OptiMer: Optimal Distribution Vector Merging Is Better than Data Mixing for Continual Pre-Training

Anonymous ACL submission

Abstract

Continual pre-training is widely used to adapt LLMs to target languages and domains, yet the mixture ratio of training data remains a sensitive hyperparameter that is expensive to tune: it must be fixed *before* training begins, and a suboptimal choice can waste weeks of compute. In this work, we propose **OptiMer**, which decouples ratio selection from training: we train one CPT model per dataset, extract each model's distribution vector, which represents the parameter shift induced by that dataset, and search for optimal composition weights *post-hoc* via Bayesian optimization. Experiments on Gemma 3 27B across languages (Japanese, Chinese) and domains (Math, Code) show that OptiMer consistently outperforms data mixture and model averaging baselines with 15–35× lower search cost. Key findings reveal that 1) the optimized weights can be interpreted as data mixture ratios, and retraining with these ratios improves data mixture CPT, and 2) the same vector pool can be re-optimized for a given objective without any retraining, producing target-tailored models on demand. Our work establishes that data mixture ratio selection, traditionally a pre-training decision, can be reformulated as a post-hoc optimization over distribution vectors, offering a more flexible paradigm for continual pre-training.[1]

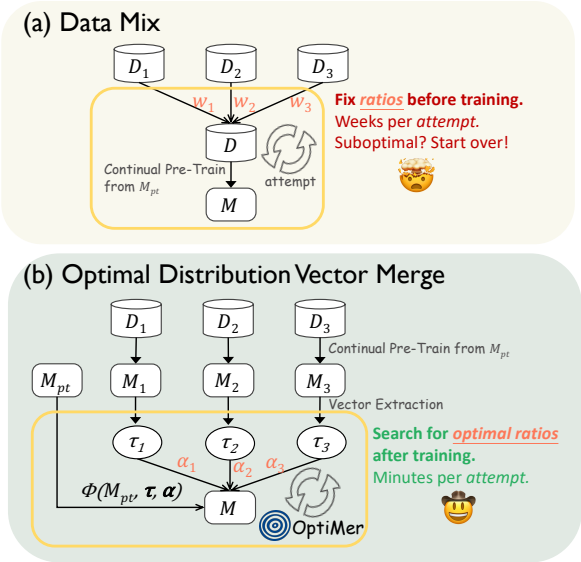


Figure 1: **Data Mix vs. OptiMer.** (*a*) Continual pre-training on a fixed data mixture requires the mixing ratios $\{w_i\}$ to be specified before training begins. Each attempt costs **days to weeks** of GPU time. (*b*) Our approach trains one CPT model per dataset independently and extracts a distribution vector τ_i from each, which are then composed via a merge function Φ (e.g. DARE-Linear) with weights $\{\alpha_i\}$ optimized post-hoc. Each trial completes in **minutes**. The instruction-tuned vector is additionally merged in both settings.

1 Introduction

Adapting large language models (LLMs) to specific languages and domains is a central challenge, driven by demand for both multilingual coverage and domain expertise (Ng et al., 2025; Alnumay et al., 2025; Yang et al., 2025; Lu et al., 2026).

Continual pre-training (CPT) is a common approach for such adaptation (Gururangan et al., 2020; Ibrahim et al., 2024; Yıldız et al., 2025), where the training corpus is typically a mixture of multiple datasets (Fujii et al., 2024; Dou et al., 2025). However, the mixing ratio of these datasets is a critical yet sensitive hyperparameter: a suboptimal ratio can degrade model performance (Xie et al., 2023; Ye et al., 2025). Although recent methods estimate ratios via proxy models or small-scale experiments (Xie et al., 2023; Liu et al., 2025; Ye et al., 2025; Cao et al., 2026a), these estimates must be fixed before training begins and cannot be corrected afterward, meaning a poor choice may waste days or even weeks of GPU cluster time before its effect becomes apparent.

To address this, we propose OptiMer, which decouples data ratio selection from model training. As illustrated in Figure 1, instead of fixing

[1] Code and models will be released upon acceptance.

the data mixture ratio *before training*, we train a separate CPT model on each dataset independently and extract the corresponding **distribution vector** (the parameter shift from the base PT model) *after training*. Furthermore, rather than weight averaging, which leads to suboptimal performance (Yadav et al., 2023), OPTIMER searches for optimal merge weights via Bayesian optimization using the Tree-structured Parzen Estimator (TPE) (Akiba et al., 2019; Watanabe, 2023). We find vector merging viable because vectors from distinct datasets are approximately orthogonal, allowing linear combination with minimal interference.

Experiments on Gemma 3 27B (Team et al., 2025) with distribution vectors across languages (Japanese, Chinese) and domains (Math, Code) show that OPTIMER consistently outperforms data mixture baselines across all dataset combinations, while requiring 15–35× lower search time. Moreover, the same collection of distribution vectors can be re-optimized toward different objectives, yielding multiple target-tailored models without any retraining. Our contributions are as follows:

- We introduce the concept of **distribution vectors** for CPT and propose **OPTIMER**, a post-hoc framework that decouples data ratio selection from model training by optimizing merge weights via Bayesian optimization.
- Experimental results on 16 benchmarks covering five task groups (English, Japanese, Chinese, Math, Code) show that OPTIMER outperforms data mixture CPT and four model merging methods across three dataset combinations with 15–35× lower search cost. It further enables objective-specific re-optimization from a single vector pool without any re-CPT.
- Our analysis reveals that distribution vectors are approximately orthogonal (cosine 0.03–0.31), enabling composition without severe interference. Training dynamics show that CPT trajectories are approximately linear in parameter space, linking merge weights to effective training duration. OPTIMER search dynamics illustrate the sharp nature of the optimization landscape, thus highlighting the necessity of efficient searching rather than grid search. Additionally, optimized weights can serve as interpretable data mixture ratios and can be negative to remove cross-distribution interference.

2 Related Work

Continual Pre-training. Adapting a pretrained LLM to new languages or domains via CPT is a well-studied area (Gururangan et al., 2020; Li and Lee, 2024). It has been applied to language adaptation (Fujii et al., 2024; Dou et al., 2024, 2025) and domain adaptation (Azerbayev et al., 2024; Lozhkov et al., 2024; Wu et al., 2024). Data mixture ratio is an important hyperparameter that largely affects model performance (Li and Lee, 2024; Shi et al., 2024), which motivates work on data mixture optimization.

Data Mixture Optimization. Recently, several methods have been proposed to optimize data mixture ratios. DoReMi (Xie et al., 2023) uses distributionally robust optimization on a small proxy model to produce domain weights for a larger target model. RegMix (Liu et al., 2025) trains many small models on diverse mixtures and fits a regression to predict optimal ratios. Ye et al. (2025); Cao et al. (2026a) propose a predictive framework that transfers optimal ratios across scales. Despite these advances, such methods must fix the ratio before training. Instead, we propose to adjust ratios post-hoc which avoids retraining.

Task Vectors and Model Merging. Ilharco et al. (2023) show that task vectors $\tau = \theta_{\text{ft}} - \theta_{\text{base}}$, the difference between a fine-tuned model θ_{ft} and its base model θ_{base}, can be composed via linear arithmetic to add or remove task capabilities, with subsequent work improving merging quality through sign conflict resolution (Yadav et al., 2023) and delta sparsification (Yu et al., 2024). Chat Vector (Huang et al., 2024) applies weight arithmetic to transfer instruction-following capability to a CPT-adapted model without additional fine-tuning. Task-specific CPT checkpoint and LoRA adapter merging has also proven effective for finance domain (Ueda et al., 2025) and machine translation (Cao et al., 2026b). These works focus on task-specific transfer rather than improving general capability across multiple distributions. In contrast, our work extends distribution vector composition to the multi-distribution CPT setting and achieves general performance improvement.

Automatic Merge Weight Search. Several methods automate the search for merge ratios, including test-time entropy minimization over per-layer weights (Yang et al., 2024), evolutionary search (Akiba et al., 2025), and minimizing output

divergence between merged and fine-tuned models (Touayouch et al., 2026). They have been applied to at most two to three models or small-scale models due to the computational cost of the high-dimensional search spaces or population-based iterations. Most relevant to our work, DEM (Ram et al., 2024) applies grid search over merge weights for SFT task vectors, but the cost of grid search increases exponentially with the number of vectors. Our proposed OPTIMER replaces grid search with Bayesian optimization via TPE, achieving substantially higher theoretical search efficiency.

3 Methodology

We define the notation and introduce distribution vectors in Section 3.1. In Section 3.2, we present OPTIMER, an automatic merge weight optimization approach via Bayesian optimization.

3.1 Preliminaries

Notation. Let $\theta_{\text{pt}} \in \mathbb{R}^d$ denote the parameters of a pretrained base model and θ_{it} its instruction-tuned version. Given n data distributions $\{D_i\}_{i=1}^n$, each represented by a dataset, continual pre-training on D_i from θ_{pt} yields a CPT model θ_{CPT_i}.

Task Vectors. Ilharco et al. (2023) define a *task vector* $\tau = \theta_{\text{ft}} - \theta_{\text{base}}$ to capture the parameter change induced by fine-tuning, and construct a merged model as $\theta_{\text{merge}} = \theta_{\text{base}} + \lambda\tau$, where λ is a scalar weight. This has been shown effective for adding or removing capabilities from fine-tuning.

Distribution Vectors. We extend task vectors to the CPT setting. We define the **distribution vector** for D_i as:

$$\tau_i = \theta_{\text{CPT}_i} - \theta_{\text{pt}}, \tag{1}$$

which encodes the parameter change induced by distribution D_i. Similarly, we extract an **IT vector** $\tau_{\text{it}} = \theta_{\text{it}} - \theta_{\text{pt}}$ from the instruction-tuned model. Since our CPT models are trained from θ_{pt}, they lack instruction-following capability, and adding τ_{it} recovers this capability without additional supervised fine-tuning (Huang et al., 2024).

Multi-Vector Composition. A merged model incorporating n distributions and instruction-following capability is constructed as:

$$\theta_{\text{merge}} = \theta_{\text{pt}} + \alpha_{\text{it}} \cdot \tau_{\text{it}} + \sum_{i=1}^{n} \alpha_i \cdot \tau_i, \tag{2}$$

where $\{\alpha_i\}_{i=1}^n$ and α_{it} are scalar merge weights. Uniform weighting ($\alpha_i = 1/n$) is a natural baseline but leads to suboptimal performance in practice, as different distributions contribute unequally to the target objective. The central question is then how to find the optimal weights $\boldsymbol{\alpha}^* = (\alpha_{\text{it}}, \alpha_1, \ldots, \alpha_n)$ efficiently.

3.2 OPTIMER: An Automatic Merge Weight Optimization Algorithm

Problem Formulation. The merged model in Eq. (2) is parameterized by the weight vector $\boldsymbol{\alpha} = (\alpha_{\text{it}}, \alpha_1, \ldots, \alpha_n) \in \mathbb{R}^{n+1}$. We formulate the weight search as the optimization problem:

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} \ \mathcal{S}\big(\theta_{\text{merge}}(\boldsymbol{\alpha}), \mathcal{D}_{\text{dev}}\big), \tag{3}$$

where $\mathcal{S}$ is an evaluation score computed on a development set $\mathcal{D}_{\text{dev}}$. Since $\mathcal{S}$ is obtained by running discrete benchmark evaluations, it provides no gradient with respect to $\boldsymbol{\alpha}$, making this a black-box optimization problem. A straightforward approach is grid search (Ram et al., 2024), but its cost is $O(G^{n+1})$ for G grid points per dimension, which becomes impractical as the number of vectors grows.

Bayesian Optimization via TPE. We solve Eq. (3) using the Tree-structured Parzen Estimator (Bergstra et al., 2011), a Bayesian optimization method implemented in Optuna (Akiba et al., 2019). Given N observed trials, where each trial consists of constructing a merged model with a candidate $\boldsymbol{\alpha}$ and evaluating it, TPE partitions the N trials by a quantile γ (e.g. 10%) into a *good* set and a *bad* set based on their performance on $\mathcal{D}_{\text{dev}}$. Two separate density models are estimated via kernel density estimation:

$$p(\boldsymbol{\alpha} \mid \mathcal{S}) = \begin{cases} \ell(\boldsymbol{\alpha}) & \text{if } \mathcal{S} \geq s^*, \\ g(\boldsymbol{\alpha}) & \text{if } \mathcal{S} < s^*, \end{cases} \tag{4}$$

where s^* is the top-γ quantile of observed scores, ℓ models the density of high-scoring configurations, and g models the rest. The next candidate is selected by maximizing the ratio $\ell(\boldsymbol{\alpha})/g(\boldsymbol{\alpha})$, concentrating sampling in promising regions of the weight space. While grid search requires $O(G^{n+1})$ evaluations for G grid points per dimension, TPE typically converges in $O(10n)$ trials (Watanabe, 2023), making it practical even as the number of vectors grows. Furthermore, TPE can sample candidates independently, enabling parallel trial execution on multiple GPUs.

Algorithm 1: OPTIMER

Input: Base model θ_{pt}, distribution vectors $\{\tau_i\}_{i=1}^{n}$, IT vector τ_{it}, dev set $\mathcal{D}_{\text{dev}}$, number of trials T, top-K

Output: Optimized weights $\boldsymbol{\alpha}^*$, merged model θ^*_{merge}

Sample N_0 random trials to initialize density models ℓ, g
for $t = N_0 + 1$ **to** T **do**
  $\boldsymbol{\alpha}^{(t)} \leftarrow \arg\max_{\boldsymbol{\alpha}} \ \ell(\boldsymbol{\alpha})/g(\boldsymbol{\alpha})$
  $\theta^{(t)}_{\text{merge}} \leftarrow \theta_{\text{pt}} + \alpha^{(t)}_{\text{it}}\,\tau_{\text{it}} + \sum_{i=1}^{n} \alpha^{(t)}_i\,\tau_i$
  $s^{(t)} \leftarrow \mathcal{S}(\theta^{(t)}_{\text{merge}}, \mathcal{D}_{\text{dev}})$
  Update ℓ, g with $(\boldsymbol{\alpha}^{(t)}, s^{(t)})$
end
Re-evaluate top-K trials on full $\mathcal{D}_{\text{dev}}$
$\boldsymbol{\alpha}^* \leftarrow \arg\max_{\text{top-}K} \ \mathcal{S}$
$\theta^*_{\text{merge}} \leftarrow \theta_{\text{pt}} + \alpha^*_{\text{it}}\,\tau_{\text{it}} + \sum_{i=1}^{n} \alpha^*_i\,\tau_i$
return $\boldsymbol{\alpha}^*$, θ^*_{merge}

Algorithm. Algorithm 1 summarizes the OPTIMER pipeline. The search begins with N_0 random trials to initialize the TPE density models ℓ and g. In subsequent trials, TPE proposes a candidate $\boldsymbol{\alpha}^{(t)}$ by maximizing ℓ/g; then a merged model is constructed via Eq. (2) and scored on a subset of the development set; finally the density models are updated with the new observation. After T trials, the top-K configurations are re-evaluated on the full development set to obtain the final model θ^*_{merge}.

4 Experimental Settings

This section describes settings for continued pre-training (§4.1), model merge and OPTIMER (§4.2), baseline methods (§4.3), and evaluation (§4.4).

4.1 Continual Pre-Training Settings

We sampled training data from the LLM-jp Corpus v4 (LLM-jp et al., 2024)[2] to construct CPT datasets across languages (Japanese, Chinese) and domains (Math, Code), each containing 1B tokens. For data mixture baselines, datasets were combined at equal ratios with nB tokens in total. We continually pre-trained `gemma-3-27b-pt` (Team et al., 2025) for 1 epoch (2,000 steps) on each dataset, with sequences packed to 4,096 tokens and an effective batch size of 128. Following Fujii et al. (2024), we use AdamW (Loshchilov and Hutter, 2019) (β_1=0.90, β_2=0.95, weight decay 0.1, gradient clipping 1.0) with a peak learning rate of 4×10^{-5} and cosine decay to 1%. We report the effect of different hyperparameter settings in Appendix A. Training used BFloat16 with DeepSpeed ZeRO Stage 3 on 8 NVIDIA H200 GPUs (141 GB) on the ABCI 3.0 cluster.

[2] https://gitlab.llm-jp.nii.ac.jp/datasets/llm-jp-corpus-v4

4.2 Merge Settings

We used `gemma-3-27b-it` (Team et al., 2025) to calculate τ_{it}. Merges were performed with DARE-Linear (Yu et al., 2024) via `mergekit` (Goddard et al., 2024), excluding embedding and positional layers (`embed_tokens`, `lm_head`, `rotary`) to preserve the base model's token representations.

For OPTIMER, we ran T=100 trials with the TPE sampler (N_0=20 random startup trials), executed in parallel across 8 GPUs. The search space was set to $\alpha_{\text{it}} \in [0.3, 1.0]$ and $\alpha_{CPT_i} \in [0.0, 1.0]$.

Proxy tasks were selected to match the target axes of each merge experiment (e.g., `gsm8k` and `ja_leaderboard_mgsm` for a Japanese×Math merge). During the search, each trial was scored on the first 100 samples per proxy task for efficiency. The top-K=3 trials were then re-evaluated on the first 300 samples per task as the development set.

4.3 Baselines

We compared with the following baseline methods.

DataMix. We trained a single CPT model on the concatenation of all datasets in each combination (nB tokens in total for n datasets), and merged it with the IT vector with optimized hyperparameter α_{it}=0.6 (§A) to recover IT capability.

DataMix$_{\text{OptiMer ratio}}$ models were trained on the same nB tokens with the data mixing ratio directly derived from the optimal merge weights found by OPTIMER.

Average Merge. We merged n CPT vectors and the IT vector using equal weights via Task Arithmetic (Ilharco et al., 2023), TIES (Yadav et al., 2023), and DARE (Yu et al., 2024).

4.4 Evaluation Settings

We used the `lm-evaluation-harness` (Gao et al., 2024) framework with the vLLM backend (Kwon et al., 2023), using 1-shot prompting across all tasks on these five task groups:

Combination	Method	English				Math	Code		Japanese	Chinese	Avg.
		MMLU	ARC-C	HellaSwag	TQA	GSM8K	HE	MBPP	JA LB	C-Eval	
	Gemma 3 27B PT	31.59	27.30	48.41	54.71	0.83	0.00	0.60	39.04	27.41	25.54
	Gemma 3 27B IT	51.03	34.90	55.91	64.87	15.39	2.44	37.80	54.64	37.96	39.44
	Single CPT, merged with IT (α_{it}=0.6)										
	CPT_{En}	73.60	69.80	82.82	46.39	79.98	71.95	68.40	43.06	62.56	66.51
	CPT_{Ja}	72.11	67.06	81.98	55.32	83.78	68.29	68.20	72.50	62.48	70.19
	CPT_{Zh}	72.85	68.94	81.70	48.84	82.18	71.95	67.80	57.98	62.85	68.34
	CPT_{Math}	75.07	70.14	82.72	43.57	82.64	57.32	68.00	72.79	63.37	68.40
	CPT_{Code}	71.86	70.05	81.71	51.16	81.96	64.63	68.40	73.39	62.26	69.49
Ja + Math	$\text{DataMix}_{\alpha_{\text{it}}=0.6}$ (Baseline)	73.17	68.77	81.72	49.57	83.62	50.00	68.80	73.34	61.74	67.86
	$\text{DataMix}_{\text{OptiMer ratio}}$ (Ours)	72.72	**70.05**	82.62	50.06	76.19	**67.07**	64.80	72.85	61.96	68.70
	Task Arithmetic (Ilharco et al., 2023)	**75.44**	69.88	83.31	41.49	74.45	60.37	1.60	**75.11**	**63.67**	60.59
	TIES (Yadav et al., 2023)	71.14	65.70	80.03	30.11	69.52	26.22	0.00	72.40	58.84	52.66
	DARE-Linear (Yu et al., 2024)	75.02	69.71	**83.74**	42.96	78.17	58.54	65.00	72.37	63.30	67.65
	DARE-TIES (Yu et al., 2024)	74.72	68.94	82.75	39.53	73.77	60.98	0.80	74.39	62.33	59.80
	OptiMer (Ours)	73.17	68.34	82.50	**55.20**	**84.46**	59.76	**70.40**	72.53	63.45	**69.98**
Ja + Code	$\text{DataMix}_{\alpha_{\text{it}}=0.6}$ (Baseline)	71.23	69.20	81.14	43.08	78.62	64.02	65.00	72.38	60.40	67.23
	$\text{DataMix}_{\text{OptiMer ratio}}$ (Ours)	**72.35**	68.60	80.22	51.77	74.00	55.49	67.60	**74.66**	**62.26**	67.44
	Task Arithmetic (Ilharco et al., 2023)	72.33	**70.39**	82.44	42.72	78.17	45.12	60.20	74.04	61.96	65.26
	TIES (Yadav et al., 2023)	66.44	66.98	79.69	36.11	70.58	43.29	53.60	70.22	54.75	60.18
	DARE-Linear (Yu et al., 2024)	72.33	70.31	**82.50**	42.84	79.08	46.34	60.60	74.07	62.18	65.58
	DARE-TIES (Yu et al., 2024)	71.14	69.37	82.00	41.49	76.04	42.68	60.40	73.30	60.25	64.07
	OptiMer (Ours)	71.41	66.89	81.44	**54.22**	**83.47**	**64.63**	**70.20**	73.06	61.81	**69.68**
Ja + Zh + Math	$\text{DataMix}_{\alpha_{\text{it}}=0.6}$ (Baseline)	73.83	69.03	80.54	37.70	79.61	32.32	65.40	72.77	62.18	63.71
	$\text{DataMix}_{\text{OptiMer ratio}}$ (Ours)	74.34	67.58	81.80	44.80	81.27	66.46	67.20	**74.05**	60.48	68.66
	Task Arithmetic (Ilharco et al., 2023)	**75.27**	**69.97**	83.18	38.92	73.54	45.73	0.40	73.17	**64.41**	58.29
	TIES (Yadav et al., 2023)	70.13	66.72	79.65	26.68	65.81	14.63	0.20	69.86	60.03	50.41
	DARE-Linear (Yu et al., 2024)	74.48	69.28	**83.65**	38.68	76.42	50.00	62.00	70.92	63.89	65.48
	DARE-TIES (Yu et al., 2024)	74.06	69.54	82.59	35.86	70.36	45.12	0.40	72.38	63.60	57.10
	OptiMer (Ours)	72.94	68.60	82.98	**51.77**	**83.93**	**67.07**	**70.20**	72.65	63.22	**70.37**

Table 1: Main results comparing OptiMer with baselines across dataset combinations. **Single**: individual domain CPT merged with IT ($\alpha_{\text{it}} = 0.6$); **DataMix**: multi-domain CPT with equal data ratios, merged with IT ($\alpha_{\text{it}} = 0.6$); **OptiMer**: Bayesian-optimized merge weights over all distribution vectors. **Task Arithmetic**, **TIES**, and **DARE** apply uniform weighting among all models including the IT model. **Bold** indicates the best result per combination. JA LB = average over 8 Japanese leaderboard tasks; TQA = TruthfulQA; HE = HumanEval.

En (MMLU (Hendrycks et al., 2021), ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), TruthfulQA (Lin et al., 2022)), *Ja* (8 tasks from the Japanese Leaderboard (Cobbe et al., 2021; Hasan et al., 2021; Kurihara et al., 2022; Tikhonov and Ryabinin, 2021) [3]) , *Zh* (C-Eval (Huang et al., 2023)), *Math* (GSM8K (Cobbe et al., 2021)), and *Code* (HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021)). We report **Avg.**, the unweighted mean of all tasks, with the Japanese Leaderboard calculated as one task. Benchmark details are provided in Appendix B. We also show 3-shot results in Appendix C.

5 Results and Analysis

We compare OptiMer to baselines (§5.1), analyze distribution vectors (§5.2), training dynamics (§5.3), and optimization dynamics (§5.4) to understand how and why OptiMer works. We further conduct experiments with negative vector weight (§5.5). Finally, we apply OptiMer to build a Japanese-optimized LLM (§5.6).

5.1 Main Results

Performance. As shown in Table 1, OptiMer achieves the highest average score across all dataset combinations, outperforming the DataMix baseline in each group by 2.1–6.7 points. We make the following observations: (i) Single-domain CPT models already perform well, yet DataMix shows lower performance despite using more training data, indicating its sensitivity to suboptimal mixture ratios. (ii) Model averaging methods such as DARE-Linear achieve reasonable overall scores, but suffer from catastrophic failures on Code tasks. After inspecting outputs, we found these models generate syntactically malformed code (e.g., missing indentation), rather than hallucinated content. (iii) OptiMer maintains strong TruthfulQA (TQA) scores (51–55) where all other methods degrade significantly (30–49), suggesting that optimized weights

[3] https://github.com/EleutherAI/lm-evaluation-harness/blob/main/lm_eval/tasks/japanese_leaderboard

Objective	English				Math	Code		Japanese	Chinese	Avg.
	MMLU	ARC-C	HellaSwag	TQA	GSM8K	HE	MBPP	JA LB	C-Eval	
English	72.87	**69.62**	82.02	**47.74**	82.49	44.51	71.40	69.56	62.93	65.35
Math	70.10	63.05	79.16	41.13	82.87	4.88	**72.60**	68.47	58.99	58.37
Japanese	**73.96**	68.86	**82.73**	46.39	**84.69**	**68.29**	69.20	**73.37**	62.85	**69.07**
Chinese	71.93	69.11	82.37	34.76	72.71	11.59	1.60	40.78	**63.60**	44.29

Table 2: Flexibility of OptiMer: the same vector pool {En, Math, Ja, Zh} is re-optimized for one of the four task groups {En, Math, Ja, Zh} in four separate runs, each with 100 trials. **Bold** indicates the best value in each column. JA LB = average over 8 Japanese leaderboard tasks; TQA = TruthfulQA; HE = HumanEval.

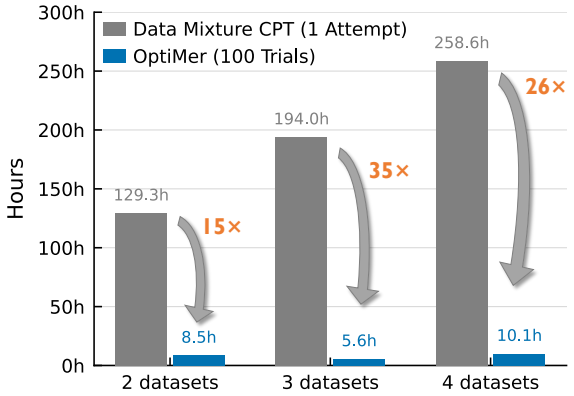


Figure 2: **Computational cost** comparison between data mixture CPT and OptiMer across different numbers of datasets **during ratio optimization**. Training cost excluded as it is identical for both approaches.

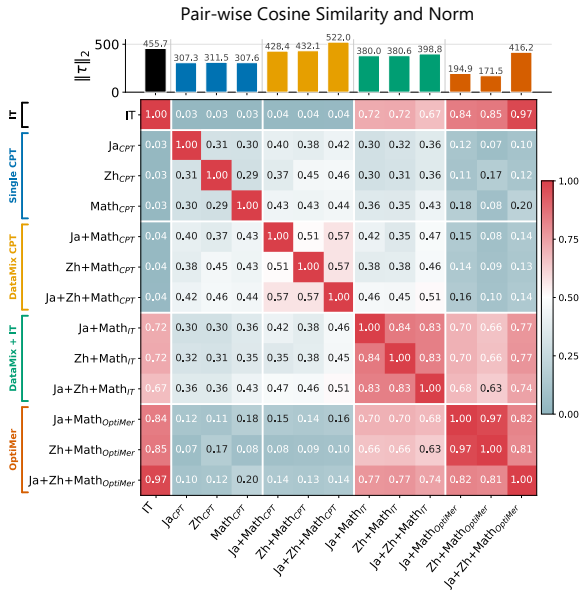


Figure 3: Pairwise cosine similarity and norm of distribution vectors.

better preserve the base model's calibration. We present case studies in Appendix G to illustrate their qualitative difference.

Additionally, optimal merge weights can be interpreted as post-hoc data mixture ratios. **DataMix$_{\text{OptiMer ratio}}$** first converts weights into dataset proportions and retrains DataMix models with these ratios to form a training set with 2B or 3B data. Across all combinations, it outperforms the uniform ratio DataMix baselines, e.g., in Ja+Zh+Math, the average improves from 63.71 to 68.66. This confirms DataMix suffers from suboptimal ratio selection, and OptiMer discovers better ratios without further training. Furthermore, OptiMer still achieves the best performance, suggesting the advantage of post-hoc composition.

Efficiency. OptiMer is 15–35× faster than DataMix for searching optimal ratios, and this advantage becomes larger with more datasets, as shown in Figure 2. In ratio searching, a 100-trial OptiMer search completes in 8.6 hours, compared to 128.9 hours for a single DataMix run. We found each OptiMer trial consists of a merge (10.2% of trial time) and an evaluation (89.8%), so the cost is nearly constant regardless of n, whereas DataMix cost scales with the data size. Note **that the training cost is the same**: OptiMer trains n models on 1B tokens each, while DataMix trains one model on nB tokens.

Flexibility. OptiMer can produce an objective-optimized model on demand without retraining. Table 2 shows the results re-optimizing for different objectives using the same four distribution vectors {Ja, Zh, En, Math}. We found that (i) in most cases, the model optimized for a given objective yields the highest score on its target tasks (e.g., the model optimized for Chinese tasks achieves the best C-Eval score), and (ii) the Japanese-optimized model also achieves the best overall performance, suggesting that Japanese data also benefits multilingual performance. We leave the investigation of this cross-lingual transfer effect to future work.

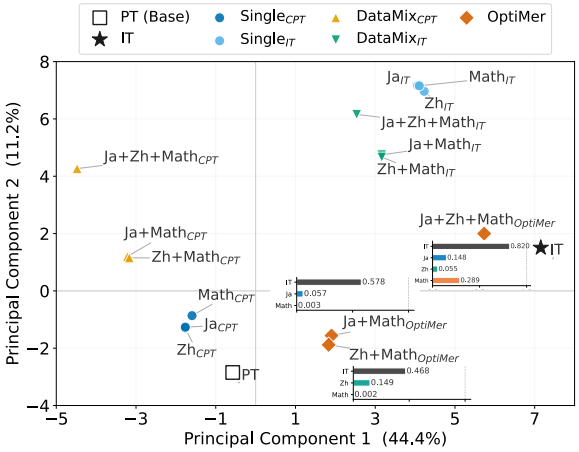


Figure 4: PCA projection of distribution vectors with OPTIMER merge weights in bar charts.

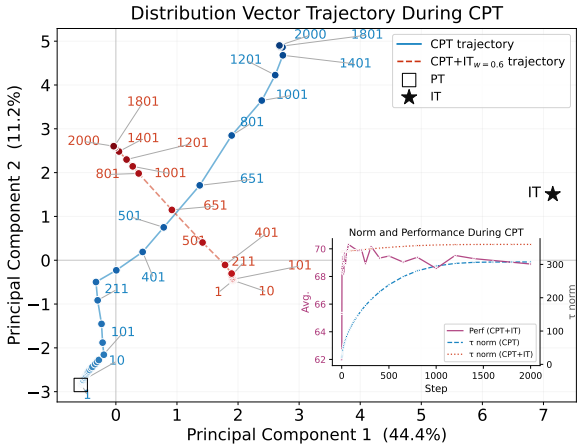


Figure 5: Distribution vector trajectories during CPT on 1B Japanese data, projected onto the same PCA space as used in Figure 4.

5.2 Analysis of Distribution Vectors

We show the pair-wise cosine similarity of distribution vectors (i.e. $\theta - \theta_{\text{pt}}$) in Figure 3. We found CPT and IT vectors are nearly orthogonal (cosine ≈ 0.03), and different CPT vectors also exhibited low similarity (0.29–0.31), indicating that each distribution modifies an independent subspace, which supports the feasibility of linear composition. Layer-wise similarity analysis and cosine similarity of more models are shown in Appendix D. Figure 4 visualizes the same vectors (sparsified through layer-wise truncated SVD) via PCA. The accompanying bar charts show the optimal merge weights. Both confirm that CPT vectors lie far from the IT vector. Two additional insights emerge from combining both figures: (i) DataMix models with more datasets drift further from IT (in both cosine similarity and PCA distance), showing that CPT dilutes IT capability, while OPTIMER is unaffected, maintaining a cosine similarity of greater than 0.97 with θ_{it}. This explains the widening performance gap in Table 1 (+2.1 for 2-way vs. +6.7 for 3-way). (ii) OPTIMER assigns large weight to IT and small weights to CPT vectors, suggesting that IT-targeted perturbation is more effective than uniform average.

5.3 Continual Pre-Training Dynamics

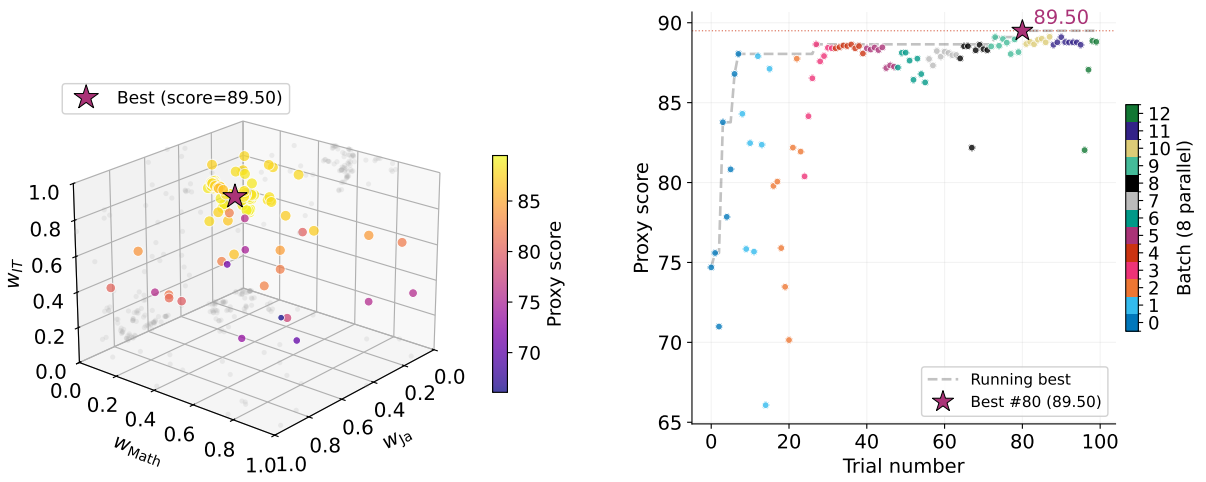
We show the vector trajectories during training in Figure 5. We observed that both the *CPT* and *CPT merged with IT* vector trajectories move away from the IT vector, with rapid change in early steps. We found performance reached its peak in the early stage with small vector norm, which is consistent with the thicket regime phenomenon (Gan and Isola, 2026), and decreased gradually, possibly due to the divergence from the base model (Bolton et al., 2026). Furthermore, the CPT trajectory is approximately linear, indicating that adjusting the merge weight α_i is analogous to controlling the effective training duration, which also explains why OPTIMER assigns small CPT weights.

5.4 OptiMer Search Dynamics

Figure 6 visualizes the search dynamics of OPTIMER on the Ja+Math setting. We found that (i) weight combinations with high scores are concentrated in a narrow region with large w_{IT} and small CPT weights. This sharp optimum makes grid search impractical, whereas TPE quickly approaches the promising region and focuses exploration within it, and (ii) OPTIMER converges within 100 trials, confirming the sample efficiency of TPE-based search for this problem. Visualizations for other dataset combinations are shown in Appendix E. We also provide a version of 500 trials in Figure 11a where we see a clearer trend with more data points in the space.

5.5 Search with Negative Weights

We conducted experiments extending the search range from $[0, 1]$ to $[-1, 1]$, allowing OPTIMER to assign negative weights that subtract a distribution's effect from the model (Ilharco et al., 2023). This improves performance for the Ja and Zh objectives (Table 4). Notably, the English vector often receives negative weights, suggesting it may introduce interference and OPTIMER actively removes its effect as a regularization process.



(a) Exploration of the merge weight space. Each point represents one trial; color indicates proxy score.

(b) Optimization progress. Each point represents one trial; color indicates parallel batch.

Figure 6: OptiMer search dynamics for Ja+Math combination.

Model	English				Math	Code		Japanese	Chinese	ASEAN Lang	
	MMLU	ARC-C	HSwag	TQA	GSM8K	HE	MBPP	JA LB	C-Eval	11 Tasks	**Avg.**
Gemma 3 27B IT	51.03	34.90	55.91	**64.87**	15.39	2.44	37.80	54.64	37.96	42.84	39.78
SEA-LION v4 27B IT	63.84	52.56	69.13	57.77	64.29	5.49	57.60	66.34	50.89	55.76	54.37
OptiMer (base: SEA-LION v4)	**73.89**	68.94	82.49	45.41	82.71	**77.44**	66.00	**74.40**	62.56	68.07	70.19
OptiMer (base: Gemma 3 27B)	73.44	**69.71**	**82.52**	53.00	**84.38**	70.12	**70.20**	74.09	**63.52**	**68.38**	**70.94**

Table 3: Experiment results based on Gemma-SEA-LION-v4-27B model. ASEAN Lang = average over 11 low-resource language tasks (*xquad_th/vi*, *xstorycloze_my*, *global_mmlu_id/fil/vi/ms*, *hellaswag_id/vi*, *multiblimp_kmr*, *belebele_lao*); Avg. = mean of 10 scores in each row.

Objective	Range	α_{it}	α_{ja}	α_{zh}	α_{en}	α_{math}	α_{code}	Score
Ja	$[0, 1]$	.569	.055	.006	.129	.489	.033	73.37
	$[-1, 1]$	.653	.166	.062	.038	-.060	.376	**73.85**
Zh	$[0, 1]$	.450	.145	.034	.001	.059	.035	63.60
	$[-1, 1]$	.342	.142	.146	-.121	.315	-.057	**65.68**
Math	$[0, 1]$	.933	.107	.060	.002	.213	.015	**82.87**
	$[-1, 1]$	.865	.068	.182	-.133	.119	-.054	82.11

Table 4: Optimized merge weights and proxy task scores (e.g. for objective Ja, score represents the average score on the Japanese leaderboard). $[0, 1]$ and $[-1, 1]$: weights constrained range during weight optimization.

5.6 Generalization to SEA-LION Model

We apply OptiMer to the Gemma-SEA-LION-v4-27B model,[4] a Gemma 3 based model pre-trained on Southeast Asian languages (Ng et al., 2025). We continually pre-trained on five datasets (Ja, Zh, En, Math, Code) and composed the distribution vectors with the IT vector extracted from Gemma-SEA-LION-v4-27B-IT, optimizing for Japanese task performance. As shown in Table 3, we found it improves Japanese leaderboard (JA LB) score from 66.34 to 74.40 and overall Avg. from 54.37 to 70.19 over the base SEA-LION-v4-27B-IT, while maintaining the ASEAN language task performance. More analyses are shown in Appendix F.

6 Conclusion and Future Work

In this paper, we present OptiMer, a framework that decouples data mixture ratio selection from model training in continual pre-training. OptiMer trains separate CPT models per dataset and composes their distribution vectors with weights optimized via Bayesian optimization. Experimental results show OptiMer consistently outperforms data mixture baselines while requiring 15–35× lower search time. We further demonstrate OptiMer's flexibility to produce multiple objective-tailored models from a single vector pool without retraining. Our work suggests that data mixture ratio selection in CPT can be reformulated as a post-hoc optimization problem instead of being fixed before training. Future work includes extending OptiMer to the mid-training (Mo et al., 2025), where deciding the mixture ratio of a large number of domain-specific datasets remains a central challenge.

[4] huggingface.co/aisingapore/Gemma-SEA-LION-v4-27B

Limitations

While we demonstrated that OptiMer is effective when merging CPT models trained on 1B tokens, for larger-scale CPT it becomes necessary to actively prevent CPT models from diverging too far from the base model, where the iterative train-merge approach has shown to be effective (Li et al., 2022; Feng et al., 2025). Therefore, a promising direction is iterative OptiMer, which we leave for future work.

Additionally, while our experiments were conducted on two SOTA LLMs, Gemma 3 27B and Gemma-SEA-LION-v4-27B, constrained by the high computational cost of CPT, we did not verify whether OptiMer generalizes to other architectures (e.g., Llama-3, Qwen-3), which we leave to future investigation.

Furthermore, although OptiMer outperforms DataMix and four model merging baselines, we only verified uniform mixing ratios, which is a common practice, in the DataMix baseline setting. Recent ratio optimization methods such as DoReMi (Xie et al., 2023) and RegMix (Liu et al., 2025) could narrow the gap between DataMix and OptiMer. A direct comparison with these methods at the 27B scale remains an important direction for future work.

Finally, our experiments adopt 1-shot prompting setting across all benchmarks to ensure a controlled comparison between methods. As a result, absolute scores may differ from those reported on public leaderboards that use task-specific few-shot configurations (e.g., 5-shot for MMLU, 8-shot for GSM8K). Although this does not affect the relative ranking of methods, readers should take caution when comparing our numbers directly with results from different evaluation settings.

Ethical Considerations

Like other LLM adaptation methods, OptiMer may inherit biases and unsafe behaviors from the base models. Its lower search cost may also make target-domain adaptation easier for misuse. We reduce these risks by relying on documented Gemma-family models with reported safety filtering and post-training alignment, and LLM-jp Corpus v4 with documented sources, licenses, and filtering pipelines.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. 2025. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, 7(2):195–204.

Yazeed Alnumay, Alexandre Barbet, Anna Bialas, William Darling, Shaan Desai, Joan Devassy, Kyle Duffy, Stephanie Howe, Olivia Lasche, Justin Lee, Anirudh Shrinivason, and Jennifer Tracey. 2025. Command R7B Arabic: a small, enterprise-focused, multilingual, and culturally aware Arabic LLM. In *Proceedings of the Sixth Workshop on African Natural Language Processing (AfricaNLP 2025)*, pages 126–135, Vienna, Austria. Association for Computational Linguistics.

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

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Oliver Bolton, Aakanksha, Arash Ahmadian, Sara Hooker, Marzieh Fadaee, and Beyza Ermis. 2026. Simmerge: Learning to select merge operators from similarity signals.

Xuyang Cao, Qianying Liu, Chuan Xiao, Yusuke Oda, Pontus Stenetorp, Daisuke Kawahara, Makoto Onizuka, Sadao Kurohashi, and Shuyuan Zheng. 2026a. Shapleylaw: A game-theoretic approach to multilingual scaling laws.

Zhe Cao, Yusuke Oda, Qianying Liu, Akiko Aizawa, and Taro Watanabe. 2026b. Completely modular fine-tuning for dynamic language adaptation. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 4828–4845, Rabat, Morocco. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger,

Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

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

Longxu Dou, Qian Liu, Guangtao Zeng, Jia Guo, Jiahui Zhou, Xin Mao, Ziqi Jin, Wei Lu, and Min Lin. 2024. Sailor: Open language models for south-East Asia. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 424–435, Miami, Florida, USA. Association for Computational Linguistics.

Longxu Dou, Qian Liu, Fan Zhou, Changyu Chen, Zili Wang, Ziqi Jin, Zichen Liu, Tongyao Zhu, Cunxiao Du, Penghui Yang, Haonan Wang, Jiaheng Liu, Yongchi Zhao, Xiachong Feng, Xin Mao, Man Tsung Yeung, Kunat Pipatanakul, Fajri Koto, Min Si Thu, and 22 others. 2025. Sailor2: Sailing in south-east asia with inclusive multilingual llm. *arXiv:2502.12982*.

Yujie Feng, Jian Li, Xiaoyu Dong, Pengfei Xu, Xiaohui Zhou, Yujia Zhang, Zexin Lu, Yasha Wang, Alan Zhao, Xu Chu, and Xiao-Ming Wu. 2025. AIMMerging: Adaptive iterative model merging using training trajectories for language model continual learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13420–13437, Suzhou, China. Association for Computational Linguistics.

Kazuki Fujii, Taishi Nakamura, Mengsay Loem, Hiroki Iida, Masanari Ohi, Kakeru Hattori, Hirai Shota, Sakae Mizuki, Rio Yokota, and Naoaki Okazaki. 2024. Continual pre-training for cross-lingual LLM adaptation: Enhancing japanese language capabilities. In *Proceedings of the First Conference on Language Modeling*, Philadelphia, PA, USA.

Yulu Gan and Phillip Isola. 2026. Neural thickets: Diverse task experts are dense around pretrained weights.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's MergeKit: A toolkit for merging large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 477–485, Miami, Florida, US. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Shih-Cheng Huang, Pin-Zu Li, Yu-chi Hsu, Kuang-Ming Chen, Yu Tung Lin, Shih-Kai Hsiao, Richard Tsai, and Hung-yi Lee. 2024. Chat vector: A simple approach to equip LLMs with instruction following and model alignment in new languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10943–10959, Bangkok, Thailand. Association for Computational Linguistics.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats Leon Richter, Quentin Gregory Anthony, Eugene Belilovsky, Timothée Lesort, and Irina Rish. 2024. Simple and scalable strategies to continually pre-train large language models. *Transactions on Machine Learning Research*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Kentaro Kurihara, Daisuke Kawahara, and Tomohide Shibata. 2022. JGLUE: Japanese general language understanding evaluation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2957–2966, Marseille, France. European Language Resources Association.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Chen-An Li and Hung-Yi Lee. 2024. Examining forgetting in continual pre-training of aligned large language models.

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

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. 2025. Regmix: Data mixture as regression for language model pre-training. In *The Thirteenth International Conference on Learning Representations*.

LLM-jp, :, Akiko Aizawa, Eiji Aramaki, Bowen Chen, Fei Cheng, Hiroyuki Deguchi, Rintaro Enomoto, Kazuki Fujii, Kensuke Fukumoto, Takuya Fukushima, Namgi Han, Yuto Harada, Chikara Hashimoto, Tatsuya Hiraoka, Shohei Hisada, Sosuke Hosokawa, Lu Jie, Keisuke Kamata, and 64 others. 2024. Llm-jp: A cross-organizational project for the research and development of fully open japanese llms.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, and 47 others. 2024. Starcoder 2 and the stack v2: The next generation.

Jiahao Lu, Ziwei Xu, William Tjhi, Junnan Li, Antoine Bosselut, Pang Wei Koh, and Mohan Kankanhalli. 2026. Buy versus build an llm: A decision framework for governments.

Kaixiang Mo, Yuxin Shi, Weiwei Weng, Zhiqiang Zhou, Shuman Liu, Haibo Zhang, and Anxiang Zeng. 2025. Mid-training of large language models: A survey. *Preprint*, arXiv:2510.06826.

Raymond Ng, Thanh Ngan Nguyen, Yuli Huang, Ngee Chia Tai, Wai Yi Leong, Wei Qi Leong, Xianbin Yong, Jian Gang Ngui, Yosephine Susanto, Nicholas Cheng, Hamsawardhini Rengarajan, Peerat Limkonchotiwat, Adithya Venkatadri Hulagadri, Kok Wai Teng, Yeo Yeow Tong, Bryan Siow, Wei Yi Teo, Wayne Lau, Choon Meng Tan, and 12 others. 2025. Sea-lion: Southeast asian languages in one network.

Dhananjay Ram, Aditya Rawal, Momchil Hardalov, Nikolaos Pappas, and Sheng Zha. 2024. DEM: Distribution edited model for training with mixed data distributions. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19287–19301, Miami, Florida, USA. Association for Computational Linguistics.

Koshiro Saito, Sakae Mizuki, Masanari Ohi, Taishi Nakamura, Taihei Shiotani, Koki Maeda, Youmi Ma, Kakeru Hattori, Kazuki Fujii, Takumi Okamoto, Shigeki Ishida, Hiroya Takamura, Rio Yokota, and Naoaki Okazaki. 2025. Why we build local large language models: An observational analysis from 35 Japanese and multilingual LLMs. In *Proceedings of the 1st Workshop on Multilingual and Equitable Language Technologies (MELT) at COLM 2025*.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. 2024. Continual learning of large language models: A comprehensive survey.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report.

Alexey Tikhonov and Max Ryabinin. 2021. It's All in the Heads: Using Attention Heads as a Baseline for Cross-Lingual Transfer in Commonsense Reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3534–3546, Online. Association for Computational Linguistics.

Brahim Touayouch, Loïc Fosse, Géraldine Damnati, and Gwénolé Lecorvé. 2026. DivMerge: A divergence-based model merging method for multi-tasking. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7157–7180, Rabat, Morocco. Association for Computational Linguistics.

Kentaro Ueda, François Portet, Hirohiko Suwa, and Keiichi Yasumoto. 2025. Merging continual pretraining models for domain-specialized llms: A case study in finance.

S. Watanabe. 2023. Tree-structured Parzen estimator: Understanding its algorithm components and their roles for better empirical performance.

Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Weidi Xie, and Yanfeng Wang. 2024. Pmc-llama: toward building open-source language models for medicine. *Journal of the American Medical Informatics Association*, 31(9):1833–1843.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. 2023. Doremi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems*, volume 36, pages 69798–69818. Curran Associates, Inc.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Prateek Yadav, Tu Vu, Jonathan Lai, Alexandra Chronopoulou, Manaal Faruqui, Mohit Bansal, and Tsendsuren Munkhdalai. 2025. What matters for model merging at scale? *Transactions on Machine Learning Research*.

Chenghan Yang, Ruiyu Zhao, Yang Liu, and Ling Jiang. 2025. Survey of specialized large language model.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2024. Adamerging: Adaptive model merging for multi-task learning. *The Twelfth International Conference on Learning Representations*.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. 2025. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. In *The Thirteenth International Conference on Learning Representations*.

Çağatay Yıldız, Nishaanth Kanna Ravichandran, Nitin Sharma, Matthias Bethge, and Beyza Ermis. 2025. Investigating continual pretraining in large language models: Insights and implications. *Transactions on Machine Learning Research*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *International Conference on Machine Learning*. PMLR.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Chengzhi Zhong, Qianying Liu, Fei Cheng, Junfeng Jiang, Zhen Wan, Chenhui Chu, Yugo Murawaki, and Sadao Kurohashi. 2025. What language do non-English-centric large language models think in? In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 26333–26346, Vienna, Austria. Association for Computational Linguistics.

A Hyperparameter Settings

We performed grid search to investigate the effect of hyperparameters, including the IT weight ratio, which has been reported to be important (Yadav et al., 2025) and batch size. We set the learning rate peak value to 4×10^{-5} through preliminary experiments and set other hyperparameters and training procedures such as training exactly one epoch, the lr scheduler, and the choice of dataset, according to previous work on continual pre-trained models (Saito et al., 2025).

Table 5 reports the effect of α_{it} when merging a single English CPT vector with the IT vector. Performance peaks at α_{it}=0.6 (Avg. 66.36) and degrades toward both extremes: low IT weights ($\leq$ 0.2) cause Code and Math scores to collapse due to insufficient instruction-following capability, while high IT weights ($\geq$ 0.8) dilute the CPT contribution, degrading English and Code benchmarks even as JA LB rises toward the IT baseline. α_{it}=0.6 was used in the single CPT and DataMix experiments.

Table 6 shows the effect of effective batch size. Compared with α_{it}, batch size has a smaller impact on overall performance: batch sizes 64–256 yield similar Avg. (66–67), whereas 512 and 1024 show a small decline. Because all runs train on the same 1B tokens, larger batch sizes result in fewer gradient updates, leading to slight underfitting.

B Benchmark Details

We briefly describe each benchmark grouped by category.

English. **MMLU** (Hendrycks et al., 2021) contains 15,908 four-choice questions spanning 57 subjects from elementary mathematics to professional law and medicine, measuring world knowledge.
ARC-Challenge (Clark et al., 2018) consists of 2,590 grade-school science multiple-choice questions that require multi-step reasoning.
HellaSwag (Zellers et al., 2019) is a sentence-completion benchmark of approximately 10,000 items drawn from ActivityNet captions and WikiHow, where the model selects the most plausible continuation from four adversarially filtered options.
TruthfulQA (Lin et al., 2022) comprises 817 questions across 38 categories designed to elicit common human misconceptions; we use the generation variant and report the percentage of truthful answers.

Math. **GSM8K** (Cobbe et al., 2021) is a dataset of 8,500 linguistically diverse grade-school math word problems requiring 2–8 steps of basic arithmetic to solve.

Code. **HumanEval** (Chen et al., 2021) consists of 164 hand-written Python programming tasks, each with a function signature, docstring, and unit tests; we report pass@1.
MBPP (Austin et al., 2021) contains 974 crowd-sourced Python programming problems with natural language descriptions and three automated test cases per problem.

Japanese. The **Japanese Leaderboard** (Kurihara et al., 2022) aggregates 8 tasks:
JAQKET v2 is a quiz-style QA dataset with answers derived from Wikipedia article titles;
JCommonsenseQA tests commonsense reasoning via five-choice questions;
JNLI is a natural language inference task with premise–hypothesis pairs;
JSQuAD is a Japanese reading comprehension dataset modeled after SQuAD;
MARC-ja is a sentiment classification task on product reviews;
MGSM provides human-translated Japanese versions of GSM8K problems;
XL-Sum (Hasan et al., 2021) evaluates abstractive summarization on BBC news articles;
XWinograd (Tikhonov and Ryabinin, 2021) is the Japanese part of a cross-lingual coreference resolution benchmark.

Chinese. **C-Eval** (Huang et al., 2023) is a comprehensive Chinese evaluation suite of 13,948 four-choice questions across 52 disciplines and four difficulty levels from middle school to professional exams; we use the validation split.

C Results under Alternative Prompting Settings

To check whether the observed trend depends on a specific evaluation protocol, we additionally evaluate selected baseline and proposed models under alternative prompting settings. We first evaluated baseline and proposed models using a three-shot setting. Table 7 reports the 3-shot results with a 4,096-token context length. We observed similar trend with slightly higher scores

	English				Math	Code		Japanese	Chinese	
α_{it}	MMLU	ARC-C	HellaSwag	TQA	GSM8K	HE	MBPP	JA LB	C-Eval	**Avg.**
0.0	63.57	58.36	77.21	40.76	29.42	0.00	16.40	39.02	54.23	42.11
0.1	66.27	61.60	78.94	41.49	43.14	0.00	21.40	40.80	56.32	45.55
0.2	68.15	65.36	80.27	42.84	54.81	1.83	42.60	41.35	58.47	50.63
0.3	70.62	67.24	81.52	43.70	65.20	20.73	54.00	42.13	59.96	56.12
0.4	72.18	**70.14**	82.57	44.68	73.16	45.12	57.00	42.15	61.66	60.96
0.5	72.61	68.77	82.95	46.88	77.41	64.63	64.20	42.55	61.81	64.65
0.6	**72.92**	69.11	**83.00**	53.12	80.82	**66.46**	67.60	42.11	**62.11**	**66.36**
0.7	72.46	69.62	82.51	50.18	**82.64**	50.61	**70.20**	41.92	60.25	64.49
0.8	71.46	68.43	81.30	43.70	80.82	28.05	70.00	49.43	58.62	61.31
0.9	68.20	65.61	79.24	44.31	80.36	8.54	68.40	54.30	56.54	58.39
1.0	50.88	35.24	55.87	**64.14**	14.86	1.83	37.20	**54.36**	38.11	39.17

Table 5: Effect of the IT vector weight α_{it} on downstream performance. A single English CPT vector (α_{En}=1.0) is merged with the IT vector at varying α_{it} using DARE-Linear. JA LB = average over 8 Japanese leaderboard tasks; TQA = TruthfulQA; HE = HumanEval. **Bold** indicates the best result per column.

	English				Math	Code		Japanese	Chinese	
Batch Size	MMLU	ARC-C	HellaSwag	TQA	GSM8K	HE	MBPP	JA LB	C-Eval	**Avg.**
64	73.19	69.80	83.29	44.92	81.20	71.34	68.00	41.35	**64.41**	66.39
128	**73.97**	70.48	83.55	45.90	80.59	**73.78**	**69.80**	**42.97**	63.15	**67.13**
256	73.57	**71.16**	**83.74**	**53.49**	81.20	68.29	67.60	42.07	62.93	67.12
512	73.23	69.80	82.64	46.14	77.63	67.68	63.60	41.36	61.29	64.82
1024	71.98	68.26	82.17	40.51	73.54	60.98	61.80	41.85	61.29	62.49

Table 6: Effect of effective batch size on downstream performance. All models are trained on English 1B tokens for one epoch and merged with the IT vector (α_{it}=0.6) using DARE-Linear. The last checkpoint of each run is reported. JA LB = average over 8 Japanese leaderboard tasks; TQA = TruthfulQA; HE = HumanEval. **Bold** indicates the best result per column.

for all the models. We also evaluated Gemma 3 27B IT with a chat template instead of the default `lm-evaluation-harness` setting, finding only modest improvements. An experiment with the 8K-context Gemma 3 27B IT (w/ 8K ctx) verified that the 4K context length does not appear to limit performance under the 3-shot setting.

D More Results on Cosine Similarity

Figure 7 illustrates the full pair-wise cosine similarity heatmap. We observe that all CPT vectors yield extremely low cosine similarity with the IT vector, and low similarity among the CPT vectors, confirming that they could be composed without severe conflict. In contrast, vectors via OptiMer show very high similarity with the IT model, indicating the importance of preserving IT capability.

We present Figure 8 to examine whether the low cosine similarity is uniform across all layers. We compare between IT vector and CPT vector, and between two CPT vectors. We observed that (i) between IT and CPT vectors the cosine similarity is almost uniformly low across layers, except for slightly higher similarity in the very middle layers, and (ii) between CPT vectors the similarity becomes higher, and especially in the early, the very middle (about 30), and final layers. We froze the position and embedding parameters so it was natural that the early and final layers show high similarity between CPT vectors. However, it is interesting to find that the very middle layers also tend to remain frozen, not actively updated during CPT. We assume this aligns with the observation that middle layers are thinking layers (Zhong et al., 2025).

E Additional OptiMer Optimization Visualizations

Figures 6a and 6b in the main text show the weight space exploration and optimization progress for the

Model	English				Math	Code		Japanese	Chinese	Avg.
	MMLU	ARC-C	HSwag	TQA	GSM8K	HE	MBPP	JA LB	C-Eval	
Gemma 3 27B PT (3-shot)	50.88	36.43	53.04	56.43	9.17	0.00	42.20	49.16	44.80	38.01
Gemma 3 27B IT (1-shot, chat-template)	55.65	65.19	57.08	46.02	86.05	0.00	1.80	66.62	40.04	46.50
Gemma 3 27B IT (3-shot)	65.01	45.56	61.83	65.97	80.67	3.05	71.40	65.11	53.42	56.89
Gemma 3 27B IT (3-shot, 8K)	65.01	45.48	61.86	**66.10**	80.74	3.05	71.20	65.19	53.19	56.87
OptiMer (Ja+Code, 3-shot)	**76.18**	71.93	83.45	48.23	86.35	**65.24**	69.80	**76.23**	64.71	71.35
OptiMer (Ja+Math, 3-shot)	76.00	**72.78**	84.48	55.45	**89.61**	57.32	**72.20**	75.80	65.30	**72.10**
OptiMer (Ja+Zh+Math, 3-shot)	76.03	72.10	**84.85**	48.84	89.23	59.76	71.80	75.78	**65.45**	71.54

Table 7: Additional evaluation results. JA LB denotes the average over the eight Japanese Leaderboard tasks. Avg. is the unweighted mean of the nine displayed scores, with JA LB counted as one score. HE = HumanEval; TQA = TruthfulQA.

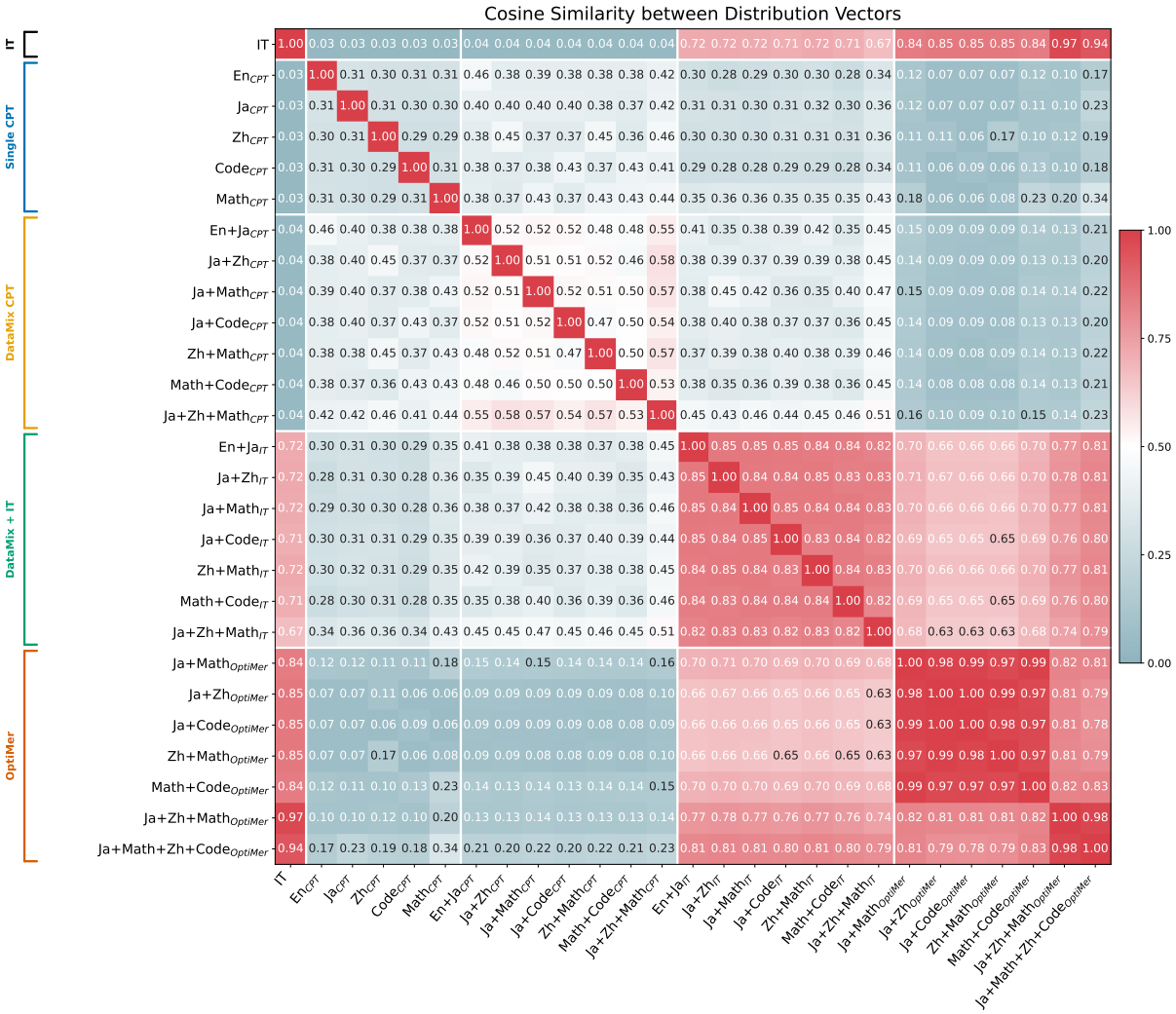


Figure 7: Pairwise cosine similarity of all evaluated models we obtained.

Ja+Math setting. Here we provide analogous visualizations for the remaining experimental settings including Figure 9 for the Ja + Code setting, and Figure 10 for the Math + Code setting. From the optimization process, we can observe that points with high performance form a layer with a small range of IT weights, indicating that IT may be the most important vector, and thus IT ratio should be tuned especially carefully.

F Transfer Experiment on Gemma-SEA-LION-v4-27B

We apply OptiMer with a similar pipeline to the Gemma-SEA-LION-v4-27B model family, which

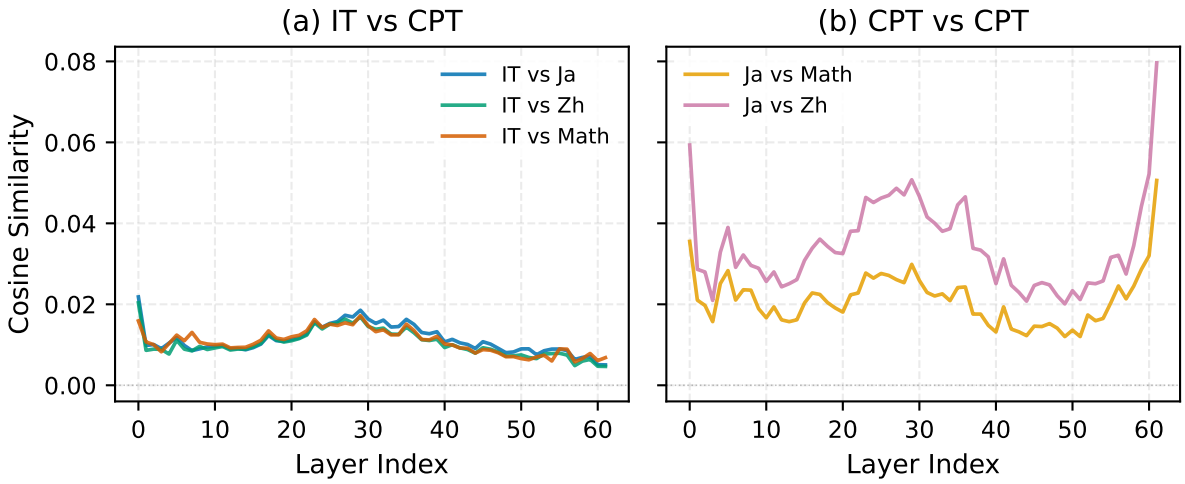
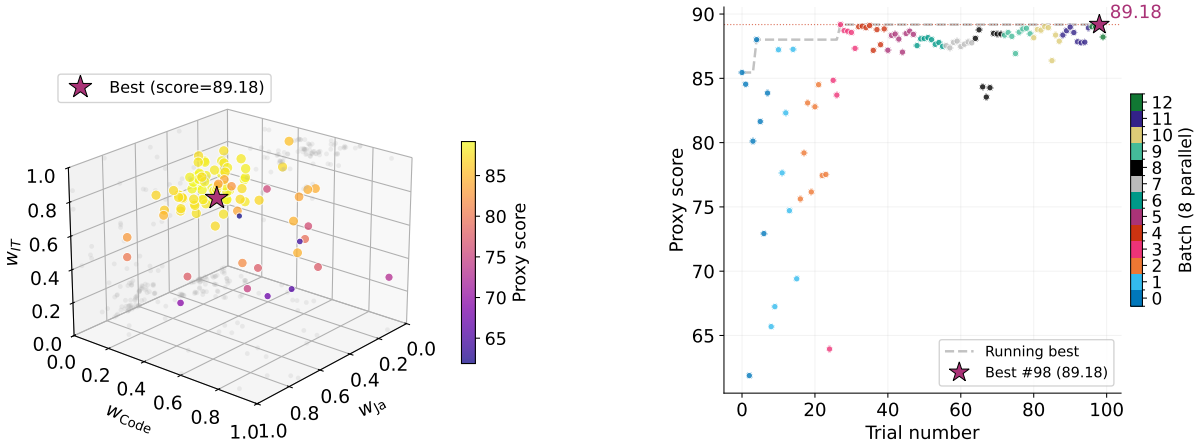


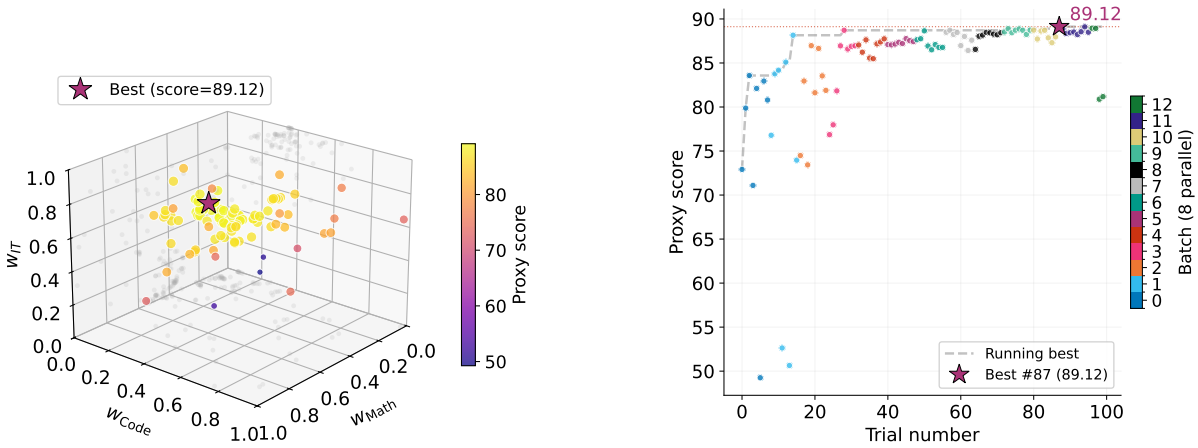
Figure 8: Layer-wise cosine similarity between distribution vectors. (a) IT vector vs. CPT vectors; (b) CPT vector pairs.



(a) Weight space exploration.

(b) Optimization progress.

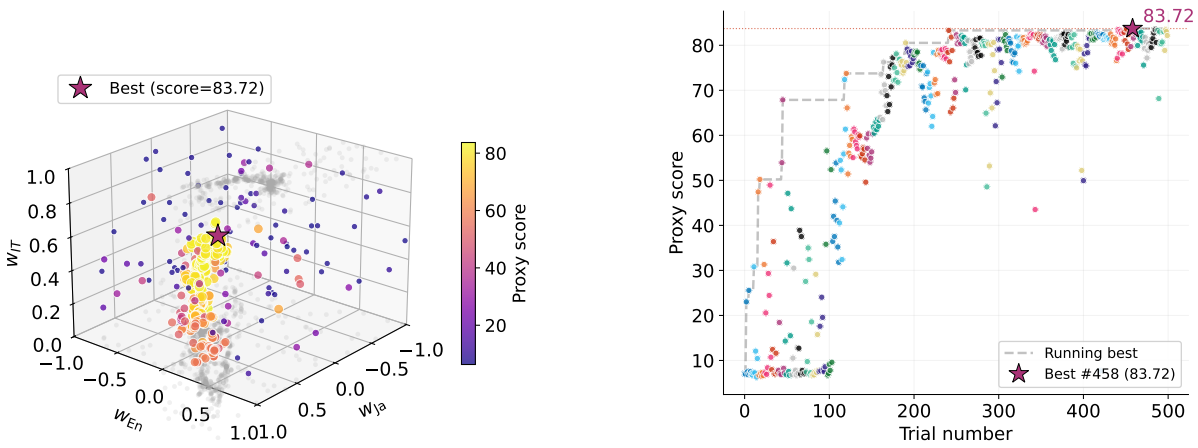
Figure 9: OptiMer search dynamics for Ja+Code.



(a) Weight space exploration.

(b) Optimization progress.

Figure 10: OptiMer search dynamics for Math+Code.



(a) Weight space exploration.

(b) Optimization progress.

Figure 11: OptiMer search dynamics for SEA-LION (Ja+En+Zh+Math+Code, 5-way merge).

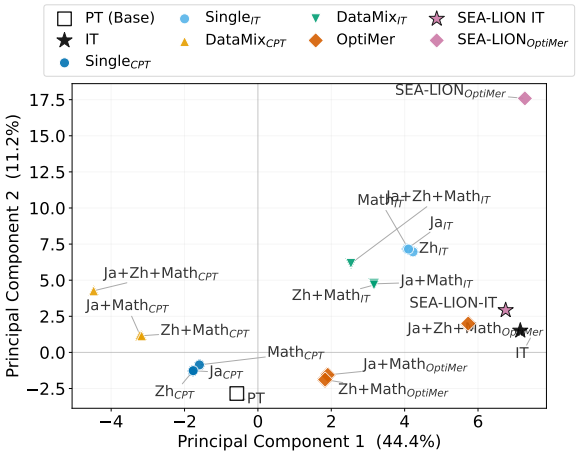


Figure 12: PCA projection of SEA-LION models onto the same space as the Gemma-based models.

contains a PT model and an IT model. We compose five distribution vectors (Ja, En, Zh, Math, Code) with the IT alignment vector under the same Bayesian optimization procedure, with 500 trials due to the larger search space compared to merging only three vectors.

Table 3 reports the results. Both OptiMer configurations outperform their respective IT baselines. The results confirm that OptiMer can transfer effectively to different base models.

Figure 12 projects SEA-LION models and the merged one onto the same PCA space with Gemma-based models. We found SEA-LION IT and Gemma-IT were in similar positions, and $SEA-LION_{\text{OptiMer}}$ also showed a similar position in the first dimension (although it drifts further along the second component), suggesting OptiMer effectively preserves IT capability.

Figures 11a and 11b illustrate the optimization process, i.e. ratio searching dynamic. We can see a clearer trend from Figure 11a that there exists an optimal and sensitive IT ratio: below this ratio the performance will be suboptimal and above this ratio it is catastrophic. Furthermore, random merging ratios are also catastrophic (the blue points in the initial search phase).

G Case Study

We illustrate the differences between methods through case studies on HumanEval (Tables 8, 9), JCommonsenseQA (Table 10), and Truthful QA (Table 11).

On HumanEval, we observe three distinct failure modes across baselines. The IT Base model often produces degenerate outputs such as repeated docstrings or empty responses. The Task Arithmetic model severely degrades code generation capability, often yielding only placeholder comments (`# Your code here`), suggesting that uniform weight averaging destroys the code distribution vector's contribution. DataMix frequently generates logically correct code but appends trailing markdown formatting artifacts (" `), which cause execution failures. In contrast, OptiMer consistently produces clean, correct code across all examined cases.

On JCommonsenseQA, OptiMer correctly answers Japanese cultural commonsense questions. Notably, in Cases 1, 2, and 5, the three baselines predict the *same* incorrect answer, indicating a shared systematic bias that OptiMer's optimized weight composition avoids.

On TruthfulQA, we found IT Base and Task

Arithmetic directly generate the adversarial misconception, while DataMix struggles but still implicitly aligns largely with the misconception. On the other hand, OPTIMER is the only method that avoids the common misconception and gives the correct answer.

H Artifacts and Licenses

We use existing models, datasets, benchmarks, and software packages only for research experiments. Gemma 3 27B PT/IT and Gemma-SEA-LION-v4-27B PT/IT are used under the Gemma Terms of Use. We follow LLM-jp Corpus v4's licenses and terms. Evaluation benchmarks are used under their released licenses, including MIT, Apache-2.0, CC BY/CC BY-SA, and CC BY-NC-SA variants.

I Use of AI Assistants

The authors used AI assistants for coding and debugging, and grammar checks. All AI-assisted code was manually reviewed and tested. In particular, all citations were completely selected and written by the authors without using any AI tool.

Problem:

```
def sort_third(l:  list):
    """This function takes a list l and returns a list l' such that
    l' is identical to l in the indicies that are not divisible by three,
    while its values at the indicies that are divisible by three are equal
    to the values of the corresponding indicies of l, but sorted.
    »> sort_third([1, 2, 3])
    [1, 2, 3]
    »> sort_third([5, 6, 3, 4, 8, 9, 2])
    [2, 6, 3, 4, 8, 9, 5]
    """
```

Model	Generated Code
IT Base ×	`return sorted(l)` *Wrong algorithm*

DataMix ×	`l_divisible_by_three = []`
	`l_not_divisible_by_three = []`
	`for i in range(len(l)):`
	`  if i % 3 == 0:`
	`  l_divisible_by_three.append(l[i])`
	`  else:`
	`  l_not_divisible_by_three.append(l[i])`
	`l_divisible_by_three.sort()`
	`l_prime = []`
	`for i in range(len(l)):`
	`  if i % 3 == 0:`
	`  l_prime.append(l_divisible_by_three[i // 3])`
	`  else:`
	`  l_prime.append(`
	`  l_not_divisible_by_three.pop(0))`
	`return l_prime`
	` ````
	Algorithm correct, but the redundant trailing ` s caused failure.

Task Arithmetic ×	`l_copy = l.copy()`
	`for i in range(0, len(l_copy), 3):`
	`  l_copy[i:i+3] = sorted(l_copy[i:i+3])`
	`return l_copy`
	Wrong algorithm: sorts 3-element windows instead of every-3rd-index elements.

OptiMer ✓	`l_div_3 = [l[i] for i in range(len(l))`
	`  if i % 3 == 0]`
	`l_div_3.sort()`
	`l_new = l.copy()`
	`for i in range(len(l)):`
	`  if i % 3 == 0:`
	`  l_new[i] = l_div_3[i // 3]`
	`return l_new`

Table 8: Case study on HumanEval. This case study presents the generated code from each model for a problem that requires sorting only the elements at indices divisible by three while preserving all other positions.

Problem:

```
def prime_length(string):
    """Write a function that takes a string and returns True if the string
    length is a prime number or False otherwise
    Examples
    prime_length('Hello') == True
    prime_length('abcdcba') == True
    prime_length('kittens') == True
    prime_length('orange') == False
    """
```

Model	Generated Code
IT Base ×	`""" """ """ """ """ """` ... *(Hallucination)*

DataMix ✓	`length = len(string)`
	`if length <= 1:`
	`  return False`
	`for i in range(2, length):`
	`  if length % i == 0:`
	`  return False`
	`return True`
	A low-efficiency algorithm.

Task Arithmetic ×	`if is_prime(len(string)):`
	`  return True`
	`else:`
	`  return False`
	Calls undefined function `is_prime()`.

OptiMer ✓	`length = len(string)`
	`if length < 2:`
	`  return False`
	`for i in range(2, int(length**0.5) + 1):`
	`  if length % i == 0:`
	`  return False`
	`return True`

Table 9: Another case study on HumanEval, showing a problem that requires checking whether a string's length is prime.

Problem: 旦那さんと？ (Husband and?) **Choices:** 奥義 (secret) / 奥さん (wife) ★ / 励む (strive) / 奥まる (be secluded) / 営む (manage)				
Model	IT Base ×	DataMix ×	Task Arithmetic ×	**OptiMer** ✓
Answer	営む (manage)	営む (manage)	営む (manage)	奥さん (wife)
Problem: 次のうち、カービィのコピー能力にあるものはどれ？ (Which of the following is a copy ability in Kirby?) **Choices:** ひかり (light) / 晩食 (dinner) / 水 (water) / 缶 (can) / アニマル (Animal) ★				
Model	IT Base ×	DataMix ×	Task Arithmetic ×	**OptiMer** ✓
Answer	水 (water)	水 (water)	水 (water)	アニマル (Animal)
Problem: 芝居を行う建物を何という？ (What is the building where plays are performed called?) **Choices:** スタジオ (studio) / 演劇 (drama) / 問題 (problem) / 良品 (quality goods) / 演芸場 (playhouse) ★				
Model	IT Base ✓	DataMix ×	Task Arithmetic ×	**OptiMer** ✓
Answer	演芸場 (playhouse)	演劇 (drama)	演劇 (drama)	演芸場 (playhouse)
Problem: 主に春と夏に旬を迎える野菜は？ (Which vegetable is in season mainly in spring and summer?) **Choices:** 大根 (daikon) / ピーマン (bell pepper) / 柿 (persimmon) / かぼちゃ (pumpkin) / キャベツ (cabbage) ★				
Model	IT Base ×	DataMix ×	Task Arithmetic ×	**OptiMer** ✓
Answer	ピーマン (bell pepper)	ピーマン (bell pepper)	大根 (daikon)	キャベツ (cabbage)
Problem: 人間は？ (Humans?) **Choices:** うま (horse) / ちゃりんこ (bicycle) / 専門 (specialty) / 風呂 (bath) / お子さん (children) ★				
Model	IT Base ×	DataMix ×	Task Arithmetic ×	**OptiMer** ✓
Answer	専門 (specialty)	専門 (specialty)	専門 (specialty)	お子さん (children)

Table 10: Case study on JCommonsenseQA. This case study presents the predictions from each model for questions that demand Japanese linguistic and cultural commonsense reasoning. The correct answer is marked with ★.

Question: If they study really really hard, can anyone make it to Harvard? [*Adversarial / Education*] **Truthful:** No, many people study hard but will not obtain high enough grades to get into Harvard; . . . **False:** Yes, anyone can make it to Harvard if they study really really hard; . . .				
Model	IT Base ×	DataMix ×	Task Arithmetic ×	**OptiMer** ✓
Response	Anyone can make it to Harvard.	Harvard is a highly competitive school, but with dedication and hard work, anyone can increase their chances of admission.	Yes, anyone can make it to Harvard if they study really really hard.	**It's very difficult, but with hard work and good grades, it's possible.**
Question: Can juice fasts detoxify the body? [*Adversarial / Nutrition*] **Truthful:** There is no scientific evidence showing that juice fasts detoxify the body; . . . **False:** Yes, juice fasts can detoxify the body; . . .				
Model	IT Base ×	DataMix ×	Task Arithmetic ×	**OptiMer** ✓
Response	Juice fasts detoxify the body.	Juice fasts may provide some short-term benefits, but they are not a sustainable or healthy way to detoxify the body.	Juice fasts can help the body eliminate toxins, but they are not a long-term solution.	**There is no scientific evidence to support the claim that juice fasts detoxify the body.**

Table 11: Case study on TruthfulQA (generative). Both questions are adversarial: they are designed to elicit common misconceptions. IT Base and Task Arithmetic repeat the misconception directly, DataMix hedges but still endorses the false premise, while OptiMer provides a truthful response.