FEATCAL: Feature Calibration for Post-Merging Models

Yanggan Gu[1*] **Shuo Cai**[1*] **Zihao Wang**[2*] **Wenjun Wang**[1*] **Yuanyi Wang**[1]
Pengkai Wang[1] **Sirui Huang**[1] **Su Lu**[1] **Jianmin Wu**[1,4] **Hongxia Yang**[1,3,4†]
[1]The Hong Kong Polytechnic University (PolyU)
[2]The Chinese University of Hong Kong
[3]Research Institute of Generative Artificial Intelligence [4]InfiX.ai
yanggangu@outlook.com
Code: github.com/egangu/featcal

Abstract

Model merging combines task experts into one model and avoids joint training, retraining, or deploying many expert models, but the merged model often still underperforms task experts. We study this performance gap through *feature drift*, the difference between features produced by the merged model and by the expert on the same input. Our theory decomposes this drift into upstream propagation and local mismatch, tracks how it propagates and combines through later layers in forward order, and links final feature drift to output drift. This view motivates FEATCAL, which uses a small calibration set to calibrate the merged model weights layer by layer in forward order, reducing feature drift while staying close to merged weights and preserving the benefits of model merging. FEATCAL uses an efficient closed-form solution to update model weights, with no gradient descent, iterative optimization, or extra modules. On the main CLIP and GLUE benchmarks, FEATCAL beats Surgery and ProbSurgery, the closest post-merging calibration baselines: 85.5% vs. 77.0%/78.8% on CLIP-ViT-B/32 Task Arithmetic (TA) and 85.2% vs. 83.7%/82.2% on FLAN-T5-base GLUE. On CLIP-ViT-B/32, 8 examples per task reach 82.9%, and 256 examples per task take 53 seconds, about 4x faster than both baselines, showing better sample efficiency and lower calibration cost.

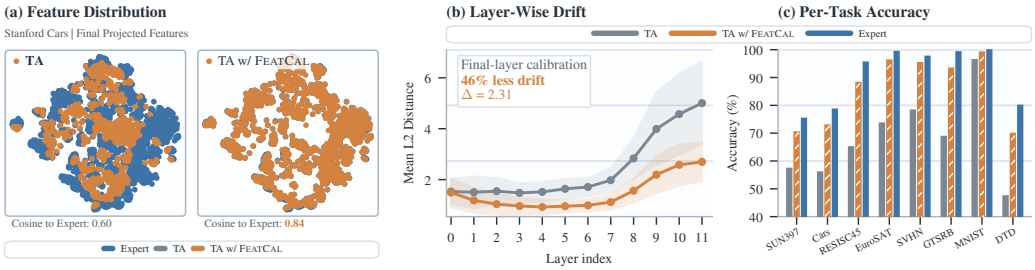


Figure 1: *Feature drift* after Task Arithmetic (TA) merging and FEATCAL calibration in CLIP-ViT-B/32. Panels (a,b) use Stanford Cars: FEATCAL moves features toward expert features, raises their mean cosine similarity from 0.60 to 0.84, and reduces mean L2 feature drift, with 46% less final-layer drift. Panel (c) reports per-task accuracy in the 8-task setting. App. B gives full 8-task feature views.

*Equal contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Model merging composes task experts into one model, avoiding joint training, retraining, or deploying a separate model for each task [1–10]. However, the merged model often still underperforms the experts it is intended to combine. This leaves a clear performance gap after merging in practice.

We analyze this gap through *feature drift*, the difference between features of the merged model and the task expert on the same input sample. Our layer by layer analysis decomposes this drift into upstream propagation and local mismatch at expert input features, then shows how local mismatches propagate through later layers in *forward order* and combine into final feature drift. We further analyze how final feature drift reaches the model output and becomes output drift, explaining how feature drift can affect output scores. Fig. 1 illustrates this behavior on CLIP-ViT-B/32 merged with Task Arithmetic (TA) [3], where final features move away from the Stanford Cars expert features and drift appears across multiple layers of the network after TA merging.

Surgery and ProbSurgery [11, 12] are related post-merging calibration methods: they identify feature drift at the final layer and train extra modules for calibration. Other intervention methods point to the same lesson: expert signals can help, but current methods often rely on task specific intervention parameters, extra modules at inference, or iterative optimization [13, 14]. These choices make calibration less direct for an already merged model and can leave the final model with an auxiliary inference path. This motivates efficient, direct calibration that preserves inference speed.

FEATCAL uses our drift analysis as a design cue for calibrating the merged model. It uses a forward-order schedule suggested by the propagation view and calibrates the model layer by layer. We introduce a regularization term that keeps the calibrated model close to the merged model, which helps preserve the benefits of model merging and reduce overfitting to the small calibration set. The resulting objective has an efficient closed form solution, so calibration needs no gradient descent, iterative optimization, or extra modules at inference. We further introduce feature interpolation and anchor regularization to balance expert signals with the merged model and improve performance.

Empirically, Fig. 1 shows the practical effect: FEATCAL moves merged features toward expert features, reduces feature drift, and raises per-task accuracy after TA merging. On CLIP-ViT-B/32 8-task TA, it reaches 85.5% versus 77.0%/78.8% for Surgery/ProbSurgery, and on FLAN-T5-base GLUE it reaches 85.2% versus 83.7%/82.2%. The same trend holds on CLIP-ViT-L/14 WUDI, FLAN-T5-large, and MergeBench Llama-family LLM merging, where FEATCAL improves TA by $+2.0$/$+2.3$ average points on 3B/8B models. On CLIP-ViT-B/32 TA, 8 examples per task reach 82.9%, and calibration with 256 examples per task takes 53 seconds, about 4x faster than both baselines under the same calibration protocol.

We summarize the main contributions of this work as follows:

❶ We develop a theory of feature drift after merging, with an exact decomposition into local mismatch and upstream propagation, forward order propagation, and a link to output drift.

❷ We introduce FEATCAL, which efficiently calibrates merged model weights in forward order with closed form updates, without gradient descent, architecture changes, or extra modules at inference.

❸ We validate FEATCAL on CLIP, FLAN-T5, and MergeBench LLM benchmarks, where it outperforms related post-merging calibration baselines while using fewer samples and lower calibration cost without adding inference-time modules.

2 Related Work

Model Merging. Model merging and knowledge fusion combine capabilities from multiple task experts [1–10, 15–20]. Methods based on feature statistics or feature drift, such as RegMean, RegMean++, and LOT Merging, are closer in mechanism: they derive layer updates during merging from feature statistics, regression objectives, or an explicit feature drift objective [21–23]. These methods define how to build the merged model. In contrast, FEATCAL treats that model as the starting point, uses a small calibration set and task experts for post-merging calibration, and controls calibration strength through regularization. This stage separation matters because calibration must work with the feature drift left by a chosen merger instead of changing the merge rule itself. It helps preserve the benefits of model merging while reducing the risk of overfitting to calibration data.

Post-Merging Feature Calibration. Representation Surgery and follow-up methods operate on merged-model features through task-specific plugins, deeper interventions, probabilistic feature-drift modeling, or parameter-efficient modules [11, 13, 12, 14]. These closest post-merging alternatives establish that expert-guided feature calibration is useful. FEATCAL differs in parameterization and deployment: instead of learning or deploying auxiliary intervention modules, it folds the calibration into the original linear module weights through closed-form regularized updates, leaving a single architecture-preserving model at inference time rather than an auxiliary intervention path.

3 Post-Merging Feature Drift: Problem Formulation and Properties

Before introducing FEATCAL, we formalize post-merging feature drift and show how local mismatch is defined at expert input features, then propagated and combined in forward order across depth.

3.1 Layer-Wise Feature Drift

We consider N task experts $\{M_i^{\exp}\}_{i=1}^N$ and a merged model M^{mer}. As in standard weight-space merging, experts are fine-tuned from a common pretrained base. All models share the same architecture and contain L layers. Let $\mathcal{D}_i$ be the data distribution of task i. For layer ℓ, $f_{i,\ell}^{\exp}$ and f_ℓ^{mer} denote the corresponding layer functions of the task expert and the merged model, respectively.

The merged encoder is shared across tasks; task dependence enters through the input $x \sim \mathcal{D}_i$ and the corresponding expert.

For an input sample x from task i, define the expert and merged layer output features recursively as

$$\begin{aligned} h_{i,0}^{\exp}(x) &= x, \quad h_{i,\ell}^{\exp}(x) = f_{i,\ell}^{\exp}(h_{i,\ell-1}^{\exp}(x)), \\ h_0^{\mathrm{mer}}(x) &= x, \quad h_\ell^{\mathrm{mer}}(x) = f_\ell^{\mathrm{mer}}(h_{\ell-1}^{\mathrm{mer}}(x)), \end{aligned} \qquad \ell = 1, \dots, L. \tag{1}$$

Definition 1 (Layer-wise feature drift)**.** For task i, sample x, and layer ℓ, the layer-wise feature drift is the difference between the merged-model feature and the corresponding task-expert feature at that layer:

$$e_{i,\ell}(x) = h_\ell^{\mathrm{mer}}(x) - h_{i,\ell}^{\exp}(x). \tag{2}$$

This pointwise drift signal is the object propagated in forward order in the analysis below.

3.2 Local Mismatch and Drift Propagation

Proposition 1 (Exact layer-wise drift decomposition)**.** *For every task i, sample x, and intermediate layer ℓ, the drift decomposes as follows. By Eq. (1), $p_{i,1}(x) = 0$.*

$$e_{i,\ell}(x) = \underbrace{f_\ell^{\mathrm{mer}}\left(h_{i,\ell-1}^{\exp}(x)\right) - f_{i,\ell}^{\exp}\left(h_{i,\ell-1}^{\exp}(x)\right)}_{\text{local mismatch } m_{i,\ell}(x)} + \underbrace{f_\ell^{\mathrm{mer}}\left(h_{\ell-1}^{\mathrm{mer}}(x)\right) - f_\ell^{\mathrm{mer}}\left(h_{i,\ell-1}^{\exp}(x)\right)}_{\text{upstream-drift propagation } p_{i,\ell}(x)}. \tag{3}$$

Proof. The algebraic identity and its regularity details are deferred to App. C.

Interpretation. Prop. 1 decomposes layer-wise feature drift into two terms: local mismatch and upstream-drift propagation. The local mismatch $m_{i,\ell}(x)$ measures the feature mismatch caused by the difference between the merged and expert layer maps at the same expert input feature. The propagation term $p_{i,\ell}(x)$ measures how feature drift inherited from earlier layers changes the input feature of layer ℓ and is then carried into the output feature of this layer.

When f_ℓ^{mer} is continuously differentiable on a neighborhood of the segment between $h_{i,\ell-1}^{\exp}(x)$ and $h_{\ell-1}^{\mathrm{mer}}(x)$, define the path-averaged local sensitivity operator as

$$A_{i,\ell}(x) = \int_0^1 J_u f_\ell^{\mathrm{mer}}\Big(h_{i,\ell-1}^{\exp}(x) + t\, e_{i,\ell-1}(x)\Big)\, dt. \tag{4}$$

Proposition 2 (Layer-wise propagation of local mismatch)**.** *Fix a task i, sample x, and layers $1, \dots, L$. Suppose that, for each layer ℓ, f_ℓ^{mer} is continuously differentiable on an open neighborhood containing the segment between $h_{i,\ell-1}^{\mathrm{exp}}(x)$ and $h_{\ell-1}^{\mathrm{mer}}(x)$. Let $A_{i,\ell}(x)$ denote the corresponding path-averaged local sensitivity operator defined in Eq. (4). Then the drift obeys the layer-by-layer recursion*

$$p_{i,\ell}(x) = A_{i,\ell}(x)e_{i,\ell-1}(x), \qquad e_{i,\ell}(x) = A_{i,\ell}(x)e_{i,\ell-1}(x) + m_{i,\ell}(x). \tag{5}$$

Since $e_{i,0}(x) = 0$, the final feature drift is

$$\begin{aligned} e_{i,L}(x) &= \sum_{\ell=1}^{L} P_{i,\ell\to L}(x)m_{i,\ell}(x), \\ P_{i,\ell\to L}(x) &= A_{i,L}(x)A_{i,L-1}(x)\cdots A_{i,\ell+1}(x), \qquad P_{i,L\to L}(x) = I. \end{aligned} \tag{6}$$

The product composes compatible local maps. See Eq. (22) for the s-to-t form.

Proof. The local sensitivity derivation and unrolled expansion are deferred to App. C.

Interpretation. Prop. 2 gives a simple forward-order view: local mismatches arise at individual layers, and their induced feature drift propagates through downstream layers to the final layer. Thus, final feature drift is a downstream combination of local mismatches from different layers. For residual networks, we further show that residual paths carry upstream feature drift through the skip connection. The drift can also grow under specific conditions. See App. D.

3.3 From Feature Drift to Output Drift

Definition 2 (Output drift for task scores)**.** For the merged model M^{mer}, let $z_i^{\mathrm{exp}}(x)$ and $z_i^{\mathrm{mer}}(x)$ be the expert and merged task scores, each represented as a score vector. Let ψ_i^{exp} and ψ_i^{mer} map final features to task scores, with $z_i^{\mathrm{exp}}(x) = \psi_i^{\mathrm{exp}}(h_{i,L}^{\mathrm{exp}}(x))$ and $z_i^{\mathrm{mer}}(x) = \psi_i^{\mathrm{mer}}(h_L^{\mathrm{mer}}(x))$. The post-merging output drift is

$$\Delta z_i^{\mathrm{mer}}(x) = z_i^{\mathrm{mer}}(x) - z_i^{\mathrm{exp}}(x). \tag{7}$$

The index i identifies the task-specific score space and candidate classes; the merged encoder remains shared. In the logit or similarity settings used below, z_i can contain class logits, CLIP candidate scores (scaled similarity scores over a fixed candidate set), or fixed-prefix decoder vocabulary logits.

Proposition 3 (Feature-to-output perturbation)**.** *Suppose ψ_i^{mer} is locally $B_i^{\mathrm{mer}}(x)$-Lipschitz on the segment between $h_{i,L}^{\mathrm{exp}}(x)$ and $h_L^{\mathrm{mer}}(x)$. Here $B_i^{\mathrm{mer}}(x)$ locally bounds how much the merged task score map can change when its final-feature input moves along this segment. Then*

$$\|\Delta z_i^{\mathrm{mer}}(x)\|_2 \le B_i^{\mathrm{mer}}(x)\,\|e_{i,L}(x)\|_2 + \delta_i^{\psi,\mathrm{mer}}(x), \tag{8}$$

where

$$\delta_i^{\psi,\mathrm{mer}}(x) = \left\|\psi_i^{\mathrm{mer}}\left(h_{i,L}^{\mathrm{exp}}(x)\right) - \psi_i^{\mathrm{exp}}\left(h_{i,L}^{\mathrm{exp}}(x)\right)\right\|_2$$

is the score map mismatch. If the task score map is shared, this term is 0.

Proof. This is the perturbation bound proved in Prop. 6.

Interpretation. Prop. 3 shows that final feature drift can propagate to output drift, thus further affect model outputs and task loss. For example, in a language model under a fixed prefix, token probabilities are obtained by applying softmax to output logits. Holding other logits fixed, a larger token logit gives a larger token probability, so output-logit drift can potentially induce probability drift and change the next-token choice. Cross-entropy loss is also tied to the probability assigned to the target token, so probability drift can induce loss drift. Detailed analysis is deferred to App. E.

4 FEATCAL: Feature Calibration for Post-Merging Models

The preceding drift analysis guides the design of a direct calibration procedure for an already merged model. FEATCAL visits layers in forward order, takes a current feature snapshot after earlier layer updates, and solves expert-guided calibration objectives for the modules in that layer with explicit regularization.

4.1 Basic Calibration Objective

Forward-order calibration. The drift analysis in Sec. 3.2 shows that feature drift arises from local mismatch terms that propagate layer by layer in forward order. FEATCAL follows the same order by calibrating the merged model layer by layer: after earlier layers are calibrated, it recollects current calibrated-model features for the next layer before fitting that layer's module objectives. This schedule also reduces mismatch between features used during calibration and features exposed by the deployed calibrated model; App. F formalizes this source/deployed feature mismatch.

Why calibrate linear modules? Under this schedule, each layer provides a shared feature snapshot. Given the cached input features and expert target features for a module in that snapshot, the local surrogate can be defined for any linear module. In practice, FEATCAL applies it to the modules configured for calibration. Linear modules are natural feature-mixing and projection points in the architectures we study, including attention and MLP linear modules. Once their calibrated-model input features are fixed, each linear module gives a tractable regularized regression problem with a closed-form update. The update replaces the existing merged weight, preserving the architecture without gradient descent, adapters, or inference-time modules. For other affine components, including bias parameters and LayerNorm affine parameters, we also design calibration updates, as detailed in Appendix G. In practice, these extra updates give limited gains: on CLIP, their average accuracy improvement is less than 0.5 percentage points. The main gains come from calibrating linear modules.

Linear module feature drift. For a fixed linear module inside the current layer, we define a module-local version of feature drift using the variables available at that module. Let $W_i, W^{\mathrm{mer}}, W^{\mathrm{base}} \in \mathbb{R}^{m\times d}$ be the task-i expert, merged, and base weights for this module. When processing the layer, we cache fixed input feature matrices X_i^{cal} and X_i^{exp} for this module on the same task-i calibration samples. Here X_i^{cal} is produced by the prefix-calibrated model and is therefore the deployed input source, while X_i^{exp} is the corresponding task-expert feature matrix.

Following the layer-wise feature drift definition in Eq. (2), we define the linear module feature drift of a candidate calibrated weight W by

$$e_i^{\mathrm{linear}}(W) = WX_i^{\mathrm{cal}} - W_iX_i^{\mathrm{exp}}. \tag{9}$$

As in the layer-wise analysis, this module-level drift can be interpreted as a combination of module-local mismatch and upstream-drift propagation.

Basic calibration objective. With input features fixed, the per-module calibration objective minimizes the overall linear module feature-drift error with a merged-weight penalty:

$$W^\star = \arg\min_{W\in\mathbb{R}^{m\times d}} \sum_{i=1}^{N} \|WX_i^{\mathrm{cal}} - W_iX_i^{\mathrm{exp}}\|_F^2 \;+\; \lambda_{\mathrm{mer}}\|W - W^{\mathrm{mer}}\|_F^2. \tag{10}$$

The quadratic penalty controls how far a single module update can move from the merged weights. This objective is a tractable module-local surrogate for reducing linear module feature drift, rather than an exact objective for end-to-end task risk or all-layer feature drift.

4.2 Feature Interpolation for Calibration Targets

Under the forward-order schedule, each layer's calibration should focus on the local mismatch at its current linear module rather than compensate for feature drift caused by upstream layers. Directly using expert input features for calibration can violate this goal. The gap between X_i^{cal} and X_i^{exp} already contains upstream feature drift, so fitting the direct target $W_iX_i^{\mathrm{exp}}$ may force the current linear module to fit drift left by earlier layers. We therefore introduce an interpolated target feature:

$$X_i^{\mathrm{tgt}} = \alpha X_i^{\mathrm{exp}} + (1-\alpha)X_i^{\mathrm{cal}}, \qquad \alpha\in[0,1]. \tag{11}$$

In Eq. (10), we replace the target term $W_iX_i^{\mathrm{exp}}$ with $W_iX_i^{\mathrm{tgt}}$. This target keeps the expert signal while making the local regression less aggressive under upstream drift.

4.3 Anchor Regularization

During calibration, the objective should use targets formed from expert features while keeping the update tied to the merged weights. The base model provides another useful reference. It is pretrained on large data before task specialization and can contain knowledge that a small calibration set does not cover. To use this reference, we include the base weight in the regularization term. The coefficients below let us control how calibration uses the merged and base references. For a linear module, we define the anchor weight as

$$W^{\mathrm{anc}} = \rho W^{\mathrm{mer}} + (1-\rho) W^{\mathrm{base}}, \qquad \rho \in \mathbb{R}. \tag{12}$$

The coefficient ρ controls the anchor used by the quadratic penalty.

Combining the target interpolation in Eq. (11) and the anchor in Eq. (12) gives the practical objective:

$$W^{\star} = \arg\min_{W \in \mathbb{R}^{m\times d}} \sum_{i=1}^{N} \|W X_i^{\mathrm{cal}} - W_i X_i^{\mathrm{tgt}}\|_F^2 \; + \; \lambda \|W - W^{\mathrm{anc}}\|_F^2. \tag{13}$$

The first term uses the interpolated target from Eq. (11) to fit the expert feature signal without using the raw expert input feature as a hard target. The second term uses the anchor from Eq. (12) to keep the update close to the chosen reference, with $\lambda > 0$ controlling the regularization strength.

4.4 Task-Wise Scale Normalization and Closed-Form Solution

This subsection describes the update applied after feature collection. At each forward-order layer, FEATCAL first caches current calibrated-model features and expert features for the modules calibrated in that layer. The cached features are then used to compute each linear module update separately, and the layer parameters are loaded after the layer's updates are formed. Thus the statistics below are module-local, even though feature collection is organized by layer.

For a fixed linear module in the current layer, let n denote the calibration sample count for each task at this module. The $1/n$ factors form per-task empirical moments and prevent each task contribution from scaling directly with the sample count. We summarize task i by the empirical feature statistics

$$G_i = \frac{1}{n} X_i^{\mathrm{cal}} X_i^{\mathrm{cal}\top} \in \mathbb{R}^{d\times d}, \qquad C_i = \frac{1}{n} X_i^{\mathrm{tgt}} X_i^{\mathrm{cal}\top} \in \mathbb{R}^{d\times d}. \tag{14}$$

Here, G_i is the input second moment and C_i is the target-input cross moment.

The objective in Eq. (13) is a matrix-valued ridge regression problem [24, 25]. To reduce scale sensitivity, we use stabilized inverse task weights with $\epsilon > 0$,

$$\nu_i = \max\{\|G_i\|_F, \epsilon\}, \qquad \omega_i = \nu_i^{-1}, \tag{15}$$

With these stabilized task weights fixed, the module-wise objective becomes

$$W^{\star} = \arg\min_{W \in \mathbb{R}^{m\times d}} \sum_{i=1}^{N} \frac{\omega_i}{n} \|W X_i^{\mathrm{cal}} - W_i X_i^{\mathrm{tgt}}\|_F^2 \; + \; \lambda \|W - W^{\mathrm{anc}}\|_F^2. \tag{16}$$

The corresponding stationary condition for this quadratic objective is

$$W\Big(\sum_{i=1}^{N} \omega_i G_i + \lambda I_d\Big) = \sum_{i=1}^{N} \omega_i W_i C_i + \lambda W^{\mathrm{anc}}, \tag{17}$$

Solving this linear system gives the closed-form update for this module

$$W^{\star} = \Big(\sum_{i=1}^{N} \omega_i W_i C_i + \lambda W^{\mathrm{anc}}\Big)\Big(\sum_{i=1}^{N} \omega_i G_i + \lambda I_d\Big)^{-1}. \tag{18}$$

Because $G_i \succeq 0$, $\omega_i > 0$, and $\lambda > 0$, the inverse is well defined. In implementation, we also add ϵI_d to the solve matrix as a numerical stabilizer.

The complete forward-order calibration procedure is given in App. H.

5 Experiments

5.1 Setup

Benchmarks. We use two public FusionBench settings [26] and one MergeBench LLM setting [27]:

❶ **CLIP image classification.** We use the FusionBench CLIP model merging benchmark with CLIP-ViT-B/32 and CLIP-ViT-L/14 [28]. The primary setting has 8 image tasks: SUN397, Stanford Cars, RESISC45, EuroSAT, SVHN, GTSRB, MNIST, and DTD [29–36]. The 14-task suite adds Flowers102, PCAM, FER2013, Oxford-IIIT Pet, STL10, and CIFAR100 [37–42]. The 20-task suite further adds CIFAR10, Food101, Fashion-MNIST, EMNIST Letters, KMNIST, and Rendered SST2 [42–48]. We follow prior merging protocols [5, 10], report top-1 accuracy and task averages, and defer full per-task extended results to App. I.

❷ **FLAN-T5 text generation.** We evaluate FusionBench FLAN-T5-base and FLAN-T5-large merging on 8 prompted GLUE tasks: CoLA, MNLI, MRPC, QNLI, QQP, RTE, SST-2, and STS-B [49–57, 47, 58]. The base experts are full fine-tuned models, while the large experts use LoRA fine-tuning [59]. We merge task experts and evaluate generated text outputs. We report exact match accuracy except for STS-B, where we report Spearman's ρ, and average the 8 task scores.

❸ **MergeBench LLM merging.** We evaluate Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct in the MergeBench domain-expert setting. The task suite covers mathematics, coding, instruction following, and general knowledge through MATH-500, GSM8K, HumanEval+, MBPP+, IFEval, and ARC-Challenge. HumanEval+ and MBPP+ report pass@1, and each table average is the mean over all 6 reported MergeBench tasks in this LLM setting.

Compared methods. We compare against pre-trained, single-task, and multi-task references when available. For CLIP, upstream mergers include Simple Averaging [1], Task Arithmetic [3], AdaMerging [5], and WUDI-Merging [10]; for FLAN-T5 and MergeBench, we use Task Arithmetic. We apply FEATCAL on top of different upstream mergers and compare with Surgery [11] and ProbSurgery [12] where available. Unless otherwise stated, upstream and baseline hyperparameters follow the FusionBench recipes.

Calibration setup. By default, FEATCAL uses 256 calibration samples per task, calibrates layers in forward order, and applies the linear-weight update in Eq. (18). When enabled, it also applies the bias and LayerNorm affine updates in Appendix G; the main CLIP runs enable both, while the FLAN-T5 runs calibrate linear-module bias parameters but not LayerNorm affine parameters. For the main CLIP accuracy tables, we fix $\lambda = 0.05$, $\rho = 2.0$, and $\alpha = 0.3$. For FLAN-T5, we use $\lambda = 10^{-5}$, $\rho = 2.0$, and $\alpha = 0.6$. For MergeBench, we use $(\lambda, \rho, \alpha) = (10^{-5}, 0.5, 0.15)$ for Llama-3.2-3B-Instruct and $(3.0, 0.5, 0.15)$ for Llama-3.1-8B-Instruct. Test sets are used only for final reporting. Sec. 5.5 includes a compact sensitivity diagnostic for the 8-task TA setting.

5.2 Results on CLIP Models

On B/32 8-task CLIP, FEATCAL raises the 4 upstream averages from 66.3/67.5/82.7/84.5 to 83.2/85.5/88.1/88.8, beating Surgery and ProbSurgery in each block. The gains are largest for the weaker upstream mergers, but FEATCAL also improves AdaMerging and WUDI-Merging, where the merged models are already close to the multi-task reference. This pattern aligns with the feature drift motivation: the same calibration step can recover large lost accuracy and still refine strong merged models without changing the merger itself. The best average, 88.8, is close to the 90.3 task expert average and above the 88.6 multi-task reference. For the TA and WUDI rows in Tab. 2, FEATCAL remains above Surgery and ProbSurgery across all extended averages. On 20-task TA, FEATCAL reaches 79.4 on B/32 and 84.7 on L/14, giving $+18.8$ and $+14.4$ points over TA. Full per-task results are in App. I.

Table 2: CLIP Extended Average Accuracy.

Method	8-Task L/14	14-Task B/32	14-Task L/14	20-Task B/32	20-Task L/14
Pre-trained	64.6	58.8	69.1	55.6	65.6
Fine-tuned (STL)	94.3	90.0	92.8	90.3	93.1
Task Arithmetic	80.5 ↑0.0	66.2 ↑0.0	77.3 ↑0.0	60.6 ↑0.0	70.3 ↑0.0
w/ Surgery	86.0 ↑5.5	76.8 ↑10.	83.8 ↑6.5	75.2 ↑14.	82.4 ↑12.
w/ ProbSurgery	87.6 ↑7.1	78.3 ↑12.	86.1 ↑8.8	77.7 ↑17.	83.3 ↑13.
w/ FEATCAL	91.6 ↑11.	81.5 ↑15.	90.7 ↑13.	79.4 ↑18.	84.7 ↑14.
WUDI-Merging	92.2 ↑0.0	78.7 ↑0.0	88.8 ↑0.0	67.1 ↑0.0	75.8 ↑0.0
w/ Surgery	92.8 ↑0.6	82.5 ↑3.8	90.3 ↑1.5	76.5 ↑9.4	84.9 ↑9.1
w/ ProbSurgery	93.0 ↑0.8	82.7 ↑4.0	90.7 ↑1.9	77.6 ↑10.	86.5 ↑10.
w/ FEATCAL	**93.5** ↑1.3	**86.2** ↑7.5	**91.5** ↑2.7	**83.5** ↑16.	**89.8** ↑14.

Table 1: Multi-task performance of CLIP-ViT-B/32 models on 8 image-classification tasks. All numbers are top-1 accuracy (%). Gray arrows show changes over upstream mergers.

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg.
Pre-trained	63.2	59.8	60.7	46.0	31.6	32.5	48.2	43.9	48.2
Fine-tuned (STL)	75.0	78.3	95.2	99.0	97.3	98.9	99.6	79.7	90.3
Traditional MTL	72.3	76.6	92.2	97.9	95.5	97.7	99.3	77.7	88.6
Simple Averaging	65.4 ↑0.0	62.4 ↑0.0	70.6 ↑0.0	75.7 ↑0.0	64.5 ↑0.0	55.0 ↑0.0	86.3 ↑0.0	50.6 ↑0.0	66.3 ↑0.0
w/ Surgery	67.4 ↑2.0	63.5 ↑1.1	80.5 ↑9.9	94.7 ↑19.	70.8 ↑6.3	79.7 ↑24.	96.8 ↑10.	64.7 ↑14.	77.3 ↑11.
w/ ProbSurgery	69.2 ↑3.8	65.8 ↑3.4	85.8 ↑15.	93.4 ↑17.	70.4 ↑5.9	87.0 ↑32.	96.5 ↑10.	69.1 ↑18.	79.7 ↑13.
w/ FEATCAL	69.7 ↑4.3	70.4 ↑8.0	85.0 ↑14.	95.4 ↑19.	92.6 ↑28.	87.1 ↑32.	97.9 ↑11.	67.4 ↑16.	83.2 ↑16.
Task Arithmetic	57.0 ↑0.0	55.7 ↑0.0	64.7 ↑0.0	73.3 ↑0.0	77.9 ↑0.0	68.5 ↑0.0	96.1 ↑0.0	47.1 ↑0.0	67.5 ↑0.0
w/ Surgery	59.9 ↑2.9	59.9 ↑4.2	76.1 ↑11.	92.4 ↑19.	83.6 ↑5.7	84.9 ↑16.	98.0 ↑1.9	61.1 ↑14.	77.0 ↑9.5
w/ ProbSurgery	60.6 ↑3.6	61.1 ↑5.4	81.0 ↑16.	93.8 ↑20.	85.3 ↑7.4	87.1 ↑18.	97.9 ↑1.8	63.8 ↑16.	78.8 ↑11.
w/ FEATCAL	70.1 ↑13.	72.5 ↑16.	88.1 ↑23.	96.3 ↑23.	95.0 ↑17.	93.0 ↑24.	98.8 ↑2.7	69.8 ↑22.	85.5 ↑18.
AdaMerging	67.9 ↑0.0	71.2 ↑0.0	84.0 ↑0.0	92.3 ↑0.0	87.6 ↑0.0	93.1 ↑0.0	98.2 ↑0.0	66.9 ↑0.0	82.7 ↑0.0
w/ Surgery	69.8 ↑1.9	72.1 ↑0.9	88.7 ↑4.7	95.3 ↑3.0	90.5 ↑2.9	95.7 ↑2.6	98.7 ↑0.5	73.3 ↑6.4	85.5 ↑2.8
w/ ProbSurgery	70.6 ↑2.7	72.9 ↑1.7	90.4 ↑6.4	95.8 ↑3.5	90.2 ↑2.6	95.0 ↑1.9	98.8 ↑0.6	73.8 ↑6.9	85.9 ↑3.2
w/ FEATCAL	**73.0** ↑5.1	**77.4** ↑6.2	91.3 ↑7.3	96.8 ↑4.5	94.2 ↑6.6	96.7 ↑3.6	99.0 ↑0.8	76.4 ↑9.5	88.1 ↑5.4
WUDI-Merging	68.0 ↑0.0	72.5 ↑0.0	85.0 ↑0.0	94.6 ↑0.0	94.8 ↑0.0	95.0 ↑0.0	99.3 ↑0.0	66.7 ↑0.0	84.5 ↑0.0
w/ Surgery	69.0 ↑1.0	71.7 ↓0.8	89.1 ↑4.1	97.2 ↑2.6	95.6 ↑0.8	96.7 ↑1.7	99.3 ↑0.0	72.7 ↑6.0	86.4 ↑1.9
w/ ProbSurgery	69.5 ↑1.5	72.7 ↑0.2	90.6 ↑5.6	97.3 ↑2.7	95.6 ↑0.8	97.1 ↑2.1	99.3 ↑0.0	73.3 ↑6.6	86.9 ↑2.4
w/ FEATCAL	72.2 ↑4.2	76.0 ↑3.5	**93.0** ↑8.0	**98.1** ↑3.5	**96.7** ↑1.9	**97.9** ↑2.9	**99.4** ↑0.1	**77.1** ↑10.	**88.8** ↑4.3

5.3 Results on FLAN-T5 Models

Table 3: FLAN-T5 GLUE generation results. Scores are percentages: exact-match accuracy except STS-B, which reports Spearman's ρ; arrows show changes over Task Arithmetic, and bold marks best non-reference entries.

Model	Method	GLUE Tasks								Avg.
		CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	
Base	Pre-trained	69.1	56.5	76.2	88.4	82.1	80.1	91.2	62.2	75.7
	Fine-tuned (STL)	75.0	83.4	87.5	91.5	85.4	85.9	93.6	88.7	86.4
	Task Arithmetic	70.5 ↑0.0	57.8 ↑0.0	78.4 ↑0.0	90.2 ↑0.0	83.6 ↑0.0	80.5 ↑0.0	92.3 ↑0.0	77.8 ↑0.0	78.9 ↑0.0
	w/ Surgery	70.8 ↑0.3	82.4 ↑24.	82.4 ↑4.0	89.8 ↓0.4	84.2 ↑0.6	83.0 ↑2.5	92.1 ↓0.2	85.2 ↑7.4	83.7 ↑4.8
	w/ ProbSurgery	**82.4** ↑11.	69.1 ↑11.	78.3 ↓0.1	80.6 ↓9.6	**89.8** ↑6.2	83.4 ↑2.9	81.2 ↓11.	**92.5** ↑14.	82.2 ↑3.3
	w/ FEATCAL	72.2 ↑1.7	**82.6** ↑24.	**85.1** ↑6.7	**91.1** ↑0.9	84.7 ↑1.1	**85.2** ↑4.7	**93.0** ↑0.7	87.7 ↑9.9	**85.2** ↑6.3
Large	Pre-trained	73.7	56.6	82.4	91.1	85.5	85.6	94.3	87.5	82.1
	Fine-tuned (STL)	80.2	88.5	89.2	94.4	87.2	91.7	95.2	90.9	89.6
	Task Arithmetic	76.8 ↑0.0	85.4 ↑0.0	85.3 ↑0.0	**94.0** ↑0.0	85.8 ↑0.0	88.1 ↑0.0	95.2 ↑0.0	87.7 ↑0.0	87.3 ↑0.0
	w/ Surgery	76.0 ↓0.8	**87.9** ↑2.5	86.0 ↑0.7	93.9 ↓0.1	86.2 ↑0.4	**89.9** ↑1.8	95.2 ↑0.0	89.0 ↑1.3	88.0 ↑0.7
	w/ ProbSurgery	77.2 ↑0.4	87.6 ↑2.2	85.0 ↓0.3	93.8 ↓0.2	86.0 ↑0.2	88.4 ↑0.3	95.2 ↑0.0	87.8 ↑0.1	87.6 ↑0.3
	w/ FEATCAL	**79.2** ↑2.4	87.8 ↑2.4	**89.0** ↑3.7	93.9 ↓0.1	**86.9** ↑1.1	88.4 ↑0.3	**95.3** ↑0.1	**90.8** ↑3.1	**88.9** ↑1.6

Tab. 3 shows that the GLUE gains hold for both FLAN-T5-base and FLAN-T5-large. For base, FEATCAL improves Task Arithmetic by $+6.3$ average points, exceeds Surgery and ProbSurgery, and improves all 8 tasks, with the largest gains on MNLI and STS-B. For large, where LoRA-based Task Arithmetic is already strong, FEATCAL gives smaller gains but still reaches the best post-TA average. This contrast suggests that feature calibration helps most when the merged generator has clear headroom, while still helping in the stronger setting.

5.4 Results on MergeBench LLMs

Tab. 4 shows that the gains extend to LLM domain expert merging. FEATCAL improves the 6-task average over Task Arithmetic by $+2.0$ points on the 3B model and $+2.3$ points on the 8B model, outperforming Surgery and ProbSurgery in both model blocks. The largest gain is on IFEval for the 8B setting, where FEATCAL improves Task Arithmetic by $+15$. points.

Table 4: MergeBench results on Llama-family models. All Task Arithmetic rows use scale 0.3; arrows show changes over Task Arithmetic within each model block. Bold marks the best non-reference entry for each metric within the corresponding model block.

Model	Method	MATH-500	GSM8K	HumanEval+	MBPP+	IFEval	ARC-C	Avg.
Llama-3.2-3B-Instruct	Base	47.8	72.6	49.4	56.6	67.3	73.1	61.1
	Fine-tuned (STL)	49.0	80.6	54.9	57.1	69.7	73.1	64.1
	Task Arithmetic	44.0 ↑0.0	74.8 ↑0.0	52.4 ↑0.0	53.4 ↑0.0	63.2 ↑0.0	72.6 ↑0.0	60.1 ↑0.0
	w/ Surgery	45.8 ↑1.8	74.5 ↓0.3	53.0 ↑0.6	54.5 ↑1.1	64.3 ↑1.1	72.9 ↑0.3	60.8 ↑0.7
	w/ ProbSurgery	46.0 ↑2.0	74.8 ↑0.0	**54.3** ↑1.9	54.5 ↑1.1	64.1 ↑0.9	**73.0** ↑0.4	61.1 ↑1.0
	w/ FEATCAL	**47.0** ↑3.0	**78.0** ↑3.2	53.0 ↑0.6	**55.3** ↑1.9	**66.5** ↑3.3	72.5 ↓0.1	**62.1** ↑2.0
Llama-3.1-8B-Instruct	Base	48.6	83.5	62.2	63.0	72.8	80.9	68.5
	Fine-tuned (STL)	52.6	85.3	67.1	63.8	72.8	80.9	70.4
	Task Arithmetic	49.0 ↑0.0	**85.6** ↑0.0	62.8 ↑0.0	61.6 ↑0.0	45.3 ↑0.0	76.9 ↑0.0	63.5 ↑0.0
	w/ Surgery	47.2 ↓1.8	85.2 ↓0.4	**64.6** ↑1.8	62.2 ↑0.6	47.1 ↑1.8	77.6 ↑0.7	64.0 ↑0.5
	w/ ProbSurgery	**49.6** ↑0.6	85.3 ↓0.3	62.8 ↑0.0	61.9 ↑0.3	48.8 ↑3.5	77.8 ↑0.9	64.4 ↑0.9
	w/ FEATCAL	47.6 ↓1.4	84.5 ↓1.1	59.8 ↓3.0	**62.7** ↑1.1	**60.8** ↑15.	**79.5** ↑2.6	**65.8** ↑2.3

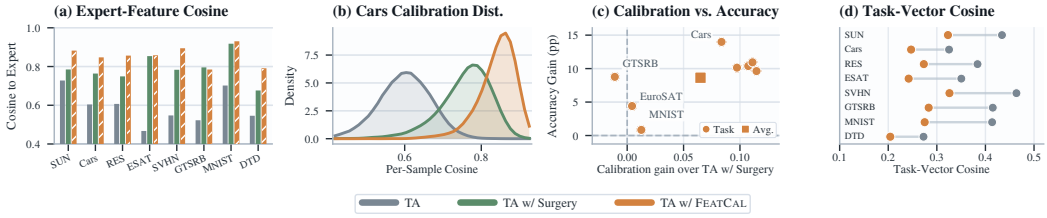


Figure 2: Feature-calibration diagnostics for TA on CLIP-ViT-B/32. (a) Task-wise final-feature cosine to experts; (b) per-sample cosine on Cars; (c) expert-cosine gain versus accuracy gain over TA w/ Surgery; (d) backbone task-vector cosine to same-task experts.

5.5 Analysis

We collect 4 diagnostics that probe the mechanism, practical cost, robustness, and stability of post-merging feature calibration.

Feature-calibration diagnostics. Fig. 2 tests whether TA w/ FEATCAL gains coincide with final features closer to task experts. Panels (a) and (b) show larger expert cosine than Surgery in this setting: the macro average rises from 0.785 to 0.850 over TA w/ Surgery, and Stanford Cars has a mean per-sample gain of 0.084. This matches the intended mechanism because FEATCAL calibrates the feature distribution reached by the deployed merged model. Panel (c) shows that expert-feature cosine is a diagnostic rather than a complete explanation: larger cosine gains often track accuracy gains, with an average $+8.64$-point gain over TA w/ Surgery, but EuroSAT and MNIST improve with small cosine changes, while GTSRB improves despite a slight decrease. Panel (d) highlights a feature-space and parameter-space mismatch: although FEATCAL moves final features closer to experts, the calibrated backbone has lower same-task expert task-vector cosine than the TA baseline on every task. Thus, FEATCAL calibrates the deployed feature distribution without bringing the backbone parameters into closer task-vector alignment, and the ProbSurgery comparison in App. J shows the same pattern with a stronger adapter baseline.

Calibration cost. All methods start from the same Task Arithmetic model and differ only in the post-merging calibration stream. Tab. 5 fixes $n = 256$: excluding final evaluation, the closed-form update takes 53s, $4.1\times$ faster than Surgery and $4.2\times$ faster than ProbSurgery, with lower GPU energy and much less peak CPU RSS. Full per-budget accuracy, resource averages, hardware, and logging details are in App. K.

Table 5: Post-TA runtime at $n = 256$, excluding final evaluation. Speedup is relative to Surgery.

Method	Speedup (Time)	GPU Energy (Wh)	CPU RSS (GiB)
Surgery	1.0× (217s)	18.1	69.3
ProbSurgery	1.0× (224s)	18.3	87.0
FEATCAL	**4.1× (053s)**	**1.9**	**22.8**

Sample efficiency. Fig. 3 reports post-TA sample efficiency on CLIP-ViT-B/32 TA with matched per-task budgets n. With 8 examples per task, FEATCAL already reaches 82.9% average accuracy and then saturates near 85.5%, while Surgery and ProbSurgery rise more slowly.
Together, FEATCAL reaches most of its TA gain with few examples and keeps calibration cost low at $n = 256$.

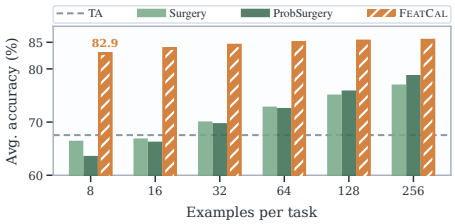


Figure 3: Sample Efficiency.

Robustness to corrupted calibration data. We corrupt only the images used for post-merging calibration and evaluate on clean test sets from 8-task TA, isolating calibration data quality rather than test-time corruption robustness. Protocol details are in App. L.

FEATCAL remains strongest in every reported setting, with a 78.0% average over clean, Gaussian noise, motion blur, and fog, compared with 71.5% for Surgery and 71.6% for ProbSurgery. Corrupted runs remain below the clean reference, but the stable ordering in Fig. 4 suggests that FEATCAL extracts a reliable expert-feature calibration signal from imperfect data.

Table 6: 8-task TA accuracy under corrupted calibration. Avg. includes clean, Gaussian, blur, and fog.

Method	Clean	Gauss.	Blur	Fog	Avg.
Task Arithmetic	67.5	–	–	–	–
w/ Surgery	76.8	72.4	67.5	69.4	71.5
w/ ProbSurgery	79.1	73.2	65.8	68.2	71.6
w/ FEATCAL	**85.5**	**76.6**	**74.2**	**75.6**	**78.0**



Figure 4: Calibration examples under corruptions.

Ablation study. Fig. 5 reports single-factor sensitivity in the 8-task TA setting. FEATCAL works best in a conservative calibration regime. Overly aggressive targets or regularization can degrade performance. The ridge sweep is flat over small positive values, so the closed-form solve is not tied to a narrow regularization choice, while very large ridge strength suppresses the update. The interpolation sweep has a sharper target-side tradeoff: small and medium α values help, whereas the hard-expert endpoint can force local updates to absorb upstream drift that should be calibrated gradually across layers. The merged-base blend curve is most asymmetric: moderate extrapolation improves both upstream mergers, but large ρ values push calibrated weights too far from the merged-base reference and especially hurt the WUDI variant. Overall, the ablation is a stability diagnostic with a broad conservative region rather than a single sharp optimum.

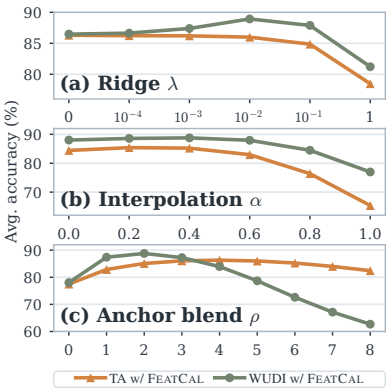


Figure 5: CLIP-ViT-B/32 TA coefficient sweeps.

6 Conclusion

Feature drift frames post-merging calibration by showing how local mismatches can propagate through a merged model and affect outputs. FEATCAL uses this signal by capturing features layer by layer and applying closed-form updates to linear modules, improving CLIP and FLAN-T5 mergers over Surgery and ProbSurgery in the main settings without adding modules at inference.

Acknowledgments

This work was supported by the Hong Kong RGC (TRS: T41-517/25-N; GRF: 15228325), ITC RAISe+ (RAI/24/1/086A), and the Research Institute for Generative Artificial Intelligence at PolyU.

Competing interests. Beyond the funding and affiliations disclosed above, the authors declare no additional competing financial interests relevant to this work.

References

[1] Mitchell Wortsman, Gabriel Ilharco, Samir Ya 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 *ICML*, 2022.

[2] Michael S. Matena and Colin A. Raffel. Merging models with fisher-weighted averaging. In *NeurIPS*, 2022.

[3] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *ICLR*, 2023.

[4] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *NeurIPS*, 2023.

[5] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. AdaMerging: Adaptive model merging for multi-task learning. In *ICLR*, 2024.

[6] 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 *ICML*, 2024.

[7] MohammadReza Davari and Eugene Belilovsky. Model Breadcrumbs: Scaling multi-task model merging with sparse masks. In *ECCV*, 2024.

[8] Derek Tam, Mohit Bansal, and Colin Raffel. Merging by matching models in task parameter subspaces. *TMLR*, 2024.

[9] Nico Daheim, Thomas Möllenhoff, Edoardo M. Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In *ICLR*, 2024.

[10] Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. Whoever started the interference should end it: Guiding data-free model merging via task vectors. In *ICML*, 2025.

[11] Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. In *ICML*, 2024.

[12] Qi Wei, Shuo He, Enneng Yang, Tingcong Liu, Haobo Wang, Lei Feng, and Bo An. Representation surgery in model merging with probabilistic modeling. In *ICML*, 2025.

[13] Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xingwei Wang, Xiaocun Cao, Jie Zhang, and Dacheng Tao. SurgeryV2: Bridging the gap between model merging and multi-task learning with deep representation surgery. *arXiv preprint arXiv:2410.14389*, 2024.

[14] Marcin Osial, Daniel Marczak, and Bartosz Zieliński. Parameter-efficient interventions for enhanced model merging. In *Proceedings of the 2025 SIAM International Conference on Data Mining*, 2025.

[15] Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In *NeurIPS*, 2023.

[16] Guodong Du and Wanyu Lin. Dynamic model merging made slim. In *NeurIPS*, 2026.

[17] Guodong Du, Zhuo Li, Xuanning Zhou, Junlin Li, Zesheng Shi, Wanyu Lin, Ho-Kin Tang, Xiucheng Li, Fangming Liu, Wenya Wang, Min Zhang, and Jing Li. Knowledge fusion of large language models via modular SkillPacks. In *ICLR*, 2026.

[18] Guodong Du, Zitao Fang, Jing Li, Junlin Li, Runhua Jiang, Shuyang Yu, Yifei Guo, Yangneng Chen, Sim Kuan Goh, Ho-Kin Tang, Daojing He, Honghai Liu, and Min Zhang. Neural parameter search for slimmer fine-tuned models and better transfer. In *ACL*, 2025.

[19] Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim Kuan Goh, Ho-Kin Tang, Daojing He, and Min Zhang. Parameter competition balancing for model merging. In *NeurIPS*, 2024.

[20] Guodong Du, Jing Li, Hanting Liu, Runhua Jiang, Shuyang Yu, Yifei Guo, Sim Kuan Goh, and Ho-Kin Tang. Knowledge fusion by evolving weights of language models. In *Findings of ACL*, 2024.

[21] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *ICLR*, 2023.

[22] The-Hai Nguyen, Huu-Tien Dang, Takeshi Suzuki, and Le-Minh Nguyen. RegMean++: Enhancing effectiveness and generalization of regression mean for model merging. *arXiv preprint arXiv:2508.03121*, 2025.

[23] Wenju Sun, Qingyong Li, Wen Wang, Yang Liu, Yangliao Geng, and Boyang Li. Towards minimizing feature drift in model merging: Layer-wise task vector fusion for adaptive knowledge integration. In *NeurIPS*, 2025.

[24] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 1970.

[25] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems*. V. H. Winston & Sons, 1977. Distributed solely by Halsted Press.

[26] Anke Tang, Li Shen, Yong Luo, Enneng Yang, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. Fusionbench: A unified library and comprehensive benchmark for deep model fusion. *JMLR*, 2025.

[27] Yifei He, Siqi Zeng, Yuzheng Hu, Rui Yang, Tong Zhang, and Han Zhao. MergeBench: A benchmark for merging domain-specialized LLMs. *arXiv preprint arXiv:2505.10833*, 2025.

[28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[29] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[30] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *ICCV Workshops*, 2013.

[31] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proc. IEEE*, 2017.

[32] 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*, 2019.

[33] 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.

[34] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, 2011.

[35] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 1998.

[36] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

[37] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

[38] Bastiaan S. Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant CNNs for digital pathology. In *Medical Image Computing and Computer Assisted Intervention*, 2018.

[39] Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: A report on three machine learning contests. *Neural Networks*, 2015.

[40] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012.

[41] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

[42] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[43] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101: Mining discriminative components with random forests. In *ECCV*, 2014.

[44] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[45] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre van Schaik. EMNIST: Extending MNIST to handwritten letters. In *International Joint Conference on Neural Networks*, 2017.

[46] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. In *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2018.

[47] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

[48] OpenAI. Rendered SST-2 Dataset. https://github.com/openai/CLIP/blob/main/data/rendered-sst2.md, 2021.

[49] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

[50] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *ICLR*, 2022.

[51] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *JMLR*, 2024.

[52] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018.

[53] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *TACL*, 2019.

[54] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, 2018.

[55] William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*, 2005.

[56] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[57] Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*. Springer, 2006.

[58] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval*, 2017.

[59] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[60] Qi Zhou, Yiming Zhang, Yanggan Gu, Yuanyi Wang, Zhaoyi Yan, Zhen Li, Chi Yung Chung, and Hongxia Yang. Model fusion for scalable and sustainable artificial intelligence: A review and outlook. *Journal of Modern Power Systems and Clean Energy*, 2026.

[61] Qi Zhou, Yiming Zhang, Yanggan Gu, Yuanyi Wang, Zhijie Sang, Zhaoyi Yan, Zhen Li, Shengyu Zhang, Fei Wu, and Hongxia Yang. Democratizing ai through model fusion: A comprehensive review and future directions. *Nexus*, 2025.

[62] Yuanyi Wang, Yanggan Gu, Yiming Zhang, Qi Zhou, Zhaoyi Yan, Congkai Xie, Xinyao Wang, Jianbo Yuan, and Hongxia Yang. Model merging scaling laws in large language models. *arXiv preprint arXiv:2509.24244*, 2025.

[63] Yuanyi Wang, Yanggan Gu, Zihao Wang, Kunxi Li, Yifan Yang, Zhaoyi Yan, Congkai Xie, Jianmin Wu, and Hongxia Yang. MergePipe: A budget-aware parameter management system for scalable LLM merging. *arXiv preprint arXiv:2602.13273*, 2026.

[64] Yanggan Gu, Yuanyi Wang, Zhaoyi Yan, Yiming Zhang, Qi Zhou, Fei Wu, and Hongxia Yang. InfiFPO: Implicit model fusion via preference optimization in large language models. *arXiv preprint arXiv:2505.13878*, 2025.

[65] Yanggan Gu, Junzhuo Li, Sirui Huang, Xin Zou, Zhenghua Li, and Xuming Hu. Capturing nuanced preferences: Preference-aligned distillation for small language models. In *Findings of ACL*, 2025.

[66] Yuanyi Wang, Zhaoyi Yan, Yiming Zhang, Qi Zhou, Yanggan Gu, Fei Wu, and Hongxia Yang. InfiGFusion: Graph-on-logits distillation via efficient gromov-wasserstein for model fusion. *arXiv preprint arXiv:2505.13893*, 2025.

[67] Yunkai Dang, Mengxi Gao, Yibo Yan, Xin Zou, Yanggan Gu, Jungang Li, Jingyu Wang, Peijie Jiang, Aiwei Liu, Jia Liu, and Xuming Hu. Exploring response uncertainty in MLLMs: An empirical evaluation under misleading scenarios. In *EMNLP*, 2025.

[68] Yifan Yang, Jinjia Li, Kunxi Li, Puhao Zheng, Yuanyi Wang, Zheyan Qu, Yang Yu, Jianmin Wu, Ming Li, and Hongxia Yang. InfiCoEvalChain: A blockchain-based decentralized framework for collaborative LLM evaluation. *arXiv preprint arXiv:2602.08229*, 2026.

[69] Wenjun Wang, Shuo Cai, Congkai Xie, Mingfa Feng, Yiming Zhang, Zhen Li, Kejing Yang, Ming Li, Jiannong Cao, and Hongxia Yang. InfiR2: A comprehensive FP8 training recipe for reasoning-enhanced language models. *arXiv preprint arXiv:2509.22536*, 2025.

A Limitations

Computational scope. Our experiments cover CLIP, FLAN-T5, and Llama models on vision and language benchmarks. The evaluation does not establish performance on other modalities or model families, or on models larger than the evaluated Llama-3.1-8B. Still, the current experiments cover more than 20 tasks across vision and language, which provides evidence that FEATCAL is not tied to a single task or benchmark.

Hyperparameter selection. FEATCAL still requires hyperparameters, including λ, ρ, and α, to be selected on a development or validation set. We do not provide an automatic selection rule. Because calibration is fast and uses closed form updates, this search is practical in our setting and adds little overhead compared with iterative calibration baselines.

Dependence on task calibration data. Like Surgery and ProbSurgery, FEATCAL needs task calibration data to fit parameter updates after merging. This requirement may be limiting when task data cannot be stored or sampled. Our results show better data efficiency than these baselines: on CLIP-ViT-B/32 TA, 8 examples per task already give strong gains, and performance saturates with fewer samples than the baselines.

B Additional Feature-Distribution Diagnostics for 8-Task TA

Fig. 6 extends the Stanford Cars visualization in panel (a) of Fig. 1 to all 8 tasks in the CLIP-ViT-B/32 TA setting. Each task uses its own joint 2D projection, so the geometry is intended for within-task comparison between the uncalibrated and calibrated merged features rather than cross-task distance comparison.

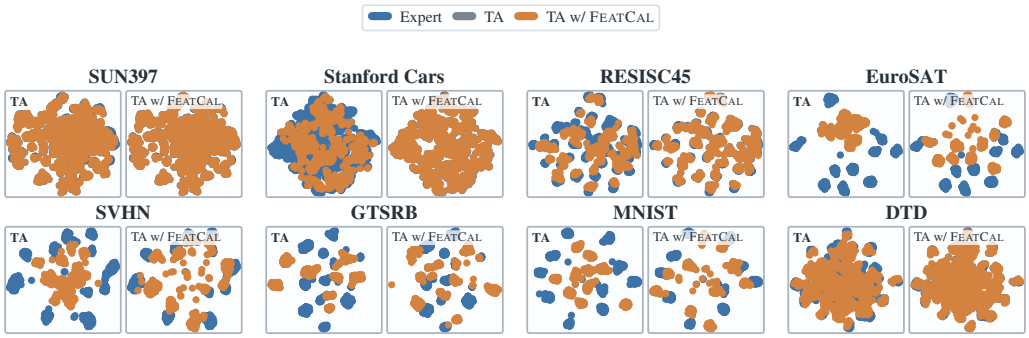


Figure 6: **8-task feature-distribution diagnostic.** Matched final projected features for the task expert, Task Arithmetic (TA), and TA w/ FEATCAL on each task in the CLIP-ViT-B/32 TA setting, following panel (a) of Fig. 1.

C Proofs for the Feature-Drift Analysis

Proof of Prop. 1. Starting from the definition of $e_{i,\ell}(x)$, add and subtract $f_\ell^{\mathrm{mer}}(h_{i,\ell-1}^{\mathrm{exp}}(x))$ to obtain

$$\begin{aligned} e_{i,\ell}(x) &= f_\ell^{\mathrm{mer}}\left(h_{\ell-1}^{\mathrm{mer}}(x)\right) - f_{i,\ell}^{\mathrm{exp}}\left(h_{i,\ell-1}^{\mathrm{exp}}(x)\right) \\ &= \left[f_\ell^{\mathrm{mer}}\left(h_{\ell-1}^{\mathrm{mer}}(x)\right) - f_\ell^{\mathrm{mer}}\left(h_{i,\ell-1}^{\mathrm{exp}}(x)\right)\right] \\ &\quad + \left[f_\ell^{\mathrm{mer}}\left(h_{i,\ell-1}^{\mathrm{exp}}(x)\right) - f_{i,\ell}^{\mathrm{exp}}\left(h_{i,\ell-1}^{\mathrm{exp}}(x)\right)\right]. \end{aligned} \tag{19}$$

The first bracket is $p_{i,\ell}(x)$ and the second is $m_{i,\ell}(x)$, proving Eq. (3). □

Proof of Prop. 2. For a fixed layer ℓ, the averaged Jacobian is defined in Eq. (4). The assumed differentiability lets us apply the fundamental theorem of calculus along the expert-to-merged feature segment to obtain

$$p_{i,\ell}(x) = A_{i,\ell}(x)\, e_{i,\ell-1}(x), \tag{20}$$

and hence

$$e_{i,\ell}(x) = A_{i,\ell}(x)\, e_{i,\ell-1}(x) + m_{i,\ell}(x). \tag{21}$$

Iterating Eq. (21) from layer $s+1$ to layer t yields

$$e_{i,t}(x) = P_{i,s\to t}(x) e_{i,s}(x) + \sum_{\ell=s+1}^{t} P_{i,\ell\to t}(x) m_{i,\ell}(x), \tag{22}$$

where $P_{i,r\to t}(x) = A_{i,t}(x)\cdots A_{i,r+1}(x)$, $P_{i,t\to t}(x) = I$, and products denote composition of compatible local linear maps. Taking $s=0$, $t=L$, and $e_{i,0}(x)=0$ yields

$$e_{i,L}(x) = \sum_{\ell=1}^{L} P_{i,\ell\to L}(x)\, m_{i,\ell}(x), \tag{23}$$

where

$$P_{i,\ell\to L}(x) = A_{i,L}(x) A_{i,L-1}(x)\cdots A_{i,\ell+1}(x), \qquad P_{i,L\to L}(x) = I. \tag{24}$$

Thus every term in the final drift expansion originates from a layer-wise local mismatch and is propagated through the downstream sensitivity chain. □

D Residual Propagation and Conditional Growth

This appendix specializes Prop. 1 to residualized Transformer layers with matching input and output dimensions:

$$f_\ell^{\mathrm{mer}}(h) = h + g_\ell^{\mathrm{mer}}(h), \qquad f_{i,\ell}^{\mathrm{exp}}(h) = h + g_{i,\ell}^{\mathrm{exp}}(h), \tag{25}$$

where g_ℓ^{mer} and $g_{i,\ell}^{\mathrm{exp}}$ collect module-internal nonlinear operations, such as self-attention or MLP together with normalization and other internal operations.

Define the residual-branch propagation term

$$r_{i,\ell}(x) = g_\ell^{\mathrm{mer}}\left(h_{\ell-1}^{\mathrm{mer}}(x)\right) - g_\ell^{\mathrm{mer}}\left(h_{i,\ell-1}^{\mathrm{exp}}(x)\right), \tag{26}$$

and the local mismatch term becomes

$$m_{i,\ell}(x) = g_\ell^{\mathrm{mer}}\left(h_{i,\ell-1}^{\mathrm{exp}}(x)\right) - g_{i,\ell}^{\mathrm{exp}}\left(h_{i,\ell-1}^{\mathrm{exp}}(x)\right). \tag{27}$$

Proposition 4 (Residual preservation)**.** *For a residualized layer of the form in Eq. (25), the propagated component satisfies*

$$p_{i,\ell}(x) = e_{i,\ell-1}(x) + r_{i,\ell}(x), \tag{28}$$

and therefore

$$e_{i,\ell}(x) = e_{i,\ell-1}(x) + r_{i,\ell}(x) + m_{i,\ell}(x). \tag{29}$$

Proof. By Eq. (25),

$$\begin{aligned} p_{i,\ell}(x) &= f_\ell^{\mathrm{mer}}\left(h_{\ell-1}^{\mathrm{mer}}(x)\right) - f_\ell^{\mathrm{mer}}\left(h_{i,\ell-1}^{\mathrm{exp}}(x)\right) \\ &= \left(h_{\ell-1}^{\mathrm{mer}}(x) + g_\ell^{\mathrm{mer}}\left(h_{\ell-1}^{\mathrm{mer}}(x)\right)\right) - \left(h_{i,\ell-1}^{\mathrm{exp}}(x) + g_\ell^{\mathrm{mer}}\left(h_{i,\ell-1}^{\mathrm{exp}}(x)\right)\right) \\ &= \underbrace{\left(h_{\ell-1}^{\mathrm{mer}}(x) - h_{i,\ell-1}^{\mathrm{exp}}(x)\right)}_{e_{i,\ell-1}(x)} + \underbrace{\left(g_\ell^{\mathrm{mer}}\left(h_{\ell-1}^{\mathrm{mer}}(x)\right) - g_\ell^{\mathrm{mer}}\left(h_{i,\ell-1}^{\mathrm{exp}}(x)\right)\right)}_{r_{i,\ell}(x)}, \end{aligned} \tag{30}$$

which proves Eq. (28). Substituting this identity into Eq. (3) gives Eq. (29). □

Interpretation. Eq. (29) makes the identity-skip propagation property explicit: the skip connection carries upstream feature drift through the additive term $e_{i,\ell-1}(x)$, while the residual branch may compensate for or amplify it.

When $g_\ell^{\mathrm{mer}}(\cdot)$ is continuously differentiable on an open neighborhood containing the line segment between $h_{i,\ell-1}^{\mathrm{exp}}(x)$ and $h_{\ell-1}^{\mathrm{mer}}(x)$, define

$$R_{i,\ell}(x) = \int_0^1 J_u g_\ell^{\mathrm{mer}}\Big(h_{i,\ell-1}^{\mathrm{exp}}(x) + t\, e_{i,\ell-1}(x)\Big)\, dt. \tag{31}$$

Then

$$r_{i,\ell}(x) = R_{i,\ell}(x)\, e_{i,\ell-1}(x), \tag{32}$$

and Eq. (29) becomes

$$e_{i,\ell}(x) = \big(I + R_{i,\ell}(x)\big)e_{i,\ell-1}(x) + m_{i,\ell}(x). \tag{33}$$

Proposition 5 (Sufficient condition for local monotone amplification)**.** *Consider a residualized layer of the form in Eq. (25), and assume g_ℓ^{mer} is continuously differentiable on an open neighborhood containing the line segment between $h_{i,\ell-1}^{\mathrm{exp}}(x)$ and $h_{\ell-1}^{\mathrm{mer}}(x)$, so $R_{i,\ell}(x)$ is well defined. Suppose $e_{i,\ell-1}(x) \neq 0$. Suppose further that there exists $\gamma_{i,\ell} > 0$ such that, for $v = e_{i,\ell-1}(x)$,*

$$\big\|\big(I + R_{i,\ell}(x)\big)v\big\|_2 \geq (1+\gamma_{i,\ell})\|v\|_2, \tag{34}$$

and suppose the local mismatch term satisfies

$$\|m_{i,\ell}(x)\|_2 \leq \eta_{i,\ell}\, \|e_{i,\ell-1}(x)\|_2 \qquad \text{for some } \eta_{i,\ell} \in [0, \gamma_{i,\ell}). \tag{35}$$

Then

$$\|e_{i,\ell}(x)\|_2 \geq (1 + \gamma_{i,\ell} - \eta_{i,\ell})\|e_{i,\ell-1}(x)\|_2 > \|e_{i,\ell-1}(x)\|_2. \tag{36}$$

Proof. By Eq. (33) and the reverse triangle inequality,

$$\begin{aligned}
\|e_{i,\ell}(x)\|_2 &= \big\|\big(I + R_{i,\ell}(x)\big)e_{i,\ell-1}(x) + m_{i,\ell}(x)\big\|_2 \\
&\geq \big\|\big(I + R_{i,\ell}(x)\big)e_{i,\ell-1}(x)\big\|_2 - \|m_{i,\ell}(x)\|_2 \\
&\geq (1+\gamma_{i,\ell})\|e_{i,\ell-1}(x)\|_2 - \eta_{i,\ell}\|e_{i,\ell-1}(x)\|_2 \\
&= (1 + \gamma_{i,\ell} - \eta_{i,\ell})\|e_{i,\ell-1}(x)\|_2.
\end{aligned} \tag{37}$$

Since $e_{i,\ell-1}(x) \neq 0$ and $\eta_{i,\ell} < \gamma_{i,\ell}$, the multiplicative factor is strictly larger than 1, proving the claim. □

Prop. 5 is intentionally conditional: feature drift need not amplify at every residualized layer, but it does grow monotonically when the residualized layer map is locally expansive along the current drift direction and the new local mismatch term is too small to offset that expansion.

Corollary 1 (Conditional cumulative monotone amplification)**.** *If there exists a consecutive range of layers $\ell_0 + 1, \ldots, L$ such that the assumptions of Prop. 5 hold for every layer in this range, then*

$$\|e_{i,L}(x)\|_2 \geq \prod_{\ell=\ell_0+1}^{L} (1 + \gamma_{i,\ell} - \eta_{i,\ell})\, \|e_{i,\ell_0}(x)\|_2. \tag{38}$$

Consequently, the feature drift norm increases strictly from a layer to the next across that range.

Proof. Apply Eq. (36) recursively from layer $\ell_0 + 1$ to layer L. □

E Output-Drift Bridge

This appendix records the standard perturbation argument connecting feature drift to output-score drift in the continuous task output score vector. For the merged model M^{mer}, define

$$e_{i,L}(x) = h_L^{\mathrm{mer}}(x) - h_{i,L}^{\mathrm{exp}}(x).$$

Let the task score maps ψ_i^{mer} and ψ_i^{exp} produce

$$z_i^{\mathrm{mer}}(x) = \psi_i^{\mathrm{mer}}(h_L^{\mathrm{mer}}(x))\,, \qquad z_i^{\mathrm{exp}}(x) = \psi_i^{\mathrm{exp}}\Big(h_{i,L}^{\mathrm{exp}}(x)\Big)\,,$$

and define

$$\Delta z_i^{\mathrm{mer}}(x) = z_i^{\mathrm{mer}}(x) - z_i^{\mathrm{exp}}(x).$$

Here z_i may denote class logits, CLIP candidate scores (scaled similarity scores over a fixed candidate set), or decoder vocabulary logits under a fixed prefix.

Proposition 6 (Output score map perturbation)**.** *Suppose ψ_i^{mer} is locally $B_i^{\mathrm{mer}}(x)$-Lipschitz on the segment between $h_{i,L}^{\mathrm{exp}}(x)$ and $h_L^{\mathrm{mer}}(x)$. Here $B_i^{\mathrm{mer}}(x)$ locally bounds how much the merged task score map can change when its final-feature input moves along this segment. Then*

$$\|\Delta z_i^{\mathrm{mer}}(x)\|_2 \le B_i^{\mathrm{mer}}(x)\,\|e_{i,L}(x)\|_2 \;+\; \left\|\psi_i^{\mathrm{mer}}\left(h_{i,L}^{\mathrm{exp}}(x)\right) - \psi_i^{\mathrm{exp}}\left(h_{i,L}^{\mathrm{exp}}(x)\right)\right\|_2. \tag{39}$$

If the task score map is shared, the second term is 0*.*

Proof. Add and subtract $\psi_i^{\mathrm{mer}}(h_{i,L}^{\mathrm{exp}}(x))$, then apply the local Lipschitz bound to the first difference:

$$\begin{aligned}
\|\Delta z_i^{\mathrm{mer}}(x)\|_2 &= \left\|\psi_i^{\mathrm{mer}}(h_L^{\mathrm{mer}}(x)) - \psi_i^{\mathrm{exp}}\left(h_{i,L}^{\mathrm{exp}}(x)\right)\right\|_2 \\
&\le \left\|\psi_i^{\mathrm{mer}}(h_L^{\mathrm{mer}}(x)) - \psi_i^{\mathrm{mer}}\left(h_{i,L}^{\mathrm{exp}}(x)\right)\right\|_2 \\
&\quad + \left\|\psi_i^{\mathrm{mer}}\left(h_{i,L}^{\mathrm{exp}}(x)\right) - \psi_i^{\mathrm{exp}}\left(h_{i,L}^{\mathrm{exp}}(x)\right)\right\|_2 \\
&\le B_i^{\mathrm{mer}}(x)\,\|e_{i,L}(x)\|_2 \;+\; \left\|\psi_i^{\mathrm{mer}}\left(h_{i,L}^{\mathrm{exp}}(x)\right) - \psi_i^{\mathrm{exp}}\left(h_{i,L}^{\mathrm{exp}}(x)\right)\right\|_2.
\end{aligned} \tag{40}$$

□

Softmax probability bridge. For a softmax output over a fixed candidate or token set, let $p(z) = \mathrm{softmax}(z)$. Its Jacobian is

$$J_{\mathrm{sm}}(z) = \nabla_z p(z) = \mathrm{diag}(p(z)) - p(z)p(z)^\top.$$

Hence, for any logit perturbation Δz,

$$p(z+\Delta z) - p(z) = \int_0^1 J_{\mathrm{sm}}(z + t\Delta z)\Delta z\, dt.$$

For cross-entropy with one-hot label vector u_y,

$$\mathcal{L}_{\mathrm{CE}}(y, z+\Delta z) - \mathcal{L}_{\mathrm{CE}}(y, z) = \int_0^1 \big(p(z+t\Delta z) - u_y\big)^\top \Delta z\, dt.$$

Thus relative logit changes induce softmax-normalized probability and loss changes through the softmax Jacobian and the cross-entropy gradient; an additive shift $c\mathbf{1}$ is a null direction for both.

Proposition 7 (Logit perturbation and loss stability)**.** *Let $\mathcal{L}_i(y, z)$ be a task loss that is differentiable along the segment between $z_i^{\mathrm{exp}}(x)$ and $z_i^{\mathrm{mer}}(x)$. Define*

$$\bar{g}_i(x) = \int_0^1 \nabla_z \mathcal{L}_i(y, z_i^{\mathrm{exp}}(x) + t\Delta z_i^{\mathrm{mer}}(x))\; dt.$$

Then

$$\mathcal{L}_i(y, z_i^{\mathrm{mer}}(x)) - \mathcal{L}_i(y, z_i^{\mathrm{exp}}(x)) = \bar{g}_i(x)^\top \Delta z_i^{\mathrm{mer}}(x), \tag{41}$$

and hence

$$|\mathcal{L}_i(y, z_i^{\mathrm{mer}}(x)) - \mathcal{L}_i(y, z_i^{\mathrm{exp}}(x))| \le \|\bar{g}_i(x)\|_2\, \|\Delta z_i^{\mathrm{mer}}(x)\|_2. \tag{42}$$

Proof. Apply the fundamental theorem of calculus to $\mathcal{L}_i(y, z_i^{\mathrm{exp}}(x) + t\Delta z_i^{\mathrm{mer}}(x))$ over $t \in [0,1]$, and then use Cauchy–Schwarz. □

Corollary 2 (Feature-to-loss perturbation)**.** *Under the conditions of Prop. 6, suppose $\mathcal{L}_i(y, z)$ is locally $G_i^{\mathrm{mer}}(x)$-Lipschitz on the segment between $z_i^{\mathrm{exp}}(x)$ and $z_i^{\mathrm{mer}}(x)$, and let $\delta_i^{\psi,\mathrm{mer}}(x)$ denote the score map mismatch term in Eq. (39). Then*

$$|\mathcal{L}_i(y, z_i^{\mathrm{mer}}(x)) - \mathcal{L}_i(y, z_i^{\mathrm{exp}}(x))| \le G_i^{\mathrm{mer}}(x)\left(B_i^{\mathrm{mer}}(x)\,\|e_{i,L}(x)\|_2 + \delta_i^{\psi,\mathrm{mer}}(x)\right). \tag{43}$$

Proof. Combine Eq. (39) with the local Lipschitz bound on $\mathcal{L}_i(y, \cdot)$. □

Decision margins. For discrete decisions, output-score drift matters through margins. For a top-1 classification output with unique expert winner $k^\star = \arg\max_k z_{i,k}^{\exp}(x)$, each competitor $k \neq k^\star$ changes its pairwise margin as

$$z_{i,k^\star}^{\mathrm{mer}}(x) - z_{i,k}^{\mathrm{mer}}(x) = \big(z_{i,k^\star}^{\exp}(x) - z_{i,k}^{\exp}(x)\big) - \big(\Delta z_{i,k}^{\mathrm{mer}}(x) - \Delta z_{i,k^\star}^{\mathrm{mer}}(x)\big).$$

Thus a decision boundary is crossed only when a competitor's relative score drift closes the corresponding expert margin. In particular, with the minimum expert margin

$$\mu_i(x) = z_{i,k^\star}^{\exp}(x) - \max_{k \neq k^\star} z_{i,k}^{\exp}(x),$$

the merged model preserves the expert top-1 decision whenever

$$\|\Delta z_i^{\mathrm{mer}}(x)\|_\infty < \mu_i(x)/2 \quad \Longrightarrow \quad \arg\max_k z_{i,k}^{\mathrm{mer}}(x) = k^\star. \tag{44}$$

Analogous pairwise margins apply to fixed-candidate retrieval items and fixed-prefix next-token choices.

F Forward-Order Calibration and Deployed Feature Sources

This appendix formalizes why FEATCAL collects current calibrated-model features at each layer before fitting the calibrated modules in that layer. The point is complementary to Prop. 2: local mismatch produced at a layer is propagated downstream in forward order, so the layer-wise calibration surrogate should be fitted on the feature source that the final calibrated model will actually expose to that layer.

Let $M^{[0]} = M^{\mathrm{mer}}$, and let $M^{[\ell-1]}$ denote the partially calibrated model after all calibrated layers before layer ℓ have been updated. Because updates to layer ℓ and later layers are downstream of the prefix before layer ℓ, the input feature to layer ℓ in the final calibrated model M^{cal} is the same as the input feature produced by $M^{[\ell-1]}$:

$$h_{i,\ell-1}^{M^{\mathrm{cal}}}(x) = h_{i,\ell-1}^{M^{[\ell-1]}}(x). \tag{45}$$

Thus the deployed layer-input distribution is the push-forward distribution

$$\mu_{i,\ell}^{\mathrm{dep}} = \big(h_{i,\ell-1}^{M^{[\ell-1]}}\big)_{\#}\mathcal{D}_i. \tag{46}$$

Within layer ℓ, FEATCAL uses this cached layer snapshot to construct the input features for all calibrated modules in the layer. It does not advance the feature source after each module update inside the same layer. Thus the deployed source statement is layer-level, while the objectives below are module-local fits computed from cached features.

Consider a calibrated linear module with candidate weight W. Let (u, y) denote the module input feature and the expert target output used by the local regression, and define

$$r_W(u, y) = \|Wu - y\|_2^2.$$

Let $\Pi_{i,\ell}^{\mathrm{dep}}$ be the deployed joint distribution of these input-target pairs induced by the prefix-calibrated feature source in Eq. (46). Let $\Pi_{i,\ell}^{\mathrm{src}}$ be any other joint distribution used to fit the same local update, such as features collected from the uncalibrated merged model, from the task expert, or from a stale partially calibrated model. For $a \in \{\mathrm{src}, \mathrm{dep}\}$, define the population ridge objective

$$J_\ell^a(W) = \sum_{i=1}^{N} \mathbb{E}_{(u,y)\sim\Pi_{i,\ell}^a} r_W(u, y) + \lambda\|W - W^{\mathrm{anc}}\|_F^2. \tag{47}$$

Proposition 8 (Source/deployed objective mismatch)**.** *Let $\mathcal{W}$ be a weight class for which the following suprema are finite, and let*

$$W_{\mathrm{src}}^\star \in \arg\min_{W\in\mathcal{W}} J_\ell^{\mathrm{src}}(W), \qquad W_{\mathrm{dep}}^\star \in \arg\min_{W\in\mathcal{W}} J_\ell^{\mathrm{dep}}(W).$$

Then

$$J_\ell^{\mathrm{dep}}(W_{\mathrm{src}}^\star) - J_\ell^{\mathrm{dep}}(W_{\mathrm{dep}}^\star) \leq 2 \sup_{W\in\mathcal{W}} \left| J_\ell^{\mathrm{dep}}(W) - J_\ell^{\mathrm{src}}(W) \right|. \tag{48}$$

If, moreover, $r_W(u, y)$ is uniformly K_ℓ-Lipschitz in (u, y) on the relevant supports for all $W \in \mathcal{W}$, then

$$J_\ell^{\mathrm{dep}}(W_{\mathrm{src}}^\star) - J_\ell^{\mathrm{dep}}(W_{\mathrm{dep}}^\star) \le 2K_\ell \sum_{i=1}^{N} W_1\left(\Pi_{i,\ell}^{\mathrm{dep}}, \Pi_{i,\ell}^{\mathrm{src}}\right), \tag{49}$$

where W_1 denotes the 1-Wasserstein distance on input-target pairs.

Proof. Add and subtract $J_\ell^{\mathrm{src}}(W_{\mathrm{src}}^\star)$ and $J_\ell^{\mathrm{src}}(W_{\mathrm{dep}}^\star)$:

$$\begin{aligned} J_\ell^{\mathrm{dep}}(W_{\mathrm{src}}^\star) - J_\ell^{\mathrm{dep}}(W_{\mathrm{dep}}^\star) = &\left[J_\ell^{\mathrm{dep}}(W_{\mathrm{src}}^\star) - J_\ell^{\mathrm{src}}(W_{\mathrm{src}}^\star)\right] \\ &+ \left[J_\ell^{\mathrm{src}}(W_{\mathrm{src}}^\star) - J_\ell^{\mathrm{src}}(W_{\mathrm{dep}}^\star)\right] \\ &+ \left[J_\ell^{\mathrm{src}}(W_{\mathrm{dep}}^\star) - J_\ell^{\mathrm{dep}}(W_{\mathrm{dep}}^\star)\right]. \end{aligned} \tag{50}$$

The middle bracket is nonpositive by optimality of $W_{\mathrm{src}}^\star$ for J_ℓ^{src}. Both remaining brackets are bounded by $\sup_{W \in \mathcal{W}} |J_\ell^{\mathrm{dep}}(W) - J_\ell^{\mathrm{src}}(W)|$, which proves Eq. (48). The quadratic regularization term is identical in both objectives, so the source/deployed difference only comes from the expected matching losses. The Kantorovich–Rubinstein duality for W_1, under the stated uniform Lipschitz condition, gives

$$\left|\mathbb{E}_{\Pi_{i,\ell}^{\mathrm{dep}}} r_W - \mathbb{E}_{\Pi_{i,\ell}^{\mathrm{src}}} r_W\right| \le K_\ell W_1\left(\Pi_{i,\ell}^{\mathrm{dep}}, \Pi_{i,\ell}^{\mathrm{src}}\right).$$

Summing over tasks and applying the previous bound proves Eq. (49). □

Implication for FEATCAL. Forward-order calibration sets the source distribution for layer ℓ to the deployed prefix distribution in Eq. (46), so the population source/deployed mismatch term in Eq. (49) vanishes for that layer snapshot. Using merged-model, expert, or stale features can fit a different objective: after earlier layers are calibrated, the final model may expose layer ℓ to a feature source that differs from the source used to estimate W. This does not prove global optimality of the forward-order procedure, but it explains why FEATCAL captures features at each layer from the current partially calibrated model rather than from a non-deployed feature source.

G Bias and LayerNorm Affine Updates

The main text writes the core update for a bias-free linear map. The implementation also calibrates a linear-module bias parameter when present. It first solves the weight $W^\star$ by Eq. (18), then solves the bias with $W^\star$ fixed. Thus the following formula is a second-stage bias update, not a joint affine solve over W and b. For readability, the equations below use uniform empirical averages; replacing these averages by normalized sample or token weights gives the implemented weighted version.

Let $b_i, b^{\mathrm{mer}}, b^{\mathrm{base}} \in \mathbb{R}^m$ be the expert, merged, and base bias parameters for the same linear module, and define

$$b^{\mathrm{anc}} = \rho b^{\mathrm{mer}} + (1 - \rho) b^{\mathrm{base}}. \tag{51}$$

Let $\mathbf{1}_n \in \mathbb{R}^n$ be the all-one vector and define the empirical input means

$$\mu_i^{\mathrm{cal}} = \frac{1}{n} X_i^{\mathrm{cal}} \mathbf{1}_n, \qquad \mu_i^{\mathrm{tgt}} = \frac{1}{n} X_i^{\mathrm{tgt}} \mathbf{1}_n. \tag{52}$$

The mean μ_i^{tgt} is taken after target-side interpolation in Eq. (11); it equals the raw expert-feature mean only when $\alpha = 1$. With $W^\star$ fixed and the task weights ω_i from Eq. (15), the bias objective is

$$b^\star = \arg\min_{b \in \mathbb{R}^m} \sum_{i=1}^{N} \omega_i \left\| W^\star \mu_i^{\mathrm{cal}} + b - \left(W_i \mu_i^{\mathrm{tgt}} + b_i\right)\right\|_2^2 + \lambda \|b - b^{\mathrm{anc}}\|_2^2. \tag{53}$$

Taking the derivative gives

$$\left(\sum_{i=1}^{N} \omega_i + \lambda\right) b = \sum_{i=1}^{N} \omega_i \left(b_i + W_i \mu_i^{\mathrm{tgt}} - W^\star \mu_i^{\mathrm{cal}}\right) + \lambda b^{\mathrm{anc}}, \tag{54}$$

and hence

$$b^\star = \frac{\sum_{i=1}^N \omega_i \left(b_i + W_i \mu_i^{\mathrm{tgt}} - W^\star \mu_i^{\mathrm{cal}}\right) + \lambda b^{\mathrm{anc}}}{\sum_{i=1}^N \omega_i + \lambda}. \tag{55}$$

The experiments in this paper provide merged and base anchors. If an anchor is unavailable in code, the corresponding λ-term is omitted from the numerator and denominator.

For a LayerNorm module, write LN_0 for LayerNorm without its affine scale and shift. For task i, FEATCAL collects the current calibrated input features and forms

$$Z_i^{\mathrm{cal}} = \mathrm{LN}_0(X_i^{\mathrm{cal}}),$$

using the module's normalized shape and its LayerNorm numerical constant. This LayerNorm update uses the current normalized features only: it matches a shared affine map to the expert affine maps evaluated on Z_i^{cal}, rather than matching expert-normalized features.

Let $\gamma_i, \beta_i \in \mathbb{R}^d$ be the expert LayerNorm scale and shift for task i, and let $\gamma^{\mathrm{mer}}, \beta^{\mathrm{mer}}, \gamma^{\mathrm{base}}, \beta^{\mathrm{base}} \in \mathbb{R}^d$ be the corresponding merged and base affine parameters. Define

$$\gamma^{\mathrm{anc}} = \rho\gamma^{\mathrm{mer}} + (1-\rho)\gamma^{\mathrm{base}}, \qquad \beta^{\mathrm{anc}} = \rho\beta^{\mathrm{mer}} + (1-\rho)\beta^{\mathrm{base}}.$$

For each task, summarize the current normalized features by the coordinate-wise mean and second moment

$$\bar{z}_i = \frac{1}{n} Z_i^{\mathrm{cal}} \mathbf{1}_n, \qquad q_i = \frac{1}{n} \left(Z_i^{\mathrm{cal}} \odot Z_i^{\mathrm{cal}}\right) \mathbf{1}_n, \tag{56}$$

where vector products with Z_i^{cal} are applied column-wise. The LayerNorm affine objective is

$$\begin{aligned}(\gamma^\star, \beta^\star) = \arg\min_{\gamma,\beta\in\mathbb{R}^d} \sum_{i=1}^N \frac{1}{n} \left\|\gamma \odot Z_i^{\mathrm{cal}} + \beta \mathbf{1}_n^\top - \left(\gamma_i \odot Z_i^{\mathrm{cal}} + \beta_i \mathbf{1}_n^\top\right)\right\|_F^2 \\ + \lambda\|\gamma - \gamma^{\mathrm{anc}}\|_2^2 + \lambda\|\beta - \beta^{\mathrm{anc}}\|_2^2.\end{aligned} \tag{57}$$

Unlike the linear-weight and bias updates, this LayerNorm affine update uses equal task weights after each task's empirical moments are formed. This objective separates across coordinates. Let $\mathbf{1}_d \in \mathbb{R}^d$ be the all-one vector and define

$$\begin{aligned} a_{11} &= \sum_{i=1}^N q_i + \lambda \mathbf{1}_d, & a_{12} &= \sum_{i=1}^N \bar{z}_i, & a_{22} &= N + \lambda, \\ r_\gamma &= \sum_{i=1}^N \left(q_i \odot \gamma_i + \bar{z}_i \odot \beta_i\right) + \lambda\gamma^{\mathrm{anc}}, & r_\beta &= \sum_{i=1}^N \left(\bar{z}_i \odot \gamma_i + \beta_i\right) + \lambda\beta^{\mathrm{anc}}. \end{aligned} \tag{58}$$

The coordinate-wise 2×2 normal equations have determinant

$$D = a_{22}a_{11} - a_{12} \odot a_{12}.$$

Thus the closed-form affine update is

$$\gamma^\star = (a_{22} r_\gamma - a_{12} \odot r_\beta) \oslash D, \qquad \beta^\star = (a_{11} \odot r_\beta - a_{12} \odot r_\gamma) \oslash D, \tag{59}$$

where $\oslash$ denotes element-wise division. In the implementation, near-singular determinants are clamped from below by the FeatCal numerical stabilizer ϵ. If an anchor is unavailable, the corresponding λ-term is omitted.

H Calibration Algorithm

Algorithm 1 FEATCAL Calibration

Input: M^{mer}, M^{base}, experts $\{M_i^{\mathrm{exp}}\}_{i=1}^{N}$, calibration sets, layers or blocks in forward order with calibrated modules inside each layer, and hyperparameters $\lambda, \alpha, \rho, \epsilon$
Output: calibrated model M^{cal}

Initialize $M^{\mathrm{cal}} \leftarrow M^{\mathrm{mer}}$
for each layer or block in forward order **do**
    Cache current calibrated-model and expert input features once for the modules calibrated in this layer
    for each module t calibrated in the current layer **do**
        if t is a linear module **then**
            Let W^{mer}, W^{base}, and W_i be the corresponding merged, base, and expert module weights
            Read cached X_i^{cal} and X_i^{exp} from the layer feature snapshot
            Construct X_i^{tgt}, G_i, C_i, and ω_i using Eqs. (11), (14) and (15)
            Form W^{anc} by Eq. (12) and store the closed-form weight from Eq. (18)
            if the linear module has a calibrated bias parameter **then**
                Store the closed-form bias from Eq. (55)
            end if
        else if t is a LayerNorm module **then**
            Read cached normalized current features and store the affine parameters from Eq. (59)
        end if
    end for
    Load all stored calibrated parameters of this layer into M^{cal}
end for
return M^{cal}

I Detailed CLIP Results

Table 7: Full 8-task CLIP-ViT-L/14 results. All numbers are top-1 accuracy (%). Parenthesized values in Avg. report changes over the corresponding upstream merger.

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg.
Reference methods									
Pre-trained	68.3	77.8	71.0	58.9	58.4	50.6	76.4	55.5	64.6
Fine-tuned (STL)	82.8	92.9	97.4	99.2	97.9	99.2	99.8	85.5	94.3
Traditional MTL	79.0	89.3	94.5	98.4	96.4	98.1	99.4	83.7	92.4
Model merging and post-merging calibration									
Simple Averaging	72.5	81.5	82.3	88.5	81.6	74.0	96.6	61.8	79.9
w/ Surgery	73.8	81.2	89.1	95.8	86.1	97.9	74.4	85.2	85.4 (+5.5)
w/ ProbSurgery	76.4	82.9	92.2	95.6	85.3	91.2	98.0	77.0	87.3 (+7.3)
w/ FEATCAL	77.4	88.4	91.0	97.1	95.3	93.7	97.3	73.6	89.2 (+9.3)
Task Arithmetic	72.0	79.0	80.6	84.6	87.5	83.5	98.0	58.5	80.5
w/ Surgery	73.4	80.5	87.9	94.6	90.6	90.3	98.7	72.1	86.0 (+5.5)
w/ ProbSurgery	74.3	80.8	90.9	95.4	90.7	93.0	98.8	76.6	87.6 (+6.9)
w/ FEATCAL	80.0	90.8	93.7	98.1	96.6	97.0	98.3	78.6	91.6 (+11.1)
AdaMerging	78.2	90.7	90.9	96.1	94.8	97.6	98.5	81.4	91.0
w/ Surgery	79.0	91.0	93.9	96.9	95.4	97.8	98.8	83.1	92.0 (+1.0)
w/ ProbSurgery	79.4	91.6	94.3	97.5	95.6	98.1	98.9	83.4	92.4 (+1.4)
w/ FEATCAL	81.9	92.6	94.9	98.2	96.4	98.5	98.0	84.1	93.1 (+2.1)
WUDI-Merging	79.8	90.9	94.0	98.4	97.0	98.1	99.3	80.0	92.2
w/ Surgery	80.5	91.3	95.3	98.5	97.1	98.4	99.4	81.6	92.8 (+0.6)
w/ ProbSurgery	80.4	91.3	95.8	98.8	97.2	98.4	99.4	82.8	93.0 (+0.7)
w/ FEATCAL	81.5	91.7	96.5	99.0	97.8	99.2	98.6	83.6	93.5 (+1.3)

Table 8: Full 14-task CLIP-ViT-B/32 results. All numbers are top-1 accuracy (%). Parenthesized values in Avg. report changes over the corresponding upstream merger.

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Flowers	PCAM	FER	Pets	STL10	C100	Avg.
Reference methods															
Pre-trained	63.2	59.6	60.3	45.0	31.6	32.5	48.3	44.2	66.5	60.6	41.2	83.3	97.1	89.8	58.8
Fine-tuned (STL)	74.9	78.5	95.1	99.1	97.3	98.9	99.6	79.7	88.5	98.0	71.6	92.5	97.5	88.4	90.0
Model merging and post-merging calibration															
Simple Averaging	64.8	60.4	67.1	67.0	50.7	45.6	76.6	46.9	67.4	65.2	51.6	84.2	97.2	70.4	65.4
w/ Surgery	66.7	61.4	80.7	92.5	61.4	74.8	94.8	63.7	78.0	78.6	60.4	88.5	97.9	72.9	76.6 (+11.2)
w/ ProbSurgery	68.7	64.7	85.0	93.3	60.6	82.9	95.9	68.6	79.9	81.1	60.2	91.3	97.9	75.0	78.9 (+13.5)
w/ FEATCAL	68.2	67.3	80.3	92.5	86.5	77.0	95.1	60.7	75.2	86.0	63.9	89.3	97.5	74.5	79.6 (+14.2)
Task Arithmetic	62.4	55.6	64.2	67.0	53.1	85.6	48.2	55.5	61.9	76.4	54.2	82.7	95.2	64.6	66.2
w/ Surgery	65.6	60.5	78.7	92.7	80.6	96.3	62.7	67.5	75.4	81.2	60.5	88.0	96.8	68.4	76.8 (+10.6)
w/ ProbSurgery	65.9	64.0	83.2	92.6	85.6	96.2	67.1	68.7	75.3	82.2	60.6	88.2	96.7	70.5	78.3 (+12.1)
w/ FEATCAL	68.5	68.6	83.7	93.6	91.7	85.4	98.0	64.4	74.1	87.4	65.3	89.2	97.0	73.5	81.5 (+15.3)
AdaMerging	66.3	70.0	82.1	92.2	86.1	89.8	98.0	61.3	73.6	51.6	65.0	87.3	96.6	68.9	77.8
w/ Surgery	68.5	69.5	88.5	95.2	88.8	94.6	98.4	70.9	80.8	79.7	65.3	91.4	97.6	71.6	82.9 (+5.1)
w/ ProbSurgery	69.0	71.1	89.8	96.0	89.2	93.9	98.8	72.6	82.0	82.7	64.5	91.4	97.6	73.3	83.7 (+5.9)
w/ FEATCAL	71.2	74.6	88.4	95.3	90.6	91.9	98.6	71.4	81.2	85.2	69.9	91.2	97.5	76.6	84.5 (+6.7)
WUDI-Merging	65.7	64.8	77.0	89.2	91.3	91.5	99.0	60.7	63.8	85.5	64.2	86.2	96.1	66.6	78.7
w/ Surgery	66.9	68.5	84.6	96.0	92.6	95.3	99.0	67.9	76.4	86.7	64.9	88.6	96.8	70.8	82.5 (+3.8)
w/ ProbSurgery	67.1	69.5	86.7	96.4	92.5	95.4	99.1	68.6	76.8	85.7	63.9	89.0	96.8	70.3	82.7 (+4.0)
w/ FEATCAL	71.8	74.6	91.2	97.1	95.9	96.4	99.3	72.8	80.5	86.8	70.6	90.7	97.9	80.7	86.2 (+7.5)

Table 9: Full 14-task CLIP-ViT-L/14 results. All numbers are top-1 accuracy (%). Parenthesized values in Avg. report changes over the corresponding upstream merger.

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Flowers	PCAM	FER	Pets	STL10	C100	Avg.
Reference methods															
Pre-trained	68.2	77.9	71.3	61.2	58.4	50.5	76.3	55.5	79.3	51.2	50.0	93.2	99.4	75.1	69.1
Fine-tuned (STL)	82.8	92.8	97.4	91.1	97.9	99.2	99.8	85.5	97.7	91.1	75.9	95.8	99.2	93.0	92.8
Model merging and post-merging calibration															
Simple Averaging	71.2	79.0	78.7	80.4	71.3	64.6	94.3	58.7	81.9	74.2	54.8	94.6	99.3	82.4	77.5
w/ Surgery	72.0	78.7	82.6	93.4	75.4	78.6	97.3	70.4	88.7	85.4	67.2	95.3	99.5	83.2	83.4 (+5.9)
w/ ProbSurgery	75.7	81.8	91.0	95.3	75.7	89.9	98.2	76.8	95.3	84.8	67.9	95.6	99.6	84.7	86.6 (+9.1)
w/ FEATCAL	74.4	85.4	86.8	95.9	91.9	86.8	96.7	68.0	91.6	84.3	67.1	95.8	99.2	84.2	86.3 (+8.8)
Task Arithmetic	71.2	75.9	77.2	76.4	70.2	95.7	58.2	72.5	80.3	73.3	57.9	94.6	98.7	79.7	77.3
w/ Surgery	72.3	78.3	81.1	93.9	83.7	97.7	70.0	79.8	88.8	85.9	67.1	95.0	99.3	80.8	83.8 (+6.5)
w/ ProbSurgery	74.8	79.8	90.2	94.7	89.9	98.1	74.3	80.2	94.0	87.2	66.1	94.9	99.3	81.7	86.1 (+8.8)
w/ FEATCAL	81.1	92.0	93.0	97.2	94.2	95.8	98.2	80.9	97.5	82.9	75.2	96.2	99.1	86.4	90.7 (+13.4)
AdaMerging	77.6	90.1	91.3	95.9	94.1	96.1	98.6	78.1	95.7	51.6	74.3	95.8	99.4	83.2	87.3
w/ Surgery	78.7	91.0	93.5	96.5	94.5	96.8	98.9	81.5	97.3	82.8	74.1	96.0	99.5	84.2	90.4 (+3.1)
w/ ProbSurgery	79.0	91.4	94.6	97.0	94.9	97.4	99.0	82.7	98.0	83.2	74.4	96.4	99.5	85.2	90.9 (+3.6)
w/ FEATCAL	81.1	91.8	93.0	97.1	94.2	95.7	98.2	80.9	97.3	82.8	75.1	96.2	99.2	86.2	90.6 (+3.3)
WUDI-Merging	76.7	87.6	90.4	95.4	94.8	95.7	99.2	71.4	95.4	86.7	70.7	96.2	99.1	84.5	88.8
w/ Surgery	78.0	89.6	92.4	97.3	95.1	96.9	99.3	77.1	96.4	89.8	72.1	96.0	99.4	85.2	90.3 (+1.5)
w/ ProbSurgery	78.4	89.3	94.3	97.9	95.5	97.0	99.2	79.3	96.6	89.4	72.2	96.0	99.3	85.3	90.7 (+1.9)
w/ FEATCAL	80.5	91.1	95.4	98.5	97.2	98.5	99.0	80.6	97.5	83.7	75.0	95.6	99.2	89.3	91.5 (+2.7)

Table 10: Full 20-task CLIP-ViT-B/32 results. All numbers are top-1 accuracy (%). Parenthesized values in Avg. report changes over the corresponding upstream merger.

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Flowers	PCAM	FER	Pets	STL10	C100	C10	Food	F-MNIST	EMNIST	KMNIST	Rendered	Avg.
Reference methods																					
Pre-trained	63.2	59.6	60.3	45.0	31.6	32.5	48.3	44.2	66.5	60.6	41.2	83.3	97.1	63.7	89.8	82.4	63.0	12.0	9.9	58.6	55.6
Fine-tuned (STL)	74.9	78.5	95.1	99.1	97.3	98.9	99.6	79.7	88.5	98.0	71.6	92.5	97.5	88.4	97.6	88.4	94.8	95.6	98.2	71.3	90.3
Model merging and post-merging calibration																					
Simple Averaging	64.2	59.6	64.8	60.9	47.3	43.1	71.8	46.4	66.5	63.9	50.2	84.1	97.0	69.8	92.7	80.4	71.3	15.0	11.5	61.8	61.1
w/ Surgery	66.9	61.1	79.1	91.2	56.8	74.0	95.3	61.5	76.8	80.4	59.3	89.1	97.9	72.3	94.6	81.6	83.1	63.4	65.5	60.7	75.5 (+14.4)
w/ ProbSurgery	68.0	64.6	83.5	92.3	58.2	82.0	95.8	67.9	81.0	79.4	59.3	90.0	97.6	74.8	94.5	82.0	84.7	73.6	66.8	63.8	78.0 (+16.9)
w/ FEATCAL	67.6	65.3	78.4	92.0	81.1	71.2	91.5	58.3	73.5	82.8	62.7	88.6	97.3	73.4	94.9	83.4	83.1	24.3	35.2	68.4	73.7 (+12.6)
Task Arithmetic	62.0	53.7	60.9	58.1	48.9	79.4	46.1	48.5	61.1	73.4	51.4	82.3	94.9	64.6	91.4	71.9	73.9	17.8	12.2	59.9	60.6
w/ Surgery	64.9	59.8	76.7	90.2	78.2	96.1	60.6	62.9	74.4	79.8	58.9	86.6	96.9	68.9	93.2	74.8	83.3	66.7	70.4	61.0	75.2 (+14.6)
w/ ProbSurgery	65.8	61.8	80.1	91.8	84.1	96.7	67.2	63.5	77.2	79.9	57.7	88.3	96.6	79.7	92.8	74.7	84.8	75.7	70.4	66.0	77.7 (+17.1)
w/ FEATCAL	67.6	67.3	83.4	94.4	91.6	88.5	97.9	63.6	73.3	84.8	65.0	87.7	97.0	76.0	95.9	82.6	89.3	42.8	66.7	72.2	79.4 (+18.8)
AdaMerging	66.2	67.0	80.0	92.1	79.1	87.2	90.5	59.5	72.7	53.2	65.5	85.2	96.4	69.3	90.8	79.7	70.0	15.4	10.0	50.6	69.0
w/ Surgery	68.9	67.1	87.3	95.4	84.2	92.7	97.0	70.0	81.5	79.6	64.0	91.4	97.9	71.8	91.9	83.0	81.4	63.2	60.0	61.1	79.5 (+10.5)
w/ ProbSurgery	69.3	67.8	89.2	95.1	85.6	92.6	98.0	71.8	81.2	81.7	64.7	91.8	97.9	73.7	92.9	84.3	82.9	72.0	59.5	62.1	80.7 (+11.7)
w/ FEATCAL	70.3	70.9	85.6	94.9	87.0	89.2	97.0	67.3	79.0	82.6	68.7	90.0	97.6	75.3	95.2	84.3	83.2	27.8	28.2	68.3	77.1 (+8.1)
WUDI-Merging	55.1	44.8	59.4	78.5	79.7	82.9	98.1	50.3	49.3	82.0	58.5	77.7	93.4	59.9	90.3	53.1	83.9	35.4	40.0	69.0	67.1
w/ Surgery	57.5	60.1	73.8	93.4	84.9	89.5	98.3	59.3	67.7	82.2	61.0	82.1	95.2	64.8	90.8	57.9	87.7	75.3	78.4	70.4	76.5 (+9.4)
w/ ProbSurgery	57.8	60.7	76.7	94.3	86.2	92.7	98.4	63.0	69.3	82.3	60.6	83.9	95.0	65.4	91.9	57.1	88.6	80.6	77.4	70.1	77.6 (+10.5)
w/ FEATCAL	70.5	71.6	89.1	96.6	93.9	93.5	98.4	69.1	77.9	85.4	68.7	90.0	97.7	79.5	96.6	85.1	91.4	58.1	85.9	71.1	83.5 (+16.4)

Table 11: Full 20-task CLIP-ViT-L/14 results. All numbers are top-1 accuracy (%). Parenthesized values in Avg. report changes over the corresponding upstream merger.

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Flowers	PCAM	FER	Pets	STL10	C100	C10	Food	F-MNIST	EMNIST	KMNIST	Rendered	Avg.
Reference methods																					
Pre-trained	68.2	77.9	71.3	61.2	58.4	50.5	76.3	55.5	79.3	51.2	50.0	93.2	99.4	75.1	95.6	91.2	67.0	12.3	9.7	68.9	65.6
Fine-tuned (STL)	82.8	92.8	97.4	91.1	97.9	99.2	99.8	85.5	97.7	91.1	75.9	95.8	99.2	93.0	99.1	94.7	95.3	95.4	98.3	80.5	93.1
Model merging and post-merging calibration																					
Simple Averaging	70.7	77.7	76.4	75.3	69.5	62.1	93.7	57.7	80.8	73.6	52.7	94.2	99.2	81.7	97.0	90.7	77.4	16.1	10.4	66.1	71.2
w/ Surgery	72.0	78.1	82.1	95.0	75.4	79.2	97.5	70.9	89.0	84.9	66.5	94.8	99.3	83.5	98.1	93.0	86.3	60.5	67.6	70.8	82.2 (+11.0)
w/ ProbSurgery	75.1	78.7	90.7	94.5	74.7	89.5	97.9	75.1	94.2	84.7	67.2	94.6	99.5	84.0	98.0	93.6	87.2	78.4	73.0	73.8	85.2 (+14.1)
w/ FEATCAL	73.4	83.1	84.8	95.6	89.7	82.2	97.2	64.9	89.8	82.9	65.7	95.8	99.1	83.1	98.0	91.6	87.4	36.4	14.4	71.9	79.3 (+8.1)
Task Arithmetic	70.4	74.1	73.9	66.3	65.6	95.1	56.6	69.9	78.6	70.4	55.7	94.2	98.6	79.1	96.6	87.6	80.8	17.6	10.6	63.6	70.3
w/ Surgery	71.7	76.5	79.6	94.3	80.6	98.1	69.3	78.5	88.6	85.8	66.2	94.6	99.0	80.5	97.5	90.1	86.4	66.2	74.3	71.5	82.4 (+12.1)
w/ ProbSurgery	76.4	86.6	89.2	96.1	92.9	91.0	98.5	68.5	93.7	82.9	70.0	95.9	98.9	84.3	98.1	91.3	90.4	50.9	37.6	73.2	83.3 (+13.0)
w/ FEATCAL	77.2	86.9	90.0	96.4	93.8	93.0	98.6	69.1	94.9	82.7	70.4	95.8	99.0	84.9	98.1	92.5	92.0	54.0	51.2	73.2	84.7 (+14.4)
AdaMerging	76.8	89.6	90.3	95.1	91.9	95.2	98.5	76.4	94.5	52.0	69.9	95.5	99.3	82.3	96.5	90.5	85.2	14.0	10.0	81.7	79.3
w/ Surgery	78.3	90.2	92.6	96.7	92.7	96.7	98.6	80.8	96.9	84.3	72.2	95.9	99.3	83.7	96.8	92.0	87.3	55.8	61.7	77.7	86.5 (+7.2)
w/ ProbSurgery	78.4	90.3	94.1	96.7	94.1	96.6	98.8	81.3	97.3	83.3	72.2	95.9	99.4	84.6	97.1	93.0	88.7	73.9	65.1	81.7	88.1 (+8.8)
w/ FEATCAL	80.3	91.1	91.7	96.7	91.7	93.3	98.7	78.7	96.9	82.4	73.0	96.1	99.0	84.8	97.9	91.6	89.0	36.7	10.3	81.8	83.1 (+3.8)
WUDI-Merging	70.3	72.1	73.3	69.7	81.8	84.3	98.1	56.8	85.9	83.7	64.2	94.5	97.5	72.9	95.7	83.5	89.6	33.2	33.3	74.8	75.8
w/ Surgery	71.7	80.5	81.7	95.3	86.4	87.7	98.7	69.2	91.3	88.6	66.2	94.1	98.2	75.1	95.7	85.1	90.7	74.4	88.7	79.0	84.9 (+9.1)
w/ ProbSurgery	72.2	80.8	88.3	95.2	86.8	92.8	98.9	73.2	93.1	88.4	66.2	93.7	98.3	76.0	95.8	85.1	91.0	85.5	89.6	79.7	86.5 (+10.7)
w/ FEATCAL	79.4	89.4	94.3	98.0	96.1	97.2	98.9	77.2	97.2	83.0	74.4	95.7	99.2	88.4	98.7	93.8	93.8	73.8	88.7	79.3	89.8 (+14.0)

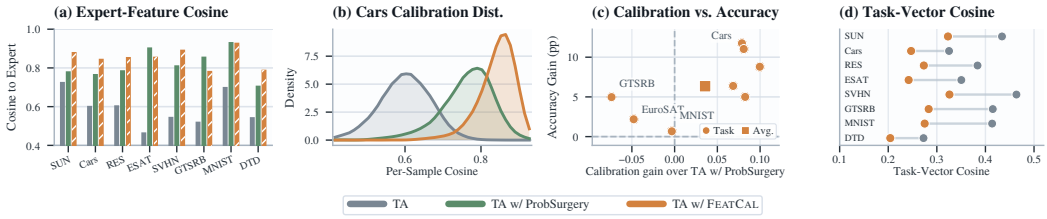


Figure 7: Feature-calibration diagnostics for TA w/ FEATCAL versus TA w/ ProbSurgery on CLIP-ViT-B/32 Task Arithmetic. (a) Task-wise mean cosine between final projected features and same-task expert features. (b) Per-sample expert-feature cosine on Stanford Cars. (c) Expert-feature cosine gain over TA w/ ProbSurgery versus top-1 accuracy gain. (d) Backbone task-vector cosine to same-task expert vectors for TA and TA w/ FEATCAL.

Table 12: Full sample-budget accuracy values corresponding to Fig. 3. Entries are mean top-1 accuracy (%) in the 8-task TA setting and standard deviation over 3 seeds.

Method	$n = 8$	$n = 16$	$n = 32$	$n = 64$	$n = 128$	$n = 256$	$n = 512$
TA w/ Surgery	66.2 ± 1.1	66.7 ± 1.2	69.9 ± 0.3	72.6 ± 0.2	74.9 ± 0.2	76.8 ± 0.2	77.8 ± 0.3
TA w/ ProbSurgery	63.4 ± 1.4	66.1 ± 1.3	69.5 ± 0.6	72.4 ± 0.5	75.7 ± 0.5	78.6 ± 0.2	80.7 ± 0.2
TA w/ FEATCAL	$\mathbf{82.9 \pm 0.3}$	$\mathbf{83.8 \pm 0.2}$	$\mathbf{84.4 \pm 0.0}$	$\mathbf{84.9 \pm 0.1}$	$\mathbf{85.2 \pm 0.1}$	$\mathbf{85.4 \pm 0.1}$	$\mathbf{85.5 \pm 0.0}$

J Additional Diagnostic: Comparison with ProbSurgery

Fig. 7 repeats the diagnostic comparison with the stronger TA w/ ProbSurgery baseline. Panels (a) and (b) show the same average feature-space pattern as in the main text: TA w/ FEATCAL raises the macro-average expert cosine from 0.814 to 0.850 over TA w/ ProbSurgery, improves 8-task average accuracy from 79.1 to 85.5, and increases the Stanford Cars mean expert cosine by 0.079.

Panel (c) again shows that expert-feature cosine is useful but not a complete accuracy explanation. TA w/ FEATCAL improves accuracy over TA w/ ProbSurgery on every task, yet its final-feature expert cosine is lower on EuroSAT, GTSRB, and MNIST; the largest negative cosine change is on GTSRB (-0.074), where accuracy still improves by $+4.97$ points. Panel (d) shows the same feature-space and parameter-space mismatch as Fig. 2: although TA w/ FEATCAL improves final-feature alignment on average, the calibrated backbone has lower same-task expert task-vector cosine than the TA baseline on every task in this run. Thus, the ProbSurgery comparison reinforces the main diagnostic message: feature calibration can improve the deployed final-feature distribution without producing closer task-vector alignment in parameter space, and neither diagnostic alone determines task accuracy.

K Efficiency Experiment Protocol

Task and model setting. The efficiency analysis in Sec. 5.5 uses the FusionBench CLIP-ViT-B/32 8-task TA setting with Task Arithmetic as the shared upstream merger. The Task Arithmetic scaling factor is 0.3, and all post-merging methods start from the same saved TA model. We evaluate Surgery, ProbSurgery, and FEATCAL on the 8 image-classification tasks listed in Sec. 5.2. We follow each method's native FusionBench calibration interface: Surgery and ProbSurgery use their test-time adaptation calibration stream, while FEATCAL uses its calibration split. Final accuracy is always measured by a separate 8-task TA evaluation pass after all calibration jobs finish, so this analysis should be read as an implementation-level post-TA efficiency comparison rather than a controlled split-ablation.

Data budgets and seeds. For each method, n denotes the exact number of calibration examples per task. We use $n \in \{8, 16, 32, 64, 128, 256, 512\}$ and report mean and standard deviation over 3 seeds, 42, 43, and 44. Surgery and ProbSurgery set `max_samples_per_task=n`; FEATCAL sets `num_regmean_examples=n` and truncates the final dataloader batch so that exactly n examples per task are used.

Table 13: Cross-budget calibration-stage resource summary. Values average over all data budgets and 3 seeds per budget; GPU energy is computed per run from average GPU power draw and calibration-stage time. Final 8-task TA evaluation and offline model search are excluded.

Method	Avg. stage time (s)	Avg. GPU energy (Wh)	Avg. peak CPU RSS (GiB)
TA w/ Surgery	209.9	17.0	102.9
TA w/ ProbSurgery	220.3	17.3	95.1
TA w/ FEATCAL	**52.1**	**1.5**	**20.4**

Optimization and dataloading. All 3 post-merging methods use calibration dataloader batch size 16, matching the original FusionBench AdaMerging/Surgery dataloader default, and the profiling run uses 8 dataloader workers. Surgery and ProbSurgery use learning rate 10^{-3}. Surgery and ProbSurgery run for 1000 calibration steps, and their internal validation/evaluation hooks are disabled for the main efficiency comparison. This avoids mixing calibration cost with intermediate model-selection cost. FEATCAL follows the same 8-task TA calibration protocol as the corresponding main result in Sec. 5.1.

Measurement protocol. The profiler separates calibration and evaluation. It first generates or reuses the TA baseline model, then runs all post-merging calibration jobs, and only after all calibration jobs finish evaluates the TA baseline and all calibrated models on the 8-task TA task pool. Calibration efficiency is therefore reported using calibration-only wall-clock time, GPU energy, and CPU RSS columns, while final accuracy is reported from the separate evaluation stage. CPU RSS is the peak resident host memory of the profiled process tree. The profiling run also evaluates saved Surgery and ProbSurgery models at steps $200, 400, 600, 800, 1000$ as an offline model-selection audit. Those model-evaluation passes are excluded from the runtime columns in Tabs. 5 and 13; if model search is reported, it is labeled as an offline model-search variant rather than the default final-model result.

Shared hardware and software. All experiments in this paper use the same single-GPU worker class, including the CLIP main and extended results, the FLAN-T5 results, the feature-calibration diagnostics, the corrupted-calibration study, the sensitivity study, and the efficiency study. Each run uses a single NVIDIA A800-SXM4-80GB GPU with driver 550.163.01, CUDA 12.4 as reported by `nvidia-smi`, an Intel Xeon Platinum 8358 CPU node with 128 logical CPUs, Python 3.12.2, and PyTorch $2.10.0 + \text{cu}128$. No experiment uses distributed training or multi-GPU parallelism. The profiler records calibration-stage wall-clock time, GPU energy, and peak CPU RSS for each sample-budget run in the efficiency study; these measurements are summarized in Tabs. 5 and 13. The logs also record the launch command, environment variables, `nvidia-smi`, CPU information, package versions, and output CSV files used to populate Fig. 3 and Tabs. 5, 12 and 13.

L Corrupted-Calibration Robustness Protocol

The corrupted-calibration analysis in Sec. 5.5 uses the FusionBench CLIP-ViT-B/32 8-task TA setting with Task Arithmetic as the shared upstream merger. All methods start from the same Task Arithmetic model, and final accuracy is measured on the clean test sets from this 8-task TA setting. Only the data consumed by the post-merging calibration stage are corrupted. For FEATCAL, this is the 256-image-per-task calibration set; for Surgery and ProbSurgery, this is each method's native calibration stream under the same per-task budget.

We apply a single ImageNet-C-style severity-3 corruption at a time: Gaussian noise, motion blur, or fog. The uncalibrated Task Arithmetic baseline has no calibration stream, so it is reported only in the clean column. All calibrated rows in Tab. 6 report the final calibrated models from the corresponding corrupted-data run.

Table 14: Individual MergeBench expert results on Llama-family models.

Series / Model	Method	MATH-500	GSM8K	HumanEval+@1	MBPP+@1	IFEval	ARC-C	Avg.
Llama-3.2-3B-Instruct	Base	47.8	72.6	49.4	56.6	67.3	73.1	61.1
	Coding Expert	27.4	55.5	54.9	57.1	49.5	71.5	52.7
	Math Expert	46.4	80.6	48.2	55.3	53.0	71.1	59.1
	Instruction Expert	49.0	73.6	54.3	56.1	69.7	72.7	62.6
	Fine-tuned (STL)	49.0	80.6	54.9	57.1	69.7	73.1	64.1
Llama-3.1-8B-Instruct	Base	48.6	83.5	62.2	63.0	72.8	80.9	68.5
	Coding Expert	21.2	53.4	66.5	59.0	43.8	76.5	53.4
	Math Expert	47.4	83.9	22.0	36.5	15.5	47.4	42.1
	Instruction Expert	52.6	85.3	67.1	63.8	72.5	78.8	70.0
	Fine-tuned (STL)	52.6	85.3	67.1	63.8	72.8	80.9	70.4

M Additional MergeBench Expert Results

We report the individual MergeBench expert references omitted from the compact main-text table. All entries in Tab. 14 are percentages, and the average is the mean over MATH-500, GSM8K, HumanEval+, MBPP+, IFEval, and ARC-Challenge. HumanEval+ and MBPP+ report pass@1.

N Broader Context and Future Directions

Our experiments focus on post-merging calibration after a base merger has already produced a single deployable model, but this setting is part of a wider model-fusion agenda. Recent surveys frame model fusion as a route toward scalable, sustainable, and more broadly accessible AI systems [60, 61]. For FEATCAL, an immediate next step is to understand how closed-form calibration scales with model size, task count, and available compute. Scaling-law and parameter-management studies for large-language-model merging provide useful tools for this question, especially when calibration must fit into a constrained memory or budget envelope [62, 63].

Another direction is to extend feature calibration beyond accuracy-oriented benchmarks. Preference-oriented fusion and distillation suggest that merged models can be shaped by preference signals rather than only task labels or expert features [64, 65], while logits-level distillation gives a complementary way to transfer relational information between models [66]. As calibrated merged models move into interactive and multimodal settings, evaluation should also account for response uncertainty and possible misleading contexts [67]. Finally, systems support remains important: decentralized evaluation pipelines and low-precision reasoning-model training recipes point to deployment settings where calibrated model fusion should be efficient, auditable, and inexpensive to run [68, 69]. These directions are complementary to the layer-wise calibration mechanism studied here and ask how post-merging feature calibration should be scaled, evaluated, and deployed.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the method, theory, scope, and main empirical claims. These claims are supported by the reported CLIP, FLAN-T5, and MergeBench experiments within the stated post-merging calibration setting.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in App. A, including the empirical scope, hyperparameter selection, and dependence on task calibration data.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper states the assumptions for its theoretical results and provides formal proofs in Apps. C to E. The main text also gives the key definitions and propositions used by the method.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the algorithm, benchmarks, calibration protocol, hyperparameters, and evaluation settings in Sec. 5.1 and Alg. 1. Additional robustness and efficiency protocol details appear in Apps. K and L.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The public code repository is linked on the first page. The experiments use public benchmarks and cited public datasets, and the repository provides implementation code and instructions for the CLIP-ViT-B/32 8-task Task Arithmetic setting.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setup specifies the model families, tasks, baselines, calibration sample budget, key hyperparameters, and final evaluation protocol in Sec. 5.1. Further details for the diagnostic and efficiency experiments are given in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The data-budget efficiency study reports mean and standard deviation over three seeds in Tab. 12. However, the main CLIP, FLAN-T5, and MergeBench result tables do not report error bars, confidence intervals, or statistical significance tests.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The efficiency protocol reports the GPU, CPU, driver, CUDA, Python, and PyTorch environment in App. K. The efficiency study also reports wall-clock time, GPU energy, and peak CPU RSS for the profiled calibration runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research uses public benchmarks and does not involve new human-subject studies or private data collection. The authors have reviewed the NeurIPS Code of Ethics and believe the work conforms to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper focuses on the methodology and empirical evaluation of post-merging calibration. It does not provide a dedicated analysis of both positive and negative societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new high-risk data or models such as scraped datasets, image generators, or pretrained language models. The work calibrates existing benchmark models for experimental study.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper cites the benchmarks, datasets, and model families used in the experiments. It does not explicitly list the licenses, versions, URLs, and terms of use for every existing asset.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces implementation code as a research asset, with the public repository linked on the first page. The repository provides documentation, usage instructions, and licensing information.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing experiments or research with human subjects. No participant instructions, screenshots, or compensation details are therefore applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing experiments or research with human subjects. IRB approval or equivalent review is therefore not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used only to polish the manuscript writing. They were not used as an important, original, or non-standard component of the core method, experiments, data construction, or scientific claims.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.