A. Theoretical Derivations

A.1. Derivation of the Quadratic Objective

We derive the quadratic objective in Eq. (2) from the layer-wise discrepancy objective in Eq. (1)

Recall the merging objective:

$$\min_{\bar{W}} \sum_{t\in[T]} \mathbb{E}_{x\sim\mathcal{D}_t} \left\| f(\bar{W}, x) - f(W_t, x) \right\|_2^2. \tag{20}$$

Under the local linear approximation $f(W, x) \approx Wx$, the objective becomes

$$\sum_{t\in[T]} \mathbb{E}_{x\sim\mathcal{D}_t} \left\| (\bar{W} - W_t)x \right\|_2^2. \tag{21}$$

For any matrix A and vector x, we have

$$\|Ax\|_2^2 = x^\top A^\top Ax = \mathrm{Tr}(Axx^\top A^\top).$$

Applying this identity with $A = \bar{W} - W_t$, we obtain

$$\begin{aligned} \mathbb{E}_{x\sim\mathcal{D}_t} \|(\bar{W} - W_t)x\|_2^2 &= \mathbb{E}_{x\sim\mathcal{D}_t} \mathrm{Tr}\big((\bar{W} - W_t)xx^\top(\bar{W} - W_t)^\top\big) &(22)\\ &= \mathrm{Tr}\big((\bar{W} - W_t)\,\mathbb{E}_{x\sim\mathcal{D}_t}[xx^\top]\,(\bar{W} - W_t)^\top\big)\,. &(23)\end{aligned}$$

Defining

$$\Sigma_t := \mathbb{E}_{x\sim\mathcal{D}_t}[xx^\top],$$

we arrive at

$$L(\bar{W}) = \sum_{t\in[T]} \mathrm{Tr}\big((\bar{W} - W_t)\Sigma_t(\bar{W} - W_t)^\top\big)\,, \tag{24}$$

which is exactly Eq. (2) in the main paper.

A.2. Derivation of the Closed-Form Solution

In this section, we derive the closed-form solution for the optimal merged weights $\bar{W}$ by minimizing the quadratic objective function from Eq. (2).

To find the minimum, we compute the gradient of $\mathcal{L}(\bar{W})$ with respect to the matrix $\bar{W}$ and set it to zero. First, let's expand the term inside the trace:

$$\begin{aligned} (\bar{W} - W_t)\Sigma_t(\bar{W} - W_t)^\top &= (\bar{W} - W_t)\Sigma_t(\bar{W}^\top - W_t^\top) \\ &= \bar{W}\Sigma_t\bar{W}^\top - \bar{W}\Sigma_t W_t^\top - W_t\Sigma_t\bar{W}^\top + W_t\Sigma_t W_t^\top. &(25)\end{aligned}$$

Using the linearity of the trace and the sum, we can differentiate the loss term by term with respect to $\bar{W}$. We use the following standard matrix calculus identities for the gradient of a trace:

- $\nabla_X \mathrm{Tr}(XAX^\top) = X(A + A^\top)$
- $\nabla_X \mathrm{Tr}(XB) = B^\top$
- $\nabla_X \mathrm{Tr}(AX^\top) = A$

Given that the covariance matrix Σ_t is symmetric (i.e., $\Sigma_t = \Sigma_t^\top$), the first identity simplifies to $\nabla_X \mathrm{Tr}(X\Sigma_t X^\top) = 2X\Sigma_t$.

Applying these rules, the gradient of the loss function is:

$$\begin{aligned}
\nabla_{\bar{W}}\mathcal{L}(\bar{W}) &= \nabla_{\bar{W}} \sum_{t\in[T]} \operatorname{Tr}\left[\bar{W}\Sigma_t\bar{W}^\top - \bar{W}\Sigma_t W_t^\top - W_t\Sigma_t\bar{W}^\top + W_t\Sigma_t W_t^\top\right] \\
&= \sum_{t\in[T]} \nabla_{\bar{W}} \operatorname{Tr}\left[\bar{W}\Sigma_t\bar{W}^\top - \bar{W}\Sigma_t W_t^\top - W_t\Sigma_t\bar{W}^\top\right] \\
&= \sum_{t\in[T]} \left(2\bar{W}\Sigma_t - (\Sigma_t W_t^\top)^\top - W_t\Sigma_t\right) \\
&= \sum_{t\in[T]} \left(2\bar{W}\Sigma_t - W_t\Sigma_t^\top - W_t\Sigma_t\right) \\
&= \sum_{t\in[T]} \left(2\bar{W}\Sigma_t - 2W_t\Sigma_t\right) \quad (\text{since } \Sigma_t = \Sigma_t^\top) \\
&= 2\sum_{t\in[T]} (\bar{W}\Sigma_t - W_t\Sigma_t). \qquad (26)
\end{aligned}$$

Setting the gradient to zero to find the minimum:

$$\begin{aligned}
2\sum_{t\in[T]} (\bar{W}\Sigma_t - W_t\Sigma_t) &= 0 \\
\sum_{t\in[T]} \bar{W}\Sigma_t &= \sum_{t\in[T]} W_t\Sigma_t. \qquad (27)
\end{aligned}$$

Since $\bar{W}$ is common to all terms in the left-hand summation, we can factor it out:

$$\bar{W}\left(\sum_{t\in[T]} \Sigma_t\right) = \sum_{t\in[T]} W_t\Sigma_t. \qquad (28)$$

Finally, to isolate $\bar{W}$, we right-multiply both sides by the inverse of the aggregated covariance matrix:

$$\bar{W} = \left(\sum_{t\in[T]} W_t\Sigma_t\right)\left(\sum_{t\in[T]} \Sigma_t\right)^{-1}. \qquad (29)$$

This gives the closed-form solution for the optimal merged weights, which matches Eq. (3) in the main paper.

A.3. Proof of Theorem 1

We provide a detailed derivation showing that the second-order structure of fine-tuning updates is governed by the input second-moment matrix.

Let $W_0 \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ denote the pretrained weights, and let W_t be the task-specific weights obtained after fine-tuning on dataset $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{N_t}$. Define the weight displacement

$$\Delta W_t = W_t - W_0.$$

Assumptions. We adopt the following standard assumptions.

(A1) **Local linearization with squared loss.** The loss is $\mathcal{L}(x, y; W) = \|Wx - y\|_2^2$, and its gradient at W_0 is

$$\nabla_W \mathcal{L}(x, y; W_0) = 2(W_0x - y)x^\top = 2e\,x^\top,$$

where $e = W_0x - y \in \mathbb{R}^{d_{\text{out}}}$.

(A2) **i.i.d. sampling.** Samples (x_i, y_i) are drawn i.i.d. from $\mathcal{D}_t$.

(A3) **Approximately zero-mean first-order update.**

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_t}[e\,x^\top] \approx 0.$$

(A4) **Residual-input decoupling.**

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_t}\left[(e^\top e)\,xx^\top\right] \approx \mathbb{E}_{(x,y)\sim\mathcal{D}_t}[e^\top e]\,\mathbb{E}_{x\sim\mathcal{D}_t}[xx^\top].$$

Let

$$\Sigma_t = \mathbb{E}_{x\sim\mathcal{D}_t}[xx^\top]$$

denote the task-specific input second-moment matrix.

Empirical validation of Assumption (A3). To validate Assumption (A3), we examine the empirical distribution of the entries of ΔW_t across architectures and layer types. As shown in Fig. 3, these distributions are consistently centered around zero and exhibit approximately Gaussian shapes.

Under the linearized update model, a near-zero mean distribution of the entries of ΔW_t is consistent with $\mathbb{E}[ex^\top] \approx 0$. This empirical observation supports Assumption (A3) and suggests that the dominant task-specific information is captured by the second-order structure of the parameter updates.

Step 1: Linearized update expression. Under the local linearization in Assumption (A1), the contribution of a single sample to the first-order update is

$$\Delta W^{(i)} = -2\eta\, e_i x_i^\top.$$

Aggregating these first-order contributions over the fine-tuning trajectory gives

$$\Delta W_t \approx -2\eta \sum_{i=1}^{N_t} e_i x_i^\top. \tag{30}$$

Let

$$G_i = e_i x_i^\top, \qquad c = -2\eta,$$

then

$$\Delta W_t = c \sum_{i=1}^{N_t} G_i.$$

Step 2: Second-order structure of the update. We analyze the column Gram matrix of the task vector:

$$\Delta W_t^\top \Delta W_t = c^2 \left(\sum_{i=1}^{N_t} G_i\right)^\top \left(\sum_{j=1}^{N_t} G_j\right) \tag{31}$$

$$= c^2 \sum_{i,j} G_i^\top G_j. \tag{32}$$

Taking expectation gives

$$\mathbb{E}\left[\Delta W_t^\top \Delta W_t\right] = c^2 \sum_{i,j} \mathbb{E}[G_i^\top G_j]. \tag{33}$$

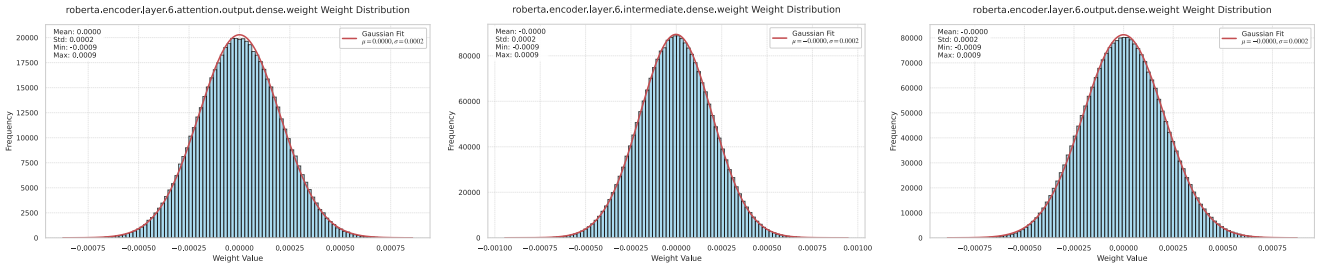
Under Assumptions (A2) and (A3), the cross terms vanish at leading order:

$$\mathbb{E}[G_i^\top G_j] \approx 0, \qquad i \neq j.$$

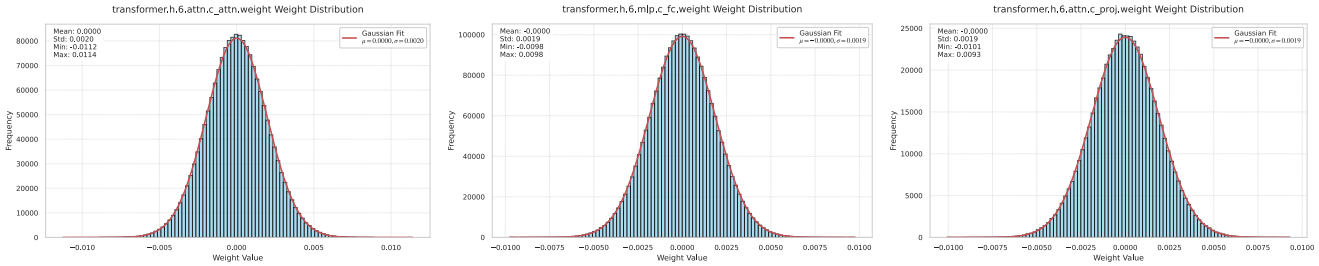
Therefore, Eq. (33) reduces to

$$\mathbb{E}\left[\Delta W_t^\top \Delta W_t\right] \approx c^2 N_t\, \mathbb{E}[G^\top G], \tag{34}$$

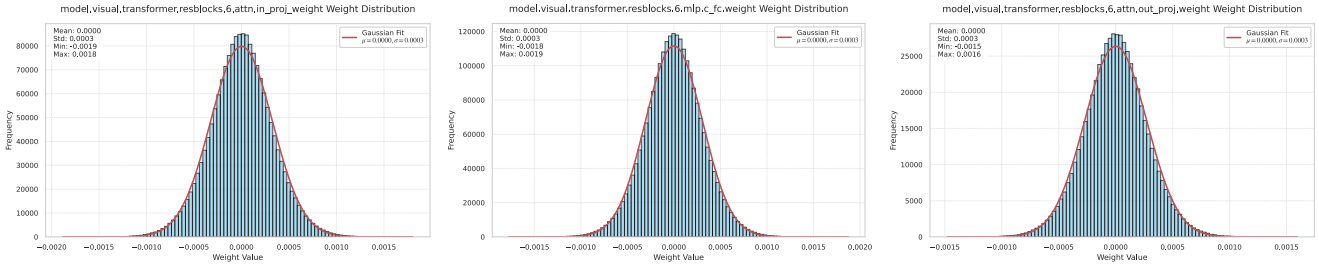
where $G = ex^\top$ denotes a generic single-sample contribution.



(a) RoBERTa-Base (Attention Out Layer) (b) RoBERTa-Base (MLP Layer) (c) RoBERTa-Base (MLP Out Layer)



(d) GPT-2 (Attention Layer) (e) GPT-2 (MLP Layer) (f) GPT-2 (Attention Proj Layer)



(g) ViT-B/16 (Attention In Layer) (h) ViT-B/16 (MLP Layer) (i) ViT-B/16 (Attention Out Layer)

Figure 3. **Empirical distributions of ΔW_t across architectures, layer types, and tasks.** Each subplot shows the histogram of the entries of ΔW_t for one representative layer from RoBERTa-Base, GPT-2, and ViT-B/16. The distributions are consistently zero-centered and approximately Gaussian, supporting the local zero-mean assumption used in our theoretical analysis and suggesting that the dominant task-specific information is captured by the second-order structure of the parameter updates.

Step 3: Single-sample second-order structure. For $G = ex^\top$, we have

$$G^\top G = (ex^\top)^\top (ex^\top) \tag{35}$$
$$= xe^\top e\, x^\top \tag{36}$$
$$= (e^\top e)\, xx^\top. \tag{37}$$

Taking expectation yields

$$\mathbb{E}[G^\top G] = \mathbb{E}\big[(e^\top e)\, xx^\top\big] \tag{38}$$
$$\approx \mathbb{E}[e^\top e]\, \mathbb{E}[xx^\top] \qquad \text{(A4)} \tag{39}$$
$$= \sigma_e^2 \Sigma_t, \tag{40}$$

where

$$\sigma_e^2 = \mathbb{E}[e^\top e].$$

Substituting into Eq. (34), we obtain

$$\mathbb{E}\big[\Delta W_t^\top \Delta W_t\big] \approx c^2 N_t \sigma_e^2 \Sigma_t. \tag{41}$$

Hence,

$$\mathbb{E}\big[\Delta W_t^\top \Delta W_t\big] \propto \Sigma_t, \tag{42}$$

which proves Theorem 1. □

Remark on the practical estimator. In practice, only a single realization of ΔW_t is available for each task. Therefore, rather than estimating the full population covariance of the fine-tuning trajectory, we directly construct a stable second-order proxy from the observed task vector.

Following Eq. (5) in the main text, we define the centered task vector

$$\widetilde{W}_t = \Delta W_t - \mathbf{1}\mu_t^\top, \qquad \mu_t = \frac{1}{d_{\text{out}}} \Delta W_t^\top \mathbf{1},$$

where $\mathbf{1} \in \mathbb{R}^{d_{\text{out}}}$ is the all-ones vector. We then use the centered column Gram matrix

$$\widehat{\Sigma}_t \propto \widetilde{W}_t^\top \widetilde{W}_t$$

as the task-specific second-order proxy.

This centering removes the rank-one mean component across output dimensions, so that $\widehat{\Sigma}_t$ better captures the task-specific second-order structure emphasized by the update. The proportionality constant affects only the overall scale and is handled later by the adaptive normalization in Sec. 4.2.

A.4. Additional statistics for adaptive covariance normalization

In practice, we compute the Frobenius norm via the trace identity

$$\|\Delta W_t\|_F^2 = \mathrm{Tr}(\Delta W_t^\top \Delta W_t).$$

Therefore, the heterogeneity coefficient introduced in Sec. 4.2 can be implemented equivalently as

$$\gamma = \frac{\mathrm{Var}_t\big[\log \mathrm{Tr}\big(\Delta W_t^\top \Delta W_t\big)\big]}{\big(\mathbb{E}_t\big[\log \mathrm{Tr}\big(\Delta W_t^\top \Delta W_t\big)\big]\big)^2}. \tag{43}$$

Since

$$\|\Delta W_t\|_F^2 \equiv \mathrm{Tr}(\Delta W_t^\top \Delta W_t),$$

Eq. (43) is mathematically identical to the original definition and aligns exactly with our implementation.

The coefficient γ serves as a lightweight statistic for diagnosing task-scale heterogeneity before adaptive normalization. Intuitively, small γ indicates that task updates have relatively aligned energy scales, whereas large γ reflects substantial variation across tasks and signals stronger second-order mismatch.

In the main text, we reported results for RoBERTa-Base (8 tasks) and ViT-B/16 (14 tasks), where γ reliably separates relatively homogeneous from highly heterogeneous task sets. To provide more comprehensive evidence for the behavior of γ across architectures and task counts, we present additional statistics below.

RoBERTa-Large on 8 tasks. Fig. 4 shows the distribution of $\{\tau_t\}$ and the resulting γ for RoBERTa-Large across the same 8 GLUE-style tasks used in the main paper. Similar to RoBERTa-Base, the heterogeneity score is high ($\gamma > 0.3$), confirming a substantial degree of task specificity and scale mismatch across tasks.

GPT-2 on 7 tasks. Fig. 5 reports the corresponding values for GPT-2, where we observe a noticeably larger spread in $\{\tau_t\}$ and a higher γ than for RoBERTa. This aligns with the empirical difficulty of merging GPT-2 task vectors (Sec. 5), as greater heterogeneity makes simple isotropic corrections less reliable and further motivates adaptive normalization.

Effect of increasing number of tasks (ViT-L/14). To examine how heterogeneity evolves as more tasks are included, we evaluate γ for ViT-L/14 under multiple task-set sizes. In Fig. 6 and Fig. 7, we show the results for 8 and 20 tasks, representing the low- and high-diversity extremes. The heterogeneity coefficient increases noticeably when moving from 8 to 20 tasks, consistent with the overall monotonic trend observed across all task counts. This behavior supports our claim that task diversity naturally grows as more datasets are merged, reinforcing the need for the adaptive scaling mechanism introduced in Sec. 4.2.

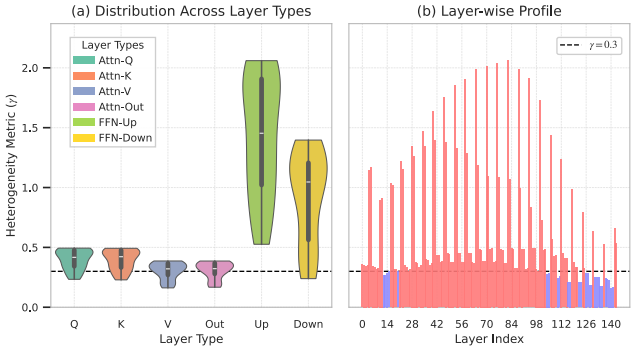


Figure 4. **RoBERTa-Large (8 tasks).** Trace statistics $\{\tau_t\}$ and heterogeneity coefficient γ.

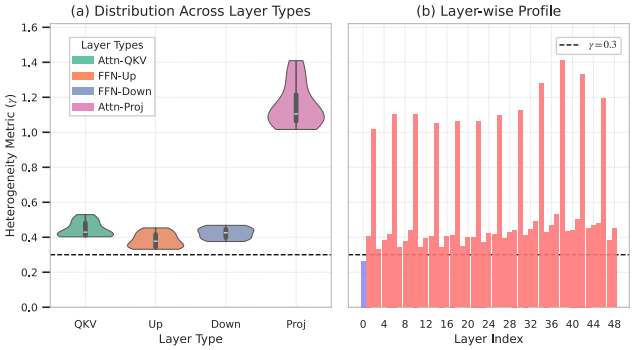


Figure 5. **GPT-2 (7 tasks).** Larger heterogeneity γ than RoBERTa, indicating stronger mismatch across task updates.

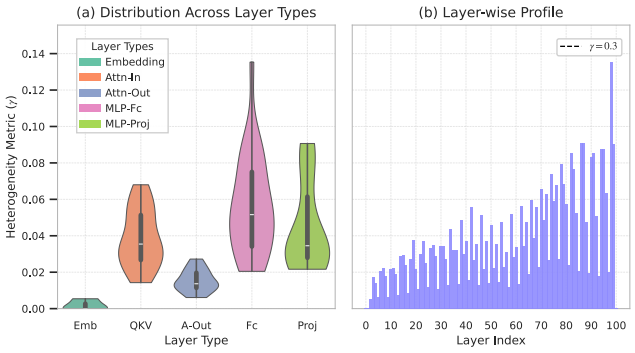


Figure 6. **ViT-L/14 (8 tasks).** Moderate heterogeneity γ, indicating manageable variation across task updates at smaller task scales.

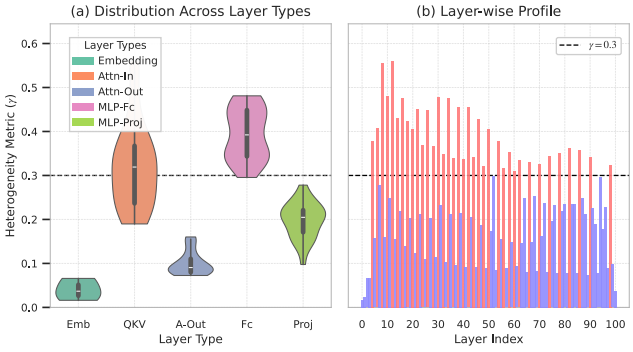


Figure 7. **ViT-L/14 (20 tasks).** Substantially larger heterogeneity γ, showing increasing mismatch across task updates as the number of merged tasks grows.

Summary. Across all architectures, γ consistently captures the heterogeneity of the task set:

- small γ indicates relatively well-aligned task scales, for which simple merging procedures are often sufficient;
- large γ reveals pronounced variability across tasks, motivating the adaptive covariance normalization used in `ACE-Merging`.

These results further support the robustness of γ as a practical criterion for activating heterogeneity-aware merging strategies.

B. Computational Complexity and Practical Overhead

We compare `ACE-Merging` with two representative data-free baselines: the SVD-based TSV-M method [19] and the gradient-based WUDI-merging approach [23]. In addition to asymptotic complexity, we also report practical merge-time and peak GPU memory, following the quantitative profiling added in the rebuttal.

We use the following notation. Let T be the number of task vectors and L be the number of merged layers. For a given layer, let each task update matrix ΔW_t have shape $d_{\text{out}} \times d_{\text{in}}$. For simplicity in big-$O$ notation, we define

$$n = \max(d_{\text{out}}, d_{\text{in}})$$

and report per-layer costs in terms of n.

TSV-M. TSV-M performs multiple singular value decompositions per layer:

- T SVDs on the task matrices $\Delta W_t \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, costing

$$O\big(T \cdot \min(d_{\text{out}}^2 d_{\text{in}},\, d_{\text{out}} d_{\text{in}}^2)\big) = O(Tn^3).$$

- Two additional SVDs on aggregated matrices, contributing a lower-order cost of $O(n^3)$.

Therefore, the total per-layer complexity is

$$O(Tn^3),$$

and the overall complexity across all L layers is

$$O(\text{TSV-M}) = O(LTn^3). \tag{44}$$

Gradient-based merging (WUDI-merging). WUDI-merging optimizes the merged weights by iterative gradient descent. Let K denote the number of optimization steps. Each step requires evaluating the loss and its gradient over all T task vectors. The dominant computation in each forward-backward pass consists of element-wise operations on tensors of size

$$T \times d_{\text{out}} \times d_{\text{in}},$$

which costs

$$O(Td_{\text{out}}d_{\text{in}}) = O(Tn^2)$$

per layer. Hence the total complexity scales as

$$O(\text{WUDI}) = O(KLTn^2). \tag{45}$$

Although the per-step cost is lower than cubic-time matrix decompositions, the dependence on the optimization horizon K makes WUDI computationally expensive in practice, especially when hundreds or thousands of iterations are required.

ACE-Merging. `ACE-Merging` avoids iterative optimization and computes the merge in closed form. Its per-layer complexity is dominated by three stages:

- **Second-order proxy aggregation.** For each task, we form the centered column Gram proxy $\widetilde{W}_t^\top \widetilde{W}_t$ (equivalently, up to centering, $\Delta W_t^\top \Delta W_t$), whose dominant matrix multiplication costs

  $$O(d_{\text{out}}d_{\text{in}}^2).$$

  Aggregating over all T tasks gives

  $$O(Td_{\text{out}}d_{\text{in}}^2).$$

- **Closed-form solve.** After aggregation, the merged weights are obtained through a single inverse (or equivalently a linear solve) on a $d_{\text{in}} \times d_{\text{in}}$ system matrix, costing

  $$O(d_{\text{in}}^3).$$

- **Spectral refinement (optional).** In the heterogeneous regime, ACE performs one additional SVD on a $d_{\text{out}} \times d_{\text{in}}$ matrix, costing

  $$O\big(\min(d_{\text{out}}^2 d_{\text{in}},\, d_{\text{out}}d_{\text{in}}^2)\big)\,.$$

  Combining these terms and using $n = \max(d_{\text{out}}, d_{\text{in}})$, the per-layer complexity is

  $$O(Tn^3 + n^3) = O(Tn^3),$$

so the total complexity across all L layers is

$$O(\texttt{ACE-Merging}) = O(LTn^3). \tag{46}$$

Asymptotic comparison. ACE-Merging matches the asymptotic complexity of SVD-based methods such as TSV-M, while retaining a true closed-form formulation. Compared with gradient-based approaches such as WUDI-merging, ACE avoids the additional multiplicative factor K and is therefore theoretically more scalable when many optimization steps are needed.

Practical overhead: time and peak memory. Beyond asymptotic complexity, we also profiled merge-time and peak GPU memory on a single NVIDIA A800 (80GB), as reported in the rebuttal. The results are summarized in Table 6. ACE-Merging is consistently the fastest method in both settings: for ViT-L/14 (8 tasks), it reduces wall-clock time from 126.68s (WUDI, 1000 iterations) and 88.39s (TSV-M) to 4.32s; for GPT-2 (7 tasks), it reduces merge time from 62.71s and 25.87s to 4.70s. Its peak memory is also much lower than iterative baselines such as WUDI and Iso-CTS, although TSV-M remains slightly more memory-efficient in these particular measurements. These empirical results are consistent with the theoretical analysis above: ACE is dominated by a one-pass second-order aggregation plus a single inversion/SVD, whereas WUDI additionally scales with the optimization horizon K.[2]

[2]The table reports these measurements on a single A800 (80GB).

Table 6. **Measured merge-time and peak GPU memory** on a single A800 (80GB), reproduced from the rebuttal.

Method	ViT-L/14 (8 tasks)		GPT-2 (7 tasks)	
	Time (s)↓	VRAM (GB)↓	Time (s)↓	VRAM (GB)↓
WUDI (1000 iters)	126.68	6.22	62.71	4.71
TSV-M	88.39	1.91	25.87	0.77
Iso-CTS	118.58	11.12	42.83	10.54
ACE (ours)	**4.32**	2.04	**4.70**	1.02

Summary. Overall, the comparison suggests the following:

- **vs. WUDI-merging:** ACE enjoys both a theoretical advantage (no dependence on the optimization horizon K) and a substantial practical speedup in measured wall-clock time.
- **vs. TSV-M:** ACE has the same asymptotic order $O(LTn^3)$, but in practice runs much faster in our experiments, while requiring only slightly higher peak memory.

Therefore, ACE-Merging offers a favorable trade-off between closed-form stability, runtime efficiency, and practical scalability.

B.1. Analysis of Limiting Cases and Hyperparameters

We analyze several limiting regimes of `ACE-Merging` to clarify how its main hyperparameters affect the resulting merge. These limiting cases should be interpreted as *sanity checks* of the formulation rather than practical operating regimes. In particular, as emphasized in the rebuttal, the regularization strength ϵ is introduced primarily for numerical stability of the closed-form solve, while the default settings used throughout the experiments are fixed to $\tau = 0.3$ and $k_{\text{frac}} = 0.3$, with robustness observed under moderate variations of these values.

Limit $\epsilon \to \infty$ (strong Tikhonov regularization). The hyperparameter `eps` (ϵ) controls the strength of Tikhonov regularization applied to each task-specific second-order proxy:

$$\widehat{\Sigma}_t \leftarrow \widehat{\Sigma}_t + \epsilon_t I.$$

As $\epsilon \to \infty$, the isotropic regularizer dominates the learned second-order structure. Two limiting behaviors arise depending on whether the task set is classified as homogeneous or heterogeneous.

- **Low heterogeneity ($\gamma \le \gamma_{\text{thresh}}$).** In this regime, the same regularization coefficient is used for all tasks, i.e. $\epsilon_t = \epsilon$, so

  $$\widehat{\Sigma}_t \approx \epsilon I.$$

  Substituting into the closed-form merge gives

  $$\bar{W} = \left(\sum_{t=1}^{T} W_t \widehat{\Sigma}_t\right)\left(\sum_{t=1}^{T} \widehat{\Sigma}_t\right)^{-1} \xrightarrow[\epsilon\to\infty]{} \frac{1}{T}\sum_{t=1}^{T} W_t.$$

  Thus, in the infinitely isotropic limit, `ACE-Merging` reduces to **simple averaging** (task arithmetic).
- **High heterogeneity ($\gamma > \gamma_{\text{thresh}}$).** In this regime, regularization is energy-adjusted across tasks, yielding

  $$\epsilon_t = \frac{\epsilon}{\tau_t}, \qquad \tau_t = \mathrm{Tr}(W_t^\top W_t),$$

  so that

  $$\widehat{\Sigma}_t \approx \frac{\epsilon}{\tau_t} I.$$

  The merge then becomes

  $$\bar{W} \xrightarrow[\epsilon\to\infty]{} \frac{\sum_{t=1}^{T} \frac{1}{\tau_t} W_t}{\sum_{t=1}^{T} \frac{1}{\tau_t}},$$

  i.e. a **weighted average** in which tasks with larger Frobenius norms receive smaller effective weights.

These two limits help connect `ACE-Merging` to simpler merging rules, but they do *not* describe the practical regime used in our experiments. As clarified in the rebuttal, ϵ is only a Tikhonov term for stable inversion, and the degenerate limits discussed here are mainly included as sanity checks; the default settings used in the paper are away from these extremes.

Role of the heterogeneity threshold γ_{thresh}. This threshold determines whether the method enters the heterogeneity-aware branch, namely whether adaptive trace normalization and spectral refinement are activated.

- If γ_{thresh} is very large, the condition $\gamma > \gamma_{\text{thresh}}$ is rarely satisfied. In that case, `ACE-Merging` remains in the low-heterogeneity branch and behaves like a regularized closed-form multi-task regression method without task-wise scale balancing.
- If $\gamma_{\text{thresh}} \to 0$, the heterogeneous branch is always selected, so trace normalization and subsequent refinement are applied for essentially all task sets.

In practice, however, γ_{thresh} is not tuned per setting. We use a fixed default threshold $\tau = 0.3$ across both vision and language experiments, and the sensitivity results show that performance remains stable under moderate variation of this threshold.

Role of the spectral fraction k_{frac}. This hyperparameter controls the rank of the spectral refinement applied after the preliminary closed-form merge. Let

$$\bar{W}_{\text{fused}} = USV^{\top}, \qquad S = \text{diag}(\sigma_1, \sigma_2, \ldots),$$

and define the low-rank refinement

$$\Delta W_{\text{refine}} = \sigma_{\text{iso}}\, U_{:,1:k} V_{:,1:k}^{\top}, \qquad \sigma_{\text{iso}} = \frac{1}{k}\sum_{i=1}^{k} \sigma_i,$$

where

$$k = k_{\text{frac}} \cdot \min(d_{\text{in}}, d_{\text{out}}).$$

- As $k_{\text{frac}} \to 0$, we have $k \to 0$, so the refinement vanishes and the final merge reduces to the preliminary closed-form solution:
$$\bar{W} \to \bar{W}_{\text{pre}}.$$
- As $k_{\text{frac}} \to 1$, the refinement spans the full singular basis, producing a full-rank spectral correction whose selected singular values are replaced by their mean. This yields a *uniformized spectral correction* rather than a full isotropization of the merged weight.

Importantly, the purpose of k_{frac} is not to introduce an arbitrary post-hoc modification, but to control how strongly the method corrects the spectral collapse observed in the preliminary solution $\bar{W}_{\text{pre}}$. As clarified in the rebuttal, the motivation is explicitly *pre-refinement*: Fig. 2 shows that $\bar{W}_{\text{pre}}$ already captures the correct dominant subspace, but exhibits an abnormally collapsed spectrum relative to the expert models. Spectral refinement is therefore a targeted correction that restores a healthier spectral profile while preserving the dominant directions, rather than a circular justification based on the final merged model.

Consistent with this interpretation, the rebuttal also notes that $k_{\text{frac}} = 0.3$ is used as a fixed default across all experiments, and that moderate changes in k_{frac} lead to only minor performance variation.

Summary. These limiting regimes clarify how `ACE-Merging` interpolates between simpler and more structured merging rules:

- strong isotropic regularization recovers simple averaging or energy-weighted averaging;
- γ_{thresh} controls whether heterogeneity-aware normalization and refinement are activated;
- k_{frac} controls the strength of the targeted spectral correction applied to $\bar{W}_{\text{pre}}$.

Together, these observations show that the hyperparameters of `ACE-Merging` have clear functional interpretations, while the practical defaults remain fixed and robust across architectures and modalities.

C. ACE-Merging Performance on ViT Models with Increasing Task Coverage

We report the full performance of `ACE-Merging` on three vision transformer backbones: ViT-B/32, ViT-B/16, and ViT-L/14, evaluated under increasing numbers of downstream tasks. These results complement the heterogeneity analysis in Sec. 4.2 and empirically demonstrate that `ACE-Merging` remains stable as task diversity grows. All values are test classification accuracies unless otherwise specified.

Each table presents side-by-side results for 8, 14, and 20 tasks, and includes the average accuracy over all tasks for each backbone and setting.

Table 7. **ACE-Merging accuracy (%) on ViT-B/32, ViT-B/16, and ViT-L/14 across increasing numbers of tasks.**

Dataset	ViT-B/32			ViT-B/16			ViT-L/14		
	8	14	20	8	14	20	8	14	20
MNIST	99.46	99.22	92.65	99.44	99.32	96.40	99.58	99.59	96.65
Cars	71.67	68.59	61.90	81.54	78.24	72.40	90.34	88.46	85.37
DTD	89.63	82.45	75.80	90.69	83.19	76.33	96.97	93.19	88.67
EuroSAT	95.81	88.37	83.70	97.78	96.96	94.56	99.52	99.30	98.22
GTSRB	92.44	85.49	76.43	92.94	88.20	80.74	97.34	97.42	93.49
RESISC45	86.79	81.51	77.92	90.73	88.75	85.97	94.76	93.25	92.11
SUN397	73.06	70.75	68.63	76.57	74.04	72.07	81.83	78.62	76.72
SVHN	94.19	90.10	82.89	95.08	93.22	88.40	96.20	95.61	91.80
PCAM	—	84.09	83.75	—	86.14	85.02	—	85.82	85.77
CIFAR100	—	72.77	68.79	—	78.41	75.64	—	86.64	83.75
STL10	—	97.50	96.73	—	98.15	97.58	—	99.41	99.34
Oxford-IIIT Pet	—	90.54	88.20	—	93.98	92.86	—	96.43	96.18
Flowers102	—	73.04	70.19	—	79.07	76.24	—	87.32	84.57
FER2013	—	67.62	63.46	—	68.04	63.81	—	73.92	69.95
CIFAR10	—	—	93.54	—	—	95.55	—	—	98.05
Food101	—	—	80.71	—	—	88.47	—	—	93.98
RenderedSST2	—	—	72.43	—	—	77.10	—	—	85.56
EMNIST	—	—	98.07	—	—	97.82	—	—	99.52
Fashion-MNIST	—	—	83.50	—	—	88.07	—	—	91.15
KMNIST	—	—	57.27	—	—	37.23	—	—	78.95
Average	87.88	82.29	78.83	90.60	86.12	82.11	94.57	91.07	89.49

Table 8. **LLaMA-3 merging results (raw scores).** Single-task fine-tuning experts and merged models evaluated on five benchmarks.

Tasks	xquad_zh	xquad_vi	gsm8k_cot	mathqa	humaneval
multilingual	27.24	42.20	–	–	–
coding	–	–	–	34.30	49.39
math	–	–	74.00	–	–
Average	8.40	32.72	73.62	34.24	46.34
Task Arithmetic	8.06	32.88	73.09	34.14	46.34
ACE (ours)	16.42	37.68	69.75	34.57	50.61

Table 9. **Normalized performance on LLaMA-3 (%).** Each value is the ratio between the merged model and the best single-task fine-tuning expert for that benchmark.

Method	xquad_zh	xquad_vi	gsm8k_cot	mathqa	humaneval	Avg.
Average	30.84	77.54	99.49	99.83	93.82	80.30
Task Arithmetic	29.59	77.91	98.77	99.53	93.82	79.92
ACE (ours)	60.28	89.29	94.26	100.79	102.47	89.42

C.1. Additional results on LLaMA-3 and out-of-domain generalization

We further evaluate `ACE-Merging` on a LLaMA-3 backbone fine-tuned on three specialized expert task groups: multilingual QA (`multilingual`), code generation (`coding`), and mathematical reasoning (`math`). Each expert is fine-tuned on a disjoint subset of benchmarks and then merged using different weight-space fusion strategies. The merged models are evaluated jointly on five downstream benchmarks: `xquad_zh`, `xquad_vi`, `gsm8k_cot`, `mathqa`, and `humaneval`. The raw scores are reported in Tab. 8.

To normalize for the different intrinsic difficulty and scale of each benchmark, we further report in Tab. 9 the performance of the merged models as a percentage of their corresponding single-task fine-tuning expert. For each column, we divide the merged score by the score of the relevant expert (`multilingual` for `xquad_zh` and `xquad_vi`, `math` for `gsm8k_cot`, and `coding` for `mathqa` and `humaneval`) and multiply by 100.

The normalized results highlight the out-of-domain behavior of ACE-Merging. On the multilingual benchmarks xquad_zh and xquad_vi, which are out-of-domain for both the coding and math experts, ACE-Merging recovers 60.28% and 89.29% of the corresponding single-task fine-tuning performance, substantially outperforming both simple averaging and task arithmetic (which remain around 30–78%). On the code benchmark humaneval, ACE-Merging even slightly exceeds the coding expert, achieving 102.47% of the fine-tuned score, while maintaining competitive accuracy on gsm8k_cot and mathqa. Overall, ACE-Merging attains the highest normalized average score (89.42%), indicating that it effectively aggregates heterogeneous experts and exhibits strong out-of-domain generalization across multilingual, coding, and mathematical reasoning tasks.