et al. (2022) validate RegMean by merging fully fine-tuned variants of DeBERTa-large (He et al., 2020), while Yadav et al. (2023) merged variants of T5-XL-LM-Adapt (Lester et al., 2021) that were fine-tuned using  $(IA)^3$  (Liu et al., 2022). Furthermore, Jin et al. (2022) merged models trained on 8 GLUE (Wang et al., 2018) datasets whereas Yadav et al. (2023) considered 11 prompted datasets from the

Public Pool of Prompts (P3) (Bach et al., 2022). While sharing an experimental setup is rare, there

are some exceptions—for example, Tam et al. (2023) and Yadav et al. (2023) both replicate the setup of Ilharco et al. (2022) for their vision experiments.

**Different Prerequisites**: Simple merging methods like parameter averaging require only the constituent model parameters to perform a merge. More sophisticated merging methods can require access to additional models or statistics. For example, Task Arithmetic requires access to the pretrained model that all the constituent models were fine-tuned from and Fisher Merging requires access to the diagonal of the Fisher Information Matrix for each constituent model. Since fine-tuned models are typically shared as parameter values alone, these prerequisites are often not available and/or must be separately computed.

171 **Different Compute:** The computational expense of different merging methods can also vary. Pa-172 rameter averaging, Task Arithmetic, TIES Merging, DARE, and Fisher Merging primarily involve 173 elementwise addition and multiplication of parameter values and therefore have relatively low com-174 putational costs. On the other hand, RegMean requires a matrix inverse for each linear layer (whose 175 cost scales cubically with the activation dimension) and MaTS solves a linear system of equations 176 with the conjugate gradient method. These operations incur a significant increase in computational 177 cost, an oft neglected consideration when new merging methods are proposed. Merging can also 178 involve significant memory costs. Naïve implementations requires loading all model parameters into 179 memory at once. For many merging methods, it is possible to load and merge each parameter "group" (e.g., a weight matrix or bias vector) individually, as done in Git-Theta (Kandpal et al., 2023). For 180 other merging methods, such as AdaMerging (Yang et al., 2024a), the entire of each model must be 181 loaded simultaneously to perform a merge, preventing its use when memory is scarce. 182

Different hyperparameter requirements: Merging methods typically have hyperparameters—for
 example, the weight of each model for simple averaging and Fisher Merging, the scale of the task
 vectors in Task Arithmetic, TIES, and DARE, the scale of the non-diagonal entries of the gram matrix
 in RegMean, and the number of conjugate gradient iterations in MaTS. The sensitivity of each method
 to its corresponding hyperparameters is an important consideration in terms of its practical utility.

- 188 189
- 3 COMPREHENSIVE AND UNIFIED EVALUATION OF MERGING
- 190 191

Given the aforementioned challenges of comparing merging methods, we propose a rigorous and 192 comprehensive evaluation setup. In this work, we evaluate different merging methods' ability to 193 both create a multitask model that retains performance on constituent model training tasks ("held-194 in") as well as generalize to new tasks or domains which are compositions of the original tasks 195 ("generalization"). Our focus on compositional generalization stems from the common use of 196 merging to compose capabilities from different models. For example, given a model fine-tuned to 197 learn skills A and B and a second model that learned skills C and D, can the merged model solve a task requiring skills A and C? We test multitask performance and compositional generalization for 199 cross-domain image classification and generation as well as cross-lingual language processing. While 200 we only experiment with a single backbone model for each modality, the models we used in each 201 settings are currently widespread for their specific settings. Additionally, the specific models we used 202 in each setting are the same as those used in past evaluations (Ilharco et al., 2021; Yadav et al., 2023; Tam et al., 2023). Wherever possible, we include the performance of three oft-neglected baselines: 203 the original pretrained model, individual-task models, and a multitask model trained on all constituent-204 model datasets simultaneously. For held-in tasks, the individual-task baseline is the performance of 205 constituent models before merging. For generalization tasks, individual-task performance represents 206 the performance attainable from *training* on held-out task-specific data (which is not available to the 207 constituent models or merging methods). We do not include the baseline evaluating the constituent 208 models on the generalization tasks as we expect poor performance due to catastrophic forgetting, 209 especially in the cross-lingual case (Vu et al., 2022). 210

211

213

212 3.1 CROSS-DOMAIN IMAGE CLASSIFICATION AND GENERATION

For our vision experiments, we use the DomainNet (Peng et al., 2019) dataset, which consists of 586K
images from 345 classes (e.g., "apple", "shovel", each of which has roughly 15 classes) grouped into 24 categories (e.g., "fruit", "tool") and 6 domains (e.g., "drawing", "clipart"), and the splits from



Figure 1: Tasks that our image classification and generation models are trained on. Each row denotes objects within a certain category (e.g., fruit, bird, and tool) and the columns denotes different domains (e.g., sketch, real, and clipart). Each (category, domain) pair forms a different task—for example, the "fruit sketch" task involves generating or classifying sketches of fruits (i.e., apples, bananas, etc). Each constituent model is trained on one of the held-in tasks along the diagonal (solid border). Compositional generalization is measured via the performance on the generalization tasks off of the diagonal (dashed border).

Muqeeth et al. (2023). We note that two classes occur in two different groups, as listed in Appendix A. Fig. 1 has examples of (task, domain) combinations for DomainNet.

For each of the 24 tasks, we train a constituent model on images from a single domain and category. We measure compositional generalization on the remaining 5 domains for each category, resulting in 120 (category, domain) combinations for evaluation. See Appendix A for the full list of categories and domains. For classification, we fine-tune the CLIP ViT-B/32 vision encoder (Radford et al., 2021), as done in previous merging work (Ilharco et al., 2022; Yadav et al., 2023). To avoid task-specific classification heads, we construct a unified classification head by stacking CLIP's text embeddings for each label. More details are available in Appendix B.1. For generation we fine-tune Stable Diffusion 2.1 (Rombach et al., 2022) using LoRA (Hu et al., 2021) as this is currently the de facto way to fine-tune image generation models. More training and evaluation details are available in Appendix B.3.

239 240 241

242

227 228

229

230

231

232

233

234

235

236

237

238

#### 3.2 CROSS-LINGUAL NATURAL LANGUAGE PROCESSING

243 For natural language processing, we consider 5 distinct tasks (e.g., question-answering, summariza-244 tion, etc.) that have datasets available in different languages. We focus on cross-lingual generalization 245 as it is only possible through compositional generalization due to differing writing systems, vocabu-246 lary, grammatical rules, etc. across languages. Table 1 shows the chosen tasks and their available 247 languages. To avoid "leakage" of language-specific capabilities, we intentionally chose disparate 248 tasks—i.e., we avoided including similar tasks such as paraphrase identification and natural language 249 inference. Not all tasks are available in all languages, so we only evaluate cross-lingual generalization 250 on unseen (task, language) pairs that are available. We use mT5-xl-lm-adapt (Xue et al., 2020) as our base model, a multilingual version of T5 (Raffel et al., 2020) that was adapted for language modeling 251 by Vu et al. (2022) since it is the only pretrained language model that supports all the languages we 252 consider and comes from the same model family as previous merging papers (Yadav et al., 2023). We 253 first fully fine-tune mT5-xl-lm-adapt on different tasks, each in a different language. After merging 254 the constituent models, we evaluate performance on the held-in (task, language) combinations in 255 addition to unseen (task, language) pairs to evaluate compositional cross-lingual generalization. For 256 all tasks, we use the standard evaluation metric and report average performance across all tasks. See 257 Appendix B.2 for more details on training procedures and evaluation metrics.

258 259 260

261 262

264 265

NLP Task $\downarrow$ / Language $ ightarrow$	English	Arabic	Thai	German	Korean
Question-Answering (SQuaD/XQuaD)	3	Ø	Ø	Ø	
Natural Language Inference (XNLI)	1 and the second	5	and the second s	and the second s	
Summarization (WikiLingua)	Ø	Ø	M	Ø	(j)
Word Sense Disambiguation (WiC/XLWiC)	1 and			5	(ji)
Is question answerable? (TyDiQA)	Ø	Ø	Ø		M

Table 1: (task, language) pairs used to evaluate cross-lingual compositional generalization. Pairs marked  $\searrow$  were used for fine-tuning; those marked  $\checkmark$  were used for evaluation. Ideally, every (task, language) pair would have been used for evaluation, but not all combinations are available.

#### 4 Results

270

271 272

273

274

275

276 277 278

279 280 281

282

283

284

285

286

287

288

289

290

291

292

293

295

296

297 298 299 Having described our experimental setup and practical considerations that differentiate merging methods, we now conduct an in-depth evaluation to provide a comprehensive picture of the field. Our evaluation covers generalization performance, method requirements, computational costs, hyperparameter sensitivity, scaling behavior, and model size.

#### 4.1 Held-in and Generalization Performance



Figure 2: Performance of different merging methods in the image classification, image generation, and natural language processing settings described in Section 3. For each method, we plot the performance on the held-in datasets against the performance on unseen datasets that require compositional generalization. Additionally, we report the performance of the pretrained model, a multitask model trained on all held-in datasets at once, and the performance attained by training on a single task's data alone. Numerical values are provided in Appendix H.

300 **Comparing Methods** The held-in and generalization performance of each merging method we 301 consider is shown in Fig. 2. For each method, we plot the performance of the merged model on 302 held-in datasets against the performance on datasets that require compositional generalization. For 303 held-in performance, RegMean and MaTS work well on image classification and NLP, matching 304 trends in previous benchmarks (Tam et al., 2023), while for image generation, TIES works well. We note that in NLP, Fisher Merging tends to generalize than RegMean and MaTS despite all three of 305 methods implicitly minimizing the same objective (Tam et al., 2023). This discrepancy could be 306 due to Fisher Merging's looser approximation of the Fisher, which could improve generalization by 307 reducing overfitting to the held-in tasks. 308

309 For image classification, we observe a positive correlation between a merging method's held-in performance and generalization (r=0.828, p=0.011), and for image generation the correlation 310 is stronger (r=0.972,  $p=5.266e^{-5}$ ). This suggests that improving multitask performance of a 311 model leads to improved general capabilities in this setting. Furthermore, for image generation, 312 many merging methods outperform multitask training in terms of both held-in and generalization 313 performance, highlighting the applicability and benefits of merging in this setting. In contrast, for NLP, 314 the held-in and generalization performance is *anticorrelated*—most merge methods underperform 315 the pretrained model in terms of generalization (r = -0.853, p = 0.007). This discrepancy could 316 stem from cross-lingual generalization being more difficult than cross-domain generalization; we 317 may expect vision models that can classify drawn images to be able to reasonably classify clipart 318 images, but we would not expect a model trained on English text to be able to generate text in Arabic 319 (which does not even share a writing system with English). In line with past work (Wortsman et al., 320 2022b), we find that while merging lags behind multitask models in terms of held-in performance, 321 merged models can exhibit better generalization to new domains than the multitask and pretrained models. Taken together, our results highlight the different behaviors of merging methods in different 322 experimental settings and elucidate which settings pose challenges that could be tackled in future 323 research on merging.

Prerequisites				Computational cost (FLOPs)		
			<b>D</b> (	- Hparams?	i	<u> </u>
	$\theta_{\mathbf{p}}$	Stats	Data	•	Merging	Statistics
Average					Mdk	
SLERP					$\mathcal{O}((5M-2)dk)$	
Task Arith.	X		X	$\checkmark$	(2M+1)dk	
DARE	X		X	$\checkmark$	(6M+1)dk	
TIES	X		<b>X</b> *	$\checkmark^*$	(4M+1)dk	$\mathcal{O}(MKdk)$
Fisher		Χ*	Χ*		(3M-1)dk	$4MTd^2k$
RegMean		<b>X</b> *	<b>X</b> *	$\checkmark$	$\mathcal{O}((M+2)d^2k)$	$MTd^2k$
MaTS		<b>X</b> *	<b>X</b> *	5	$\mathcal{O}((M+N)d^2k)$	$4MTd^2k$
1111115				•		ini i u n

334 Table 2: Practical differences between merging methods along different axes. Prerequisites: An 335 X denotes merging methods that require the pretrained model parameters, statistics from the pretrained 336 model and/or fine-tuning dataset, or access to data during the merge. Some requirements are optional, 337 denoted as  $X^*$ , e.g., if model statistics are distributed in conjunction with parameters then data is 338 not required. Hyperparameters: Merging methods with a  $\checkmark$  require tuning their hyperparameters. 339 TIES includes performant default hyperparameters, making tuning optional, denoted  $\sqrt{*}$ . The hyperparameter values we used can be found in Table 5. Computational Cost: Different methods 340 also have different computational costs. "Merging" costs are incurred during each merge, while 341 "Statistic" costs can be computed once and reused. The tables show the cost of merging a single  $d \times k$ 342 linear layer across M models. See Appendix C for exact costs. 343

344

345 346

347

#### 4.2 PREREQUISITES

348 While performance is often the main focus of new merging methods, we highlight other practical 349 requirements that make different merging methods more or less attractive or applicable. To better elucidate these requirements, in Table 2 we categorize each merging method in terms of whether it 350 requires access to the shared pretrained model, requires auxiliary model statistics (e.g., the diagonal 351 of the Fisher Information Matrix or the Gram matrix of input activations), and/or requires data 352 access to perform a merge. On the whole, the merging methods we study fall into three categories: 353 Simple averaging has no prerequisites, which may explain its continued popularity. Task Arithmetic 354 and its derivative methods (DARE and TIES) require access to the pretrained model and data to 355 tune hyperparameters (although Yadav et al. (2023) report a single value that generally works well). 356 Finally, Fisher Merging, RegMean, and MaTS require constituent model statistics or data to compute 357 required statistics. Such statistics are a form of auxiliary information that is rarely shared along with 358 fine-tuned models which may explain why these methods are rarely used in practice despite their 359 relatively strong performance. Methods that require access to data typically use a small validation 360 set of  $\sim 1.000$  examples. While the theoretical motivations of Fisher Merging require computing the 361 Fisher on the training set, we follow Tam et al. (2023) and use the validation set since performance is comparable and it simplifies comparison to methods that require a validation set. 362

363 364

366

#### 4.3 COMPUTATIONAL COSTS

Another consideration when applying model merging is the amount of compute required to perform a 367 merge. Table 2 shows the cost of various methods to merge a single linear layer. We see two classes 368 of methods emerge: ones that run in  $\mathcal{O}(dk)$  time and others that run in  $\mathcal{O}(d^2k)$ , where d and k refer to 369 the input and output dimensions of a given linear layer. Since Table 2 deals with the limiting behavior, 370 we examine the real-world costs by plotting the number of FLOPs required for each method against 371 performance in Fig. 3. We see that, loosely speaking, more expensive merging methods tend to work 372 better. In particular, the relatively expensive RegMean and MaTS methods are the most performant 373 for held-in tasks, matching previous findings that focus on multitask performance. However, when 374 measuring generalization performance, we note that the benefits of MaTS and RegMean can vanish 375 (particularly when targetting cross-lingual generalization). Notably, the cost of different merging methods varies by almost two orders of magnitude, suggesting that the computational cost of a given 376 merging method is an important consideration. Details of our FLOPs calculations can be found in 377 Appendix C.



Figure 3: The computational cost vs. performance for each merging method. For the computational cost, we report the upper bound of the number of FLOPs required to merge a single layer (see Appendix E for details).

#### HYPERPARAMETER SENSITIVITY 4.4



Figure 4: Hyperparameter sensitivity of each merging method. We plot the performance of each merging method as we sweep their respective hyperparameters. We index possible hyperparameter values from 0 to 10 as the specific hyperparameters and their ranges differ between merging methods. This captures the robustness of merging methods to different hyperparameters, regardless of the 422 specific values. See Appendix E for a description of the hyperparameters.

423 424

419

420

421

396

397

398 399

400

Merging methods with hyperparameters (highlighted in Table 2) often have different sensitivities 425 to hyperparameter choice. This can heavily impact the practical utility of a given merging method. 426 Previous works generally report performance for the best hyperparameter values, which can obscure 427 their sensitivity. Additionally, hyperparameter tuning requires more compute and access to data, 428 which may not always be available. Therefore we compare the robustness of different merging 429 methods to hyperparameter choice. We sweep the hyperparameters as described in Appendix E using values from Table 5 for each method. Note that some merging implementations introduce per-model 430 scaling hyperparameters, but we set these weights to 1/M and therefore omit their consideration as a 431 hyperparameter.

432 The results of our sweep are shown in Fig. 4. We find that merging methods vary significantly in 433 their hyperparameter sensitivity. For example, Task Arithmetic and TIES both exhibited significant 434 sensitivity to the scaling hyperparameter  $\lambda$ , whereas DARE was robust to changes in the dropout 435 probability p (provided a good  $\lambda$  is reused). Notably, held-in and generalization accuracy tend to 436 be correlated across hyperparameters, suggesting that it is safe to tune hyperparameters on heldin datasets while aiming to maximize generalization performance. Simple Averaging and Fisher 437 Merging generally attained reasonable performance, suggesting that they are a good choice when 438 hyperparameter tuning is not possible. 439

4.5 SCALING



Figure 5: Performance of merging methods as the number of constituent tasks increases. Along the x-axis, we sample a subset of tasks 10 times and report the mean held-in and generalization performance. We additionally evaluate a pretrained model and a multitask model trained on all the held-in tasks on the sampled subsets. Since the generalization datasets and the pretrained model are fixed, its generalization performance is shown as horizontal line.

466

440 441

442

Thus far, we have merged all possible models in each experiment. In practice, the number of models
being merged may vary in different applications. Therefore, we evaluate the performance of each
merging method as the number of constituent models M varies from 2 to 24. For each value of M,
we draw 10 samples of M constituent models, perform a hyperparameter sweep for each method, and
report the average performance on the held-in and applicable generalization tasks. See Appendix F
for more details. We only report the multitask performance of training on all tasks due to lack of
computational resources.

Fig. 5 shows the performance on as we scale the number of models being merged. We observe a clear 474 trend across all methods: Held-in performance tends to *decrease* as the number of constituent models 475 increases, whereas the generalization performance generally *increases*. This finding is aligned with 476 Wortsman et al. (2022a), who found that being selective about which tasks are merged can improve 477 held-in performance. This is likely because increasing the number of models results in increased 478 interference but expands the range of underlying capabilities, thereby improving generalization. This 479 suggests that merging suffers from negative interference and/or insufficient capacity for held-in tasks. 480 Conversely, merging can provide a promising way to improve generalization compared to multitask 481 performance, especially when the number of constituent models is large.

482 483

484 485 4.6 TAKEAWAYS

To summarize our empirical study, we highlight the following findings:

486
 487
 488
 488
 488
 489
 480
 480
 480
 480
 480
 481
 481
 482
 483
 484
 484
 485
 485
 486
 486
 486
 486
 486
 486
 487
 488
 488
 488
 488
 488
 488
 489
 489
 480
 480
 480
 480
 481
 481
 482
 482
 483
 484
 484
 485
 486
 486
 486
 486
 486
 486
 486
 487
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488
 488

Held-in performance and generalization performance are correlated for cross-domain generalization but anticorrelated for for cross-lingual generalization, possibly because cross-lingual generalization is more challenging.

When scaling up the number of models being merged, held-in performance decreases whereas
 generalization performance increases, though these trends plateau after around 10 models.

TIES generally provides a good trade-off between performance and practical considerations
 such as prerequisites, compute, and hyperparameter tuning.

The prerequisites and computation requirements of merging methods should be clearly stated, as they can have a large impact on its applicability in different practical scenarios. For example, though RegMean and MaTS perform well (particularly for held-in performance), their need for statistics or data and higher computational cost probably explains why they are rarely used in practice.

Some merging methods such as Task Arithmetic and TIES exhibit significant hyperparameter
 sensitivity, which should be seen as a limitation given that hyperparameter tuning requires data
 access and incurs nontrivial computational costs.

#### 505 5 RELATED WORK

506 In this work, we focus solely on the merging methods discussed in Section 2.1, which we chose based 507 on their popularity and diversity. However, the popularity of model merging has led to a larger and 508 ever-growing body of merging methods. Since we omitted many of these methods from this study 509 due to practical considerations, we briefly survey them here. Tangent Task Arithmetic (Ortiz-Jimenez 510 et al., 2023) fine-tunes models in the tangent space for better weight disentanglement when using Task 511 Arithmetic. Daheim et al. (2023) combine Fisher Merging with Task Arithmetic which is shown to help prevent the mismatch in gradients. Akiba et al. (2024) explore using evolutionary algorithms to 512 choose which layers to merge. Tang et al. (2023) learn a mask to select parameters that are important 513 for the merged model. Jiang et al. (2023) propose pruning task vectors to allow for efficiently loading 514 during inference. Ye et al. (2023) trains a gating network that predicts the weights of a weighted 515 average of examples during inference. Tang et al. (2024) train a router between the different models 516 using unlabeled data. Several works focused on merging models with different initializations via 517 permutation (Ainsworth et al., 2022; Yamada et al., 2023; Singh & Jaggi, 2020; Jordan et al., 2022). 518 These are bassed on the hypothesis that although models lie in different basins of the loss landscape 519 (i.e., are not linear mode connected (Frankle et al., 2020; Juneja et al., 2022)), once permutational 520 invariances are accounted for, the models will lie in the same basin (Entezari et al., 2021). Stoica et al. 521 (2023) extend this idea to merging models with different dimensions by expanding and permuting 522 features. Other applications of model merging include intermediate-task training (Choshen et al., 2022; Gueta et al., 2023) and merging models with different modalities (Sung et al., 2023). 523

Recent works have also measured whether models can compose skills via multitask training (Arora & Goyal, 2023; Zhao et al., 2024a) or by combining individual-task models. For example, both Pfeiffer et al. (2020) and Vu et al. (2022) tackle cross-lingual generalization by training separate "task" and "language" adapters. The adapters are swapped during inference to generalize to new (task, language) pairs. Similarly, AdaMergex uses task/language vector arithmetic for cross-lingual generalization (Zhao et al., 2024b). CALM trains a cross-attention module to compose constituent model skills, however, it requires access to a generalization dataset (Bansal et al., 2024).

531 532

533

504

#### 6 CONCLUSION

Our work has clarified the state of model merging by conducting an empirical study of merging methods in a comprehensive and rigorous experimental setting. Specifically, we evaluate eight merging methods across three settings covering natural language processing and image classification and generation. We compare both the performance of merging methods and the practical considerations that make them more or less attractive in different applications. Our findings, summarized in Section 4.6, identify important paths and best practices for future work on model merging. We hope our findings and released code will help accelerate and unify future work on model merging.

# 540 REFERENCES

- Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging Models
   Modulo Permutation Symmetries. *arXiv preprint arXiv:2209.04836*, 2022.
- Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary Optimization of Model Merging Recipes. *arXiv preprint arXiv:2403.13187*, 2024.
- Shun-Ichi Amari. Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10(2):
   251–276, 1998.
- Sanjeev Arora and Anirudh Goyal. A theory for emergence of complex skills in language models.
   *arXiv preprint arXiv:2307.15936*, 2023.
- Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the Cross-Lingual Transferability of
   Monolingual Representations. *arXiv preprint arXiv:1910.11856*, 2019.
- Stephen H Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, et al. Promptsource: An Integrated Development Environment and Repository for Natural Language Prompts. *arXiv preprint arXiv:2202.01279*, 2022.
- Rachit Bansal, Bidisha Samanta, Siddharth Dalmia, Nitish Gupta, Shikhar Vashishth, Sriram Ganapathy, Abhishek Bapna, Prateek Jain, and Partha Talukdar. LLM Augmented LLMs: Expanding Capabilities Through Composition. *arXiv preprint arXiv:2401.02412*, 2024.
- Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. Fusing Finetuned Models for Better
   Pretraining. *arXiv preprint arXiv:2204.03044*, 2022.
- Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. Tydiqa: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020.
- Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger
   Schwenk, and Veselin Stoyanov. XNLI: Evaluating Cross-Lingual Sentence Representations.
   *arXiv preprint arXiv:1809.05053*, 2018.
- Nico Daheim, Nouha Dziri, Mrinmaya Sachan, Iryna Gurevych, and Edoardo M Ponti. Elastic Weight Removal for Faithful and Abstractive Dialogue Generation. *arXiv preprint arXiv:2303.17574*, 2023.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient Finetuning of Quantized LLMs. Advances in Neural Information Processing Systems, 36, 2024.
- Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The Role of Permutation Invariance in Linear Mode Connectivity of Neural Networks. *arXiv preprint arXiv:2110.06296*, 2021.
- Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear Mode
  Connectivity and the Lottery Ticket Hypothesis. In *International Conference on Machine Learning*,
  pp. 3259–3269. PMLR, 2020.
- Yuchao Gu, Xintao Wang, Jay Zhangjie Wu, Yujun Shi, Chen Yunpeng, Zihan Fan, Wuyou Xiao, Rui
  Zhao, Shuning Chang, Weijia Wu, Yixiao Ge, Shan Ying, and Mike Zheng Shou. Mix-of-Show:
  Decentralized Low-Rank Adaptation for Multi-Concept Customization of Diffusion Models. *arXiv preprint arXiv:2305.18292*, 2023.
- Almog Gueta, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Knowledge is a Region in Weight Space for Fine-Tuned Language Models. *arXiv preprint arXiv:2302.04863*, 2023.
- <sup>593</sup> Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-Enhanced BERT with Disentangled Attention. *arXiv preprint arXiv:2006.03654*, 2020.

- Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A
   Reference-free Evaluation Metric for Image Captioning. In *Proceedings of the 2021 Conference* on Empirical Methods in Natural Language Processing, 2021.
- Magnus R. Hestenes and Eduard Stiefel. Methods of Conjugate Gradients for Solving Linear Systems
   1. Journal of Research of the National Bureau of Standards, 49(6):409–436, 1952. Research Paper 2379.
- Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter.
   GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *NeurIPS*, 2017.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*, 2021.
- Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori,
  Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali
  Farhadi, and Ludwig Schmidt. OpenCLIP, July 2021. URL https://doi.org/10.5281/
  zenodo.5143773. If you use this software, please cite it as below.
- Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt,
   Hannaneh Hajishirzi, and Ali Farhadi. Editing Models with Task Arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.
- Weisen Jiang, Baijiong Lin, Han Shi, Yu Zhang, James T Kwok, et al. Effective and ParameterEfficient Reusing Fine-Tuned Models. *arXiv preprint arXiv:2310.01886*, 2023.
- Kisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless Knowledge Fusion by
   Merging Weights of Language Models. *arXiv preprint arXiv:2212.09849*, 2022.
- Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. Repair: Renormalizing Permuted Activations for Interpolation Repair. *arXiv preprint arXiv:2211.08403*, 2022.
- Jeevesh Juneja, Rachit Bansal, Kyunghyun Cho, João Sedoc, and Naomi Saphra. Linear Connectivity
   Reveals Generalization Strategies. *arXiv preprint arXiv:2205.12411*, 2022.
- Nikhil Kandpal, Brian Lester, Mohammed Muqeeth, Anisha Mascarenhas, Monty Evans, Vishal Baskaran, Tenghao Huang, Haokun Liu, and Colin Raffel. Git-Theta: A Git Extension for Collaborative Development of Machine Learning Models. In *International Conference on Machine Learning, ICML*, July 2023. URL https://arxiv.org/abs/2306.04529.
- Minchul Kim, Shangqian Gao, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. Token fusion: Bridging
   the gap between token pruning and token merging. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1383–1392, 2024.
- Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2017.

- Faisal Ladhak, Esin Durmus, Claire Cardie, and Kathleen McKeown. WikiLingua: A New Benchmark
   Dataset for Cross-Lingual Abstractive Summarization. *arXiv preprint arXiv:2010.03093*, 2020.
- Brian Lester, Rami Al-Rfou, and Noah Constant. The Power of Scale for Parameter-Efficient Prompt Tuning. *arXiv preprint arXiv:2104.08691*, 2021.
- Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.
   URL https://aclanthology.org/W04-1013.
- Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and
   Colin A Raffel. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context
   Learning. Advances in Neural Information Processing Systems, 35:1950–1965, 2022.
- 647 Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*, 2017.

648 649	Michael S Matena and Colin A Raffel. Merging Models with Fisher-Weighted Averaging. Advances in Neural Information Processing Systems, 35:17703–17716, 2022.
651 652 653	Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In <i>Artificial intelligence and statistics</i> , 2017.
654 655	Mohammed Muqeeth, Haokun Liu, and Colin Raffel. Soft Merging of Experts with Adaptive Routing. arXiv preprint arXiv:2306.03745, 2023.
656 657 658	Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task Arithmetic in the Tangent Space: Improved Editing of Pre-Trained Models. <i>arXiv preprint arXiv:2305.12827</i> , 2023.
659 660 661	Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment Matching for Multi-Source Domain Adaptation. In <i>Proceedings of the IEEE/CVF international conference on computer vision</i> , pp. 1406–1415, 2019.
662 663	Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. Mad-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. arXiv preprint arXiv:2005.00052, 2020.
665 666	Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: The Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations. <i>arXiv preprint arXiv:1808.09121</i> , 2018.
667 668 669	Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis. <i>arXiv preprint arXiv:2307.01952</i> , 2023.
670 671 672 673 674	Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models from Natural Language Supervision. In <i>International</i> <i>conference on machine learning</i> , pp. 8748–8763. PMLR, 2021.
675 676	Colin Raffel. Building Machine Learning Models like Open Source Software. <i>Communications of the ACM</i> , 66(2):38–40, 2023.
677 678 679	Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. <i>The Journal of Machine Learning Research</i> , 21(1):5485–5551, 2020.
681 682 683 684	Alessandro Raganato, Tommaso Pasini, Jose Camacho-Collados, and Mohammad Taher Pilehvar. XL-WiC: A Multilingual Benchmark for Evaluating Semantic Contextualization. In <i>Proceedings</i> of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 7193–7206, 2020.
685 686	Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ Questions for Machine Comprehension of Text. <i>arXiv preprint arXiv:1606.05250</i> , 2016.
687 688 689	Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High- Resolution Image Synthesis with Latent Diffusion Models. In <i>Proceedings of the IEEE/CVF</i> <i>conference on computer vision and pattern recognition</i> , pp. 10684–10695, 2022.
691 692 693	Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask Prompted Training Enables Zero-Shot Task Generalization. <i>arXiv preprint arXiv:2110.08207</i> , 2021.
694 695	Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/ pytorch-fid, August 2020. Version 0.3.0.
696 697 698 699	Viraj Shah, Nataniel Ruiz, Forrester Cole, Erika Lu, Svetlana Lazebnik, Yuanzhen Li, and Varun Jampani. Ziplora: Any Subject in Any Style by Effectively Merging LoRAs. <i>arXiv preprint arXiv:2311.13600</i> , 2023.
700 701	Ken Shoemake. Animating rotation with quaternion curves. <i>SIGGRAPH Comput. Graph.</i> , 19(3): 245–254, jul 1985. ISSN 0097-8930. doi: 10.1145/325165.325242. URL https://doi.org/10.1145/325165.325242.

702 703 704	Sidak Pal Singh and Martin Jaggi. Model Fusion via Optimal Transport. Advances in Neural Information Processing Systems, 33:22045–22055, 2020.
705 706	George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. ZipIt! Merging Models from Different Tasks without Training. <i>arXiv preprint arXiv:2305.03053</i> , 2023.
707 708 709 710	Yi-Lin Sung, Linjie Li, Kevin Lin, Zhe Gan, Mohit Bansal, and Lijuan Wang. An Empirical Study of Multimodal Model Merging. <i>arXiv preprint arXiv:2304.14933 [cs.CV]</i> , 2023. doi: 10.48550/arXiv.2304.14933. URL https://arxiv.org/abs/2304.14933.
711 712	Derek Tam, Mohit Bansal, and Colin Raffel. Merging by Matching Models in Task Subspaces. <i>arXiv</i> preprint arXiv:2312.04339, 2023.
713 714 715 716	Anke Tang, Li Shen, Yong Luo, Liang Ding, Han Hu, Bo Du, and Dacheng Tao. Concrete Sub- space Learning based Interference Elimination for Multi-task Model Fusion. <i>arXiv preprint</i> <i>arXiv:2312.06173</i> , 2023.
717 718 719	Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. Merging Multi-Task Models via Weight-Ensembling Mixture of Experts. <i>arXiv preprint arXiv:2402.00433</i> , 2024.
720 721 722	Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and Efficient Foundation Language Models. <i>arXiv preprint arXiv:2302.13971</i> , 2023a.
723 724 725 726	Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. <i>arXiv preprint arXiv:2307.09288</i> , 2023b.
727 728 729	Tu Vu, Aditya Barua, Brian Lester, Daniel Cer, Mohit Iyyer, and Noah Constant. Overcoming Catastrophic Forgetting in Zero-Shot Cross-Lingual Generation. <i>arXiv preprint arXiv:2205.12647</i> , 2022.
730 731 732 733	Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. <i>arXiv preprint arXiv:1804.07461</i> , 2018.
734 735 736 737 738	Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model Soups: Averaging Weights of Multiple Fine-Tuned Models Improves Accuracy Without Increasing Inference Time. In <i>International Conference on Machine Learning</i> , pp. 23965–23998. PMLR, 2022a.
739 740 741 742 743	Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust Fine-Tuning of Zero-Shot Models. In <i>Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition</i> , pp. 7959–7971, 2022b.
744 745 746	Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A Massively Multilingual Pre-Trained Text-to-Text Transformer. <i>arXiv preprint arXiv:2010.11934</i> , 2020.
747 748 749	Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Resolving Interference When Merging Models. <i>arXiv preprint arXiv:2306.01708</i> , 2023.
750 751 752 753	Masanori Yamada, Tomoya Yamashita, Shin'ya Yamaguchi, and Daiki Chijiwa. Revisiting Permuta- tion Symmetry for Merging Models Between Different Datasets. <i>arXiv preprint arXiv:2306.05641</i> , 2023.
754 755	Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. AdaMerging: Adaptive Model Merging for Multi-Task Learning. <i>The Twelfth International Conference on Learning Representations</i> , 2024a.

756 757 758	Yang Yang, Wen Wang, Liang Peng, Chaotian Song, Yao Chen, Hengjia Li, Xiaolong Yang, Qinglin Lu, Deng Cai, Boxi Wu, and Wei Liu. LoRA-Composer: Leveraging Low-Rank Adaptation for Multi-Concept Customization in Training-Free Diffusion Models, 2024b.
759 760 761	Peng Ye, Chenyu Huang, Mingzhu Shen, Tao Chen, Yongqi Huang, Yuning Zhang, and Wanli Ouyang. Merging Vision Transformers from Different Tasks and Domains. <i>arXiv preprint arXiv:2312.16240</i> , 2023.
762 763 764 765	Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch. <i>arXiv preprint arXiv:2311.03099</i> , 2023.
766 767 768	Haoyu Zhao, Simran Kaur, Dingli Yu, Anirudh Goyal, and Sanjeev Arora. Can models learn skill composition from examples? <i>arXiv preprint arXiv:2409.19808</i> , 2024a.
769 770 771	Yiran Zhao, Wenxuan Zhang, Huiming Wang, Kenji Kawaguchi, and Lidong Bing. AdaMergeX: Cross-Lingual Transfer with Large Language Models via Adaptive Adapter Merging. arXiv preprint arXiv:2402.18913, 2024b.
772 773 774 775	Ming Zhong, Yelong Shen, Shuohang Wang, Yadong Lu, Yizhu Jiao, Siru Ouyang, Donghan Yu, Jiawei Han, and Weizhu Chen. Multi-LoRA Composition for Image Generation. <i>arXiv preprint arXiv:2402.16843</i> , 2024.
776 777 778	
779 780	
781 782 783	
784 785	
786 787 788	
789 790	
791 792 793	
794 795	
796 797 798	
799 800	
801 802 803	
804 805	
807 808	
809	

## A TASKS AND DOMAINS

811 812

In DomainNet, the 24 tasks are: furniture, mammal, tool, cloth, electricity, building, office, human\_body, road\_transportation, food, nature, cold\_blooded, music, fruit, sport, tree, bird, vegetable, shape, kitchen, water\_transportation, sky\_transportation, insects, and others.

816 The 6 domain are: clipart, infograph, painting, quickdraw, real, and sketch.

Models-to-be-merged are trained on the following (task, domain) pairs: (cloth, clipart), (furniture, clipart), (mammal, clipart), (tool, clipart), (building, infograph), (electricity, infograph), (human\_body, infograph), (office, infograph), (cold\_blooded, painting), (food, painting), (nature, painting), (road\_transportation, painting), (fruit, quickdraw), (music, quickdraw), (sport, quickdraw), (tree, quickdraw), (bird, real), (kitchen, real), (shape, real), (vegatable, real), (insect, sketch), (others, sketch), (sky\_transportation, sketch), (water\_transportation, sketch).

- There are 144 possible (task, domain) combinations, 24 tasks × 6 domains. Removing the 24 (task, domain) pairs used for training leaves 120 (task, domain) combinations to use for evaluation of compositional generalization.
- We also note the bandage and nail classes are in both the tool task and the office task.

Bue to the paucity of data in the cross-lingual domain, all training and evaluation (task, language)
pairs are enumerated in Table 1.

SQuAD (Rajpurkar et al., 2016) is released under a CC BY-SA 4.0 license. WikiLingua (Ladhak
et al., 2020) is released under a CC0-1.0 license. XQuAD (Artetxe et al., 2019) is released under
a CC-BY-SA 4.0 license. XNLI (Conneau et al., 2018) is released under a CC BY-NC 4.0 DEED
license. WiC (Pilehvar & Camacho-Collados, 2018) is released under a CC BY-NC 4.0 License.
XLWiC (Raganato et al., 2020) is released under a CC BY-NC 4.0 License.
XLWiC (Raganato et al., 2020) is released under a CC BY-NC 4.0 License.
Z020) is released under an Apache license. DomainNet (Peng et al., 2019) was released under a fair
use notice.

837 838

#### **B** TRAINING DETAILS AND EVALUATION METRICS

839 840 841

Models used and the training details vary based on the setting; they are outlined below. Models were trained and merged on a combination of NVIDIA A6000 and 80GB NVIDIA A100 GPUs.

846

847

B.1 CROSS-DOMAIN IMAGE CLASSIFICATION

We fine-tune the CLIP vision encoder from open\_clip (Ilharco et al., 2021; Radford et al., 2021).

Previous works fine-tune task-specific classifier heads on top of the CLIP representation. This means that the classifier head cannot be "merged" as different tasks have different output classes and classifier heads during evaluation. We want a classifier head that can "merged" by having all tasks use the same classifier head during evaluation. To do so, we create a frozen linear classifier head where each row vector is a representation of a class. These rows come from CLIP's textual embedding representation of the class label. Thus, loading a merged classifier head is done by embedding the text representations of each label across all tasks.

We use AdamW (Loshchilov & Hutter, 2017) with a learning rate  $2e^{-5}$  for 1,000 steps using a batch size 128. Training is done using open\_clip's fp16 setting. We checkpoint every 50 batches, with early stopping if validation performance does not improve after 5 checkpoints. We use accuracy as our evaluation metric.

For classification, we simply report the accuracy on held-out data. For generation, we follow standard
practice and compute both the CLIP Score (CLIP-S) (Hessel et al., 2021) to measure the alignment
between the generated image and prompt as well as the Frechet Inception Distance (FID) (Heusel
et al., 2017) to measure perceptual quality. Since CLIP-S more directly measures what we aim to
evaluate and we found that CLIP-S and FID were generally highly correlated in practice, we only
report CLIP-S in the main text and include FID results in Appendix I.

# 864 B.2 CROSS-LINGUAL LANGUAGE TASKS

<sup>866</sup> For all tasks, we use the standard evaluation metric and report average performance across all tasks.

867 We fine-tune mT5-xl-lm-adapt (Xue et al., 2020; Vu et al., 2022) using AdamW (Loshchilov & 868 Hutter, 2017) with a learning rate  $5e^{-4}$  for 5,000 steps using a batch size 1024. We checkpoint every 100 steps, with early stopping if validation performance does not improve after 5 checkpoints. 870 For multiple-choice language tasks, i.e., natural language inference, word understanding, and "is 871 question answerable", we use accuracy as the evaluation metric. For question-answering tasks, we 872 use the average of exact-match metric (Rajpurkar et al., 2016) and F1. Like (Vu et al., 2022), we 873 use SP-ROUGE to evaluate summarization tasks. SP-ROUGE is a variant of ROUGE (Lin, 2004) 874 that uses language independent tokenization instead of the naïve white space character. We use the average score of SP-ROUGE variants of Rouge-1, Rouge-2, and Rouge-L. 875

876 877

878

#### B.3 CROSS-DOMAIN IMAGE GENERATION

We fine-tune Stable Diffusion 2.1 (Rombach et al., 2022) using Low-Rank Adaptation (LoRA) (Hu et al., 2021) on the 24 held-in (task, domain) pairs. We fine-tune rank 64 LoRA adapters for 10K steps using the denoising objective from the original work. We train with a batch size of 4 and a learning rate of  $1e^{-4}$  with cosine decay using Adam (Kingma & Ba, 2017).

When merging models, we merge pre-multiplied A and B matrices, instead of merging the A matrices
and B matrices separately, since we found this improved performance. This also requires computing
model statistics are on the pre-multiplied A and B matrices.

To evaluate generated images, we use the CLIP-score (Hessel et al., 2021) and FID (Heusel et al., 2017) metrics. To compute held-in FID, we randomly select 3 images from each of the 345 (task, domain, class) tuples. This yields 1,035 images. Similarly, we compute generalization FID by sampling 1 image from each of the 1,722 (task, domain, class) pairs. As we have more than 1,000 images in each setting, our FID metric provides a good capture of the distribution. We use pytorch-fid Seitzer (2020) to compute FID scores with 192-dimensional features from Inception.

 To select the best hyper-parameters, we use CLIP-score as an indicator for performance and sweep the same ranges described in Table 5.

894 895

#### C COMPUTATIONAL COSTS

896 897

Table 2 shows the estimated number of FLOPs required for different merging methods, as some implementations are not yet optimized. For example, MaTS uses the conjugate gradient method which requires many matrix-vector products. These are faster on GPU, but we are not aware of any linear conjugate gradient implementations on GPU, thus the time is inflated by many GPU  $\leftrightarrow$  CPU transfers. However, we do include some preliminary timing results in Appendix D.

We see in Table 3 that two classes of merge methods emerge, ones that run in  $\mathcal{O}(d^2k)$  and those that run in  $\mathcal{O}(dk)$ . Methods that run in  $\mathcal{O}(d^2k)$  require a matrix multiplication while the others do not. This difference is clearer when we consider that in many transformer architectures d = k and therefore these costs become  $\mathcal{O}(d^3)$  and  $\mathcal{O}(d^2)$ .

907 As the majority of parameters in a transformer are from the linear layers—Attention QKV, Feed 908 Forward layers, etc.—and some methods fallback to simple averaging for other parameters, we 909 calculate the amount of compute required to merge a *single* linear layer. Each linear layer has an 910 input dimension of d and an output dimension of k and we merge M models. The conjugate gradient 911 optimization used in MaTs is run for N iterations.

- When computing model statistics, we estimate the required FLOPs per token as 1 matrix multiplication in the forward pass and 3 matrix multiplications in the backward pass, following previous works which assume backward pass is  $3 \times$  the forward pass (Liu et al., 2022). To avoid memory issues, we
- 915 pre-compute the trimming of low magnitude parameters in TIES and only keep the top K parameters.
- 916 More details on this can be found in Appendix G. While statistic computation can be costly, it only 917 needs to be done once per task. Thus statistics can be reused and the cost can be amortized across
  - many different merges.

918	Method	Merging FLOPs	Statistics FLOPs
919	memou	interging i Horis	Stutistics I HOI 5
010	Average	Mdk	-
920	Task Arith.	(2M+1)dk	-
921	DARE	(6M+1)dk	-
922	TIES	(4M+1)dk	$MKdk + Mdk\log(K)$
923	Fisher	(3M-1)dk	$4MTd^2k$
924	RegMean	$(M+2)d^2k + (3M-2)dk$	$MTd^2k$
925	MaTS	$(M+N)d^{2}k + (2M+5N-2)dk$	$4MTd^2k$
926	SLERP	$(5M-2)dk + (M+1)\log(dk)$	-
927	MLERP	$(2M+3)dk + (M+1)\log(dk) + \log(M)$	-

Table 3: Comparing compute cost of merging a linear layer between different methods. We merge Mmodels and calculate the FLOPs required to merge a single  $d \times k$  parameter. Two classes of methods emerge, methods that run in  $\mathcal{O}(dk)$  vs. ones that run in  $\mathcal{O}(d^2k)$ . Precomputed statistics are calculated over T tokens and often require  $\mathcal{O}(MTd^2k)$  FLOPs, however, this only needs to be done once per task and can be amortized across many different merges. Note that the MLERP is the extension of SLERP used when M > 2.

<sup>936</sup> <sup>937</sup> In our calculations, reduction operations across models—such as sums—require (M - 1)dk FLOPs <sup>938</sup> and element-wise operations, such as scaling by  $\lambda$ , require dk FLOPs. Some element-wise operations <sup>939</sup> are applied to the parameter for each model independently, these require Mdk FLOPs. Thus are <sup>940</sup>

Average—Mdk FLOPs. Averaging requires a sum across models and a division by the number of models.

**Task Arithmetic**—(2M + 1)dk FLOPs. Mdk to compute the task vectors, the sum across task vectors, and two element-wise operations, scaling by  $\lambda$  and adding the pretrained parameters.

945<br/>946<br/>947DARE—(6M + 1)dk FLOPs. Assuming for simplicity that it requires 1 FLOP to generate a random<br/>number, DARE's addition of dropout requires an extra 2Mdk FLOPs to generate the dropout mask<br/>for each task vector—Mdk FLOPs to generates the random numbers and Mdk FLOPs to binarize<br/>it—Mdk FLOPs to apply the masks to the task vectors, and Mdk FLOPs to rescale parameters that<br/>were not dropped out. This it adds 4Mdk FLOPs on top of Task Arithmetic.

**TIES**—(4M + 1)dk FLOPs. TIES requires a sum of the trimmed parameters across models, 3dk to compute the sign for each parameter, find the majority sign, and replace zeros with the majority sign. Mdk is required to mask each parameter, and 2(M - 1)dk to sum the selected parameters, and the count of selected parameters, across models. The final division requires another dk FLOPs.

**Fisher**—(3M - 1)dk FLOPs. Each model's parameters are weighted by their Fisher Mdk, the Fishers are summed across models as are the weighted parameters 2(M - 1)dk, and finally dk FLOPs as the sum of the weighted parameters are divided by the summed Fishers.

**RegMean**— $(M + 2)d^2k + (3M - 2)dk$  FLOPs. The non-diagonal elements of each model's gram matrix is scaled.  $Md^2k$  FLOPs are required to multiply each parameter by its respective gram matrix. These are then summed across models, as are the gram matrices.  $d^2k$  FLOPs are used to invert the sum of the gram matrices and another  $d^2k$  FLOPs are used to multiple the scaled parameters and the inverted sum of gram matrices.

963 **MaTS**— $(M + N)d^2k + (2M + 5N - 2)dk$  FLOPs.  $Md^2k$  FLOPs are required to multiply the 964 Fishers and the parameters for each model and 2(M - 1)dk FLOPs are needed to sum the Fishers and 965 scaled parameters. Each iteration of the conjugate gradient method has 1 matrix vector multiplication 966  $(d^2k$  FLOPs), 2 inner products (2dk FLOPs), and 3 vector updates 3dk FLOPs). If a practitioner is 967 committed to only using MaTS merging, the Fisher-parameter multiplication can be folded into the 968 statistics calculation and lowers the computational cost to  $Nd^2k + (2M + 5N - 2)dk$ .

**SLERP**— $(5M-2)dk + (M+1)\log(dk)$  FLOPs.  $dk + \log(dk)$  FLOPs are used to calculate the norm (dk for the squaring of each parameter and  $\log(dk)$  for a parallelized sum of squares. The square root is constant can be ignored.). This is repeated for each of the M models. Then Mdk FLOPs are used to apply the calculated norms to each model. The dot product is calculated by multiplying

Merging Method	Time (Seconds)
Average	$1.2e^{-3} \pm 000.65e^{-3}$
Task Arithmetic	$1.9e^{-3} \pm 000.14e^{-3}$
DARE	$2.6e^{-3} \pm 000.16e^{-3}$
TIES	$2.1e^{-3} \pm 000.52e^{-3}$
Fisher	$0.8e^{-3} \pm 000.27e^{-3}$
RegMean	$49.9e^{-3} \pm 033.16e^{-3}$
<b>A</b> aTS	$4,280.5e^{-3} \pm 784.38e^{-3}$

Table 4: Time required to merge a single feed=forward layer of mt5-xl-lm. Timing from 144 merges with M=5, d=5,120, k=2,048, and N=50 were collected and we present the mean and standard deviation here. Again, the MaTS implementation is currently unoptimized and does many GPU to host transfers. SLERP reuslts are omitted as we no longer have the original hardware the used, making comparisons meaningless.

981

982

983

987 each parameter of the two models—2dk, (or more generally a multiplication of the parameters from 988 each constituent model, (M-1)dk—followed by a summation ( $\log(dk)$ ). The calculations based 989 on that dot product angle are constant, O(1), with respect to the number of parameters and can be ignored. Finally Mdk FLOPs are used to scale each model and (M-1)dk FLOPs are used to 990 sum the resulting models. When M=2, this cost is  $8dk + 3\log(dk)$  FLOPs. Some calculations, 991 such as the models norm, require information from the whole model to be aggregated. In some 992 implementations, these could be considered model statistics that are pre-computed and reused. This 993 would result in a statistic cost of  $Mdk + \log(dk)$  FLOPs and a merge cost of  $(3M - 2)dk + \log(dk)$ 994 FLOPs. 995

**MLERP**— $(2M + 3)dk + (M + 1)\log(dk) + \log(M)$  FLOPS. Again,  $Mdk + M\log(dk)$  FLOPs are used to compute the norm of each model. The average model is calculated in Mdk FLOPs. Then the norm of the average model is computed in  $dk + \log(dk)$  FLOPs and actual normalization is applied in dk FLOPs. Finally scaling by the maximum norm (found in  $\log(M)$  FLOPs with a parallel implementation) is done in dk FLOPs. As they are reusable across merges, the model norm calculations could be considered statistics that are pre-computed and reused. This yields a  $Mdk + M \log(dk)$  FLOPs statistic cost and a  $(M + 3)dk + \log(dk) + \log(M)$  merging cost.

In Fig. 3, we use the size of the transformer feed-forward layers to estimate the number of FLOPs required per layer. Feed-forward layers are generally larger than the linear layers used in attention, thus they create a upper bound on the amount of compute used to merge any linear layer. For DomainNet, we use d=3,072, k=768, M=24, and N=50. Similarly, we used d=5,120, k=2,048, M=5, and N=50 for the cross-lingual graphs.

1007 1008

1009

## D MERGING TIMES

Table 4 shows the amount of time required to merge a single feed-forward layer of mt5-xl-lm. We merge 5 models. The feed-forward layers are  $5,120 \times 2,048$ . 50 iterations of conjugate gradient were used for MaTS. We show the mean and standard deviation over 144 merges. We reiterate that the MaTS implementation is currently especially unoptimized. Despite that outlier, we see that RegMean, the only other  $\mathcal{O}(d^2k)$  method, is clearly much slower than the other methods, but is still much faster than fine-tuning.

Timings were recorded on a server with 2 Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz (12 cores/24 threads each), 256 Gigabytes of DDR4 RAM running at 2400 MT/s, and 4 NVIDIA RTX A6000 GPUs—driver version 535.129.03—connected via PCIe 3.0×16.

1020

#### 1021 E HYPERPARAMETER DETAILS

1022

1023 Several merging methods can be extended by including hyperparameters that scale each model-to-be-1024 merged, i.e., a shared  $\lambda$  becomes a model specific  $\lambda_m$ . This results in exponential growth of possible 1025 hyperparameters as more models are merged. Therefore, we do not explore per-model scaling terms; we use single, shared values when an algorithm includes a scaling hyperparameter.

Method	Hyperparameters	Values
Average	-	-
LERP	-	-
ask Arith.	$\lambda$ : scales the task vectors	[0.1, 1.0] by $0.1$
ARE	$\lambda$ : scales the task vectors	Reused
	<i>p</i> : dropout probability	[0.0, 0.9] by $0.1$
IES	$\lambda$ : scales the TIES task vectors	[0.1, 1.0] by $0.1$
isher	-	-
egMean	$\lambda$ : scales non-diagonal elements of the gram matrices	[0.0, 1.0] by $0.1$
ЛаTS	N: number of iterations to run conjugate gradient	[10, 100] by 10

1040

Table 5: Hyperparameters considered. We sweep hyperparameter values and select the best ones based on validation set performance. We reuse the best  $\lambda$  value from Task Arithmetic for DARE.

1041 1042 Similarly, some merging methods are built on top of others. For example, MaTs is initialized with 1043 Task Arithmetic, which requires running Task Arithmetic and selecting the best  $\lambda$  and DARE requires 1044 two hyperparameters: the dropout probability p and the Task Arithmetic scaling parameter  $\lambda$ . To reduce the space of possible hyperparameters, we first select  $\lambda$ —the one that works best for Task 1045 Arithmetic—and then vary the dropout probability p.

#### 1046 1047

1048 1049

#### F SAMPLING PROCEDURE FOR SCALING THE NUMBER OF TASKS

When we sample the m tasks from our set of T tasks to merge, we ensure that it contains all the tasks 1050 from the sample of m-1 tasks, i.e., the sample of m tasks is the previous sample of m-1 tasks 1051 and a newly sampled task. For example, if the sample of 2 tasks is  $\{A, C\}$ , then the sample of 3 1052 tasks will be  $\{A, C, X\}$  where  $X \sim T$  is a newly drawn sample. We repeat this iterative sampling 1053 procedure 20 times and end up with 20 different samples for each number of tasks. For example, the 1054 first sample for 2 tasks might consist of  $\{A, B\}$  and the first sample for 3 tasks might consists of 1055  $\{A, B, C\}$ . Meanwhile, the second sample for 2 tasks might consist of  $\{A, D\}$  and the first sample 1056 for 3 tasks might consists of  $\{A, D, C\}$ . 1057

We use this sampling procedure to try to avoid cases where the average performance on 3 tasks is stronger than for 2 tasks simply because the 3 sampled tasks were "easier" than the 2 that were sampled.

1061

#### 1062 G TIES IMPLEMENTATION

1063

1064 It its original form, TIES is the only method we evaluate which does not operate on each parameter block independently. We make a few modifications to allow for parameter block independence. 1066 First, the "trim" step zeros out the parameters with the smallest magnitude across the whole model. The original implementation does this during the merge itself; however, this would require loading 1067 all of the parameters, for all constituent models, at once. To make it possible to merge 5 3.7B 1068 parameter models, we treat the "trimmed" model as a model statistic which is precomputed for 1069 each model individually, avoiding the need to load them all together. Given this statistic, ours TIES 1070 implementation merges each parameter block of all of the trimmed models independently. This is 1071 the second slight difference in our TIES implementation. In the original implementation, during the 1072 "elect" phase, parameters without an elected sign—that is, parameters whose sum across models is 1073 zero—use the majority elected sign across the *whole model*, thus ensuring that every elected sign is 1074 either positive or negative. Instead of replacing signs of zero with the majority sign across the whole 1075 model, we place it with the majority elected sign across the parameter block. The majority elected 1076 sign across the whole model cannot be pre-computed as a model statistic as it depends on all of the constituent models in the merge. It would be possible to compute the majority elected sign across the 1077 whole model by keeping a running tally as each parameter block is loaded, but it would require a 1078 second pass over the parameter blocks to apply it. Such a large change would make TIES hard to 1079 compare to other methods in terms of computational cost and time, thus we opt to make this small

change in implementation to allow TIES to operate per-parameter block and make it feasible to run on our hardware. 

#### Η FULL RESULTS

Below we include the numerical values used in the various graphs above. Table 6, Table 7, and Table 8 are the numerical values from the left, center, and right graphs in Fig. 2. 

Merge Method	Held-In	Held-Out
Average	63.2	60.7
SLERP	63.1	60.7
Task Arithmetic	62.8	59.6
TIES	63.2	60.7
DARE	62.9	59.6
Fisher	63.5	60.5
RegMean	66.2	61.5
MaTS	66.4	61.5
Pretrained	60.8	59.4
Multitask	78.3	55.0
Individual Models	77.7	76.0

Table 6: Performance of different merging methods for image classification. These are the numerical values from Fig. 2 (left). 

11(

)4			
)5	Merge Method	Held-In	Held-Out
)6	Average	32.99	32.85
)7	SLERP	33.02	32.65
8	Task Arithmetic	31.71	31.59
	TIES	33.28	32.87
	DARE	31.72	31.56
	Fisher	32.9	32.77
	RegMean	32.94	32.89
	MaTS	32.66	32.65
	Pretrained	31.89	31.62
	Multitask	32.21	32.28
	Individual Models	32.26	32.39
	-		

Table 7: Performance of different merging methods for image generation. These are the numerical values from Fig. 2 (center). 

120			
121	Merge Method	Held-In	Held-Out
1122	Average	41.4	16.3
123	SLERP	41.3	16.4
1124	Task Arithmetic	44.7	18.3
125	TIES	47.6	18.4
125	DARE	44.8	18.3
126	Fisher	26.7	25.4
127	RegMean	50.5	15.6
128	MaTS	52.6	15.9
129	Pretrained	25.5	19.8
130	Multitask	68.1	18.8
131	Individual Models	72.8	55.2

Table 8: Performance of different merging methods for cross-lingual NLP. These are the numerical values from Fig. 2 (right).

Note that methods with no hyperparameter have their performance listed under index 5. Table 9 and Table 10 contain the numerical values used in Fig. 4 while Table 13 and Table 14 contain the values from Fig. 4. 

	Hyperparameter Index										
Method	0	1	2	3	4	5	6	7	8	9	10
Pretrained						60.8					
Average						63.2					
SLERP						63.1					
Task Arith.		62.9	55.0	36.2	13.6	2.5	0.4	0.3	0.3	0.3	0.3
TIES		62.3	63.2	63.8	63.9	63.6	63.0	62.1	61.0	59.5	57.6
DARE	62.9	62.9	62.9	62.9	62.9	62.8	62.9	62.9	62.9	62.9	
Fisher						63.5					
RegMean	63.5	64.4	64.8	65.1	65.3	65.5	65.6	65.8	66.0	66.2	0.3
MaTS		66.1	66.3	66.3	66.3	66.4	66.4	66.4	66.4	66.4	66.4
Multitask						78.6					

Table 9: Accuracy of different merging methods on the held-in tasks in the image classification setup for different hyperparameters. These are the numerical values from Fig. 4. See Section 4.4 for a descriptions of the hyperparameters. For methods without hyperparameters, we set the hyperpa-rameter index to 5.

Hyperparameter Index											
Method	0	1	2	3	4	5	6	7	8	9	10
Pretrained						59.4					
Average						60.7					
SLERP						60.7					
Task Arith.		59.6	52.0	35.1	13.9	2.5	0.4	0.2	0.2	0.2	0.2
TIES		60.4	60.8	60.9	60.7	60.2	59.5	58.6	57.4	56.0	54.2
DARE	59.6	59.6	59.6	59.6	59.6	59.6	59.6	59.6	59.6	59.5	
Fisher						60.5					
RegMean	60.8	61.2	61.2	61.3	61.3	61.4	61.4	61.4	61.4	61.5	0.3
MaTS	61.4	61.4	61.5	61.5	61.5	61.5	61.5	61.5	61.5	61.5	
Multitask						55.0					

Table 10: Accuracy of different merging methods on the generalization tasks in the image classification setup for different hyperparameters. These are the numerical values for Fig. 4. See Section 4.4 for a descriptions of the hyperparameters. For methods without hyperparameters, we set the hyperparameter index to 5. 

Table 15, Table 17, and Table 16 include the numerical values for Fig. 3. 

Table 18, Table 19, Table 20, Table 21, Table 22, and Table 23 contain the numerical values from Fig. 5. 

Hyperparameter Index										
Method	0	1 2	3	4	5	6	7	8	9	10
Pretrained					31.8					
Average					33.0					
SLERP					33.02					
Task Arith.	31.	7 27.4	24.8	23.4	22.5	22.5	23.5	23.5	23.3	23.7
TIES	33.	0 33.2	33.0	32.9	32.6	32.0	31.5	30.8	30.2	
DARE	31.	5 31.6	31.6	31.6	31.6	31.7	31.6	31.6	31.5	31.5
Fisher					32.9					
RegMean	32.	9 32.9	32.9	32.7	32.7	32.7	32.5	32.6	32.4	32.5
MaTS	32.	6 32.3	32.2	32.3	32.3	32.3	32.2	32.3	32.2	32.2
Multitask					32.2					

Table 11: CLIP score of different merging methods on the held-in tasks in the image generation setup for different hyperparameters. These are the numerical values from Fig. 4. See Appendix E for a descriptions of the hyperparameters. For methods without hyperparameters, we set the hyperparameter index to 5.

					Нуре	param	eter In	dex			
Method	0	1	2	3	4	5	6	7	8	9	10
Pretrained						31.6					
Average						32.8					
SLERP						32.6					
Task Arith.		31.5	26.9	24.8	23.3	22.7	22.8	23.7	23.6	23.6	24.0
TIES		32.8	32.8	32.9	32.6	32.4	31.9	31.2	30.4	29.7	
DARE		31.5	31.7	31.5	31.4	31.5	31.5	31.5	31.5	31.3	31.4
Fisher						32.7					
RegMean		32.8	32.7	32.8	32.7	32.8	32.8	32.7	32.7	32.6	32.5
MaTS		32.6	32.4	32.3	32.2	32.3	32.2	32.1	32.2	32.2	32.0
Multitask						32.2					

Table 12: CLIP score of different merging methods on the generalization tasks in the image generation setup for different hyperparameters. These are the numerical values for Fig. 4. See Appendix E for a descriptions of the hyperparameters. For methods without hyperparameters, we set the hyperparameter index to 5. 

				]	Hyperp	arame	ter Ind	ex			
Method	0	1	2	3	4	5	6	7	8	9	10
Pretrained						25.5					
werage						41.4					
SLERP						41.3					
ask Arith.		44.7	41.5	40.1	39.8	38.7	38.1	38.0	38.0	38.0	37.5
ΓIES		47.0	47.3	47.6	47.0	46.2	46.5	47.3	47.3	47.6	50.0
DARE	44.7	44.8	44.7	44.7	44.6	44.7	44.7	44.5	44.5	42.3	
Fisher						26.7					
RegMean	39.4	38.0	39.9	42.5	44.3	45.7	47.0	48.1	49.4	50.5	19.2
MaTS	47.0	50.6	51.9	52.2	52.6	52.7	52.7	52.7	51.6	52.5	
Multitask						68.1					

Table 13: Accuracy of different merging methods on the held-in tasks in the cross-lingual setup for different hyperparameters. These are the numerical values from Fig. 4 See Appendix E for a descriptions of the hyperparameters. 

I						Hyperparameter Index						
Method	0	1	2	3	4	5	6	7	8	9	10	
Pretrained						19.8						
Average						16.3						
SLERP						16.4						
Task Arith.		18.3	16.2	15.7	16.6	14.4	14.4	14.3	14.1	14.0	14.1	
TIES		26.1	25.1	23.7	23.3	23.2	22.9	22.3	21.6	20.7	20.0	
DARE	18.3	18.2	18.3	18.2	18.4	18.4	18.1	18.4	18.3	18.2		
Fisher						25.4						
RegMean	15.3	15.0	15.1	15.2	15.2	15.4	15.5	15.5	15.6	15.6	18.2	
MaTS	15.6	15.5	15.8	15.8	15.9	15.9	15.8	15.0	15.8	15.9		
Multitask						18.8						

Table 14: Accuracy of different merging methods on the generalization tasks in the cross-lingual setup for different hyperparameters. These are the numerical values from Fig. 4. See Appendix E for a descriptions of the hyperparameters. For methods without hyperparameters, we set the hyperparameter index to 5.

Method	<b>Compute Cost</b>	Held-in Acc.	Generalization Acc.
Average	1,843,200	63.2	60.7
SLERP	3,917,084	63.1	60.7
Task Arith.	3,763,200	62.8	59.6
TIES	7,449,600	63.2	60.7
DARE	11,136,000	62.9	59.6
Fisher	5,452,800	63.5	60.5
RegMean	1,538,918,400	66.2	61.5
MaTS	183,792,691,200	66.4	61.5

Table 15: Computational Cost and Performance for image classification on DomainNet. These are the numerical values for Fig. 3. 

Method	<b>Compute Cost</b>	Held-in CLIP-S	<b>Generalization CLIP-S</b>
Average	1,843,200	32.99	32.85
SLERP	3,917,084	33.02	32.65
Task Arith.	3,763,200	31.71	31.59
TIES	7,449,600	33.28	32.87
DARE	11,136,000	31.72	31.56
Fisher	5,452,800	32.9	32.77
RegMean	1,538,918,400	32.94	32.89
MaTS	47,012,505,600	32.66	32.65

Table 16: Computational Cost and Performance for DomainNet generation. These are the numerical values for Fig. 3.

Method	Compute Cost	Held-in Perf.	Generalization Perf.
Average	512,000	41.4	16.3
SLERP	1,331,270	41.3	16.4
Task Arith.	1,126,400	44.7	18.1
TIES	2,150,400	47.6	17.5
DARE	3,174,400	44.8	18.1
Fisher	1,433,600	26.7	24.8
RegMean	1,469,337,600	50.5	15.6
MaTS	1,077,412,659,200	52.6	15.8

Table 17: Computational Cost and Performance in the cross-lingual setting. These are the numerical values for Fig. 3.

				Ν	Merge M	lethod			
#T	Pre.	Avg.	SLERP	TA	TIES	DARE	Fisher	RM	MaTS
2	58.9	68.9	68.9	70.5	70.7	70.7	70.8	74.2	74.7
3	58.2	64.8	64.8	66.9	67.6	67.0	67.2	71.7	72.0
4	59.5	65.1	65.1	66.6	66.8	66.6	66.6	70.9	71.2
5	57.2	62.3	62.3	63.9	64.1	63.9	62.9	68.1	68.3
6	56.1	61.1	61.1	62.4	62.4	62.5	61.2	66.8	67.0
7	57.3	61.6	61.6	62.5	62.5	62.6	61.5	66.8	67.1
8	58.7	62.6	62.6	63.2	63.3	63.4	62.4	67.4	67.5
9	58.5	62.2	62.2	62.8	62.8	62.7	61.9	66.7	66.8
10	57.5	61.1	61.2	61.8	61.8	61.8	61.1	65.5	65.8
11	58.2	61.8	61.8	62.3	62.5	62.3	61.7	66.0	66.1
12	58.3	61.8	61.8	62.0	62.5	62.0	61.8	65.7	65.9
13	59.6	62.8	62.8	63.1	63.4	63.1	62.9	66.4	66.6
14	59.7	62.7	62.7	63.1	63.3	63.1	62.7	66.2	66.4
15	59.4	62.4	62.4	62.8	63.0	62.8	62.4	65.8	66.0
16	59.8	62.6	62.6	63.1	63.2	63.1	62.7	65.9	66.0
17	59.8	62.5	62.5	62.9	63.0	62.9	62.6	65.6	65.8
18	60.3	62.8	62.8	63.2	63.3	63.2	63.0	65.9	66.0
19	60.1	62.7	62.7	63.0	63.2	63.0	62.8	65.9	66.0
20	60.5	63.0	63.0	63.2	63.6	63.3	63.2	66.3	66.4
21	60.9	63.3	63.3	63.5	64.0	63.5	63.6	66.5	66.6
22	61.1	63.4	63.4	63.5	64.1	63.5	63.8	66.5	66.7
23	60.7	63.1	63.1	63.0	63.9	63.0	63.5	66.2	66.4

1333Table 18: Average accuracy (across 10 different samples) of different merging methods on1334the held-in tasks in the image classification setup when merging various number of tasks1335(#T). These are the numerical values from Fig. 5. The multitask performance and pretrained model1336performance can be found in Table 10. TA stands for Task Arithmetic and RM for RegMean.

Under review	as a conference	paper at ICLR 2025
	-	

55									
6									
7									
8									
59									
60									
61									
52									
3									
4					Merg	Method			
5	#T		~		mining				
6		Avg.	SLERP	TA	TIES	DARE	Fisher	RM	MaTS
7	2	57.2	57.2	55.8	51.4	56.4	56.8	56.8	52.4
	3	58.0	58.0	56.1	53.7	55.8	57.9	57.9	54.1
	4	58.9	58.9	57.4	55.6	56.3	58.8	58.8	55.6
)	5	59.4	59.4	57.9	57.1	56.9	59.3	59.3	57.1
	6	59.7	59.7	58.5	57.5	57.2	59.8	59.8	57.3
	7	60.0	60.0	58.9	57.9	57.9	60.1	60.1	57.8
	8	60.1	60.1	59.0	58.2	57.8	60.3	60.3	58.7
	9	60.2	60.2	59.3	59.0	58.8	60.5	60.5	58.7
	10	60.3	60.3	59.6	59.0	58.8	60.6	60.6	59.0
	11	60.3	60.3	59.7	58.9	59.2	60.7	60.7	58.9
	12	60.4	60.4	59.8	59.1	59.5	60.8	60.8	59.0
	13	60.4	60.4	59.9	60.3	59.8	61.0	60.9	60.1
	14	60.5	60.5	59.9	60.3	59.9	61.0	61.0	60.3
	15	60.5	60.5	60.0	60.3	60.2	61.1	61.1	60.3
	16	60.5	60.6	60.0	60.3	60.1	61.1	61.1	60.3
	17	60.6	60.6	60.1	60.3	60.3	61.2	61.2	60.3
	18	60.6	60.6	60.2	60.2	60.4	61.2	61.3	60.2
	19	60.6	60.6	60.2	60.1	60.5	61.3	61.3	60.1
	20	60.6	60.6	60.3	60.0	60.5	61.3	61.4	60.0
	21	60.6	60.6	60.4	60.0	60.6	61.4	61.4	60.0
	22	60.6	60.6	60.4	59.9	60.7	61.4	61.5	59.9
			() (	() =	50.0	60.9	615	615	50.8
	23	60.6	60.6	60.5	39.8	00.8	01.5	01.5	39.0
Tabla 10.	23	60.6	60.6	00.5	J9.8	omnlac) (	of differen	01.5	jana mot

1387Table 19: Average accuracy (across 10 different samples) of different merging methods on the1388generalization tasks in the image classification setup when merging various number of tasks1389(#T). These are the numerical values for Fig. 5. The multitask performance and pretrained model1390performance can be found in Table 9. TA stands for Task Arithmetic and RM for RegMean.

1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423

8		Merge Method										
	#T	Avg.	SLERP	ТА	TIES	DARE	Fisher	RM	MaTS			
	2	33.1	31.8	33.0	34.4	33.2	32.6	32.9	32.9			
	3	33.7	33.4	34.4	34.5	34.4	33.8	33.6	33.3			
	4	33.3	32.9	34.1	34.0	34.0	33.4	33.0	32.8			
	5	33.5	33.7	33.8	33.8	33.7	33.4	33.4	32.8			
	6	33.1	33.1	33.5	33.2	33.3	33.1	33.0	32.5			
	7	32.8	33.3	33.0	32.7	32.9	32.8	32.9	32.4			
	8	32.6	32.8	32.8	32.6	32.9	32.7	32.5	32.1			
	9	32.9	32.5	32.9	32.7	32.9	32.8	32.7	32.3			
	10	32.8	33.3	32.9	32.9	32.9	33.0	32.8	32.5			
	11	32.8	33.0	32.8	32.8	32.8	32.8	32.8	32.4			
	12	33.0	32.8	32.8	32.9	32.8	32.9	32.9	32.6			
	13	33.0	32.9	32.8	32.9	32.8	33.0	33.0	32.7			
	14	33.1	32.1	33.0	33.0	32.9	33.0	33.1	32.5			
	15	33.0	33.0	32.8	33.1	32.6	33.1	33.1	32.5			
	16	33.3	33.4	32.8	33.1	32.8	33.1	33.2	32.6			
	17	33.1	33.2	32.6	33.0	32.6	32.9	33.1	32.6			
	18	33.0	33.3	32.5	33.0	32.5	33.0	33.1	32.6			
	19	32.9	33.1	32.3	32.9	32.2	32.9	33.1	32.4			
	20	32.9	32.5	32.1	32.9	32.1	32.9	33.0	32.3			
	21	32.9	32.5	32.0	32.9	32.1	33.0	32.8	31.7			
	22	33.1	33.1	31.9	32.8	31.8	33.0	33.0	31.7			
	23	33.0	33.5	31.9	32.8	31.7	32.9	33.0	31.6			
	24	33.0	33.0	31.7	33.3	31.7	32.9	32.9	32.7			

Table 20: CLIP score of different merging methods on the held-in tasks in the image generation setup when merging various number of tasks (#T). These are the numerical values from Fig. 5. The multitask performance and pretrained model performance can be found in Table 11. TA stands for Task Arithmetic and RM for RegMean.

1459					Merg	e Method			
1460	#T				merg	e memou			
1461	" -	Avg.	SLERP	TA	TIES	DARE	Fisher	RM	MaTS
1462	2	32.4	32.5	32.4	32.7	32.4	32.2	32.3	32.1
1463	3	32.4	32.5	32.6	32.6	32.8	32.5	32.4	32.2
1464	4	32.5	31.8	32.7	32.6	32.7	32.5	32.4	32.2
1465	5	32.7	31.6	32.6	32.7	32.7	32.6	32.6	32.3
1400	6	32.7	32.4	32.8	32.7	32.7	32.6	32.6	32.2
1466	7	32.7	31.9	32.9	32.7	32.7	32.6	32.7	32.3
1467	8	32.7	32.4	32.8	32.7	32.8	32.7	32.7	32.4
1468	9	32.7	33.3	32.8	32.7	32.7	32.7	32.6	32.4
1469	10	32.8	32.4	32.8	32.7	32.8	32.7	32.7	32.5
1470	11	32.8	33.0	32.7	33.0	32.6	32.8	32.7	32.5
1471	12	32.8	32.7	32.7	32.8	32.7	32.8	32.9	32.5
1472	13	32.9	32.6	32.6	32.8	32.7	32.9	32.8	32.5
1/173	14	32.8	32.8	32.6	32.9	32.7	32.9	32.8	32.4
1475	15	32.9	32.0	32.7	32.8	32.5	32.8	32.8	32.3
1474	16	32.8	33.2	32.5	32.8	32.4	32.9	32.8	32.4
1475	17	32.9	32.2	32.5	32.8	32.4	32.9	32.7	32.4
1476	18	32.9	33.2	32.4	32.9	32.3	32.8	32.9	32.5
1477	19	32.8	33.4	32.2	32.7	32.2	32.8	32.9	32.4
1478	20	32.7	33.0	32.2	32.7	32.2	32.9	32.8	32.4
1479	21	32.9	33.0	32.0	32.7	32.0	32.8	32.8	31.8
1480	22	32.8	33.3	31.9	32.7	31.8	32.9	32.8	31.9
1481	23	32.8	32.9	31.7	32.7	31.7	32.9	32.8	31.9
1482	24	32.9	32.7	31.6	32.9	31.6	32.8	32.9	32.7

Table 21: CLIP score of different merging methods on the generalization tasks in the image generation setup when merging various number of tasks (#T). These are the numerical values from Fig. 5. The multitask performance and pretrained model performance can be found in Table 12.
 TA stands for Task Arithmetic and RM for RegMean.

		Merge Method											
#T	Pre.	Avg.	SLERP	ТА	TIES	DARE	Fisher	RM	MaTS				
2	25.5	49.8	50.0	59.2	60.1	59.5	54.1	62.0	63.1				
3	25.5	45.4	45.3	50.9	55.1	51.0	41.2	58.4	60.5				
4	25.5	42.5	42.5	47.5	51.0	47.6	31.2	53.1	55.2				

Table 22: Average performance (across 5 different samples) of different merging methods on the held-in tasks in the cross-lingual setup when merging various number of tasks (#T). These are the numerical values from Fig. 5. The multitask performance and pretrained model performance can be found in Table 13. TA stands for Task Arithmetic and RM for RegMean.

	Merge Method											
#T	Avg.	SLERP	TA	TIES	DARE	Fisher	RM	MaTS				
2	17.1	17.1	17.2	19.7	21.3	16.8	16.9	18.9				
3	17.9	18.1	20.2	19.9	22.2	16.1	16.1	20.1				
4	17.3	17.4	23.2	18.8	22.1	15.8	16.1	18.7				

Table 23: Average performance (across 5 different samples) of different merging methods on
the generalization tasks in the cross-lingual setup when merging various number of tasks
(#T). These are the numerical values for Fig. 5. The multitask performance and pretrained model
performance can be found in Table 14. TA stands for Task Arithmetic and RM for RegMean.

# <sup>1512</sup> I FID RESULTS FROM IMAGE GENERATION ON DOMAINNET

Along with CLIP-score, we also evaluated our models using FID metric as detailed in Appendix B.3.
Since there is a high correlation between both these metrics, we use CLIP-score as primary metric that captures the alignment between individual image-caption pairs, as compared to general statistics of image distribution captured by FID. We put the plots and tables corresponding to FID results below, please note that lower FID is better.



Figure 6: **FID of different merging methods across various hyperparameters in DomainNet generation.** These FID results complement the results provided in Fig. 2 and Fig. 4.

	Hyperparameter Index											
Method	1	2	3	4	5	6	7	8	9	10		
Pretrained					93.8							
Average					64.7							
Task Arith.	99.5	92.1	95.7	182.7	323.8	263.6	574.5	569.1	545.4	612.5		
TIES	47.7	54.7	66.3	74.2	91.7	99.5	105.0	109.2	111.2			
DARE	99.5	97.9	99.6	97.0	96.6	97.5	96.8	96.6	96.8	96.3		
Fisher					32.9							
RegMean	64.4	67.1	74.7	73.6	77.0	73.2	75.5	75.2	78.4	81.7		
MaTS	75.2	76.0	79.5	86.5	88.7	89.3	96.8	98.8	102.4	101.0		
Multitask					75.4							

Table 24: FID of different merging methods on the held-in tasks in the DomainNet generation setup for different hyperparameters. See Appendix E for a descriptions of the hyperparameters. For methods without hyperparameters, we set the hyperparameter index to 5.

	Hyperparameter Index									
Method	1	2	3	4	5	6	7	8	9	10
Pretrained					93.9					
Average					69.2					
Task Arith.	100.3	105.3	106.9	188.9	321.5	292.6	603.6	606.7	580.2	653.7
TIES	56.1	62.9	70.9	79.0	92.0	102.4	108.8	120.4	120.3	
DARE	97.9	99.4	100.6	101.4	97.9	97.5	98.3	98.3	95.5	101.7
Fisher					67.3					
RegMean	66.2	70.2	70.6	71.4	72.6	71.1	72.0	74.4	74.2	78.0
MaTS	72.3	78.6	78.3	84.9	84.3	90.4	93.3	96.7	94.6	100.5
Multitask					75.8					

Table 25: **FID of different merging methods on the generalization tasks in the DomainNet generation setup for different hyperparameters.** See Appendix E for a descriptions of the hyperparameters. For methods without hyperparameters, we set the hyperparameter index to 5.

#### J QUALITATIVE RESULTS OF IMAGE GENERATION

1565 We provide qualitative samples generated by our merged models from the experiments in Appendix B.3. For this, we sample 6 unique captions from the held-in and generalization splits, and



Figure 7: **Qualitatively comparing merging methods across captions in held-in set.** We use the best hyperparameters found by sweeping ranges mentioned in Appendix E.

visualize generated images from the merged models below Fig. 7 and Fig. 8. Please find more samples in supplementary.



Figure 8: **Qualitatively comparing merging methods across captions in generalization set.** We use the best hyperparameters found by sweeping ranges mentioned in Appendix E.