IS FLAT MERGING ENOUGH? EVOLUTIONARY SEARCH FOR TREE-STRUCTURED MODEL MERGING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model merging provides an efficient way to integrate independently fine-tuned models without costly retraining or access to their original training data. Most existing methods, however, focus primarily on designing merging operators while combining a fixed set of candidate experts through a typically flat aggregation structure. This leaves a largely overlooked structural design space: how experts should be organized, which experts should participate, and whether experts should be reused throughout the merging process. Through systematic empirical studies, we show that, even when the candidate experts and merging operator are held fixed, changing only the merging structure can substantially affect performance, with markedly different sensitivities across operator families. Identifying effective configurations in this structural space is challenging, as merging topology, expert selection, and expert reuse jointly define a large discrete and combinatorial search space. We therefore propose EvoTM, an evolutionary framework based on genetic programming for optimizing tree-structured model merging. EvoTM represents a complete merging procedure as a hierarchical tree, where leaves correspond to expert models and internal nodes apply a given merging operator. Structure-aware evolutionary operators jointly explore tree topology and expert composition, naturally accommodating expert selection and reuse within a unified framework. Experiments across diverse merging operators and both vision and language models show that EvoTM can improve over conventional flat merging in multiple settings. Further analyses reveal operator-dependent structural sensitivities and provide additional insights into how different classes of merging operators respond to tree-structured composition. These results highlight merging structure as a complementary and previously underexplored design dimension of model merging.

1 INTRODUCTION

Model merging provides an efficient paradigm for integrating independently fine-tuned models without joint retraining or access to their original training data (Wortsman et al., 2022; Yadav et al., 2023). Recent work has developed increasingly sophisticated merging operators, including parameter-space aggregation, conflict resolution, statistics-based estimation, subspace manipulation, and optimization-based approaches (Tam et al., 2024; Wei et al., 2025b). Despite this progress, most existing methods adopt the same structural convention: a fixed set of candidate experts is combined through a single, typically flat, aggregation. This operator-centric formulation leaves a complementary question largely unexplored: *how should the merging process itself be organized?* Beyond choosing the merging operator, one may also determine how experts are grouped and composed, which experts participate, and whether an expert can be reused at different stages of the merging process. As illustrated in Fig. 1, these choices define a structural design space that is largely absent from the conventional formulation of model merging.

Why should structure matter when the merging operator remains unchanged? For linear aggregation, recursive composition may collapse to an equivalent linear combination of the original experts, possibly with different implicit weights. Many modern merging operators, however, are not invariant to the order in which intermediate models are formed. Conflict resolution, statistical estimation, pruning, and subspace construction can all depend on the particular models presented at each merging step. The output of an intermediate merge therefore changes the operands encountered by subsequent operations, making the overall procedure potentially path dependent; in general,

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

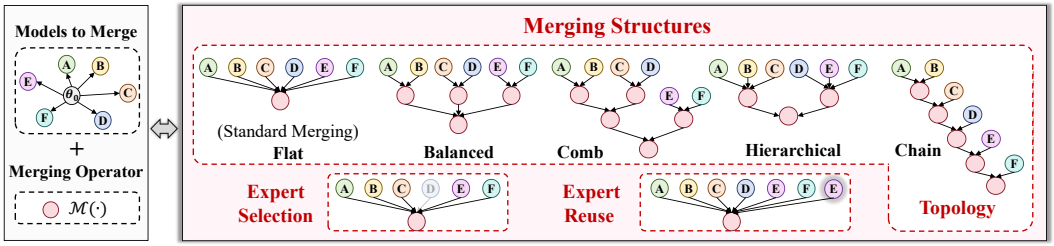


Figure 1: A structural view of model merging. Beyond model and operator design, we explore merging structure through topology, expert selection, and expert reuse.

$\mathrm{Merge}(\mathrm{Merge}(A, B), C)$ need not equal $\mathrm{Merge}(A, \mathrm{Merge}(B, C))$. Our empirical study confirms this distinction: with both the candidate experts and merging operator fixed, changing only the hierarchical organization can substantially alter the resulting performance, while the magnitude of this effect differs markedly across operators. Merging structure is therefore not merely an implementation detail, but a meaningful design variable whose importance depends on the mechanism of the underlying operator.

Once merging structure is exposed as a design variable, finding an effective configuration becomes nontrivial. A complete merging procedure may involve not only the hierarchical topology, but also which experts participate and whether some experts are reused. These choices are inherently coupled: the topology determines when and with whom an expert interacts, while selection and reuse alter the models propagated through the hierarchy. Together, they induce a large discrete and combinatorial search space whose size grows rapidly with the number of candidate experts. Existing sequential or curriculum-style approaches provide partial departures from flat aggregation (Tang et al., 2026; He et al., 2026), but typically constrain models to predefined or progressively constructed orders rather than optimizing general hierarchical structures. Moreover, local heuristics such as pairwise similarity provide only limited guidance for global structural decisions, since the effect of grouping two experts can depend on both the surrounding composition and the underlying merging operator. Rather than prescribing a single structural rule, these characteristics motivate treating merging structure as an explicit optimization problem.

To address this problem, we propose **EvoTM** (**Evo**lutionary **T**ree-structured **M**odel Merging), a genetic-programming framework for optimizing model-merging structures. EvoTM represents a merging procedure as a rooted tree, where leaves correspond to expert models and internal nodes apply a given merging operator. This tree representation naturally captures the three structural dimensions of interest: tree organization determines *merging topology*, leaf composition determines *expert selection*, and repeated leaves enable *expert reuse*. Genetic programming then jointly optimizes these dimensions through structure-aware tree variation, while leaving the underlying merging operator unchanged. EvoTM can therefore be applied on top of existing merging methods to improve model merging from the structural dimension. Experiments across diverse merging operators and both vision and language models show improvements over conventional flat merging in multiple settings, while further analyses reveal pronounced operator-dependent structural sensitivities and preferences.

Our main contributions are summarized as follows:

- We identify *merging structure* as a largely overlooked design dimension and show that different hierarchical organizations can substantially affect performance even when the candidate experts and merging operator are fixed.
- We formulate structural model merging as a discrete combinatorial optimization problem and propose **EvoTM**, a genetic programming-based framework that jointly explores tree topology, expert selection, and expert reuse under existing merging operators.
- We conduct extensive experiments across diverse merging operators and vision and language models, demonstrating improvements over conventional flat merging across multiple settings and revealing operator-dependent structural sensitivity as an additional empirical insight.

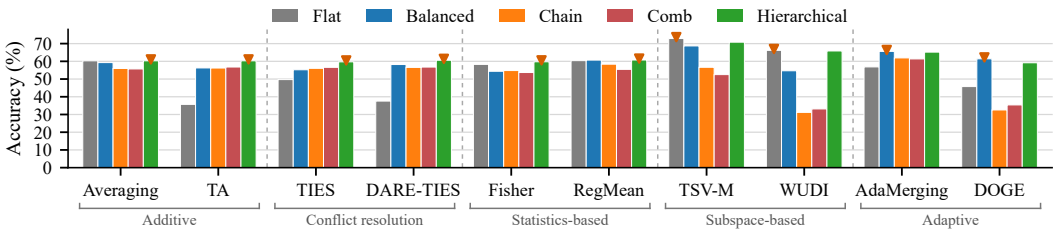


Figure 2: Effect of merging topology on 20 ViT-B/32 experts across ten representative merging operators. The expert set and merging operator are fixed, and only the topology is changed.

2 Related Work

Merging Operators. Existing model merging methods mainly differ in how model parameters or task-specific knowledge are integrated. Linear approaches average or compose model updates in parameter space (Ilharco et al., 2022; Ortiz-Jimenez et al., 2023; Jang et al., 2024), while alignment-based methods first match parameters or representations across independently trained models (Ainsworth et al., 2022; Stoica et al., 2024; Horoi et al., 2024). To reduce task interference, methods such as DARE, DELLA, Model Breadcrumbs, and CAT Merging sparsify or modify conflicting updates (Yu et al., 2024; Deep et al., 2024; Davari & Belilovsky, 2024; Sun et al., 2025). Other approaches exploit parameter statistics or task subspaces (Matena & Raffel, 2022; Jin et al., 2023; Daheim et al., 2024; Tang et al., 2023; Gargiulo et al., 2025; Marczak et al., 2025; Cheng et al., 2025), while adaptive methods optimize merging coefficients or merged representations (Yang et al., 2024b; Huang et al., 2024; Lu et al., 2024; Wang et al., 2025; Yang et al., 2024a; Wei et al., 2025a). Despite this diversity, most existing work focuses on improving the merging rule while treating the organization of the merging process as fixed.

Structures of Model Merging. In contrast, considerably less attention has been paid to *how merging operations are organized*. Conventional methods typically aggregate all candidate experts in a single flat step, whereas hierarchical merging recursively combines expert subsets and introduces additional choices over grouping, order, and participation. Continual model merging partially departs from flat aggregation by progressively incorporating newly arriving experts (Yang et al., 2026; Qiu et al., 2026b;a), but the resulting structure is largely determined by arrival order and is therefore typically chain-like. More recent studies relax this constraint from different perspectives: SimMerge considers model subsets, merging operators, and merge orders using similarity signals (Bolton et al., 2026), while FlexMerge progressively combines model components to trade model size for accuracy (Dhasade et al., 2026). Hierarchical formulations have also been explored in gradient-based fusion, multi-objective merging, and sparse expert compression (Yang et al., 2024c; Zhou et al., 2026; Chen et al., 2024).

3 From Flat Merging to Structural Optimization

3.1 A Structural View of Model Merging

Consider N expert models $\Theta = \{\theta_1, \ldots, \theta_N\}$ and a fixed merging operator $\mathcal{M}$. Conventional model merging combines all experts in a single step,

$$\theta_{\text{flat}} = \mathcal{M}(\theta_1, \ldots, \theta_N), \tag{1}$$

which corresponds to a flat structure in which all experts are directly connected to a single merging node. We generalize this formulation by representing the merging process as a rooted tree T. Each leaf v_i corresponds to an expert model, $\theta_{v_i} = \theta_i$, while each internal node recursively applies $\mathcal{M}$ to the outputs of its children:

$$\theta_v = \mathcal{M}\left(\{\theta_c \mid c \in \mathcal{C}(v)\}\right), \tag{2}$$

where $\mathcal{C}(v)$ denotes the children of node v. The final merged model is given by the root output, $\theta_T = \theta_{r(T)}$. This formulation separates the *local merging rule* from its *global composition*: $\mathcal{M}$ determines how a set of models is merged at each node, whereas T determines how these operations

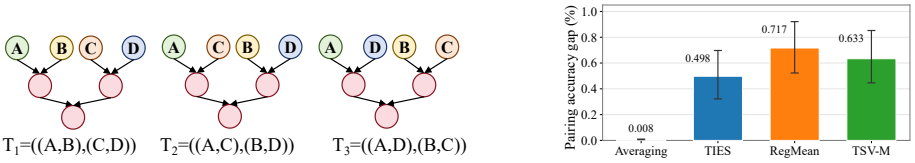


Figure 3: Analysis of expert grouping. **Left:** For the same four experts, we compare all three balanced pairings while keeping the tree shape, depth, and leaf weights fixed. **Right:** Accuracy gap between the best and worst pairing over 20 expert quartets, evaluated on the four corresponding expert tasks. Error bars denote 95% bootstrap confidence intervals.

are organized. Flat merging is therefore a special case of tree-structured merging. Fig. 1 illustrates several representative structures, including Flat, Balanced, Chain, Comb, and Hierarchical merging.

3.2 Does Merging Structure Matter?

We first ask whether topology alone can affect model merging when all other factors are fixed. Specifically, we fix the candidate experts and merging operator, require every expert to appear exactly once, and compare the five representative structures illustrated in Fig. 1. Our primary study merges 20 ViT-B/32 experts using ten representative operators spanning five families and evaluates multi-task mean accuracy across the corresponding tasks.

As shown in Fig. 2,changing only the merging topology can produce markedly different outcomes. The magnitude and even the direction of this effect vary substantially across operators: conflict-resolution and adaptive methods often favor non-flat compositions, statistics-based methods exhibit relatively milder variation, while subspace-based methods can be highly sensitive to topology yet still prefer flat aggregation in some cases. More importantly, no single handcrafted structure performs consistently best across operators. The same qualitative behavior is also observed when merging Qwen2.5-1.5B LoRA experts and evaluating on unseen language tasks (Appendix A.2.1). These results indicate that topology is not merely an execution detail. By changing which intermediate models are formed and subsequently merged, the tree alters how a fixed operator is repeatedly applied. The preferred structure is therefore coupled with the merging mechanism and the particular expert set, motivating a closer examination of how recursive composition changes the effective merging process.

3.3 How Recursive Composition Changes Merging

Why can the same operator produce different results under different trees? Rather than attempting an exhaustive characterization, we examine two direct effects of recursive composition: *implicit reweighting* and *path-dependent expert interactions*.

Implicit reweighting. Even simple additive operators can become structure-dependent when applied recursively. For example, averaging four experts under $((A, B), (C, D))$ yields $\frac{1}{4}(A + B + C + D)$, whereas $(((A, B), C), D)$ yields $\frac{1}{8}A + \frac{1}{8}B + \frac{1}{4}C + \frac{1}{2}D$. Thus, the tree can alter the effective contribution of each expert even when every internal node uses the same averaging rule. We verify this effect through controlled variants that remove the weighting induced by recursion. For Averaging, normalizing each merge by the number of descendant experts reduces the accuracy range across the five structures from 4.59 to 0.01 points. Similarly, applying the Task Arithmetic scaling factor only once at the root reduces its range from 24.58 to 0.01 points. These controls suggest that the structural sensitivity of additive merging can largely arise from implicit weighting and repeated scaling (Table 6 in Appendix A.2.2).

Path-dependent expert interactions. Reweighting alone cannot explain operators that transform intermediate merged models. For TSV-M, each internal merge constructs and truncates a task-vector subspace before propagating its output upward; consequently, changing the tree alters both which experts interact and which intermediate representations are subsequently merged. When TSV-M is instead applied only once at the root, eliminating these recursive transformations, its cross-topology accuracy range drops from 20.37 points to zero.

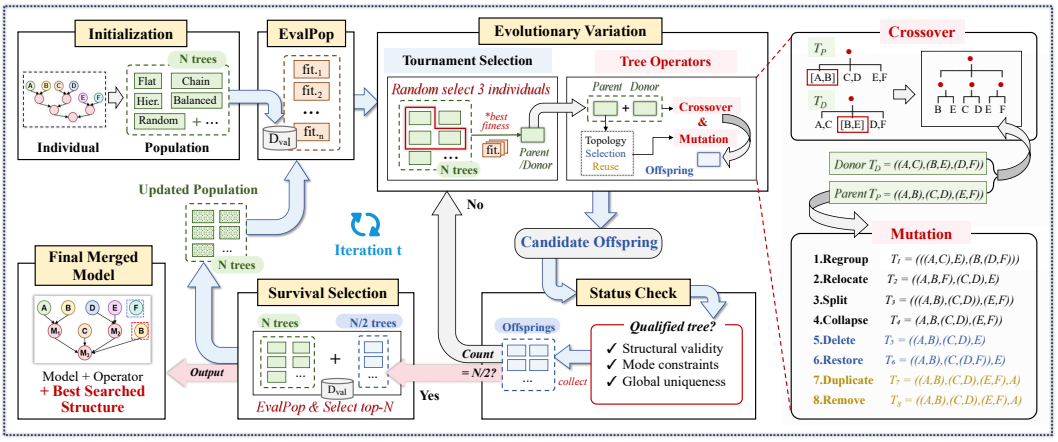


Figure 4: EvoTM: Evolutionary search for tree-structured model merging.

To further isolate the effect of expert grouping, we consider a quartet of expert models A, B, C, D and compare the three balanced pairings

$$((A, B), (C, D)), \quad ((A, C), (B, D)), \quad ((A, D), (B, C)), \tag{3}$$

which have identical tree depth, numbers of merge operations, and final expert weights. Across 20 unique sampled quartets, the mean accuracy gap between the best and worst pairing is only 0.008 points for Averaging, but increases to 0.498, 0.717, and 0.633 points for TIES, RegMean, and TSV-M, respectively (Fig. 3). Thus, even after controlling for tree shape and effective weights, the intermediate grouping of experts can change the outcome of path-dependent merging. We further test whether these grouping preferences can be predicted from simple static expert relations on the same 20 quartets, including task-vector cosine dissimilarity and operator-specific sign, Gram, and subspace measures. None of these measures consistently aligns with the best-performing pairing, indicating that effective grouping is difficult to infer from pairwise statistics alone (Appendix A.2.3).

4 Methodology

The preceding analysis shows that effective merging structures can be difficult to determine from simple expert relations. Once topology, expert selection, and expert reuse are jointly considered, structural model merging becomes a discrete combinatorial optimization problem over variable-size hierarchical configurations, for which exhaustive enumeration is impractical and no direct gradient is available. We therefore propose **EvoTM**, a genetic-programming framework that optimizes model merging at the structural level while keeping the underlying merging operator fixed. EvoTM encodes each merging strategy as a tree, allowing genetic programming to directly manipulate hierarchical organization and expert participation through subtree-level recombination and structure-aware mutation. Fig. 4 illustrates the framework; the full procedure is given in Algorithm 1.

4.1 Structural Search Space

Let $\Theta = \{\theta_1, \ldots, \theta_N\}$ denote the candidate expert models and $\mathcal{M}$ a fixed merging operator. EvoTM represents a merging strategy as an unordered rooted tree T, where each leaf is labeled by an expert index and each internal node applies $\mathcal{M}$ to the models produced by its children. Executing the tree from leaves to root yields the final merged model θ_T. For expert i, let $c_i(T)$ denote the number of leaves labeled by i. This representation exposes three complementary structural dimensions:

- **Topology:** the hierarchical grouping and order of merge operations;
- **Selection:** $c_i(T) = 0$ excludes expert i from the merge;
- **Reuse:** $c_i(T) > 1$ allows expert i to participate at multiple positions in the tree.

Accordingly, the structural search space is defined as

$$\mathcal{S} = \{T \mid L_{\min} \le |L(T)| \le L_{\max},\ 0 \le c_i(T) \le c_{\max},\ 2 \le |\mathcal{C}(v)| \le a_{\max}\}, \tag{4}$$

where $L(T)$ denotes the leaf set, $\mathcal{C}(v)$ the children of node v, and $V_{\text{int}}(T)$ the set of internal nodes. Given a validation set containing K tasks, EvoTM searches for

$$T^\star = \arg\max_{T \in \mathcal{S}} F(T), \qquad F(T) = \frac{1}{K} \sum_{k=1}^{K} s_k(\theta_T), \tag{5}$$

where $s_k(\theta_T)$ denotes the validation performance of the merged model on task k. Throughout each search, $\mathcal{M}$ and its hyperparameters remain fixed; only the structural configuration T is optimized.

4.2 Evolutionary Search over Merge Trees

Population initialization. We initialize a structurally diverse population to provide both strong reference configurations and broad coverage of the search space. For topology-only search, canonical structures such as Flat, Balanced, Chain, Comb, and Hierarchical trees are included as seeds, while the remaining individuals are randomly generated valid trees. When selection or reuse is enabled, expert multiplicities are sampled under the corresponding search constraints before being organized into random multi-ary trees. Structurally equivalent individuals are removed to avoid redundant initial evaluations.

Parent selection and subtree crossover. For each offspring proposal, EvoTM first selects a parent T_p by tournament selection, favoring high-fitness individuals while maintaining stochastic exploration. If crossover is triggered, an independent tournament selects a donor tree T_d. Crossover then performs directed subtree replacement: a non-root subtree of T_p is replaced by a non-root subtree from T_d. Importantly, the two subtrees are not required to contain the same number or identities of experts. Subtree crossover can therefore change not only the hierarchical topology, but also the leaf count and expert multiplicities. In the full search space, a single replacement may consequently introduce expert selection and reuse together with a topology change. Proposals that violate the constraints of the active search space are discarded.

Structure-aware mutation. Mutation further explores the neighborhood of an offspring through local tree edits. EvoTM uses four topology mutations that preserve the multiset of expert labels: REGROUP reconstructs a local subtree, RELOCATE moves a leaf to another position, SPLIT introduces an additional intermediate grouping, and COLLAPSE removes an internal level. These operations alter how the same experts are hierarchically composed without changing expert participation. Four additional mutations explicitly modify expert multiplicity. DELETE removes an expert that currently appears once, while RESTORE reintroduces an absent expert. DUPLICATE adds another occurrence of an expert below the multiplicity limit, whereas REMOVE-DUPLICATE removes one repeated occurrence. The available mutation set is determined by the active search space, so topology, selection, and reuse can be optimized either separately or jointly. Crossover and mutation are triggered independently, allowing an offspring to undergo crossover, mutation, or both.

Candidate validation and fitness evaluation. As structural variation may generate infeasible, unchanged, or previously visited trees, EvoTM therefore screens each proposal before performing model merging. A candidate must satisfy the active constraints on leaf count, expert multiplicity, and node arity. Because sibling order does not distinguish two merging strategies, each tree is converted to a canonical representation before comparison. To avoid repeatedly evaluating equivalent structures, EvoTM maintains a global archive of all canonical trees evaluated throughout the search. A proposal identical to its parent, already present in the archive, or duplicated within the current offspring batch is rejected without consuming an evaluation. Only feasible and globally novel trees are executed with $\mathcal{M}$ and assigned the validation fitness $F(T)$. Candidate generation continues until the required offspring batch has been collected.

Elitist survival. Once λ offspring have been evaluated, EvoTM applies elitist survival. The μ current individuals and λ offspring are pooled and ranked by validation fitness, and the best μ trees form the next generation. This preserves strong existing structures while allowing improved offspring to enter the population. The process continues until the prescribed evaluation budget is exhausted, after which the highest-fitness tree in the global archive is returned as the final merging structure.

Table 1: Results on merging 20 ViT-B/32 experts. M0 optimizes topology, M1/M2 additionally enable selection/reuse, and M3 enables both. Mean accuracy (%↑) across 20 tasks. M0–M3$_{20}$ use a matched 20-generation budget; M3$_{30}$ extends full search to 30 generations. Δ_{Flat} denotes the best matched-budget gain over Flat. Best/second-best searched results are in **bold**/underlined.

Merging Operator	Flat		M0 Topology		M1 +Selection		M2 +Reuse		M3$_{20}$ Full		M3$_{30}$ Full		Δ_{Flat}	
	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
Averaging	60.34	60.27	**62.19**	**61.63**	61.69	61.45	61.67	61.50	61.81	61.54	62.04	61.56	+1.85	+1.36
TA	36.06	35.66	61.70	61.46	62.10	61.75	61.93	61.57	62.37	61.70	**62.56**	**62.16**	+26.31	+26.09
TIES	49.90	49.67	62.57	61.90	62.57	61.68	62.84	62.10	62.80	62.16	**62.95**	**62.26**	+12.94	+12.49
Fisher	58.20	58.26	62.05	61.79	61.87	61.78	61.91	61.60	62.15	61.78	**62.45**	**62.08**	+3.95	+3.53
RegMean	60.11	60.41	62.64	**62.49**	62.31	62.33	62.09	61.79	62.39	61.91	**62.74**	62.41	+2.53	+2.08
TSV-M	73.18	72.83	74.79	74.27	74.93	74.56	75.46	74.94	75.59	**75.04**	**75.67**	74.75	+2.41	+2.21
WUDI	67.83	67.25	74.97	74.76	74.41	74.29	**76.29**	**75.83**	74.50	73.61	75.57	74.34	+8.46	+8.58
DOGE	46.38	46.29	**68.45**	68.15	67.24	66.54	67.64	67.28	67.08	66.37	67.92	**68.19**	+22.07	+21.90

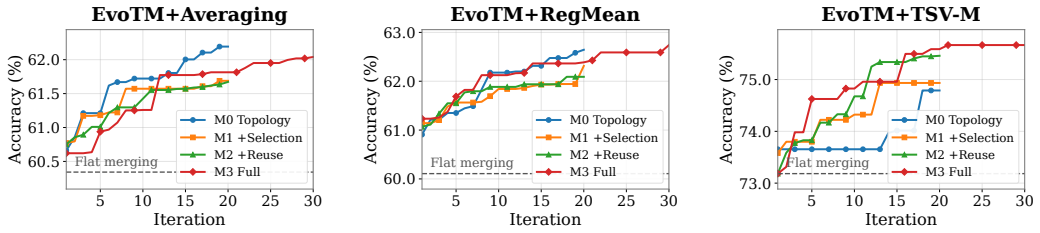


Figure 5: Best-so-far validation accuracy during structural search on ViT-B/32. M3 curve is extended to 30 generations to illustrate continued search in the full structural space.

5 Experiments

5.1 Experimental Setup

Models and Tasks. We consider two complementary evaluation settings. For *expert-task evaluation*, we merge 20 task-specific vision experts based on CLIP (Radford et al., 2021), using ViT-B/32, ViT-B/16, and ViT-L/14 backbones, reporting mean accuracy across tasks. For *cross-task evaluation*, we merge ten Qwen2.5-1.5B (Yang et al., 2025) LoRA experts fine-tuned on source tasks and evaluate the resulting model on eight disjoint target tasks, again reporting mean accuracy. For structural search, we reserve 10% of the evaluation data for validation and use the remaining 90% exclusively for final testing.

Merging Operators. Across our experiments, we consider representative merging operators spanning five types. These include additive methods, such as parameter averaging (Wortsman et al., 2022) and Task Arithmetic (Ilharco et al., 2022); conflict-resolution methods, including TIES (Yadav et al., 2023) and DARE-TIES (Yu et al., 2024; Yadav et al., 2023); statistics-based methods, including Fisher Merging (Matena & Raffel, 2022) and RegMean (Jin et al., 2023); subspace-based methods, including TSV-M (Gargiulo et al., 2025) and WUDI (Cheng et al., 2025); and adaptive merging methods such as AdaMerging (Yang et al., 2024b).

5.2 Main Results

Performance on Expert Tasks. We first evaluate EvoTM by merging 20 ViT-B/32 experts and testing on their corresponding tasks. As shown in Table 1, structural optimization improves over flat merging for all reported operators under the matched search budget, although the magnitude of the gain varies substantially across merging mechanisms. For example, TIES improves from 49.67 to 62.16 test accuracy, while the already strong TSV-M is further improved from 72.83 to 75.04. Even larger gains are observed for highly structure-sensitive operators such as TA, WUDI, and DOGE, confirming that the effectiveness of a merging rule can depend strongly on how it is recursively composed. A notable pattern is that topology optimization alone captures much of the available gain for

Table 2: Results on merging 10 Qwen2.5-1.5B experts. Mean validation/test accuracy (%↑) across eight disjoint target tasks. M0–M3_{20} use a matched 20-generation budget; M3_{30} extends full search to 30 generations. Δ_{Flat} is the best matched-budget gain over Flat for each split. Best/second-best searched results are in **bold**/underlined.

Merging Operator	Flat		M0 Topology		M1 +Selection		M2 +Reuse		M3_{20} Full		M3_{30} Full		Δ_{Flat}	
	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
Averaging	33.82	31.85	37.12	**32.45**	37.11	31.70	37.11	32.05	37.41	31.87	**37.61**	32.07	+3.59	+0.60
TA	23.99	21.40	37.48	32.34	**37.73**	32.42	37.45	32.49	37.20	32.45	37.20	**32.58**	+13.74	+11.08
TIES	32.02	29.45	37.45	32.61	37.57	**32.80**	**37.85**	32.75	37.62	32.45	37.79	32.54	+5.83	+3.35
DARE-TIES	26.78	22.85	37.98	32.24	38.05	31.98	37.77	**32.71**	**38.71**	32.37	**38.71**	32.37	+11.94	+9.86
TSV-M	36.32	32.53	38.44	34.31	38.90	32.68	39.13	34.69	38.28	33.32	**40.05**	**35.08**	+2.80	+2.16

several operators. For Averaging, M0 improves test accuracy from 60.27 to 61.63, while selection and reuse provide little additional benefit. This is consistent with our earlier observation that recursive averaging mainly induces implicit expert reweighting, which topology search already explores effectively. In contrast, TIES, TSV-M, and WUDI benefit more from additional structural flexibility. Overall, the useful structural degrees of freedom are operator dependent: topology provides a strong basis, while selection and reuse can offer complementary gains for more path-dependent merging mechanisms. The stronger M3_{30} validation results further suggest that the larger full search space can benefit from additional search budget.

Cross-Task Generalization. We next evaluate whether structural optimization transfers beyond the experts' original tasks. Ten Qwen2.5-1.5B LoRA experts trained on source tasks are merged using the validation splits of eight disjoint target tasks and evaluated on the corresponding held-out test splits. As shown in Table 2, structural search remains effective: Task Arithmetic improves from 21.40 to 32.49 test accuracy, TIES from 29.45 to 32.80, and TSV-M from 32.53 to 34.69 under the matched budget. Compared with expert-task evaluation, validation gains transfer less uniformly to test performance; for example, Averaging gains 3.59 points on validation but only 0.60 on test, and M3 is not consistently preferred. Since structural search is optimized directly on the target validation split, a larger search space may capture validation-specific interactions that do not fully generalize to held-out examples. This effect can be more pronounced when performance is aggregated across heterogeneous language tasks with limited validation data. Nevertheless, every evaluated operator improves over flat merging with at least one searched structure, showing that the benefit of structural optimization extends to disjoint downstream tasks.

5.3 Analysis

Properties of Evolved Merge Trees. Table 3 shows that the validation-best M3_{20} solutions exhibit markedly different structural patterns across operators. Averaging, TA, and Fisher retain only 11–12 of the 20 experts, whereas TSV-M preserves 19 and uses 23 leaves; TIES falls between these extremes. For additive methods, such selection and reuse can be viewed as an extension of the implicit reweighting already induced by recursive topology, further suppressing or amplifying individual expert contributions. TIES, in contrast, may use expert participation to alter the sign-conflict patterns encountered during recursive merging. Tree shape is likewise operator dependent. RegMean evolves a deep, nearly binary structure, while TSV-M favors a shallow and highly multi-ary tree. A plausible interpretation is that RegMean benefits from progressively resolving interactions through small local groups, whereas TSV-M may benefit from preserving a broader global task-vector subspace and avoiding excessive repeated truncation along deep paths. Overall, these results indicate that different operators exploit selection, reuse, and hierarchy in qualitatively different ways, arguing against a universal hand-designed merging structure.

Convergence and Optimization Efficiency. Fig. 5 shows the best-so-far validation accuracy during structural search. Because Flat and several canonical structures are included in the initial population, the curves reflect improvement from a strong initialization rather than recovery from a weak starting point. Across operators, EvoTM typically discovers substantial gains within the 20-generation budget, while the best fitness continues to improve in several cases. Topology-only search

Table 3: Structural statistics of the validation-best $M3_{20}$ solutions on 20 ViT-B/32 experts. **Selected/Omitted** count distinct included/excluded experts, **Reused** counts experts occurring multiple times, and **Leaves** counts total expert occurrences. $a_{\max}$ denotes maximum node arity.

Operator	Selected	Omitted	Reused	Leaves	Depth	#Internal	$a_{\max}$
Averaging	12	8	4	16	4	4	9
TA	11	9	4	15	3	5	7
TIES	16	4	5	21	3	7	10
Fisher	11	9	5	16	3	5	5
RegMean	12	8	3	15	5	14	2
TSV-M	19	1	4	23	3	6	18

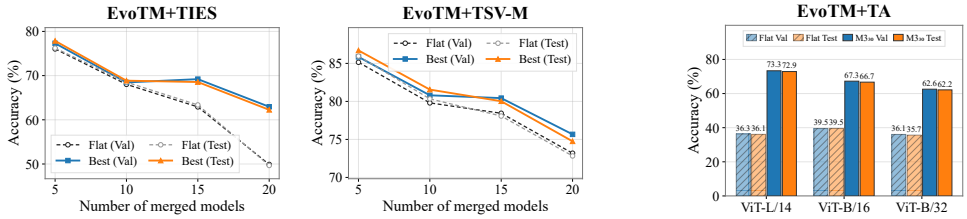


Figure 6: Scaling behavior of EvoTM. **Left:** Performance when merging different numbers of ViT-B/32 experts. **Right:** Performance across different ViT backbones when merging 20 experts.

often progresses rapidly in the early generations, whereas the larger M3 space improves more gradually and can benefit from the extended 30-generation budget. This behavior suggests a trade-off between structural flexibility and search difficulty: richer search spaces contain stronger solutions, but may require additional evaluations to exploit them effectively. The search is also robust to population size: increasing the population improves exploration and can yield slightly better solutions, but the performance gain saturates under the same generation budget (Appendix A.4.4).

Scaling Across Expert Pools and Model Backbones. We further examine whether the benefit of structural optimization persists across larger expert pools and different model backbones. As shown in the left panels of Fig. 6, EvoTM consistently outperforms flat merging when the number of ViT-B/32 experts increases from 5 to 20. For TIES, the gap widens with larger pools, as flat merging degrades markedly while searched structures remain more robust. TSV-M shows a smaller but persistent advantage, consistent with its stronger flat baseline and lower structural sensitivity. This suggests that structural optimization becomes especially valuable when larger expert pools amplify harmful expert interactions. The right panel shows a similar benefit across ViT-B/32, ViT-B/16, and ViT-L/14. For Task Arithmetic, EvoTM substantially improves over flat merging on all three backbones, while absolute performance also increases with model capacity. Thus, structural optimization is complementary to backbone scaling rather than specific to a particular model size. Complete results are provided in Appendix A.4.3 and A.4.1.

6 Conclusion

We revisit the common assumption of flat aggregation in model merging and identify merging structure as an important optimization dimension. We propose EvoTM, a genetic-programming framework that searches tree-structured merging procedures while supporting expert selection and reuse. Across diverse merging operators and both vision and language models, EvoTM improves conventional flat merging in multiple settings and reveals pronounced operator-dependent structural preferences. These findings broaden model merging from designing *how models are merged* to also optimizing *how merging operations are composed*, opening a complementary direction for improving model merging beyond operator design alone.

AI USE STATEMENT

We used generative AI tools for improving writing clarity, proofreading, and organizing the presentation of ideas. All scientific claims, experimental results, methodology, and conclusions were independently verified by the authors.

REFERENCES

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries, 2023. *URL https://arxiv. org/abs/2209.04836*, 2022.

Oliver Bolton, Arash Ahmadian, Sara Hooker, Marzieh Fadaee, Beyza Ermis, et al. Simmerge: Learning to select merge operators from similarity signals. *arXiv preprint arXiv:2601.09473*, 2026.

I Chen, Hsu-Shen Liu, Wei-Fang Sun, Chen-Hao Chao, Yen-Chang Hsu, Chun-Yi Lee, et al. Retraining-free merging of sparse moe via hierarchical clustering. *arXiv preprint arXiv:2410.08589*, 2024.

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 Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10121–10143. PMLR, 13–19 Jul 2025.

Nico Daheim, Thomas Möllenhoff, Edoardo M Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In *International Conference on Learning Representations*, volume 2024, pp. 11836–11854, 2024.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision*, pp. 270–287. Springer, 2024.

Pala Tej Deep, Rishabh Bhardwaj, and Soujanya Poria. DELLA-merging: Reducing interference in model merging through magnitude-based sampling. *arXiv preprint arXiv:2406.11617*, 2024.

Akash Dhasade, Divyansh Jhunjhunwala, Milos Vujasinovic, Gauri Joshi, and Anne-Marie Kermarrec. Navigating the accuracy-size trade-off with flexible model merging. In *International Conference on Learning Representations*, volume 2026, pp. 81010–81041, 2026.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodola. Task singular vectors: Reducing task interference in model merging. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18695–18705. IEEE, 2025.

Baoyi He, Luotian Yuan, Ying Wei, and Fei Wu. Curriculum model merging: Harmonizing chemical llms for enhanced cross-task generalization. *Advances in Neural Information Processing Systems*, 38:39201–39226, 2026.

Stefan Horoi, Albert Manuel Orozco Camacho, Eugene Belilovsky, and Guy Wolf. Harmony in diversity: Merging neural networks with canonical correlation analysis. *arXiv preprint arXiv:2407.05385*, 2024.

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. Emr-merging: Tuning-free high-performance model merging. *Advances in Neural Information Processing Systems*, 37:122741–122769, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models. In *European Conference on Computer Vision*, pp. 207–223. Springer, 2024.

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

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. *Advances in Neural Information Processing Systems*, 37:78905–78935, 2024.

Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost Van De Weijer. No task left behind: isotropic model merging with common and task-specific subspaces. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25, Vancouver, Canada, 2025. JMLR.org.

Michael S Matena and Colin Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36:66727–66754, 2023.

Zihuan Qiu, Lei Wang, Yang Cao, Runtong Zhang, Bing Su, Yi Xu, Fanman Meng, Linfeng Xu, Qingbo Wu, and Hongliang Li. Null-space filtering for data-free continual model merging: Preserving stability, promoting plasticity. In *International Conference on Learning Representations*, volume 2026, pp. 128492–128518, 2026a.

Zihuan Qiu, Yi Xu, Chiyuan He, Fanman Meng, Linfeng Xu, Qingbo Wu, and Hongliang Li. MINGLE: Mixture of null-space gated low-rank experts for test-time continual model merging. *Advances in Neural Information Processing Systems*, 38:143841–143878, 2026b.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *International Conference on Learning Representations*, volume 2024, pp. 29215–29237, 2024.

Wenju Sun, Qingyong Li, Yangliao Geng, and Boyang Li. CAT merging: A training-free approach for resolving conflicts in model merging. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57523–57543. PMLR, 13–19 Jul 2025.

Derek Tam, Mohit Bansal, and Colin Raffel. Merging by matching models in task parameter subspaces. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Anke Tang, Li Shen, Yong Luo, Liang Ding, Han Hu, Bo Du, and Dacheng Tao. Concrete subspace learning based interference elimination for multi-task model fusion. *arXiv preprint arXiv:2312.06173*, 2023.

Anke Tang, Enneng Yang, Li Shen, Yong Luo, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. Merging on the fly without retraining: A sequential approach to scalable continual model merging. *Advances in Neural Information Processing Systems*, 38:82081–82118, 2026.

Ke Wang, Nikos Dimitriadis, Alessandro Favero, Guillermo Ortiz-Jimenez, Francois Fleuret, and Pascal Frossard. Lines: Post-training layer scaling prevents forgetting and enhances model merging. In *International Conference on Learning Representations*, volume 2025, pp. 27100–27133, 2025.

Qi Wei, Shuo He, Enneng Yang, Tingcong Liu, Haobo Wang, Lei Feng, and Bo An. Representation surgery in model merging with probabilistic modeling. In *Forty-second International Conference on Machine Learning*, 2025a.

Yongxian Wei, Anke Tang, Li Shen, Zixuan Hu, Chun Yuan, and Xiaochun Cao. Modeling multi-task model merging as adaptive projective gradient descent. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66178–66193. PMLR, 13–19 Jul 2025b.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model Soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. Pmlr, 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56332–56356. PMLR, 21–27 Jul 2024a.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. AdaMerging: Adaptive model merging for multi-task learning. In *International Conference on Learning Representations*, volume 2024, pp. 22743–22763, 2024b.

Enneng Yang, Anke Tang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, and Jie Zhang. Continual model merging without data: Dual projections for balancing stability and plasticity. *Advances in Neural Information Processing Systems*, 38:39275–39305, 2026.

Haoyu Yang, Zheng Zhang, and Saket Sathe. SuperMerge: An approach for gradient-based model merging. *arXiv preprint arXiv:2412.10416*, 2024c.

Linyan 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*, volume 2, pp. 21, 2024.

Yu Zhou, Xingyu Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. Hm3: Hierarchical multi-objective model merging for pretrained models. *Advances in Neural Information Processing Systems*, 38: 114876–114910, 2026.

A Appendix

A.1 Experimental Details

A.1.1 Models to Merge

Vision Models. We consider three CLIP vision backbones (Radford et al., 2021): ViT-B/32, ViT-B/16, and ViT-L/14. ViT-B/32 and ViT-B/16 contain 12 Transformer blocks with a hidden dimension of 768 and 12 attention heads, differing in patch sizes of 32×32 and 16×16, respectively. ViT-L/14 contains 24 Transformer blocks with a hidden dimension of 1024, 16 attention heads, and a 14×14 patch size.

For each backbone, task-specific experts are independently fine-tuned from the same pretrained CLIP initialization. Following the standard model-merging setting, we merge the shared vision encoder while retaining task-specific classification heads for evaluation. ViT-B/32 is used in the

primary experiments, while ViT-B/16 and ViT-L/14 are used to evaluate generalization across model scales. The fine-tuned expert checkpoints and corresponding task datasets are publicly available at `huggingface.co/tanganke/models`.

Language Models. For language-model experiments, we use ten Qwen2.5-1.5B (Yang et al., 2025) LoRA experts following the setup of EvoGM. All experts share the same Qwen2.5-1.5B base model and are independently adapted on ten supervised fine-tuning domains from the Tulu-v2 collection. These domains cover FLAN, chain-of-thought data, OpenAssistant-1, ShareGPT, Code Alpaca, LIMA, WizardLM Evol-Instruct-v2, Open-Orca, scientific literature, and distilled Gemini-Alpaca data. The experts are trained for five epochs with an initial learning rate of 2×10^{-4} and an effective batch size of 32. We directly use the released expert checkpoints without additional fine-tuning. The base model and LoRA experts are available at `Qwen/Qwen2.5-1.5B` and `TaoJiangCN/qwen2.5-1.5b-tulu-v2-lora-experts`, respectively.

A.1.2 Merging Operators

We consider representative merging operators spanning additive, conflict-resolution, statistics-based, subspace-based, and adaptive approaches. Different experiments use different subsets of these operators. For tree-structured merging, the same operator is recursively applied at every internal node, while the corresponding flat baseline applies it once to all candidate experts.

- **Parameter Averaging** (Wortsman et al., 2022) computes the element-wise mean of model parameters. Under recursive composition, different tree structures can induce different effective weights on the leaf experts.
- **Task Arithmetic (TA)** (Ilharco et al., 2022) represents each expert by a task vector—the parameter difference from the shared pretrained model—and combines task knowledge by linearly composing these vectors.
- **TIES** (Yadav et al., 2023) reduces interference among task vectors by trimming small updates, resolving parameter-wise sign conflicts, and aggregating updates consistent with the elected direction.
- **DARE-TIES** (Yu et al., 2024; Yadav et al., 2023) first sparsifies and rescales task vectors through DARE and then applies the conflict-resolution procedure of TIES.
- **Fisher Merging** (Matena & Raffel, 2022) uses Fisher information to estimate parameter importance and performs parameter-wise weighted averaging across experts.
- **RegMean** (Jin et al., 2023) incorporates layer-wise activation statistics and input Gram matrices to obtain a closed-form merged solution that approximately preserves the experts' functional behavior.
- **TSV-M** (Gargiulo et al., 2025) decomposes task-specific updates into singular-vector subspaces and mitigates interference among overlapping task directions before reconstructing the merged update.
- **WUDI** (Cheng et al., 2025) merges models from a task-subspace perspective, manipulating task-specific update directions to reduce interference while preserving complementary information.
- **AdaMerging** (Yang et al., 2024b) adaptively optimizes merging coefficients using validation feedback, rather than relying on manually specified expert weights.

A.1.3 Device and Parameter Settings

Computing environment. All experiments were implemented in PyTorch and conducted on eight NVIDIA A100 GPUs.

EvoTM configuration. Unless otherwise specified, EvoTM uses a population size of $\mu = 32$ and generates $\lambda = 16$ offspring per generation. Tournament selection is used with tournament size 3. Subtree crossover and mutation are independently applied with probabilities $p_c = 0.8$ and $p_m = 0.2$, respectively, followed by $(\mu + \lambda)$ elitist survival. The maximum node arity is set to the number of available experts, i.e., 20 for the vision experiments and 10 for the language-model experiments. For

Table 4: Fixed-topology merging of 20 ViT-B/32 experts. Results report mean accuracy (%↑) across 20 expert tasks. Δ_{gain} denotes the gain of the best non-flat topology over Flat, and Δ_{range} the accuracy range across all five topologies. Best results are in **bold**.

Type	Operator	Flat	Balanced	Chain	Comb	Hier.	Δ_{gain}	Δ_{range}
Additive	Averaging	60.27	59.28	55.96	55.69	**60.28**	+0.01	4.59
	TA	35.70	56.26	56.21	56.78	**60.28**	+24.58	24.58
Conflict Resolution	TIES	49.69	55.23	56.04	56.60	**59.70**	+10.01	10.01
	DARE-TIES	37.56	58.19	56.56	56.75	**60.60**	+23.04	23.04
Statistics-based	Fisher	58.26	54.38	54.79	53.68	**59.74**	+1.48	6.06
	RegMean	60.38	60.69	58.38	55.46	**60.74**	+0.36	5.28
Subspace-based	TSV-M	**72.86**	68.68	56.63	52.50	70.80	-2.06	20.36
	WUDI	**66.18**	54.66	31.18	33.13	65.84	-0.34	35.00
Adaptive	AdaMerging	56.84	**65.60**	61.92	61.35	65.19	+8.76	8.76
	DOGE	45.75	**61.41**	32.55	35.39	59.17	+15.66	28.86

Table 5: Fixed-topology merging of 10 Qwen2.5-1.5B LoRA experts. Results report mean accuracy (%↑) across eight disjoint target tasks. Δ_{gain} denotes the gain of the best non-flat topology over Flat, and Δ_{range} the accuracy range across all five topologies. Best results are in **bold**.

Operator	Flat	Balanced	Chain	Comb	Hier.	Δ_{gain}	Δ_{range}
Averaging	34.79	34.87	**35.52**	33.52	35.03	+0.73	2.00
TA	34.16	34.58	**35.16**	34.23	35.03	+1.00	1.00
TIES	33.70	34.78	34.76	34.42	**35.23**	+1.53	1.53
DARE-TIES	33.86	**35.42**	35.32	33.89	34.97	+1.56	1.56
Fisher	**35.30**	35.02	34.88	33.60	**35.30**	+0.00	1.70
TSV-M	28.61	31.29	**35.99**	32.84	30.99	+7.38	7.38
WUDI	**21.79**	10.00	20.00	16.82	18.70	-1.79	11.79

the 20-expert vision setting, the number of leaves is restricted to $[15, 25]$; for the 10-expert language setting, it is restricted to $[8, 12]$. When expert reuse is enabled, each expert may appear at most twice. The four search settings are configured as follows:

- **M0 (Topology):** every expert appears exactly once;
- **M1 (+Selection):** experts may be omitted but not repeated;
- **M2 (+Reuse):** every expert appears at least once and may appear at most twice;
- **M3 (Full):** experts may be both omitted and repeated, subject to the leaf-count and multiplicity constraints above.

A.2 Preliminary Analysis

A.2.1 Effect of Merge Topology

To isolate the effect of topology from expert selection and reuse, we conduct a controlled topology-only analysis. The candidate experts and merging operators are fixed, every expert appears exactly once, and only the recursive organization of experts is changed. Specifically, we compare five representative structures: Flat, Balanced, Chain, Comb, and Hierarchical. The complete results on vision and language models are reported in Table 4 and Table 5, respectively. For each operator, we report the improvement of the best non-flat topology over flat merging and the performance variation across different topologies. These results show that structural effects already emerge without expert selection or reuse. Therefore, the benefit of hierarchical merging is not solely caused by changing expert participation, but also comes from modifying how experts are recursively composed.

Table 6: Controlled variants of recursive merging. **Native** reports the accuracy range across the five structures under the original execution, while **Controlled** removes the indicated intermediate effect.

Operator	Control	Native	Controlled
Averaging	Equal final leaf weights	4.59	0.01
Task Arithmetic	Root-only scaling	24.58	0.01
TSV-M	Root-only transformation	20.37	0.00

A.2.2 Recursive Composition and Expert Interaction

While topology changes the organization of experts, different merging operators may respond to recursive composition through different mechanisms. To disentangle these effects, we perform controlled experiments that remove specific properties introduced by intermediate merging steps. Table 6 summarizes the results. For linear operators such as Averaging and Task Arithmetic, controlling the effective contribution of leaf experts largely eliminates the topology variation. This indicates that hierarchical structures mainly act as implicit reweighting mechanisms for these operators. In contrast, operators involving intermediate transformations exhibit stronger path dependency. For TSV-M, each internal node performs subspace construction and filtering before producing an intermediate model. When this intermediate transformation is removed and TSV-M is applied only at the root, the structural variation is reduced from 20.37 points to zero. These results suggest that recursive composition influences merging through multiple mechanisms: it can modify effective expert contribution for linear operators, while additionally changing intermediate model interactions for non-linear or transformation-based operators.

A.2.3 Limitations of Static Pairwise Metrics for Expert Pairing

The previous analysis shows that expert interactions during intermediate merging stages can affect final performance. A natural question is whether such interactions can be predicted from static relationships between expert models. We perform a controlled quartet analysis. For each sampled quartet of experts $\mathcal{Q} = \{A, B, C, D\}$, we evaluate all three balanced pairings:

$$((A, B), (C, D)), \quad ((A, C), (B, D)), \quad ((A, D), (B, C)). \tag{6}$$

All candidate trees have identical depth, merge operations, and final expert weights; therefore, only the first-stage grouping differs. The pairing achieving the highest merging accuracy is treated as the empirical optimal interaction.

We compare these optimal pairings with rankings obtained from several static pairwise metrics, including cosine dissimilarity, sign conflict, Gram-matrix distance, and subspace distance. For a candidate pairing, pairwise scores are aggregated over its two expert groups:

$$S((A, B), (C, D)) = m(A, B) + m(C, D), \tag{7}$$

where $m(\cdot, \cdot)$ denotes one of the pairwise measurements. The complete protocol consists of 900 pairing evaluations over sampled quartets, with three candidate pairings per quartet. The corresponding experimental records and metric measurements are summarized in Table 7.

Figure 7 shows the top-1 identification accuracy of different metrics. The results demonstrate that static pairwise relations provide limited and operator-dependent guidance. Although certain metrics capture specific aspects of task similarity or interference, they cannot consistently identify the most effective expert grouping. This suggests that useful expert interactions depend not only on isolated pairwise relationships, but also on the recursive context in which experts are composed. Therefore, effective merging structures require optimization over the complete composition process rather than manually designed pairwise heuristics.

Table 7: Summary of the quartet pairing analysis.

Component	Setting
Expert group size	4
Candidate pairings	3 balanced pairings
Tree structure	Depth-2 balanced tree
Controlled factors	Depth, merge count, expert weights
Number of evaluations	900
Pairwise metrics	Cosine, Sign, Gram, Subspace

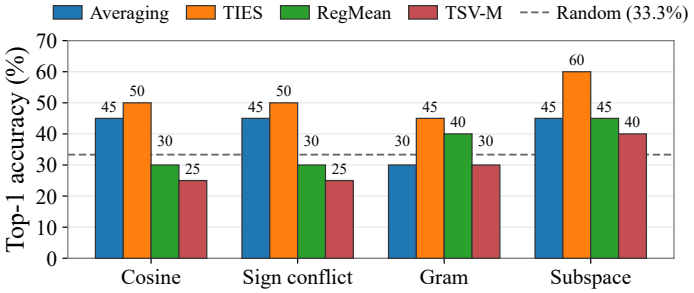


Figure 7: Top-1 pairing identification accuracy of static pairwise metrics in quartet experiments. For each quartet, three balanced expert pairings are evaluated, and the pairing with the highest merging performance is treated as the optimal interaction. Bars report the percentage of quartets where each metric correctly identifies the optimal pairing for different merging operators. The dashed line indicates random selection among three candidates (33.3%).

A.3 Details of Method

A.3.1 EvoTM Search Algorithm

Algorithm 1 summarizes the evolutionary optimization procedure of EvoTM. Each individual is represented as a merge tree, and its fitness is evaluated by recursively applying the predefined merging operator over the tree structure.

A.3.2 Structure-Aware Variation Operators

EvoTM explores the structural design space through subtree crossover and structure-aware mutation. Unlike conventional evolutionary search over fixed-length vectors, tree variation must preserve the validity of merge procedures. The proposed operators modify three structural dimensions: topology, expert selection, and expert reuse.

Subtree crossover. Given two parent trees T_p and T_d, EvoTM randomly selects a non-root subtree from each parent and exchanges them. Different from conventional crossover on fixed-size representations, the exchanged subtrees may contain different numbers and identities of experts. Therefore, subtree crossover can simultaneously alter hierarchical organization and expert composition. After crossover, the resulting tree is checked against the constraints of the active search space.

Topology mutation. Topology mutation modifies the organization of existing experts while preserving the expert occurrence set. Given a selected subtree T_s, the following operations are applied:

- **Regroup** rebuilds the local subtree by changing the grouping of child nodes, modifying expert interaction order.
- **Relocate** moves an expert leaf to another position, changing its merging context while preserving expert participation.
- **Split** introduces an additional internal merge node, creating a finer-grained hierarchical composition.

Algorithm 1 EvoTM: Evolutionary Search over Merge Trees

Require: Expert models Θ; merging operator $\mathcal{M}$; search space $\mathcal{S}$; population size μ; offspring size λ; generations G
Ensure: Optimal merge tree $T^\star$

$\mathcal{P} \leftarrow \textsc{Initialize}(\mathcal{S}, \mu)$ ▷ **Population initialization**
Evaluate $F(T)$ for all $T \in \mathcal{P}$
$\mathcal{A} \leftarrow \mathcal{P}$
for $g = 1, \ldots, G$ **do**
  $\mathcal{O} \leftarrow \emptyset$
  while $|\mathcal{O}| < \lambda$ **do** ▷ **Evolutionary variation**
    $T_p \leftarrow \textsc{TournamentSelect}(\mathcal{P})$
    $\widetilde{T} \leftarrow T_p$
    if crossover is applied **then**
      $T_d \leftarrow \textsc{TournamentSelect}(\mathcal{P})$
      $\widetilde{T} \leftarrow \textsc{SubtreeCrossover}(T_p, T_d)$
    end if
    if mutation is applied **then**
      $\widetilde{T} \leftarrow \textsc{StructureMutation}(\widetilde{T}, \mathcal{S})$
    end if
    $\widetilde{T} \leftarrow \textsc{Canonicalize}(\widetilde{T})$
    if $\widetilde{T} \notin \mathcal{S}$ **or** $\widetilde{T} \in \mathcal{A} \cup \mathcal{O}$ **then**
      continue
    end if
    Evaluate $F(\widetilde{T})$ ▷ **Fitness Evaluation**
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{\widetilde{T}\}$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\widetilde{T}\}$
  end while
  $\mathcal{P} \leftarrow \textsc{Top}_{(}\mu(\mathcal{P} \cup \mathcal{O})$ ▷ **Elitist survival**
end for
$T^\star \leftarrow \arg\max_{T \in \mathcal{A}} F(T)$
return $T^\star$

- **Collapse** removes an intermediate merge node and simplifies the local hierarchy.

Selection and reuse mutation. To optimize expert participation, EvoTM introduces additional mutations when the corresponding search dimensions are enabled.

- **Delete** removes an expert occurrence, allowing the search to exclude potentially harmful experts.
- **Restore** reintroduces an omitted expert into the tree.
- **Duplicate** creates an additional occurrence of an existing expert, enabling expert reuse at multiple merging locations.
- **Remove-Duplicate** removes repeated expert occurrences when reuse is no longer beneficial.

The available mutation operators are determined by the active search space. Therefore, M0 optimizes topology only, while M1, M2, and M3 additionally enable selection, reuse, and their combination, respectively.

A.4 Experimental Results

A.4.1 Results on ViT-L/14 and ViT-B/16

To evaluate whether structural optimization generalizes across different vision backbones, we further conduct experiments on ViT-B/16 and ViT-L/14 with 20 task-specific experts. The experimental

Table 8: Structure-aware mutation operators in EvoTM.

Dimension	Operator	Effect
Topology	Regroup	Reorganize local grouping
	Relocate	Move expert leaf position
	Split	Add intermediate merge level
	Collapse	Remove intermediate merge level
Selection	Delete	Remove expert occurrence
	Restore	Reintroduce omitted expert
Reuse	Duplicate	Add expert occurrence
	Remove-Duplicate	Remove repeated occurrence

Table 9: Results on 20 ViT-L/14 experts. Mean validation/test accuracy (%↑) across 20 tasks. M0–M3$_{20}$ use a matched 20-generation budget, while M3$_{30}$ extends full search to 30 generations. Δ_{Flat} denotes the gain over Flat using the best matched-budget searched result for each split. Best and second-best searched results are shown in **bold** and underlined, respectively.

Merging Operator	Flat		M0 Topology		M1 +Selection		M2 +Reuse		M3$_{20}$ Full		M3$_{30}$ Full		Δ_{Flat}	
	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
Averaging	70.31	70.33	72.14	72.14	72.06	72.02	71.86	71.85	72.24	72.32	**72.55**	**72.46**	+1.93	+1.99
TA	36.31	36.10	72.07	71.78	72.14	71.85	72.32	71.99	73.19	72.87	**73.35**	**72.90**	+36.88	+36.77
TIES	55.25	55.23	**73.51**	73.19	72.22	72.11	72.86	72.72	72.93	73.12	73.23	**73.48**	+18.26	+17.96
RegMean	70.26	70.41	72.42	72.28	**73.40**	**73.23**	72.27	72.17	72.81	72.50	72.90	72.52	+3.14	+2.82

settings follow the main ViT-B/32 benchmark, including the same search spaces and evaluation protocol. The complete results are reported in Table 10 and Table 9.

A.4.2 Convergence Curves

We provide additional convergence curves for different merging operators in Fig. 8. Each curve reports the best-so-far validation accuracy during evolutionary search.

A.4.3 Scaling with the Number of Merged Experts

We further investigate how structural optimization scales with the number of merged experts. Figure 9 reports the performance of EvoTM when merging different numbers of ViT-B/32 experts for several representative operators.

A.4.4 Hyperparameter Sensitivity

We evaluate the sensitivity of EvoTM to two search-related hyperparameters: population size and maximum node arity. The population size is varied among $\mu \in \{8, 16, 32, 64\}$, while the maximum node arity is varied among $a_{\max} \in \{5, 10, 15, 20\}$. Experiments are conducted on ViT-B/32 experts with Averaging and TIES merging operators. Results are shown in Figure 10.

Table 10: Results on 20 ViT-B/16 experts. Mean validation/test accuracy (%↑) across 20 tasks. M0–$M3_{20}$ use a matched 20-generation budget, while $M3_{30}$ extends full search to 30 generations. Δ_{Flat} denotes the gain over Flat using the best matched-budget searched result for each split. Best and second-best searched results are shown in **bold** and underlined, respectively.

Merging Operator	Flat		M0 Topology		M1 +Selection		M2 +Reuse		$M3_{20}$ Full		$M3_{30}$ Full		Δ_{Flat}	
	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
Averaging	64.54	64.24	66.23	65.75	66.54	66.02	66.42	66.09	66.43	66.02	**66.79**	**66.37**	+1.99	+1.85
TA	39.49	39.55	66.13	65.56	66.58	65.99	66.62	66.18	67.10	66.56	**67.26**	**66.69**	+27.60	+27.02
TIES	55.24	54.88	67.67	67.50	67.57	67.18	67.56	66.91	66.85	66.39	**68.07**	**67.67**	+12.43	+12.62
Fisher	63.20	63.28	**66.74**	**66.41**	66.16	65.81	66.24	66.03	66.40	65.92	66.61	66.19	+3.54	+3.13
RegMean	64.55	64.39	66.27	65.72	66.25	65.97	66.02	65.85	66.94	66.47	**67.12**	**66.77**	+2.39	+2.08
TSV-M	78.33	77.89	79.83	79.23	**80.37**	**79.78**	80.20	79.75	79.92	79.37	80.28	79.75	+2.04	+1.89

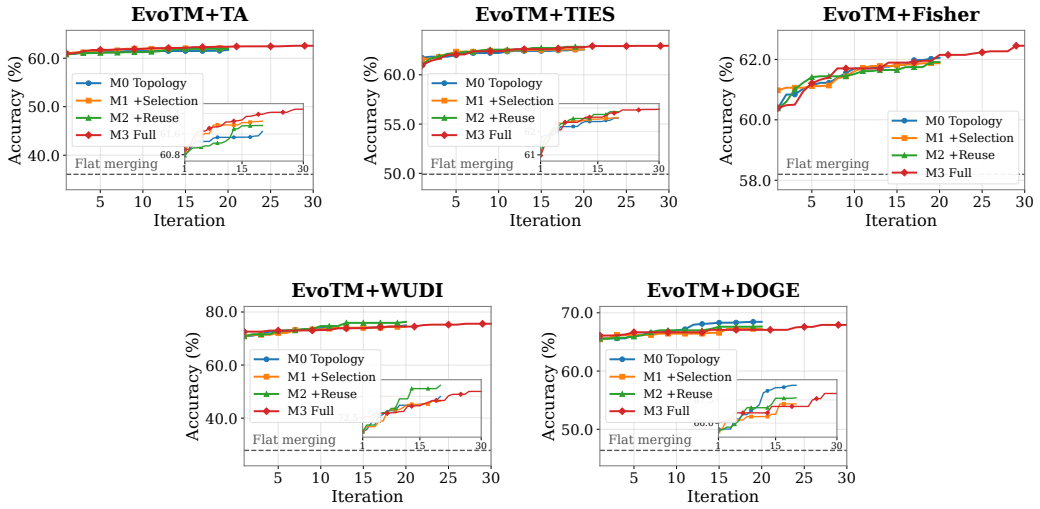


Figure 8: Best-so-far validation accuracy during structural search on ViT-B/32.

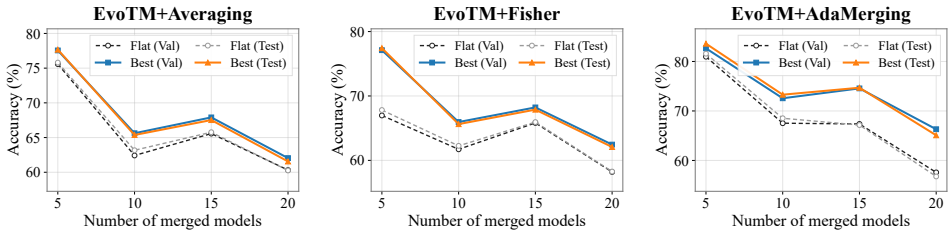


Figure 9: Performance when merging different numbers of ViT-B/32 experts.

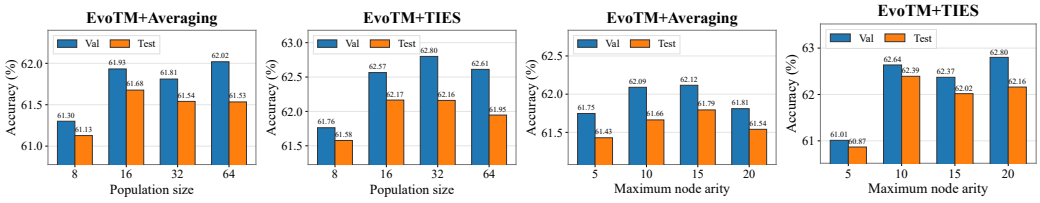


Figure 10: Hyperparameter sensitivity of EvoTM on ViT-B/32 experts. The first two panels show the effect of population size, and the last two panels show the effect of maximum node arity.