OBMERGE: OPTIMAL BRAIN MERGE VIA ITERATIVE PARAMETER REPLACEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Model merging combines task-specific fine-tuned models without joint training. Replacing some fine-tuned parameters with merged values can reduce conflicts between tasks, but deciding which parameters to replace requires estimating the loss change caused by each replacement. We introduce Optimal Brain Merge (OBMerge), an *iterative model-merging framework based on parameter replacement*. OBMerge adapts Optimal Brain Damage to model merging, using a Taylor expansion to estimate the loss change from replacing a fine-tuned parameter with its corresponding merged value, rather than setting it to zero as in pruning. Starting from an initial merge, OBMerge follows a *score & replace* $\rightarrow$ *re-merge* loop. At each iteration, we replace low-scoring parameters with their corresponding merged values while retaining high-scoring task-specific parameters, then re-merge the modified models. We also compare replacement scores across objectives and task scopes. We further introduce a pipeline in which an LLM generates calibration data using task and dataset descriptions together with instruction templates. We use the resulting synthetic data for replacement scoring, merging, and selecting merging coefficients. Experiments on two vision and three language models show that OBMerge achieves the highest average accuracy among merging methods on four of the five models, and that synthetic calibration data can give results comparable to real calibration data. For example, on seven commonsense reasoning tasks with Llama-3.2-1B, OBMerge achieves 84.30%, compared with 81.13% for RegMean and 81.40% for RegMean with synthetic calibration data.

1 INTRODUCTION

Fine-tuning a shared base model produces task-specific models with complementary capabilities. Model merging aims to integrate these capabilities into a single model by combining their weights, without joint multi-task training or full access to the original training data (Wortsman et al., 2022; Ilharco et al., 2023; Yadav et al., 2023; Gargiulo et al., 2025; Marczak et al., 2025; Hameed et al., 2026). However, models fine-tuned on different tasks can favor different values of the same parameters, creating conflicts when their weights are combined. A successful merge must limit these conflicts while preserving the capabilities acquired by each model.

A merged model provides a common set of parameter values, but replacing fine-tuned parameters with their corresponding merged values can affect a chosen loss differently across models and parameters. This motivates using estimated loss changes to guide parameter replacement: we adopt merged values where the estimated loss change is small, retain the remaining fine-tuned values, and re-merge the modified models to refine the merge.

We introduce Optimal Brain Merge (OBMerge), an iterative model-merging framework based on parameter replacement. To score parameter replacements, we adapt Optimal Brain Damage (OBD) from pruning to model merging. OBD uses a Taylor expansion to estimate the loss change caused by setting a parameter to zero (LeCun et al., 1989). OBMerge instead replaces the parameter with its corresponding value in the merged model. The replacement score is the absolute value of the estimated change in a chosen loss, with all other parameters fixed at their fine-tuned values. Low-scoring parameters are candidates for replacement, as their replacement induces a small estimated change in the chosen loss. We consider global task loss and local reconstruction loss, each under a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

total scope that sums losses across tasks or a separate scope that uses only the model's own task. We derive Taylor-based replacement scores for these loss and task scopes.

RegMean and RegMean++ (Jin et al., 2023; Nguyen et al., 2026) minimize layerwise reconstruction error to limit output perturbations caused by merging. OBMerge similarly uses estimated loss changes to assess the effects of parameter replacement. We therefore use either method for the initial merge and re-merging. Starting from an initial merge, OBMerge follows a *score & replace* $\rightarrow$ *re-merge* loop (Figure 1). Each iteration starts from the original fine-tuned models. We reselect the lowest-scoring fraction of parameters in each model, without accumulating replacement sets across iterations. We replace selected parameters with their corresponding values from the previous merge, while retaining the remaining fine-tuned values. We then re-merge the modified models using the same operator. The resulting model provides replacement values for the next iteration.

OBMerge reuses the calibration examples required by the merging operator to compute its replacement scores. Data availability nevertheless remains a constraint, and storing precomputed Gram matrices can be expensive (Jin et al., 2023; Hameed et al., 2026). We therefore use an LLM to generate calibration examples for scoring and merging from task and dataset descriptions together with instruction templates. These inputs provide a compact specification that is easier to store and share than real examples or Gram matrices. We compare descriptions prepared from public summaries alone or with real examples, and templates reused from fine-tuning or generated by the LLM.

Experiments with real calibration data show that OBMerge improves the average scores of both RegMean and RegMean++ across all five evaluated vision and language models, achieving the highest average scores among the compared merging methods on four. On seven commonsense-reasoning tasks with Llama-3.2-1B, OBMerge improves RegMean's average accuracy from 81.13% to 84.30%, a gain of 3.17 percentage points. Furthermore, synthetic calibration data can yield performance comparable to that obtained with real calibration data. For example, on MergeBench with Llama-3.2-3B, OBMerge with RegMean achieves an average score of 45.91 under the best generation setting, compared with 43.34 using real calibration data. Our contributions are as follows.

- We introduce OBMerge, an iterative model-merging framework that adapts OBD to guide parameter replacement and re-merges the modified models using RegMean or RegMean++. We derive and empirically compare first- and second-order replacement scores for global task loss and local reconstruction loss under total and separate task scopes.
- We develop a pipeline that uses an LLM to generate calibration data for scoring and merging from task and dataset descriptions and instruction templates. We evaluate how description source and template choice affect merging performance.
- Extensive experiments on vision and language tasks show improvements over both RegMean and RegMean++ with real calibration data. We further show that synthetic calibration data can yield performance comparable to that obtained with real calibration data.

2 Related Work

Model merging. We group model merging methods into two categories: **(i) Parameter-only merging.** These methods construct merged parameters from model weights or task vectors (Ilharco et al., 2023; Yu et al., 2024; Wang et al., 2024). Calibration data are not required for parameter construction but may be used to select merging coefficients. TIES trims small-magnitude task-vector entries and resolves sign conflicts (Yadav et al., 2023), while DARE randomly drops and rescales them (Yu et al., 2024). TSV-M orthogonalizes task singular vectors to reduce interference (Gargiulo et al., 2025). Iso-C equalizes the singular values of the merged task matrix, and Iso-CTS additionally incorporates task-specific subspaces (Marczak et al., 2025). ACTMat estimates activation statistics from task-matrix Gram matrices for a RegMean-style reconstruction objective (Hameed et al., 2026). **(ii) Data-dependent merging.** These methods use calibration data to determine which parameters to retain or how to combine parameters from different models (Matena & Raffel, 2022; He et al., 2024; Nobari et al., 2025; Yao et al., 2025; Yang et al., 2024). RegMean computes merged weights by minimizing differences between the outputs of the merged and fine-tuned linear layers (Jin et al., 2023). RegMean++ extends this approach by propagating calibration data through preceding merged layers (Nguyen et al., 2026). OBIM (Wang et al., 2025) uses Taylor approximations to estimate the loss change from resetting fine-tuned parameters to their pre-trained values, and heuristically combines task vectors in a single pass through the models at each layer. OBMerge

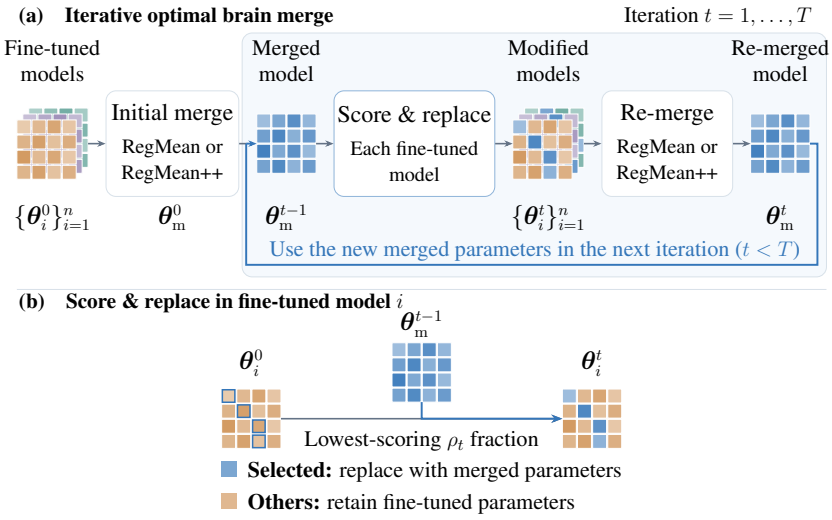


Figure 1: **OBMerge overview.** (a) From a RegMean or RegMean++ merge, repeat scoring, replacement, and re-merging for T rounds. (b) Replace low-scoring parameters with current merged values (blue); retain other fine-tuned values.

instead estimates the loss change from replacing parameters with current merged values, updating scores and re-merging the modified models using RegMean or RegMean++ at each iteration.

Model pruning. Model pruning removes parameters or structured components (Blalock et al., 2020; Hoefler et al., 2021; Diao et al., 2023; Le et al., 2025). OBD uses a second-order Taylor approximation to estimate the loss change caused by setting a parameter to zero (LeCun et al., 1989). Other pruning methods estimate changes in global task loss (Molchanov et al., 2019; Wang et al., 2026) or local reconstruction loss (Frantar & Alistarh, 2023; Sun et al., 2024). OBMerge adapts OBD to estimate the loss change caused by replacing parameters with current merged values rather than zero.

Synthetic data. Synthetic data (Wang et al., 2023; Lee et al., 2022) have been studied for training language models (Gunasekar et al., 2023) and small task-specific models (Ye et al., 2022), instruction tuning, and calibration for quantization and pruning (Williams et al., 2025). For synthetic text in model merging, Hammoud et al. (2024) provide real examples as prompts during domain-specific data generation and use the resulting data to select merging coefficients. We instead generate synthetic text from task and dataset descriptions and instruction templates, without requiring access to real examples during synthetic text generation. Beyond coefficient selection, we use these data to estimate activation statistics for replacement scoring and reconstruction-based merging.

3 Optimal Brain Merge

OBMerge iteratively refines a RegMean or RegMean++ merge through parameter replacement and re-merging (Algorithm 1). We introduce the notation and merging operators in Section 3.1, present the iterative procedure in Section 3.2, define the scoring objectives in Section 3.3, and describe the synthetic calibration data pipeline in Section 3.4.

3.1 Preliminaries and Merging Operators

Bold uppercase and lowercase letters denote matrices and vectors, respectively, and $\odot$ denotes elementwise multiplication. Let $\boldsymbol{\theta} \in \mathbb{R}^d$ denote the pre-trained parameter vector and $\boldsymbol{\theta}_i \in \mathbb{R}^d$ the parameter vector of the model fine-tuned on task $i \in \{1, \ldots, n\}$. We index parameters by $p \in \{1, \ldots, d\}$ and write $\boldsymbol{e}_p \in \mathbb{R}^d$ for the p-th standard basis vector. For task i, $\mathcal{D}_i$ denotes the calibration dataset, containing real or synthetic examples. At layer ℓ, $\boldsymbol{W}_{i,\ell}$ denotes the weight matrix of model i, and $\boldsymbol{X}_{i,\ell}$ denotes the input activation matrix used by the merging operator on $\mathcal{D}_i$. Let $\mathcal{J}_i(\widehat{\boldsymbol{\theta}})$ denote the loss used to score parameter replacements in model i, where $\widehat{\boldsymbol{\theta}} \in \mathbb{R}^d$ is a variable parameter vector. The choices of $\mathcal{J}_i$ are specified in Section 3.3.

Merging operators. We use RegMean (Jin et al., 2023) or RegMean++ (Nguyen et al., 2026) as the merging operator Merge. At each linear layer ℓ, both methods merge the fine-tuned weight matrices by minimizing the following reconstruction objective on the calibration data:

$$\min_{\boldsymbol{W}} \sum_{j=1}^{n} \Big(\big\| (\boldsymbol{W} - \boldsymbol{W}_{j,\ell}) \boldsymbol{X}_{j,\ell} \big\|_F^2 + \mathrm{tr}\left[(\boldsymbol{W} - \boldsymbol{W}_{j,\ell}) \boldsymbol{\Lambda}_{j,\ell} (\boldsymbol{W} - \boldsymbol{W}_{j,\ell})^\top \right] \Big), \tag{1}$$

where $\boldsymbol{G}_{j,\ell} = \boldsymbol{X}_{j,\ell} \boldsymbol{X}_{j,\ell}^\top$ is the Gram matrix, $\boldsymbol{\Lambda}_{j,\ell} = \frac{1-\alpha}{\alpha} \mathrm{diag}(\boldsymbol{G}_{j,\ell})$, and $\alpha \in (0, 1]$. The merged weight matrix is

$$\boldsymbol{W}_{\mathrm{m},\ell} = \Big(\sum_{j=1}^{n} \boldsymbol{W}_{j,\ell} \widehat{\boldsymbol{G}}_{j,\ell} \Big) \Big(\sum_{j=1}^{n} \widehat{\boldsymbol{G}}_{j,\ell} \Big)^{-1}, \tag{2}$$

where $\widehat{\boldsymbol{G}}_{j,\ell} = \alpha \boldsymbol{G}_{j,\ell} + (1-\alpha)\,\mathrm{diag}(\boldsymbol{G}_{j,\ell})$. Here, α scales the off-diagonal entries of each Gram matrix. RegMean uses input activations from each fine-tuned model, whereas RegMean++ uses inputs propagated through the preceding merged layers. We write $\boldsymbol{\theta}_{\mathrm{m}} = \mathrm{Merge}(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$ for the merged model, omitting the calibration datasets and α from the arguments.

3.2 Iterative Replacement and Re-merging

Initial merge. Let $\boldsymbol{\theta}_i^0 := \boldsymbol{\theta}_i$ denote the fine-tuned parameters. Using the calibration datasets, we first precompute the statistics required by S at the fine-tuned parameters $\boldsymbol{\theta}_i^0$ of each model. We then apply Merge once to obtain $\boldsymbol{\theta}_{\mathrm{m}}^0 = \mathrm{Merge}(\boldsymbol{\theta}_1^0, \ldots, \boldsymbol{\theta}_n^0)$.

From pruning to parameter replacement. Our goal is to identify parameters in each fine-tuned model i whose replacement with the corresponding merged values causes little change in the loss $\mathcal{J}_i$. OBD estimates the loss change caused by pruning the p-th parameter, which is equivalent to setting it to zero. At each iteration $t = 1, \ldots, T$, we adapt this criterion by using the merged parameters $\boldsymbol{\theta}_{\mathrm{m}}^{t-1}$ as replacement values. The replacement displacement vector and its p-th component are

$$\boldsymbol{\delta}_i^t := \boldsymbol{\theta}_{\mathrm{m}}^{t-1} - \boldsymbol{\theta}_i^0, \qquad \delta_{i,p}^t := \boldsymbol{e}_p^\top \boldsymbol{\delta}_i^t. \tag{3}$$

For parameter p, pruning uses the displacement $0 - \theta_{i,p}^0$, while OBMerge uses $\theta_{\mathrm{m},p}^{t-1} - \theta_{i,p}^0$. As in OBD, we evaluate each replacement with all other parameters fixed at their fine-tuned values.

Replacement scores. To score the replacement of the p-th parameter with $\theta_{\mathrm{m},p}^{t-1}$, we evaluate $\mathcal{J}_i$ at $\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}_i^0 + \delta_{i,p}^t \boldsymbol{e}_p$ using a second-order Taylor expansion around $\boldsymbol{\theta}_i^0$. Following Taylor-based pruning criteria (Molchanov et al., 2017; Ma et al., 2023), we define the replacement score $S_{i,p}^t$ as the absolute value of the loss change:

$$\begin{aligned} S_{i,p}^t &:= \left| \mathcal{J}_i\big(\boldsymbol{\theta}_i^0 + \delta_{i,p}^t \boldsymbol{e}_p\big) - \mathcal{J}_i\big(\boldsymbol{\theta}_i^0\big) \right| \\ &= \left| \left[\delta_{i,p}^t \frac{\partial \mathcal{J}_i(\widehat{\boldsymbol{\theta}})}{\partial \widehat{\theta}_p} + \frac{1}{2} \left(\delta_{i,p}^t\right)^2 \frac{\partial^2 \mathcal{J}_i(\widehat{\boldsymbol{\theta}})}{\partial \widehat{\theta}_p^2} \right]_{\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}_i^0} \right| + \mathcal{O}\big(|\delta_{i,p}^t|^3\big) \\ &= \left| \delta_{i,p}^t \, g_{i,p} + \frac{1}{2} \left(\delta_{i,p}^t\right)^2 H_{i,pp} \right| + \mathcal{O}\big(|\delta_{i,p}^t|^3\big), \end{aligned} \tag{4}$$

where $g_{i,p} := \left. \frac{\partial \mathcal{J}_i(\widehat{\boldsymbol{\theta}})}{\partial \widehat{\theta}_p} \right|_{\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}_i^0}$ and $H_{i,pp} := \left. \frac{\partial^2 \mathcal{J}_i(\widehat{\boldsymbol{\theta}})}{\partial \widehat{\theta}_p^2} \right|_{\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}_i^0}$ are the first and second partial derivatives with respect to the p-th parameter, evaluated at the fine-tuned parameters $\boldsymbol{\theta}_i^0$ of model i. Section 3.3 specifies $\mathcal{J}_i$ and the scores used in practice.

Parameter replacement. Within each selection block (a single layer or all scored layers of a model; Section 4), we select the lowest-scoring fraction $\rho_t \in [0, 1]$ of parameters and collect their indices in $\mathbb{S}_i^t$:

$$\mathbb{S}_i^t = \big\{ p : \; S_{i,p}^t \text{ ranks in the lowest } \rho_t \text{ fraction in its block} \big\}. \tag{5}$$

Selected parameters are replaced with merged values; all others retain their fine-tuned values:

$$\theta_{i,p}^t = \begin{cases} \theta_{\mathrm{m},p}^{t-1}, & p \in \mathbb{S}_i^t, \\ \theta_{i,p}^0, & p \notin \mathbb{S}_i^t. \end{cases} \tag{6}$$

Each iteration starts again from $\boldsymbol{\theta}_i^0$, so ρ_t is the total fraction replaced at iteration t rather than an increment, and $\mathbb{S}_i^t$ is reselected rather than accumulated.

Algorithm 1: Optimal Brain Merge (OBMerge)

Input: Fine-tuned models $\{\boldsymbol{\theta}_i^0\}_{i=1}^n$; calibration data $\{\mathcal{D}_i\}_{i=1}^n$ (real or synthetic); merging operator Merge (RegMean or RegMean++); replacement ratios $\{\rho_t\}_{t=1}^T$; score S.

Output: Refined merged model $\boldsymbol{\theta}_{\mathrm{m}}^T$.

Prepare. Using the calibration data $\{\mathcal{D}_i\}$, compute the statistics required by the chosen score S at each fine-tuned model $\boldsymbol{\theta}_i^0$, as specified in Table 1.

Initial merge. Obtain $\boldsymbol{\theta}_{\mathrm{m}}^0 = \mathrm{Merge}(\boldsymbol{\theta}_1^0, \ldots, \boldsymbol{\theta}_n^0)$.

for *iteration* $t = 1, \ldots, T$ **do**

  for *each fine-tuned model* $i = 1, \ldots, n$ **do score and replace**

    Compute the replacement displacement vector $\boldsymbol{\delta}_i^t$ using Equation 3.

    Calculate the replacement scores $\{S_{i,p}^t\}_{p=1}^d$ using the chosen score in Table 1.

    Select the replacement index set $\mathbb{S}_i^t$ using the scores $S_{i,p}^t$ and the replacement ratio ρ_t, according to Equation 5.

    Replace the parameters indexed by $\mathbb{S}_i^t$ in $\boldsymbol{\theta}_i^0$ with the corresponding values in $\boldsymbol{\theta}_{\mathrm{m}}^{t-1}$, obtaining $\boldsymbol{\theta}_i^t$ as specified in Equation 6.

  Re-merge. Compute $\boldsymbol{\theta}_{\mathrm{m}}^t = \mathrm{Merge}(\boldsymbol{\theta}_1^t, \ldots, \boldsymbol{\theta}_n^t)$.

return $\boldsymbol{\theta}_{\mathrm{m}}^T$

Re-merge. Re-merging the modified models gives $\boldsymbol{\theta}_{\mathrm{m}}^t = \mathrm{Merge}(\boldsymbol{\theta}_1^t, \ldots, \boldsymbol{\theta}_n^t)$. The new merged parameters become the replacement values for the next iteration. After T iterations, we return $\boldsymbol{\theta}_{\mathrm{m}}^T$.

What re-merging computes. At iteration t, the modified fine-tuned models differ from the previous merged model only at retained parameters. These differences determine the update when the models are re-merged. Let $\boldsymbol{M}_j^t$ be a binary mask with ones for parameters retained by model j at iteration t and zeros for parameters in $\mathbb{S}_j^t$. For RegMean, the Gram matrices remain fixed across iterations. Substituting Equation 6 into Equation 2 gives, for each linear layer,

$$\boldsymbol{W}_{\mathrm{m}}^t = \boldsymbol{W}_{\mathrm{m}}^{t-1} + \sum_{j=1}^n \left[\boldsymbol{M}_j^t \odot (\boldsymbol{W}_j^0 - \boldsymbol{W}_{\mathrm{m}}^{t-1})\right] \widehat{\boldsymbol{G}}_j \Big(\sum_{k=1}^n \widehat{\boldsymbol{G}}_k \Big)^{-1}. \tag{7}$$

We use a simple case to explain the effect of re-merging. Consider three fine-tuned models with identical positive diagonal Gram matrices and values 1, 0, and 0 at one parameter. RegMean gives an initial merged value of $1/3$. At every iteration, we retain the first model's value and replace the other two with the previous merged value. Re-merging gives $5/9$, then $19/27$, and the merged value converges to 1. This example shows how repeated re-merging can better preserve selected fine-tuned values than a single RegMean merge. We derive Equation 7 in Appendix A.3.

3.3 Scoring Objectives

We consider total and separate scopes, each with a global task loss or a local reconstruction loss.

Task scopes. The merging operator Merge sums reconstruction losses across tasks, motivating a total task scope for replacement scoring. Let $\mathcal{L}_j$ denote the global task loss or local reconstruction loss for task j. For model i, total scope uses $\mathcal{J}_i = \sum_{j=1}^n \mathcal{L}_j$, with all task losses differentiated at the fine-tuned parameters $\boldsymbol{\theta}_i^0$. Separate scope uses only the loss of the model's own task, $\mathcal{J}_i = \mathcal{L}_i$.

Loss functions. The global task loss is the average cross-entropy on task j's calibration dataset:

$$\mathcal{L}_{j,\mathrm{glob}}(\widehat{\boldsymbol{\theta}}) := \frac{1}{|\mathcal{D}_j|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_j} \ell_{\mathrm{CE}}\Big(f(\boldsymbol{x};\widehat{\boldsymbol{\theta}}), \boldsymbol{y}\Big), \tag{8}$$

where f is the network and ℓ_{CE} is the cross-entropy loss. For the local reconstruction loss, let $\widehat{\boldsymbol{W}}$ denote the weight matrix of a fixed layer in the model parameterized by $\widehat{\boldsymbol{\theta}}$. We omit the layer index. Let $\boldsymbol{W}_j^0$ be the fine-tuned weights of model j and $\boldsymbol{X}_j^0$ its input activations calculated on $\mathcal{D}_j$. We define the local reconstruction loss as

$$\mathcal{L}_{j,\mathrm{loc}}(\widehat{\boldsymbol{W}}) := \Big\| (\widehat{\boldsymbol{W}} - \boldsymbol{W}_j^0)\boldsymbol{X}_j^0 \Big\|_F^2. \tag{9}$$

Table 1: Replacement scores with $\delta := \delta_{i,p}^t$ and derivatives evaluated at $\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}_i^0$. For local scores, (r, c) denotes the position of parameter p in the weight matrix.

Scope	Loss	Scoring objective $\mathcal{J}_i$	First-order score	Second-order score
Total	Global	$\sum_{j=1}^{n} \mathcal{L}_{j,\mathrm{glob}}(\widehat{\boldsymbol{\theta}})$	$\left\lvert \delta \sum_{j=1}^{n} \frac{\partial \mathcal{L}_{j,\mathrm{glob}}(\widehat{\boldsymbol{\theta}})}{\partial \widehat{\theta}_p} \right\rvert$	$\frac{1}{2}\delta^2 \left\lvert \sum_{j=1}^{n} \frac{\partial^2 \mathcal{L}_{j,\mathrm{glob}}(\widehat{\boldsymbol{\theta}})}{\partial \widehat{\theta}_p^2} \right\rvert$
Total	Local	$\sum_{j=1}^{n} \mathcal{L}_{j,\mathrm{loc}}(\widehat{\boldsymbol{W}})$	$\left\lvert 2\delta \left[\sum_{j \neq i} (\boldsymbol{W}_i^0 - \boldsymbol{W}_j^0) \boldsymbol{X}_j^0 (\boldsymbol{X}_j^0)^\top \right]_{rc} \right\rvert$	$\delta^2 \left[\sum_{j=1}^{n} \boldsymbol{X}_j^0 (\boldsymbol{X}_j^0)^\top \right]_{cc}$
Separate	Global	$\mathcal{L}_{i,\mathrm{glob}}(\widehat{\boldsymbol{\theta}})$	$\left\lvert \delta \frac{\partial \mathcal{L}_{i,\mathrm{glob}}(\widehat{\boldsymbol{\theta}})}{\partial \widehat{\theta}_p} \right\rvert$	$\frac{1}{2}\delta^2 \left\lvert \frac{\partial^2 \mathcal{L}_{i,\mathrm{glob}}(\widehat{\boldsymbol{\theta}})}{\partial \widehat{\theta}_p^2} \right\rvert$
Separate	Local	$\mathcal{L}_{i,\mathrm{loc}}(\widehat{\boldsymbol{W}})$	0	$\delta^2 \left[\boldsymbol{X}_i^0 (\boldsymbol{X}_i^0)^\top \right]_{cc}$

For scoring, we evaluate derivatives at $\widehat{\boldsymbol{W}} = \boldsymbol{W}_i^0$ and keep $\boldsymbol{W}_j^0$ and $\boldsymbol{X}_j^0$ fixed across iterations. For separate scope ($j = i$), the factor $\widehat{\boldsymbol{W}} - \boldsymbol{W}_i^0$ in the gradient becomes $\boldsymbol{W}_i^0 - \boldsymbol{W}_i^0 = 0$, so the first-order term vanishes. Together, these choices of loss and task scope specify $\mathcal{J}_i$ in the replacement-score formulation of Section 3.2. Table 1 summarizes the resulting scores, with derivations in Appendix A.

3.4 Synthetic Calibration Data

To compute replacement scores, OBMerge reuses the calibration data already required by RegMean or RegMean++, so it needs no additional calibration examples. However, real calibration data may be unavailable, and precomputed Gram matrices can incur substantial storage costs (Jin et al., 2023; Hameed et al., 2026). In such cases, we use an LLM to generate calibration examples for RegMean, RegMean++, and OBMerge. The LLM takes a *description of the task and dataset* and, when available, the *instruction template* used for fine-tuning. These compact textual inputs are easier to store and share than real data or Gram matrices. We separate description preparation from example generation to simulate a practical setting where users provide the task and dataset description and the LLM generates calibration examples from it. We use DeepSeek-V4.1-Flash for all generation steps. We vary the description source and instruction template in a 2×2 design:

1. **Task and dataset description.** For each task, we simulate a user-provided description by asking the LLM to write one from either (i) a public summary of the task and dataset alone or (ii) the same summary together with 256 real examples, presented as field-value pairs using the dataset's original field names. These real examples are used only to prepare the description and are not passed to subsequent generation or merging.
2. **Instruction template.** The template specifies the input and answer format. We either (i) ask the LLM to generate it from the description or (ii) reuse the template used to fine-tune the corresponding model.

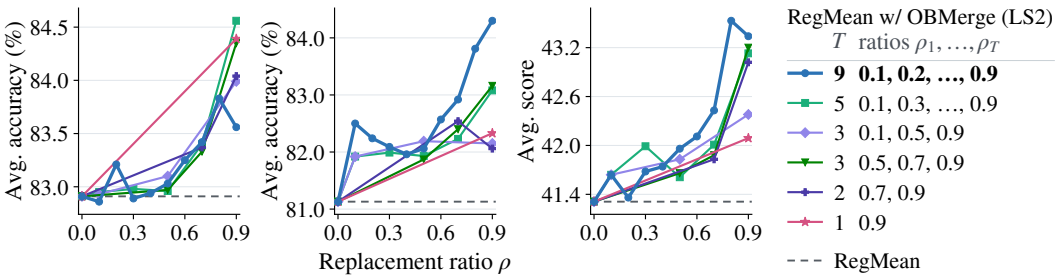
For each description–template combination, we generate calibration examples. We discard a synthetic example if its ROUGE-L F1 (Wang et al., 2023) or exact word 5-gram Jaccard similarity (Lee et al., 2022) with any available benchmark example is at least 0.7. We format the retained examples with the selected template. We provide further details in Appendix D.

4 Experiments

Experimental setup. We evaluate CLIP (Radford et al., 2021) ViT-B/32 and ViT-L/14 on eight vision tasks (Tang et al., 2024), Qwen3-0.6B (Yang et al., 2025) and Llama-3.2-1B (Grattafiori et al., 2024) on seven commonsense-reasoning (CSR) tasks, and Llama-3.2-3B (Grattafiori et al., 2024) on five MergeBench domains (He et al., 2025). Baselines in Table 2 use default hyperparameters. RegMean and RegMean++ use 512 calibration examples per vision/CSR task or MergeBench domain, with $\alpha = 0.95$ for vision and $\alpha = 0.9$ for language models. LS2 and GS1 denote local second-order and global first-order scores, respectively. In score names, the letters S and T indicate separate and

Table 2: Merging results: average accuracy (%) on vision and CSR, and average MergeBench score. Parentheses show gains over each run's initial merge; bold marks the best merging result per model.

	Vision (Avg.)		CSR (Avg.)		MergeBench (Avg.)
Method	ViT-B/32	ViT-L/14	Qwen3-0.6B	Llama-3.2-1B	Llama-3.2-3B
Pre-trained	47.96	65.28	63.67	49.75	27.54
Individual	90.46	94.38	84.95	83.96	57.12
Parameter-only merging					
AVG	66.17	80.01	69.43	65.66	35.23
TIES	71.79	83.61	74.82	64.81	41.04
TSV-M	83.05	90.61	82.56	83.47	42.81
Iso-C	80.47	90.76	81.19	79.07	36.40
Iso-CTS	78.90	90.43	79.17	76.54	33.87
ACTMat	82.97	**92.15**	81.86	82.11	44.28
Data-dependent merging					
OBIM	72.68	83.84	72.01	61.41	42.97
RegMean	82.74	90.54	82.91	81.13	41.31
w/ OBMerge (LS2)	83.95 (+1.21)	91.20 (+0.66)	83.56 (+0.65)	84.30 (+3.17)	43.34 (+2.03)
w/ OBMerge (GS1)	83.68 (+0.94)	91.08 (+0.54)	83.60 (+0.72)	83.31 (+0.80)	42.59 (+1.09)
RegMean++	84.55	91.20	83.30	82.80	43.42
w/ OBMerge (LS2)	85.31 (+0.76)	91.89 (+0.69)	84.15 (+0.85)	**84.57** (+1.77)	45.57 (+2.15)
w/ OBMerge (GS1)	85.25 (+0.70)	91.73 (+0.53)	83.93 (+0.50)	83.68 (+0.78)	44.32 (+0.97)
w/ OBMerge (LS2*)	**85.33** (+0.78)	91.90 (+0.70)	**84.36** (+1.06)	84.46 (+1.66)	**45.84** (+2.42)



(a) Qwen3-0.6B CSR (b) Llama-3.2-1B CSR (c) Llama-3.2-3B MergeBench

Figure 2: RegMean with OBMerge (LS2): six schedules on CSR and MergeBench.

total task scopes. LS2* uses RegMean++'s propagated inputs. For global task loss, we rank parameters jointly across layers within each model; for local reconstruction loss, we rank parameters separately within each layer. OBMerge uses linear replacement schedules $\rho_t = \rho_{\text{start}} + (t-1)\Delta\rho$ for $t = 1, \ldots, T$, where T is the number of iterations. By default, $\rho_{\text{start}} = \Delta\rho = 0.1$ and $T = 9$ for all models, and we report results from the final iteration. Experimental details are in Appendix C. Limitations are discussed in Appendix B.

Main results. OBMerge improves both RegMean and RegMean++ on all five models using either LS2 or GS1 with real calibration data, as shown in Table 2. LS2 improves vision/CSR accuracy by 0.65–3.17 percentage points and the MergeBench score by 2.03–2.15 points over its initial merges. Using LS2 and the default schedule for all models, RegMean++ with OBMerge outperforms all evaluated merging baselines on four of five models. ACTMat leads on ViT-L/14 (92.15% versus 91.89%). With RegMean++, LS2* gives further gains over LS2 on four models. Per-task results are in Tables 7–11. We also evaluate OBMerge with LS2 using ACTMat, TSV-M, Iso-C, and Iso-CTS as merging operators on ViT-B/32 and ViT-L/14 (Appendix C.4).

Replacement trajectory. RegMean with OBMerge (LS2) achieves higher final scores than RegMean under all six tested schedules on all three language models (Figure 2), showing that OBMerge is robust across the tested replacement schedules. Even the single-iteration schedule improves on RegMean on all three models. All schedules end at $\rho_T = 0.9$, but their final scores differ. The final ratio specifies the fraction of parameters replaced, while intermediate merges determine the re-

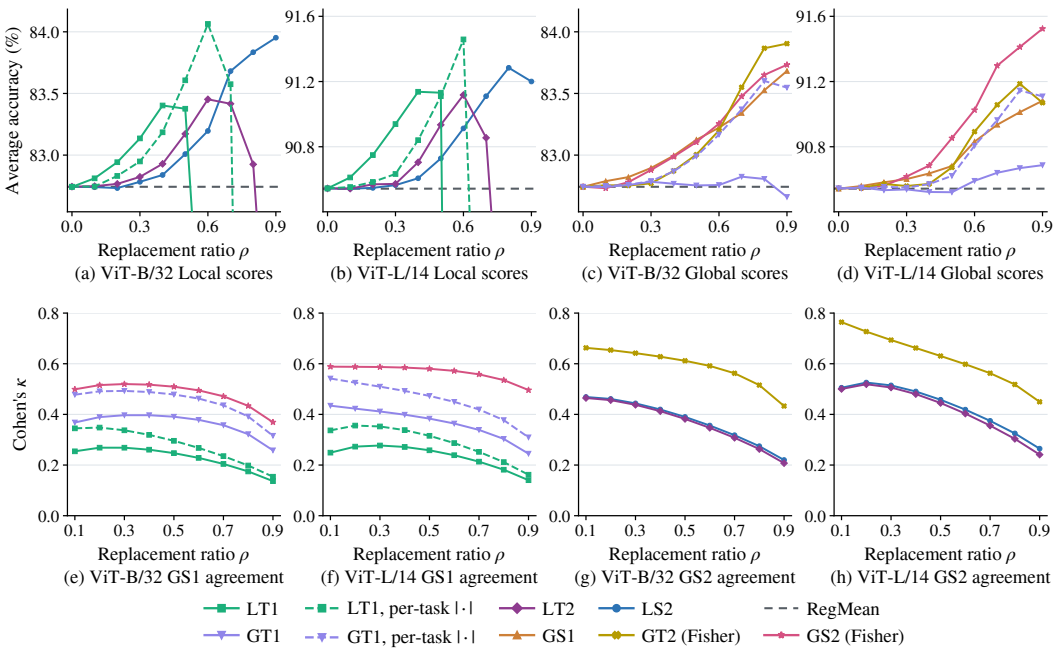


Figure 3: Replacement score comparison for RegMean with OBMerge. Top: average test accuracy (%); bottom: replacement-set agreement (Cohen's κ).

placement values used to compute LS2 scores. Different schedules can therefore produce different replacement values and parameter selections even at the same final ratio. The nine-iteration schedule achieves the highest final scores on both Llama models. On Qwen3-0.6B, the five-iteration schedule outperforms the nine-iteration schedule, motivating further study of schedules that maintain or improve performance with fewer iterations. Fewer iterations reduce the number of scoring, parameter replacement, and re-merging rounds.

Replacement scores. Global scores generally maintain high accuracy on both vision models under the default schedule, as shown in Figure 3(c,d). To calculate replacement-set agreement, we compare each score's replacement set A with the GS1 or GS2 reference set B at each iteration. Both sets contain a fraction ρ_t of each fine-tuned model's parameters and are obtained using the same merged model from the previous iteration of the reference run. Cohen's κ (Cohen, 1960) is defined as

$$\kappa = \frac{|A \cap B|/|A| - \rho_t}{1 - \rho_t}. \tag{10}$$

We average κ over eight fine-tuned models, while accuracy is reported from each score's own run. GS2 and GT2 use Fisher diagonals to approximate global Hessian diagonals because exact computation is impractical (Martens, 2020). This approximation can introduce errors in the scores and resulting replacement sets. At large replacement ratios, higher replacement-set agreement with GS1 or GS2 generally accompanies higher accuracy, as shown in Figure 3. We suspect that replacement choices that differ from the global references may adversely affect subsequent merging iterations, contributing to the accuracy declines of LT1 and LT2 at large replacement ratios. We also evaluate LT1 and GT1 variants that take the absolute value of each task's first-order contribution before summing, preventing cancellation across tasks. Both per-task variants improve final accuracy over the original scores, but the LT1 variant still suffers a sharp accuracy decline at large replacement ratios.

Data-dependent merging with synthetic calibration data. OBMerge improves RegMean++ on all three language base models under all four synthetic calibration settings, as shown in Table 3. Gains for RegMean are consistent on MergeBench but mixed on CSR. On CSR, fine-tuning templates improve average accuracy over LLM-generated templates, with little additional benefit from example-guided descriptions. On MergeBench, example-guided descriptions improve scores by 2.11–3.77 points across templates and methods. With these descriptions and LLM-generated templates, RegMean with OBMerge achieves an average score of 45.91, compared with 43.34 using real calibration data. Per-task results are in Tables 16–18.

Table 3: Average CSR test accuracy (%) and MergeBench score with synthetic calibration data. Bold: better in each pair. † 256 real examples guide descriptions; generation prompts exclude them.

Template	Description preparation	Method	CSR (Avg.) Qwen3-0.6B	CSR (Avg.) Llama-3.2-1B	MergeBench (Avg.) Llama-3.2-3B
Real-data reference		RegMean	82.91	81.13	41.31
		RegMean++	83.30	82.80	43.42
LLM-generated	Summary only	RegMean	**73.51**	68.50	41.16
		w/ OBMerge	73.16 (−0.35)	**70.55** (+2.05)	**42.14** (+0.97)
		RegMean++	75.65	72.07	41.74
		w/ OBMerge	**76.93** (+1.28)	**75.66** (+3.59)	**42.37** (+0.63)
	Summary with example guidance†	RegMean	**76.06**	70.52	44.30
		w/ OBMerge	74.67 (−1.39)	**72.75** (+2.23)	**45.91** (+1.61)
		RegMean++	76.78	73.38	44.23
		w/ OBMerge	**76.91** (+0.13)	**76.24** (+2.86)	**45.81** (+1.58)
Fine-tuning	Summary only	RegMean	82.08	**81.40**	39.13
		w/ OBMerge	**82.45** (+0.37)	81.06 (−0.34)	**40.91** (+1.78)
		RegMean++	82.74	80.69	41.99
		w/ OBMerge	**82.99** (+0.25)	**81.68** (+0.99)	**43.11** (+1.12)
	Summary with example guidance†	RegMean	**82.45**	80.71	41.48
		w/ OBMerge	82.13 (−0.32)	**81.67** (+0.96)	**43.02** (+1.54)
		RegMean++	82.75	80.71	44.33
		w/ OBMerge	**83.38** (+0.63)	**81.81** (+1.10)	**45.68** (+1.34)

Table 4: CSR coefficient selection on Llama-3.2-1B: average accuracy (%) on real test data and selected λ. Δ: gain over Default ($\lambda = 1$) using synthetic calibration data (percentage points).

Method	**Default**	**Real calib.**		**Synthetic calib.**		**Δ vs. Default**
	Acc.	λ	Acc.	λ	Acc.	Acc.
TIES	64.87	1.9	72.46	2.3	72.21	+7.34
Iso-C	79.27	1.7	83.44	1.2	82.48	+3.21
Iso-CTS	76.17	2.2	82.09	2.1	82.39	+6.22
TSV-M	83.90	2.1	83.37	1.7	84.75	+0.85
Average	**76.05**		**80.34**		**80.46**	**+4.41**

Coefficient selection with synthetic calibration data. Selecting merging coefficients with synthetic calibration data raises average CSR test accuracy to 80.75%, a gain of 3.05 percentage points over the default setting ($\lambda = 1$) across two base models and four methods, as shown in Table 19. The average accuracy is 0.50 percentage points below that obtained with real calibration data. Qwen3-0.6B with Iso-C shows the largest gap from coefficient selection using real calibration data, with test accuracy lower by 3.79 percentage points. In this experiment, we select $\lambda \in \{0.1, 0.2, \ldots, 3.0\}$ using 100 real or synthetic examples per task. We use synthetic examples generated using example-guided descriptions and LLM-generated templates. Experimental details are in Appendix D.

5 Conclusion

We introduce OBMerge, an iterative model-merging framework based on parameter replacement. With real calibration data, OBMerge improves RegMean and RegMean++ on all five models and achieves the highest average scores among the compared merging methods on four. Our method also improves RegMean and RegMean++ in most tested synthetic calibration settings. With suitable task descriptions and instruction templates, synthetic calibration data can yield good performance in data-dependent model merging and coefficient selection. These results suggest that task descriptions and instruction templates can support model merging and coefficient selection without requiring users to provide real calibration examples in advance. Future work could explore different scheduling strategies and more accurate replacement scores.

Reproducibility Statement

We describe OBMerge in Section 3 and derive its scores in Appendix A. Section 4 and Appendix C document the datasets, model checkpoints, experimental settings, and detailed results. Appendix D provides the calibration-data generation pipeline, model settings, filtering procedure, and complete prompts. Anonymized code is included in the supplementary material.

AI Use Statement

Generative AI tools assisted with manuscript drafting and language editing, code implementation and debugging, and figure preparation. We used DeepSeek-V4.1-Flash to generate task descriptions, instruction templates, and calibration examples, as described in Appendix D. The authors reviewed all AI-assisted content and take responsibility for the manuscript, code, and results.

References

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? In *Proceedings of Machine Learning and Systems*, volume 2, pp. 129–146, 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/hash/6c44dc73014d66ba49b28d483a8f8b0d-Abstract.html.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote Sensing Image Scene Classification: Benchmark and State of the Art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017. URL https://arxiv.org/abs/1703.00121.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing Textures in the Wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3606–3613, 2014. doi: 10.1109/CVPR.2014.461. URL https://doi.org/10.1109/CVPR.2014.461.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

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. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

Enmao Diao, Ganghua Wang, Jiawei Zhang, Yuhong Yang, Jie Ding, and Vahid Tarokh. Pruning deep neural networks from a sparsity perspective. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=i-DleYh34BM.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning (ICML)*, 2023.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task Singular Vectors: Reducing Task Interference in Model Merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. URL https://arxiv.org/abs/2412.00081.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023. doi: 10.48550/arXiv.2306.11644. URL https://arxiv.org/abs/2306.11644.

Marawan Gamal Abdel Hameed, Derek Tam, Pascal Jr Tikeng Notsawo, Colin Raffel, and Guillaume Rabusseau. Model merging via data-free covariance estimation. *arXiv preprint arXiv:2604.01329*, 2026. URL https://arxiv.org/abs/2604.01329.

Hasan Abed Al Kader Hammoud, Umberto Michieli, Fabio Pizzati, Philip Torr, Adel Bibi, Bernard Ghanem, and Mete Ozay. Model merging and safety alignment: One bad model spoils the bunch. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 13033–13046, 2024. doi: 10.18653/v1/2024.findings-emnlp.762. URL https://aclanthology.org/2024.findings-emnlp.762/.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. WildGuard: Open One-Stop Moderation Tools for Safety Risks, Jailbreaks, and Refusals of LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2406.18495.

Yifei He, Yuzheng Hu, Yong Lin, Tong Zhang, and Han Zhao. Localize-and-stitch: Efficient model merging via sparse task arithmetic. *arXiv preprint arXiv:2408.13656*, 2024.

Yifei He, Siqi Zeng, Yuzheng Hu, Rui Yang, Tong Zhang, and Han Zhao. MergeBench: A Benchmark for Merging Domain-Specialized LLMs. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=rw50iUoyLu.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019. URL https://arxiv.org/abs/1709.00029.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2009.03300.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021. URL https://www.jmlr.org/papers/v22/21-0366.html.

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

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. QASC: A Dataset for Question Answering via Sentence Composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. URL https://arxiv.org/abs/1910.11473.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D Object Representations for Fine-Grained Categorization. In *IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013. doi: 10.1109/ICCVW.2013.77. URL https://doi.org/10.1109/ICCVW.2013.77.

Viet Lai, Chien Nguyen, Nghia Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan Rossi, and Thien Nguyen. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 318–327, 2023. doi: 10.18653/v1/2023.emnlp-demo.28. URL https://aclanthology.org/2023.emnlp-demo.28/.

Qi Le, Enmao Diao, Ziyan Wang, Xinran Wang, Jie Ding, Li Yang, and Ali Anwar. Probe Pruning: Accelerating LLMs through dynamic pruning via model-probing. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=WOt1owGfuN.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 2, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL https://doi.org/10.1109/5.726791.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8424–8445, 2022. doi: 10.18653/v1/2022.acl-long.577. URL https://aclanthology.org/2022.acl-long.577/.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The Winograd Schema Challenge. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2012. URL https://cdn.aaai.org/ocs/4492/4492-21843-1-PB.pdf.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.01210.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the Structural Pruning of Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.11627.

Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost Van De Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 43177–43199, 2025. URL https://proceedings.mlr.press/v267/marczak25a.html.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL https://arxiv.org/abs/1412.1193.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2402.04249.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning Convolutional Neural Networks for Resource Efficient Inference. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1611.06440.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance Estimation for Neural Network Pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. URL https://arxiv.org/abs/1906.10771.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 839–849, 2016. doi: 10.18653/v1/N16-1098. URL https://aclanthology.org/N16-1098/.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. URL https://storage.googleapis.com/pub-tools-public-publication-data/pdf/37648.pdf.

The-Hai Nguyen, Dang Huu-Tien, Takeshi Suzuki, and Le-Minh Nguyen. Regmean++: Enhancing effectiveness and generalization of regression mean for model merging. *Transactions on Machine Learning Research (TMLR)*, 2026.

Amin Heyrani Nobari, Kaveh Alimohammadi, Ali ArjomandBigdeli, Akash Srivastava, Faez Ahmed, and Navid Azizan. Activation-informed merging of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pranav Shyam, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5377–5400, 2024. doi: 10.18653/v1/2024.naacl-long.301. URL https://aclanthology.org/2024.naacl-long.301/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. URL https://arxiv.org/abs/1907.10641.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "Do Anything Now": Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2024. URL https://arxiv.org/abs/2308.03825.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A Multi-Class Classification Competition. In *International Joint Conference on Neural Networks (IJCNN)*, pp. 1453–1460, 2011. doi: 10.1109/IJCNN.2011.6033395. URL https://doi.org/10.1109/IJCNN.2011.6033395.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. QuaRTz: An open-domain dataset of qualitative relationship questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5941–5946, 2019. doi: 10.18653/v1/D19-1608. URL https://aclanthology.org/D19-1608/.

Anke Tang, Li Shen, Yong Luo, Han Hu, Bo Du, and Dapeng Tao. FusionBench: A comprehensive benchmark of deep model fusion. *arXiv preprint arXiv:2406.03280*, 2024.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. In *International Conference on Machine Learning (ICML)*, 2024.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, 2023. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754/.

Zhixiang Wang, Zhenyu Mao, Yixuan Qiao, Yunfang Wu, and Biye Li. Optimal brain iterative merging: Mitigating interference in LLM merging. *arXiv preprint arXiv:2502.12217*, 2025. URL https://arxiv.org/abs/2502.12217.

Ziyan Wang, Enmao Diao, Qi Le, Pu Wang, Minwoo Lee, Shu-ping Yeh, Evgeny Stupachenko, Hao Feng, and Li Yang. From local to global: Revisiting structured pruning paradigms for large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35720–35739, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1653. URL https://aclanthology.org/2026.acl-long.1653/.

Miles Williams, George Chrysostomou, and Nikolaos Aletras. Self-calibration for language model quantization and pruning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10149–10167, 2025. doi: 10.18653/v1/2025.naacl-long.509. URL https://aclanthology.org/2025.naacl-long.509/.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, 2022.

Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. SUN Database: Exploring a Large Collection of Scene Categories. *International Journal of Computer Vision*, 119:3–22, 2016. doi: 10.1007/s11263-014-0748-y. URL https://doi.org/10.1007/s11263-014-0748-y.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2013–2018, 2015. doi: 10.18653/v1/D15-1237. URL https://aclanthology.org/D15-1237/.

Yuxuan Yao, Shuqi Liu, Zehua Liu, Qintong Li, Mingyang Liu, Xiongwei Han, Zhijiang Guo, Han Wu, and Linqi Song. Activation-guided consensus merging for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. ZeroGen: Efficient zero-shot learning via dataset generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11653–11669, 2022. doi: 10.18653/v1/2022.emnlp-main.801. URL https://aclanthology.org/2022.emnlp-main.801/.

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. In *International Conference on Machine Learning (ICML)*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Yuan Zhang, Jason Baldridge, and Luheng He. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1298–1308, 2019. doi: 10.18653/v1/N19-1131. URL https://aclanthology.org/N19-1131/.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-Following Evaluation for Large Language Models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

Appendix

A Derivation of the Replacement Scores

We derive the replacement scores in Table 1 from the objectives in Section 3.3. Fix model i, iteration t, and parameter p, and write $\delta := \delta_{i,p}^t$ as defined in Equation 3. Using the partial derivatives $g_{i,p}$ and $H_{i,pp}$ defined after Equation 4, the first- and second-order scores are $|\delta g_{i,p}|$ and $\frac{1}{2}\delta^2|H_{i,pp}|$, respectively. Each score evaluates a single-parameter replacement at $\boldsymbol{\theta}_i^0$, with all other parameters fixed at their fine-tuned values.

A.1 Global Task Loss

In total scope, differentiating $\mathcal{J}_i = \sum_{j=1}^n \mathcal{L}_{j,\text{glob}}$ at $\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}_i^0$ gives

$$\begin{aligned} g_{i,p} &= \sum_{j=1}^{n} \frac{\partial \mathcal{L}_{j,\text{glob}}(\widehat{\boldsymbol{\theta}})}{\partial \widehat{\theta}_p}\bigg|_{\widehat{\boldsymbol{\theta}}=\boldsymbol{\theta}_i^0}, \\ H_{i,pp} &= \sum_{j=1}^{n} \frac{\partial^2 \mathcal{L}_{j,\text{glob}}(\widehat{\boldsymbol{\theta}})}{\partial \widehat{\theta}_p^2}\bigg|_{\widehat{\boldsymbol{\theta}}=\boldsymbol{\theta}_i^0}. \end{aligned} \tag{11}$$

Separate scope keeps only $j = i$ in these sums. Both scores use absolute values to measure the magnitude of the estimated loss change.

A.2 Local Reconstruction Loss

For a fixed layer, let (r, c) denote the position of parameter p in the weight matrix. With the fine-tuned weights $\boldsymbol{W}_j^0$ and input activations $\boldsymbol{X}_j^0$ fixed, differentiating Equation 9 gives

$$\begin{aligned} \frac{\partial \mathcal{L}_{j,\text{loc}}(\widehat{\boldsymbol{W}})}{\partial \widehat{W}_{rc}} &= 2\big[(\widehat{\boldsymbol{W}} - \boldsymbol{W}_j^0)\boldsymbol{X}_j^0(\boldsymbol{X}_j^0)^\top\big]_{rc}, \\ \frac{\partial^2 \mathcal{L}_{j,\text{loc}}(\widehat{\boldsymbol{W}})}{\partial \widehat{W}_{rc}^2} &= 2\big[\boldsymbol{X}_j^0(\boldsymbol{X}_j^0)^\top\big]_{cc}. \end{aligned} \tag{12}$$

Under the total task scope, evaluating at $\widehat{\boldsymbol{W}} = \boldsymbol{W}_i^0$ and summing over tasks yields

$$\begin{aligned} g_{i,p} &= 2\Big[\sum_{j\neq i}(\boldsymbol{W}_i^0 - \boldsymbol{W}_j^0)\boldsymbol{X}_j^0(\boldsymbol{X}_j^0)^\top\Big]_{rc}, \\ H_{i,pp} &= 2\Big[\sum_{j=1}^{n}\boldsymbol{X}_j^0(\boldsymbol{X}_j^0)^\top\Big]_{cc}. \end{aligned} \tag{13}$$

Since $H_{i,pp} \geq 0$, the second-order score needs no absolute value. Under the separate task scope, $g_{i,p} = 0$, giving scores 0 and $\delta^2[\boldsymbol{X}_i^0(\boldsymbol{X}_i^0)^\top]_{cc}$. With fixed activations, the local reconstruction objectives under total and separate scopes are quadratic in the weights, so the first- and second-order Taylor terms together give the exact loss change for a single-parameter replacement. Taking its absolute value gives the score $|\delta g_{i,p} + \frac{1}{2}\delta^2 H_{i,pp}|$. In separate scope, $g_{i,p} = 0$, so the second-order score alone is exact. In total scope, the gradient can be nonzero, so the first-order term also contributes to the exact score.

A.3 Re-merging with Parameter Replacement

Fix a linear layer and omit its index. Using the mask $\boldsymbol{M}_j^t$, the replacement rule in Equation 6 gives

$$\boldsymbol{W}_j^t = \boldsymbol{W}_{\text{m}}^{t-1} + \boldsymbol{M}_j^t \odot (\boldsymbol{W}_j^0 - \boldsymbol{W}_{\text{m}}^{t-1}). \tag{14}$$

RegMean uses fixed Gram matrices $\widehat{\boldsymbol{G}}_j$ from the original fine-tuned models. Substituting Equation 14 into Equation 2 and expanding gives

$$\begin{aligned}
\boldsymbol{W}_{\mathrm{m}}^{t} &= \Big(\sum_{j=1}^{n} \boldsymbol{W}_j^{t} \widehat{\boldsymbol{G}}_j\Big)\Big(\sum_{k=1}^{n} \widehat{\boldsymbol{G}}_k\Big)^{-1} \\
&= \Big(\sum_{j=1}^{n} \big[\boldsymbol{W}_{\mathrm{m}}^{t-1} + \boldsymbol{M}_j^{t} \odot (\boldsymbol{W}_j^{0} - \boldsymbol{W}_{\mathrm{m}}^{t-1})\big] \widehat{\boldsymbol{G}}_j\Big)\Big(\sum_{k=1}^{n} \widehat{\boldsymbol{G}}_k\Big)^{-1} \\
&= \boldsymbol{W}_{\mathrm{m}}^{t-1}\Big(\sum_{j=1}^{n} \widehat{\boldsymbol{G}}_j\Big)\Big(\sum_{k=1}^{n} \widehat{\boldsymbol{G}}_k\Big)^{-1} \\
&\quad + \sum_{j=1}^{n} \big[\boldsymbol{M}_j^{t} \odot (\boldsymbol{W}_j^{0} - \boldsymbol{W}_{\mathrm{m}}^{t-1})\big] \widehat{\boldsymbol{G}}_j \Big(\sum_{k=1}^{n} \widehat{\boldsymbol{G}}_k\Big)^{-1} \\
&= \boldsymbol{W}_{\mathrm{m}}^{t-1} + \sum_{j=1}^{n} \big[\boldsymbol{M}_j^{t} \odot (\boldsymbol{W}_j^{0} - \boldsymbol{W}_{\mathrm{m}}^{t-1})\big] \widehat{\boldsymbol{G}}_j \Big(\sum_{k=1}^{n} \widehat{\boldsymbol{G}}_k\Big)^{-1}.
\end{aligned} \tag{15}$$

The sum of Gram matrices and its inverse cancel in the first term, giving Equation 7. The same identity holds for RegMean++ using the Gram matrices of the current iteration.

B Limitations

The replacement scores in OBMerge evaluate parameters individually and neglect interactions among jointly replaced parameters. Selecting parameters with low scores therefore does not guarantee improved task performance after re-merging. For global second-order scores under separate and total task scopes, we approximate global curvature using Fisher diagonals. Future work could account for parameter interactions and estimate curvature more accurately. This may help identify which parameters to replace.

We evaluate synthetic calibration data generated by one LLM on three language base models. Performance varies with the task descriptions and instruction templates. In some configurations, it is lower than with real calibration data. The example-guided setting simulates users preparing task and dataset descriptions from real examples. These examples are not passed to subsequent generation or merging. We discard a synthetic example if its ROUGE-L F1 score or exact word 5-gram Jaccard similarity with a benchmark example meets or exceeds the filtering threshold. However, this filtering may miss paraphrased benchmark content. In addition, the LLM used to generate calibration data or the evaluated base models may have seen benchmark test examples during training. Future work could use other LLMs to generate calibration data. Benchmarks created after both the LLMs and the evaluated base models were trained could help assess whether the results generalize.

We evaluate OBMerge with RegMean and RegMean++ on five models. Future work could extend this evaluation to other merging operators and architectures. The best tested schedule varies across models. Future work could study how to adjust replacement schedules and when to stop iterating.

C Experimental Details and Full Results

C.1 Datasets and Evaluation Suites

Table 5 lists the datasets. The MergeBench (He et al., 2025) suite is also used by RegMean++ (Nguyen et al., 2026).

Table 5: Datasets. de, es, fr, and ru denote German, Spanish, French, and Russian.

Group	Datasets and configurations
MergeBench: evaluation	**Instruction following:** IFEval (Zhou et al., 2023). **Mathematics:** GSM8K (Cobbe et al., 2021), 8-shot chain-of-thought. **Multilingual reasoning:** mMMLU, mARC, and mHellaSwag from Okapi (Lai et al., 2023), translated from MMLU (Hendrycks et al., 2021), ARC (Clark et al., 2018), and HellaSwag (Zellers et al., 2019); each in de, es, fr, and ru ($3 \times 4 = 12$ configurations). **Code generation:** HumanEval+ and MBPP+ (Liu et al., 2023). **Safety:** WildGuardTest (Han et al., 2024), HarmBench (Mazeika et al., 2024), DoAnythingNow (Shen et al., 2024), and XSTest (Röttger et al., 2024).
MergeBench: calibration	One set per domain (He et al., 2025): mb-instruction, mb-math, mb-multilingual, mb-coding, and mb-safety.
ViT: eight merging tasks	Cars (Krause et al., 2013), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), GTSRB (Stallkamp et al., 2011), RESISC45 (Cheng et al., 2017), SUN397 (Xiao et al., 2016), SVHN (Netzer et al., 2011), and MNIST (LeCun et al., 1998).
CSR: seven merging tasks	PAWS (Zhang et al., 2019), QASC (Khot et al., 2020), QuaRTz (Tafjord et al., 2019), StoryCloze (Mostafazadeh et al., 2016), WikiQA (Yang et al., 2015), WinoGrande (Sakaguchi et al., 2020), and WSC (Levesque et al., 2012).

Vision checkpoints. The fine-tuned CLIP (Radford et al., 2021) vision encoders are the FusionBench (Tang et al., 2024) checkpoints used by RegMean++ (Nguyen et al., 2026) (Table 6). Only the vision encoder is merged; the text encoder and both projection layers come from the pre-trained model. Each task uses a fixed zero-shot head built from the pre-trained text encoder with the task's class names and prompt templates, shared by all merged models.

Table 6: Vision checkpoints on Hugging Face. {task} is one of mnist, stanford-cars, dtd, eurosat, gtsrb, resisc45, sun397, or svhn.

Base model	Role	Hugging Face repository
ViT-B/32	Pre-trained	openai/clip-vit-base-patch32
	Fine-tuned	tanganke/clip-vit-base-patch32_{task}
ViT-L/14	Pre-trained	openai/clip-vit-large-patch14
	Fine-tuned	tanganke/clip-vit-large-patch14_{task}

Language checkpoints. For each CSR task, we fine-tune all parameters of Qwen/Qwen3-0.6B-Base (Yang et al., 2025) and meta-llama/Llama-3.2-1B (Grattafiori et al., 2024) using AdamW with a learning rate of 2×10^{-5} and weight decay of 5×10^{-4}. We train for 10 epochs with a batch size of 20, a sequence length of 256, and a cosine learning rate schedule with 5% warmup. We evaluate models on each task's test split using the same instruction template as for fine-tuning. For MergeBench, we use the released fine-tuned checkpoints MergeBench/Llama-3.2-3B_{domain} (He et al., 2025), where {domain} is one of instruction, math, multilingual, coding, or safety. These checkpoints were all fine-tuned from meta-llama/Llama-3.2-3B (Grattafiori et al., 2024). We evaluate models using the MergeBench evaluators and RegMean++'s metric definitions (Appendix C.2).

C.2 MERGEBENCH METRICS AND SCORE AGGREGATION

We select one metric per dataset (in $[0, 1]$), weight datasets equally within each domain regardless of test-set size, and report domain scores on a 0–100 scale (higher is better).

Instruction following. We use IFEval prompt-level loose accuracy a_{IFEval} (`prompt_level_loose_acc,none`), so $S_{\text{inst}} = 100 a_{\text{IFEval}}$.

Mathematics. Following the dataset scope of RegMean++ (Nguyen et al., 2026), we use GSM8K alone, with 8-shot chain-of-thought prompting and `exact_match,strict-match` from `gsm8k_cot`: $S_{\text{math}} = 100 a_{\text{GSM8K}}$. Extraction failures count as incorrect.

Multilingual reasoning. Let $\mathcal{L} = \{\text{de}, \text{es}, \text{fr}, \text{ru}\}$. For each language k, we use `acc,none` for mMMLU and `acc_norm,none` for mARC and mHellaSwag, and average the twelve dataset–language scores:

$$S_{\text{multi}} = \frac{100}{12} \sum_{k \in \mathcal{L}} \left(a_{\text{mMMLU},k} + a^{\text{norm}}_{\text{mARC},k} + a^{\text{norm}}_{\text{mHellaSwag},k} \right). \tag{16}$$

Code generation. We use `pass@1` for HumanEval+ and MBPP+:

$$S_{\text{code}} = \frac{100}{2} \left(p^{@1}_{\text{HumanEval+}} + p^{@1}_{\text{MBPP+}} \right). \tag{17}$$

Safety. We use WildGuardTest's harmful-response rate h_{WG}, the attack success rates r_{HB} on HarmBench and r_{DAN} on DoAnythingNow, and XSTest's overall accuracy a_{XS}. We subtract each of the first three rates from 1 so that higher values indicate better safety, then average the four scores:

$$S_{\text{safety}} = \frac{100}{4} \left[(1 - h_{\text{WG}}) + (1 - r_{\text{HB}}) + (1 - r_{\text{DAN}}) + a_{\text{XS}} \right]. \tag{18}$$

Overall average. The five domains have equal weight:

$$S_{\text{avg}} = \frac{1}{5} \left(S_{\text{inst}} + S_{\text{math}} + S_{\text{multi}} + S_{\text{code}} + S_{\text{safety}} \right). \tag{19}$$

Aggregation uses unrounded scores; values are rounded to two decimal places only for presentation.

C.3 BASELINE SETTINGS AND FULL RESULTS

Hyperparameters. The main tables use fixed baseline defaults and no validation- or test-based hyperparameter selection. Task-vector methods (TIES, TSV-M, Iso-C, Iso-CTS): scaling coefficient $\lambda = 1$. TIES: keep ratio 20%. TSV-M (Gargiulo et al., 2025): rank $\lfloor r/K \rfloor$ per task, with $r = \min(d_{\text{out}}, d_{\text{in}})$ for a $d_{\text{out}} \times d_{\text{in}}$ matrix and $K = n$ fine-tuned models (8 vision, 7 CSR, 5 MergeBench). Iso-CTS: common-space fraction 0.8. OBIM (Wang et al., 2025): each model scores entries by layer-wise saliency $[\boldsymbol{X}\boldsymbol{X}^\top]_{cc}\, \delta^2_{rc}$ on RegMean's calibration examples and keeps a $1/K$ share of every weight matrix, assigned mutually exclusively in a merging order rotated by one position per transformer block; no scaling coefficient. Appendix D.6 compares fixed scaling with scaling selected on real or synthetic calibration data.

Merged modules. AVG and TIES merge all parameters. All other methods, including OBMerge via its RegMean or RegMean++ operator, merge the attention and MLP weight matrices of every layer (vision: `q/k/v/out_proj`, `fc1`, `fc2`; CSR: `q/k/v/o_proj`, `gate/up/down_proj`); TSV-M, Iso-C, and Iso-CTS also merge the vision positional embedding. Remaining parameters are averaged.

Per-task results. Tables 7–10 report per-task accuracy. *Pre-trained* denotes the base model before fine-tuning; *Individual* denotes each fine-tuned model evaluated on its own task.

Table 7: **ViT-B/32.** Test accuracy (%) on eight vision tasks. Parentheses: gains over initial merges (percentage points); bold: best average among merging methods.

Method	MNIST	Cars	DTD	EuroSAT	GTSRB	RESISC45	SUN397	SVHN	Avg.
Pre-trained	48.25	59.59	44.26	43.93	32.56	60.30	63.21	31.59	47.96
Individual	99.58	78.56	79.68	99.63	98.91	95.14	74.89	97.27	90.46
Parameter-only merging									
AVG	86.26	62.42	50.53	74.63	54.92	70.67	65.44	64.52	66.17
TIES	97.53	58.15	47.34	76.07	79.56	68.90	60.32	86.46	71.79
TSV-M	98.86	71.67	63.83	93.19	92.52	84.71	67.69	91.90	83.05
Iso-C	96.69	73.49	65.21	88.59	84.59	84.79	71.69	78.70	80.47
Iso-CTS	95.36	73.61	65.00	87.56	80.89	84.05	71.70	73.04	78.90
ACTMat	99.19	70.48	60.80	92.11	94.83	83.86	68.14	94.35	82.97
Data-dependent merging									
OBIM	97.11	60.83	52.87	80.96	77.45	70.41	62.84	78.95	72.68
RegMean	98.45	70.65	66.49	95.00	83.71	85.89	69.28	92.47	82.74
w/ OBMerge (LS2)	98.91	69.26	69.20	94.70	88.20	87.40	68.66	95.28	83.95 (+1.21)
w/ OBMerge (GS1)	98.55	69.97	68.40	95.19	85.59	88.02	70.17	93.58	83.68 (+0.94)
RegMean++	98.97	71.01	69.47	94.52	90.66	87.24	69.91	94.63	84.55
w/ OBMerge (LS2)	99.23	69.97	71.01	94.63	93.41	89.25	69.61	95.39	85.31 (+0.76)
w/ OBMerge (GS1)	99.11	70.23	71.17	94.30	92.09	88.94	70.82	95.32	85.25 (+0.70)
w/ OBMerge (LS2*)	99.25	69.93	71.12	94.56	93.57	89.14	69.77	95.27	**85.33** (+0.78)

Table 8: **ViT-L/14.** Test accuracy (%) on eight vision tasks. Parentheses: gains over initial merges (percentage points); bold: best average among merging methods.

Method	MNIST	Cars	DTD	EuroSAT	GTSRB	RESISC45	SUN397	SVHN	Avg.
Pre-trained	76.36	77.84	55.53	64.00	50.53	71.32	68.21	58.47	65.28
Individual	99.76	92.77	85.48	99.70	99.24	97.40	82.77	97.92	94.38
Parameter-only merging									
AVG	96.64	81.48	61.70	89.81	73.99	82.29	72.53	81.63	80.01
TIES	98.16	80.43	59.95	86.22	91.31	85.13	74.90	92.80	83.61
TSV-M	99.08	89.80	75.27	97.04	96.48	93.48	78.17	95.57	90.61
Iso-C	98.77	90.76	76.76	97.37	95.39	94.37	79.75	92.93	90.76
Iso-CTS	98.64	91.01	77.07	97.41	94.54	93.90	79.43	91.41	90.43
ACTMat	99.22	90.15	78.99	98.07	98.32	94.60	80.85	97.00	**92.15**
Data-dependent merging									
OBIM	98.33	82.90	62.45	90.44	87.74	84.57	74.26	90.05	83.84
RegMean	98.74	89.63	77.23	97.70	94.40	92.92	77.20	96.53	90.54
w/ OBMerge (LS2)	98.53	89.01	79.57	97.74	97.27	93.05	77.35	97.08	91.20 (+0.66)
w/ OBMerge (GS1)	98.73	90.14	78.62	97.78	95.41	93.67	77.39	96.93	91.08 (+0.54)
RegMean++	99.09	90.01	78.99	97.26	96.42	93.24	77.59	96.99	91.20
w/ OBMerge (LS2)	99.26	90.18	80.64	97.85	97.69	93.92	78.19	97.40	91.89 (+0.69)
w/ OBMerge (GS1)	99.18	90.37	80.00	97.59	97.21	94.35	77.76	97.38	91.73 (+0.53)
w/ OBMerge (LS2*)	99.28	90.20	80.48	97.89	97.72	93.95	78.27	97.44	91.90 (+0.70)

Table 9: **Qwen3-0.6B.** Test accuracy (%) on seven CSR tasks. Parentheses: gains over initial merges (percentage points); bold: best average among merging methods.

Method	PAWS	QASC	QuaRTz	WikiQA	StoryCloze	WSC	WinoGrande	Avg.
Pre-trained	76.30	60.69	51.79	89.86	53.87	64.42	48.78	63.67
Individual	92.91	96.44	85.59	96.17	92.46	67.31	63.77	84.95
Parameter-only merging								
AVG	80.74	78.62	57.78	94.34	59.11	65.38	50.04	69.43
TIES	85.72	90.06	68.88	92.26	66.92	64.42	55.49	74.82
TSV-M	89.35	94.60	82.53	94.86	88.67	68.27	59.67	82.56
Iso-C	87.59	93.30	80.61	95.46	88.67	67.31	55.41	81.19
Iso-CTS	85.00	91.25	77.68	95.56	89.31	63.46	51.93	79.17
ACTMat	89.96	94.49	80.10	94.94	84.55	65.38	63.61	81.86
Data-dependent merging								
OBIM	83.40	83.15	64.67	93.98	64.30	65.38	49.17	72.01
RegMean	91.06	95.90	84.95	95.57	89.74	64.42	58.72	82.91
w/ OBMerge (LS2)	92.22	96.54	84.06	95.44	90.01	65.38	61.25	83.56 (+0.65)
w/ OBMerge (GS1)	91.90	96.44	86.22	95.78	90.86	62.50	61.48	83.60 (+0.72)
RegMean++	91.19	96.00	85.59	95.56	89.42	65.38	59.98	83.30
w/ OBMerge (LS2)	92.51	96.44	84.44	95.56	89.74	68.27	62.12	84.15 (+0.85)
w/ OBMerge (GS1)	91.74	96.44	86.48	95.77	90.86	65.38	60.85	83.93 (+0.50)
w/ OBMerge (LS2*)	92.47	96.22	84.82	95.67	90.01	69.23	62.12	**84.36** (+1.06)

Table 10: **Llama-3.2-1B.** Test accuracy (%) on seven CSR tasks. Parentheses: gains over initial merges (percentage points); bold: best average among merging methods.

Method	PAWS	QASC	QuaRTz	WikiQA	StoryCloze	WSC	WinoGrande	Avg.
Pre-trained	56.71	77.75	51.15	21.39	55.00	36.54	49.72	49.75
Individual	92.75	97.30	87.37	95.86	84.29	57.69	72.45	83.96
Parameter-only merging								
AVG	61.18	84.56	55.74	80.31	56.97	70.19	50.67	65.66
TIES	64.42	91.68	62.12	84.38	56.65	36.54	57.85	64.81
TSV-M	89.03	96.44	85.59	93.80	76.32	75.00	68.11	83.47
Iso-C	88.09	94.82	83.16	93.59	71.46	58.65	63.69	79.07
Iso-CTS	85.26	93.52	81.89	95.15	73.54	49.04	57.38	76.54
ACTMat	89.30	96.22	84.44	95.67	74.72	63.46	70.96	82.11
Data-dependent merging								
OBIM	64.76	89.74	55.48	51.29	56.81	60.58	51.22	61.41
RegMean	90.18	96.54	86.35	95.57	79.21	51.92	68.11	81.13
w/ OBMerge (LS2)	91.79	96.65	87.37	95.51	82.90	66.35	69.53	84.30 (+3.17)
w/ OBMerge (GS1)	91.33	96.98	87.88	95.65	82.47	58.65	70.24	83.31 (+0.80)
RegMean++	90.58	96.87	86.61	95.43	79.85	60.58	69.69	82.80
w/ OBMerge (LS2)	92.05	96.87	86.73	95.31	82.42	68.27	70.32	**84.57** (+1.77)
w/ OBMerge (GS1)	91.56	97.08	88.27	95.51	81.88	61.54	69.93	83.68 (+0.78)
w/ OBMerge (LS2*)	92.10	96.98	86.73	95.34	82.36	67.31	70.40	84.46 (+1.66)

Table 11: **Llama-3.2-3B.** Scores on five MergeBench domains. Parentheses: gains over initial merges (points); bold: best average among merging methods.

Method	Instruction	Math	Multilingual	Coding	Safety	Avg.
Pre-trained	7.76	28.28	46.38	26.77	28.53	27.54
Individual	48.24	69.45	47.69	44.20	76.03	57.12
Parameter-only merging						
AVG	12.01	41.77	47.87	37.35	37.17	35.23
TIES	28.28	50.27	47.81	39.03	39.80	41.04
TSV-M	23.48	54.66	47.98	40.25	47.68	42.81
Iso-C	10.35	47.38	47.74	36.02	40.48	36.40
Iso-CTS	6.47	43.75	47.55	34.52	37.05	33.87
ACTMat	27.91	60.58	47.92	37.76	47.24	44.28
Data-dependent merging						
OBIM	34.20	50.11	47.96	41.59	40.99	42.97
RegMean	15.53	56.33	48.02	40.71	45.94	41.31
w/ OBMerge (LS2)	19.22	58.07	47.91	40.62	50.84	43.34 $(+2.03)$
w/ OBMerge (GS1)	18.67	56.94	48.19	40.57	48.57	42.59 $(+1.09)$
RegMean++	21.63	56.79	48.02	41.41	49.25	43.42
w/ OBMerge (LS2)	25.32	58.98	48.02	41.81	53.70	45.57 $(+2.15)$
w/ OBMerge (GS1)	24.03	56.79	47.97	40.96	51.85	44.32 $(+0.97)$
w/ OBMerge (LS2*)	27.36	58.83	47.94	41.82	53.24	**45.84** $(+2.42)$

C.4 OBMerge with Other Merging Operators

Table 12 reports results for OBMerge with LS2 using different merging operators on ViT-B/32 and ViT-L/14. With RegMean, accuracy remains stable across iterations and is higher at the final iteration than at the initial merge on both models. This may be because both RegMean and LS2 account for output perturbations: RegMean minimizes layerwise reconstruction error, while LS2 estimates the change in local reconstruction loss caused by parameter replacement. ACTMat, Iso-C, and Iso-CTS improve in early iterations. By the final iteration, however, ACTMat and TSV-M have slightly lower accuracy than their initial merges, while Iso-C and Iso-CTS collapse to below 5% accuracy on both models. These results suggest that further research is needed on replacement schedules and how to choose merging operators for OBMerge.

Table 12: Average test accuracy (%) on eight vision tasks across iterations of OBMerge with LS2. The listed methods serve as merging operators. Bold marks row maxima for each method.

Model	Method	Iteration									
		0	1	2	3	4	5	6	7	8	9
ViT-B/32	ACTMat	82.96	83.18	83.82	84.06	**84.07**	83.66	82.68	81.92	81.01	80.40
	TSV-M	**83.04**	82.97	**83.04**	82.96	82.76	82.31	82.22	81.86	81.72	81.44
	Iso-C	80.47	80.62	81.69	**82.29**	60.00	7.13	4.55	4.35	4.07	4.05
	Iso-CTS	78.90	79.09	80.02	**81.82**	75.62	49.93	4.72	4.46	4.14	4.46
	RegMean	82.74	82.74	82.73	82.78	82.84	83.01	83.20	83.68	83.83	**83.95**
ViT-L/14	ACTMat	92.15	92.16	**92.24**	92.12	91.97	91.55	91.03	90.54	90.02	89.61
	TSV-M	90.61	90.65	90.72	**90.76**	90.68	90.59	90.49	90.39	90.35	90.56
	Iso-C	90.76	90.82	91.14	**91.41**	75.63	4.42	4.27	4.85	4.79	4.71
	Iso-CTS	90.43	90.44	90.72	**91.51**	90.89	5.84	4.74	4.39	4.08	4.54
	RegMean	90.54	90.54	90.55	90.56	90.61	90.73	90.91	91.11	91.28	**91.20**

D SYNTHETIC CALIBRATION DATA: PIPELINE AND PROMPTS

D.1 GENERATION AND CALIBRATION PIPELINE

We implement the 2×2 design of Section 3.4 in five steps.

1. **Description.** From a public task and dataset summary, alone or with 256 real examples from the fine-tuning split, the LLM writes `task_info` and `dataset_info` descriptions; only these are passed to later LLM calls.
2. **Instruction template.** Generated from the description or reused from fine-tuning, the template specifies the example fields, the answer field, and the rendering format.
3. **Example generation.** The LLM generates JSON records from the description and template. No real examples are provided to the LLM in this step.
4. **Reference-overlap filtering.** Our combined rule rejects a record if its ROUGE-L F1 (Wang et al., 2023) or exact word 5-gram Jaccard similarity (Lee et al., 2022) with any reference for its task (Appendix D.2) is at least 0.7.
5. **Rendering and calibration.** For CSR, we sample 512 calibration examples per task from the first 650 retained records, with a maximum sequence length of 256 tokens. For MergeBench, real and synthetic calibration both use 512 examples per domain, with a maximum sequence length of 2048 tokens. Synthetic examples are sampled from pools of 1100 per domain. RegMean, RegMean++, and their OBMerge variants share the same calibration pools.

D.2 REFERENCE DATA FOR OVERLAP CHECKS

Table 13: CSR overlap references, with row counts of the project splits. Configurations are in parentheses.

Task	Hugging Face source (configuration)	Train	Valid.	Test
PAWS	`google-research-datasets/paws` (`labeled_final`)	49,401	8,000	8,000
QASC	`allenai/qasc`	6,507	1,627	926
QuaRTz	`allenai/quartz`	2,696	384	784
StoryCloze	`MoE-UNC/story_cloze`	1,496	375	1,871
WikiQA	`microsoft/wiki_qa`	20,360	2,733	6,165
WinoGrande	`allenai/winogrande` (`winogrande_xl`)	32,318	8,080	1,267
WSC	`aps/super_glue` (`wsc.fixed`)	443	111	104

Table 14: MergeBench overlap reference counts; multilingual benchmarks are in Table 15. Columns use upstream split names; "–": split not provided by the dataset; *: evaluation split.

Domain	Source (configuration)	Train	Valid.	Test
Instruction	`MergeBench/instruction_val`	1,000	–	–
Math	`MergeBench/math_val`	1,000	–	–
Multilingual	`MergeBench/multilingual_val`	1,000	–	–
Coding	`MergeBench/coding_val`	1,000	–	–
Safety	`MergeBench/safety_val`	1,000	–	–
Instruction	`google/IFEval`	541*	–	–
Math	`openai/gsm8k` (`main`)	7,473	–	1,319*
Coding	`evalplus/humanevalplus`	–	–	164*
Coding	`evalplus/mbppplus`	–	–	378*
Safety	`allenai/wildguardmix` (`wildguardtrain`)	86,759	–	–
Safety	`allenai/wildguardmix` (`wildguardtest`)	–	–	1,725*
Safety	HarmBench text behaviors	–	80	320*
Safety	DoAnythingNow (processed evaluation records)	–	–	300*
Safety	XSTest	–	–	450*

Table 15: Okapi overlap reference counts. Repositories use the `alexandrainst/` prefix, with one configuration per language; "–": split not provided by the dataset; *: evaluation split.

Repository	Language	Train	Val	Test
`m_mmlu`	`de`	277	1,449	13,258*
	`es`	272	1,465	13,334*
	`fr`	270	1,417	13,091*
	`ru`	279	1,450	13,007*
`m_arc`	`de`	1,116	298	1,169*
	`es`	1,118	297	1,170*
	`fr`	1,118	298	1,169*
	`ru`	1,118	297	1,169*
`m_hellaswag`	`de`	–	9,368*	–
	`es`	–	9,374*	–
	`fr`	–	9,338*	–
	`ru`	–	9,272*	–

D.3 Full Prompts

The prompts used in our experiments are shown below. Text in <. . .> is replaced with the input for each task. In P1B, each of the 256 real examples starts with `Sample N`, followed by one `field: value` pair per line. In P3, `fields` lists the field names and descriptions, `input_format` gives the template, and `answer_field` gives the name of the answer field.

P1: Description from a public summary

Public summary of the task and dataset:

`<public task and dataset summary>`

Return only a JSON object with two string fields:
`"task_info"`: one sentence describing the task requirements.
`"dataset_info"`: one to three sentences describing the data characteristics.

Use your own words so the description can guide generation of new examples. Cover subject matter, language, input/output structure, wording, and answer length, but include only details explicitly stated in the summary. Briefly note missing task or data characteristics in the corresponding field; do not infer details or pad the description. Omit dataset sizes, split counts, benchmark scores, citations, licensing information, and dataset, paper, organisation, or model names and URLs. Do not reproduce example instances.

P1B: Description with real examples

Public summary of the task and dataset, followed by real samples:

`<public task and dataset summary>`

`<256 real samples serialized as Sample N followed by field: value lines>`

Return only a JSON object with two string fields:
`"task_info"`: one sentence describing the task requirements.
`"dataset_info"`: one to three sentences describing the data characteristics.

Use your own words so the description can guide generation of new examples. Describe characteristics shared across the samples, using the summary as context. Cover subject matter, language, input/output structure, wording, and answer length only when stated in the summary or consistently observed in the samples. Prefer the samples if they conflict with the summary, and briefly note variation among samples. Do not infer unsupported details or pad the description. Omit dataset sizes, split counts, benchmark scores, citations, licensing information, and dataset, paper, organisation, or model names and URLs. Do not copy phrases or examples, or include names, numbers, topics, entities, libraries, or specific problems that occur in only one or a few samples.

P2: Instruction template generation

Task description: `<task_info>`
Dataset description: `<dataset_info>`

Design an instruction template specifying the JSON fields of a data sample and its plain-text format for a language model. Return only a JSON object with:
`"fields"`: an object mapping lowercase field names to short descriptions, including the answer field;
`"answer_field"`: the name of the answer field;
`"input_format"`: a string using every field as a {`field`} placeholder and ending with the answer placeholder.

The field named by `"answer_field"` must contain the complete target output specified by the task and dataset descriptions. All other fields must contain only information provided as input to the task, without any part of the target response.

Include every task-required input described above; do not omit any information needed to determine the target output.

Do not include example content, Markdown, or additional keys.

P3: Calibration example generation

Task description: `<task_info>`
Dataset description: `<dataset_info>`
JSON fields (field names and descriptions; generate string values):
`<fields JSON object>`
Instruction template (rendering reference only; do not fill or return it):
`"<input_format>"`

Return each example as a JSON object with exactly the listed fields, all filled with string values. Each value must contain only the content described for that field. The application will insert these values into the instruction template exactly once. Do not copy the surrounding template text, headers, or unfilled placeholders into field values, and do not return a fully rendered example in a field. Put the generated answer or response in the `"<answer_field>"` field. Do not embed a completed instruction/response exchange in an input field or append the generated response to it. Keep task-required answer choices in their designated fields.

Generate exactly 50 distinct examples, following the language or languages specified in the task and dataset descriptions. Each example must be at most 2048 tokens in total, including all input fields and the answer; when a template is provided, include its rendered text in this limit. Keep each example concise and complete. Vary topics, settings, names, and wording within the described task. Do not repeat examples within this batch. Check answers internally. Do not copy dataset examples. Return only a JSON object with one key, `"examples"`, whose value is an array of the generated records, without surrounding Markdown fences or commentary. Preserve task-required formatting inside field values.

D.4 Per-Task CSR Results with Synthetic Calibration

Tables 16 and 17 break down the CSR averages of Table 3 by task. OBMerge uses LS2 with the schedule $0.1, \ldots, 0.9$ and is reported at $\rho_T = 0.9$. Real-data references are RegMean and RegMean++ without OBMerge.

Table 16: Qwen3-0.6B: per-task CSR accuracy (%). Parentheses: gains over initial merges (percentage points); bold: higher average in each pair.

Template	Description preparation	Method	PAWS	QASC	QuaRTz	WikiQA	SCloze	WSC	WinoG	**Avg.**
Real-data reference		RegMean	91.06	95.90	84.95	95.57	89.74	64.42	58.72	82.91
		RegMean++	91.19	96.00	85.59	95.56	89.42	65.38	59.98	83.30
LLM-generated	Summary only	RegMean	84.54	79.37	65.05	94.05	77.98	63.46	50.12	**73.51**
		w/ OBMerge	81.47	82.29	65.69	94.23	74.72	63.46	50.28	73.16 (−0.35)
		RegMean++	84.45	78.29	73.72	94.94	84.07	64.42	49.64	75.65
		w/ OBMerge	86.39	82.61	74.62	94.71	86.69	63.46	50.04	**76.93** (+1.28)
	Summary with example guidance†	RegMean	87.21	84.77	69.39	93.77	77.39	69.23	50.67	**76.06**
		w/ OBMerge	86.01	87.80	68.24	92.17	70.44	66.35	51.70	74.67 (−1.39)
		RegMean++	87.79	85.85	70.03	94.66	81.72	67.31	50.12	76.78
		w/ OBMerge	89.64	90.82	71.30	94.84	83.16	58.65	49.96	**76.91** (+0.13)
Fine-tuning	Summary only	RegMean	89.84	93.52	84.06	95.43	90.27	65.38	56.04	82.08
		w/ OBMerge	90.72	93.95	82.14	95.54	89.26	67.31	58.25	**82.45** (+0.37)
		RegMean++	89.83	93.52	84.69	95.51	90.54	67.31	57.77	82.74
		w/ OBMerge	90.80	94.71	84.44	95.39	89.15	67.31	59.12	**82.99** (+0.25)
	Summary with example guidance†	RegMean	89.74	95.25	83.29	95.47	90.06	66.35	56.99	**82.45**
		w/ OBMerge	90.46	94.60	82.14	95.43	89.31	63.46	59.51	82.13 (−0.32)
		RegMean++	89.65	95.36	84.18	95.51	90.17	66.35	58.01	82.75
		w/ OBMerge	90.71	95.68	84.06	95.46	89.42	68.27	60.06	**83.38** (+0.63)

Table 17: Llama-3.2-1B: per-task CSR accuracy (%). Parentheses: gains over initial merges (percentage points); bold: higher average in each pair.

Template	Description preparation	Method	PAWS	QASC	QuaRTz	WikiQA	SCloze	WSC	WinoG	**Avg.**
Real-data reference		RegMean	90.18	96.54	86.35	95.57	79.21	51.92	68.11	81.13
		RegMean++	90.58	96.87	86.61	95.43	79.85	60.58	69.69	82.80
LLM-generated	Summary only	RegMean	67.58	86.07	74.11	93.63	61.46	46.15	50.51	68.50
		w/ OBMerge	75.55	87.58	77.81	94.58	64.35	42.31	51.70	**70.55** (+2.05)
		RegMean++	69.14	85.53	76.53	93.48	63.28	66.35	50.20	72.07
		w/ OBMerge	78.03	85.96	82.14	95.13	69.27	67.31	51.78	**75.66** (+3.59)
	Summary with example guidance†	RegMean	73.80	89.09	76.15	93.41	60.61	50.00	50.59	70.52
		w/ OBMerge	81.84	92.55	79.72	94.73	64.46	44.23	51.70	**72.75** (+2.23)
		RegMean++	76.19	89.96	76.91	93.40	61.68	65.38	50.12	73.38
		w/ OBMerge	84.88	92.66	82.78	95.10	66.65	59.62	52.01	**76.24** (+2.86)
Fine-tuning	Summary only	RegMean	83.08	94.82	84.31	95.34	79.42	66.35	66.46	**81.40**
		w/ OBMerge	87.25	96.76	86.48	95.78	83.59	49.04	68.51	81.06 (−0.34)
		RegMean++	82.35	96.22	85.46	95.54	80.01	59.62	65.67	80.69
		w/ OBMerge	86.28	96.87	87.37	95.78	83.81	53.85	67.80	**81.68** (+0.99)
	Summary with example guidance†	RegMean	83.51	96.00	83.67	95.49	79.37	60.58	66.38	80.71
		w/ OBMerge	87.59	97.08	86.35	95.70	83.59	51.92	69.46	**81.67** (+0.96)
		RegMean++	83.23	96.76	84.06	95.56	80.17	57.69	67.48	80.71
		w/ OBMerge	86.95	97.19	87.37	95.83	83.70	51.92	69.69	**81.81** (+1.10)

D.5 Per-Domain MergeBench Results with Synthetic Calibration

Table 18 gives the per-domain MergeBench scores. Real-data references are from Table 11; scoring follows Appendix C.2.

Table 18: MergeBench (Llama-3.2-3B): per-domain scores. Parentheses: gains over initial merges (points); bold: higher average in each pair.

Template	Description preparation	Method	Instruction	Math	Multilingual	Coding	Safety	**Avg.**
Real-data reference		RegMean	15.53	56.33	48.02	40.71	45.94	41.31
		RegMean++	21.63	56.79	48.02	41.41	49.25	43.42
LLM-generated	Summary only	RegMean	26.06	46.63	47.80	38.73	46.61	41.16
		w/ OBMerge	26.99	48.60	47.80	38.35	48.97	**42.14** (+0.97)
		RegMean++	26.25	46.40	47.81	38.26	49.98	41.74
		w/ OBMerge	26.99	47.99	47.97	37.46	51.46	**42.37** (+0.63)
	Summary with example guidance†	RegMean	29.76	52.54	47.95	41.02	50.25	44.30
		w/ OBMerge	34.38	53.60	48.04	41.03	52.49	**45.91** (+1.61)
		RegMean++	30.68	50.80	48.18	41.18	50.31	44.23
		w/ OBMerge	34.20	52.54	48.27	41.81	52.24	**45.81** (+1.58)
Fine-tuning	Summary only	RegMean	16.27	45.11	47.91	41.32	45.05	39.13
		w/ OBMerge	19.78	44.50	47.86	41.14	51.26	**40.91** (+1.78)
		RegMean++	22.37	45.34	47.94	42.15	52.14	41.99
		w/ OBMerge	24.58	46.17	47.95	42.55	54.27	**43.11** (+1.12)
	Summary with example guidance†	RegMean	20.52	50.80	47.83	41.04	47.20	41.48
		w/ OBMerge	23.11	51.33	47.80	41.83	51.05	**43.02** (+1.54)
		RegMean++	28.84	51.86	47.90	41.47	51.61	44.33
		w/ OBMerge	31.79	52.16	47.95	42.79	53.69	**45.68** (+1.34)

D.6 Coefficient Selection with Synthetic Calibration Data

We compare three ways to set the merging coefficient of TIES, Iso-C, Iso-CTS, and TSV-M on the seven CSR tasks with Qwen3-0.6B and Llama-3.2-1B (Table 19): **Default** fixes it to 1; **Real calibration** and **Synthetic calibration** select it on real or synthetic examples, respectively. Default results are from the runs used in this coefficient-selection experiment and may differ slightly from those in Table 2.

Calibration data and scoring. Real calibration uses 100 examples per task sampled from the real validation partitions. Synthetic calibration uses only the *public summary* + 256 *real examples* + *LLM-generated template* condition. It provides enough examples to evaluate all seven tasks. Other conditions may omit fields needed to compute accuracy. For example, QASC records generated with the fine-tuning template lack the alternative answer choices required by the multiple-choice evaluator.

Search protocol. For each method, let $\boldsymbol{\theta}_{\mathrm{m}}$ denote its merged checkpoint at coefficient 1. We evaluate

$$\boldsymbol{\theta}_{\mathrm{m}}(\lambda) = \boldsymbol{\theta}_{\mathrm{base}} + \lambda\big(\boldsymbol{\theta}_{\mathrm{m}} - \boldsymbol{\theta}_{\mathrm{base}}\big), \qquad \lambda \in \{0.1, 0.2, \ldots, 3.0\}. \tag{20}$$

All other merging hyperparameters stay fixed.

Table 19: CSR coefficient selection: selected λ and average accuracy (%) on seven full real test splits. Δ: test accuracy gain over Default ($\lambda = 1$) using synthetic calibration data (percentage points).

Model	**Method**	**Default**	**Real calib.**		**Synthetic calib.**		**Δ vs. Default**
		Acc.	λ	Acc.	λ	Acc.	Acc.
Qwen3-0.6B	TIES	74.82	1.6	77.80	1.6	77.85	+3.03
	Iso-C	81.17	1.8	83.82	2.5	80.03	−1.14
	Iso-CTS	78.99	2.8	84.22	2.9	84.12	+5.13
	TSV-M	82.43	1.2	82.83	1.6	82.19	−0.24
Llama-3.2-1B	TIES	64.87	1.9	72.46	2.3	72.21	+7.34
	Iso-C	79.27	1.7	83.44	1.2	82.48	+3.21
	Iso-CTS	76.17	2.2	82.09	2.1	82.39	+6.22
	TSV-M	83.90	2.1	83.37	1.7	84.75	+0.85
Average		**77.70**		**81.25**		**80.75**	**+3.05**

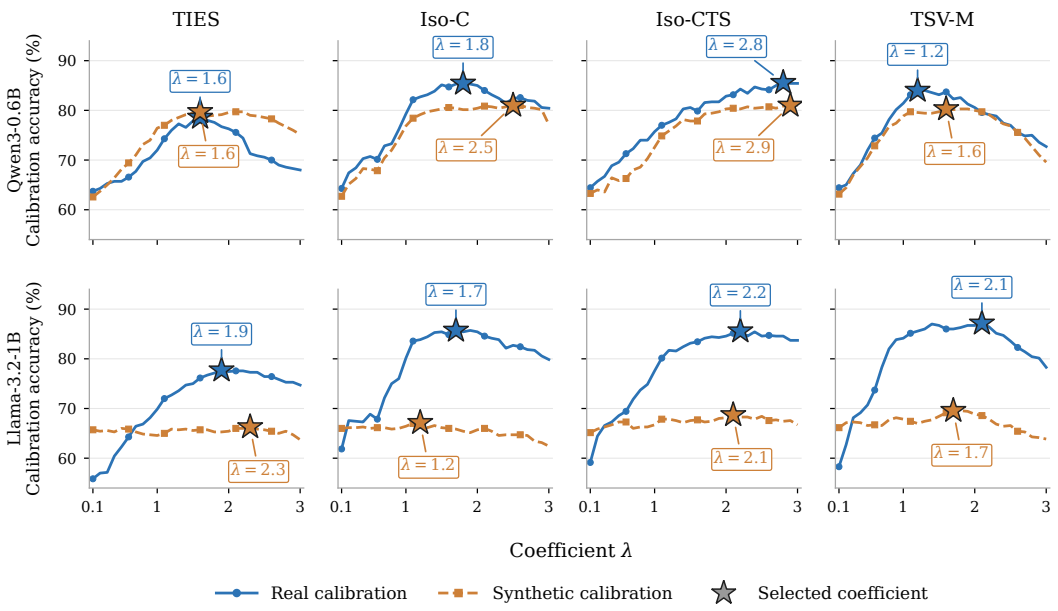


Figure 4: Average CSR calibration accuracy (%) versus merging coefficient λ on real and synthetic data. Stars mark the selected coefficients.

Synthetic calibration raises average test accuracy from 77.70% with Default ($\lambda = 1$) to 80.75%, a gain of 3.05 percentage points across the eight settings (Table 19). The average test accuracy with synthetic calibration is only 0.50 percentage points below that with real calibration (81.25%) and exceeds that with the default setting by 0.85–7.34 percentage points in six of the eight settings. The two decreases are on Qwen3-0.6B: 1.14 points for Iso-C and 0.24 points for TSV-M. Figure 4 shows the coefficient search curves. Real and synthetic calibration generally select similar values of λ.