HETEROWEAVE: VARIABLE-CARDINALITY BLOCK COMPOSITION IN PRETRAINED MODEL REASSEMBLY

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing heterogeneous pretrained model reassembly methods usually retain one block at each aligned position, limiting the potential for different pretrained sources to contribute jointly. Allowing multiple blocks to participate at the same position increases reassembly flexibility, but turns block identity and participation cardinality into coupled architectural decisions, substantially expanding the composition space. Meanwhile, additional parameter count and computation do not translate monotonically into performance gains, creating explicit tradeoffs between performance and resource cost. To address these challenges, we propose HeteroWeave to reformulate heterogeneous pretrained model reuse as a variable-cardinality, multi-objective reassembly problem that explicitly accounts for performance–resource trade-offs. It jointly determines block identity and participation cardinality within a unified composition space, while shared interfaces and lightweight adaptation make heterogeneous compositions executable. To avoid training every candidate, we design a low-cost proxy that estimates performance from block-level activation-pattern statistics, using square-root aggregation to reduce score inflation as the activation space expands. HeteroWeave then optimizes the proxy, parameter count, and FLOPs as three objectives via Pareto-guided evolutionary search, retaining diverse nondominated architectures for different training and inference preferences. We validate HeteroWeave across four tasks, covering image classification, vision-language retrieval, semantic segmentation, and object detection. On classification, one nondominated representative improves accuracy by 1.08 percentage points over the single-block reassembly baseline, while using 26.7% fewer parameters and 11.9% fewer FLOPs.

1 INTRODUCTION

With continued advances in pretraining and downstream adaptation, models covering diverse scales and architectures have accumulated, providing a rich foundation for model reuse (Kadasi et al., 2025). Existing model reuse methods generally operate on complete models or checkpoints, including model selection (Zheng et al., 2026), downstream adaptation (Guo et al., 2019), and parameter merging (Wortsman et al., 2022; Xu et al., 2024). In contrast, pretrained model reassembly works at a finer granularity by reusing consecutive-layer blocks from heterogeneous pretrained models to construct architectures for target tasks (Bansal et al., 2021; Pan et al., 2023). This extends model reuse beyond complete models to the reorganization of internal computational structures.

Existing heterogeneous model reassembly methods can already provide multiple heterogeneous pretrained blocks for selection at each aligned functional position (Bandarkar et al., 2025; Mai et al., 2026). However, most methods still select only one block per position, treating pretrained computation from different sources as mutually exclusive alternatives (Yang et al., 2022; He et al., 2024; Pan et al., 2024). This limits cross-source composition because useful pretrained computation may be distributed across multiple sources that cannot be fully captured by any single block alone. Consequently, the diversity of candidate sources cannot be fully translated into equally diverse participation structures, limiting compositional flexibility. Although existing methods enable multiple paths through auxiliary branches (Zhang et al., 2020), adaptation modules (Chen et al., 2022), or routing mechanisms (Fang et al., 2023; Wang et al., 2025), these paths typically stem from structures attached to a predefined backbone rather than reusable blocks from independently pretrained models. Therefore, existing heterogeneous model reassembly has not yet fully explored variable-cardinality

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.

participation structures, where different functional positions in a reassembled architecture can involve different numbers of heterogeneous pretrained blocks that jointly contribute to computation.

Allowing variable numbers of pretrained blocks per position shifts heterogeneous model reassembly from single-block selection to the design of variable-cardinality participation structures. A block's utility is composition-dependent, as its contribution is determined by both its own computational transformation and other participating blocks, making isolated block evaluation insufficient to capture its role in a given composition (Xiao et al., 2022). Accordingly, block identity and participation cardinality become coupled architectural decisions. Higher cardinality can draw on more pretrained sources but also increases parameter count and computation. Moreover, the performance gain from adding blocks depends on their interaction with the existing composition and does not increase monotonically with resource use. Balancing performance and resource cost becomes a central architectural challenge in variable-cardinality heterogeneous reassembly (Sukthanker et al., 2024).

Efficiently exploring this reassembly space remains a key challenge. This difficulty arises because joint decisions over block identity and cardinality cause the search space to grow combinatorially, complicating the discovery of high-performing architectures. It is computationally infeasible to exhaustively enumerate and fully train all candidates to obtain their true performance (Tybl & Neumann, 2025). Although low-cost proxies provide a practical way to avoid full training for every candidate (Lin & Luo, 2026; Yang et al., 2024), variable-cardinality reassembly exacerbates scale-induced evaluation bias. As cardinality increases, the activation space expands, causing some activation-based proxy signals to favor larger compositions structurally even without corresponding performance gains. Therefore, efficient search in this space requires both low-cost candidate evaluation and reliably comparable performance estimates across compositions and cardinalities.

To address these challenges, we propose HeteroWeave, a variable-cardinality heterogeneous block reassembly framework with explicit performance and resource tradeoffs. At each position, HeteroWeave represents participation as a variable-size nonempty block set, jointly determining block identity and cardinality within a unified architecture space. Shared boundary interfaces and lightweight adaptation make cross-source, variable-cardinality compositions executable. To compare candidates reliably across scales, we design a tailored low-cost performance proxy. The proxy preserves module-specific response differences through activation-pattern statistics, and applies square-root aggregation to reduce score inflation as cardinality expands the activation space. HeteroWeave treats the performance proxy, parameter count, and FLOPs as independent objectives and optimizes them through Pareto-guided evolutionary search, retaining nondominated architectures with different performance-resource tradeoffs. A single search yields multiple nondominated architectures for later selection under different resource constraints and application requirements, avoiding repeated search for each budget. Our main contributions are summarized as follows:

(1) We reformulate single-block reassembly as variable-cardinality participation design, jointly determining block identity and participation cardinality under performance-resource tradeoffs.

(2) We design a training-free activation proxy based on module-level activation statistics that mitigates scale-induced bias and improves ranking reliability across different cardinalities.

(3) We jointly optimize performance proxy, parameters, and FLOPs in one Pareto-guided search, yielding multiple nondominated architectures for different resource constraints.

(4) Cross-task experiments show that performance varies nonmonotonically with cardinality and across tasks, validating the need to search rather than maximize participation.

2 Related Work

Pretrained Model Reassembly and Block Composition. Pretrained model blocks can support downstream tasks in different ways. One line of work extends a pretrained backbone with auxiliary branches, adapters, or routing mechanisms to enable multi-path computation (Zhang et al., 2020; Pfeiffer et al., 2021; Chen et al., 2022; Fang et al., 2023; Wang et al., 2025). Another directly selects and assembles blocks from multiple pretrained sources to construct new architectures (Bansal et al., 2021; Pan et al., 2023; Yang et al., 2022; Sanyal et al., 2026). Most reassembly methods operate within architecturally compatible models and retain a single block per functional position (Bansal et al., 2021; Pan et al., 2023; 2024; He et al., 2024). Subsequent search-based methods broaden

the source scope of candidate blocks while still preserving single-block participation. For example, DeRy searches cross-architecture block assignments under constraints on parameter count and FLOPs (Yang et al., 2022), whereas KLAS uses representation similarity to search cross-model stitching configurations (Sanyal et al., 2026). Existing multi-block formulations either prescribe the number of participating blocks (Mai et al., 2026) or enable variable-cardinality search within compatible models under objectives centered on task performance and fairness (Wu et al., 2026). Separately, existing studies have established the importance of resource objectives in architecture search (Dong et al., 2018; Tan et al., 2019; Cai et al., 2019; Zhao et al., 2024). HeteroWeave therefore studies variable-cardinality block reassembly across heterogeneous architectures, jointly searching block identity and participation cardinality while explicitly accounting for the resulting performance-resource trade-offs. Table 1 provides a multidimensional comparison.

Training-Free Candidate Evaluation. Training-free proxies enable low-cost ranking of candidate architectures in large search spaces (Abdelfattah et al., 2021; Krishnakumar et al., 2022; Lee & Ham, 2024). Activation-based methods, in particular, use response-pattern diversity as a signal of candidate performance (Mellor et al., 2021; Peng et al., 2024; Husistein et al., 2025). Under variable cardinality, however, increasing participation changes both the block composition and the size of the activation space, making proxy scores sensitive to structural scale. Consequently, increases in proxy scores may arise from structural expansion rather than corresponding improvements in actual performance. HeteroWeave targets this cross-cardinality comparability issue with a low-cost performance proxy designed to reduce scale-induced scoring bias.

Table 1: Comparison of representative pretrained model reuse methods. *Same*, *Compatible*, and *Different* denote reuse within the same pretrained model, across distinct models with compatible interfaces, and across heterogeneous architectures, respectively.

Method	Source Architecture	Block Cardinality	Selection Way	Search Objective
Side-Tuning (Zhang et al., 2020)	Same	> 1	Fixed	–
AdapterFusion (Pfeiffer et al., 2021)	Same	> 1	Fixed	–
AdaptFormer (Chen et al., 2022)	Same	> 1	Fixed	–
PatchNAS (Fang et al., 2023)	Same	> 1	Fixed	–
SEMA (Wang et al., 2025)	Same	> 1	Fixed	–
Model Stitching (Bansal et al., 2021)	Compatible	1	Fixed	–
SN-Net (Pan et al., 2023)	Compatible	1	Fixed	–
Stitched ViTs (Pan et al., 2024)	Compatible	1	Fixed	–
ESTA (He et al., 2024)	Compatible	1	Fixed	–
Layer Swapping (Bandarkar et al., 2025)	Compatible	1	Fixed	–
VFM Stitch Tree (Mai et al., 2026)	Compatible	> 1	Fixed	–
LEGO-LLM (Wu et al., 2026)	Compatible	> 1	Search	Performance $\uparrow$
KLAS (Sanyal et al., 2026)	Different	1	Search	Performance $\uparrow$
DeRy (Yang et al., 2022)	Different	1	Search	Performance $\uparrow$
HeteroWeave (Ours)	**Different**	> 1	**Search**	**Performance $\uparrow$ Params. $\downarrow$, FLOPs $\downarrow$**

3 METHOD

3.1 OVERVIEW OF HETEROWEAVE

We propose HeteroWeave, a variable-cardinality reassembly framework that constructs architectures from heterogeneous pretrained blocks. As shown in Figure 1(a), pretrained models are first partitioned into reusable blocks and organized into position-specific candidate pools, forming the basis for subsequent composition. Based on these pools, Figure 1(b) defines each position through a nonempty subset of candidate blocks, jointly encoding block identity and participation cardinality within a unified composition decision. Interface adaptation and fusion then connect the selected blocks, inducing a mapping from each composition encoding to an executable reassembled architecture. This mapping defines the architecture space explored in Figure 1(c). Evolutionary variation

proposes candidate architectures, while a low-cost performance proxy, parameter count, and FLOPs provide the three objectives for Pareto-guided search. Multiobjective selection progressively retains nondominated candidates across different performance–resource trade-offs, yielding a Pareto set from which representative architectures are selected for full training and evaluation.

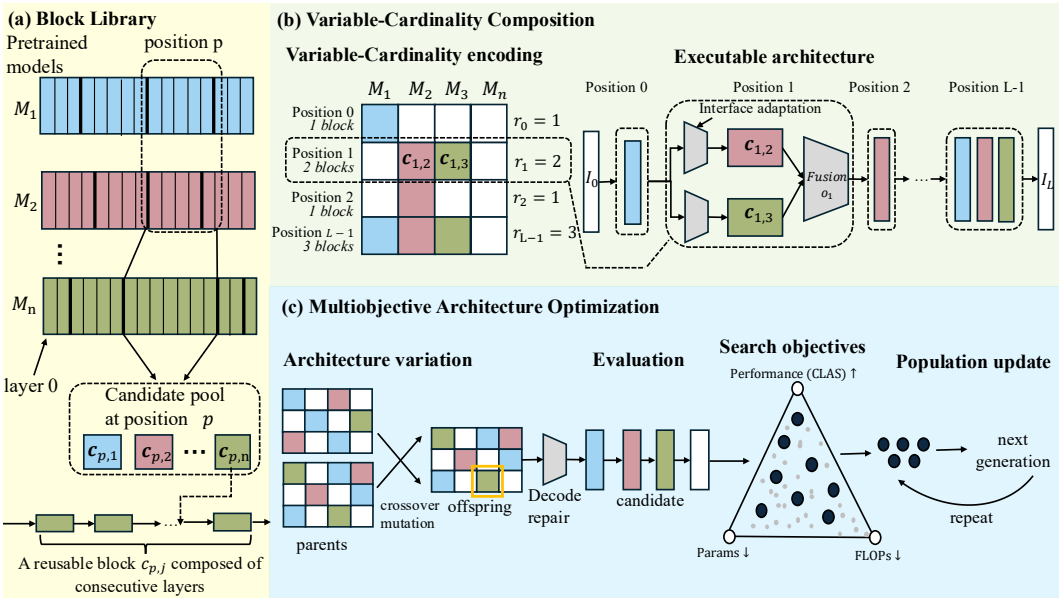

Figure 1: Overview of HeteroWeave. (a) Heterogeneous pretrained blocks are organized into position-specific candidate pools. (b) Variable-cardinality composition selects nonempty block subsets to construct an architecture. (c) Multiobjective search jointly optimizes a performance proxy, parameters, and FLOPs to obtain nondominated architectures and select representatives.

3.2 Variable-Cardinality Composition

3.2.1 Block Library and Composition Encoding

Let $\mathcal{M} = \{M_n\}_{n=1}^{N}$ denote a library of N heterogeneous pretrained source models, where M_n denotes the n-th source model. Following a previously validated strategy for cross-model partitioning and positional alignment (Yang et al., 2022), we decompose the source models into contiguous reusable blocks and align them across models by functional position. This procedure yields L position-specific candidate pools $\mathcal{C}_p = \{c_{p,j}\}_{j=1}^{N_p}$ for $p \in \{0, \ldots, L-1\}$, where N_p denotes the number of candidate blocks at position p and $c_{p,j}$ denotes the j-th candidate block.

At each position p, HeteroWeave represents variable-cardinality participation by a nonempty subset $S_p \subseteq \mathcal{C}_p$, whose cardinality $r_p = |S_p|$ specifies the number of participating blocks. Let $K_p \leq N_p$ denote the maximum participation cardinality allowed at position p, such that $1 \leq r_p \leq K_p$. This defines the admissible composition space at position p as

$$\mathcal{Z}_p = \{S_p \subseteq \mathcal{C}_p : 1 \leq |S_p| \leq K_p\}. \tag{1}$$

A complete composition is encoded by $\mathbf{z} = (S_0, \ldots, S_{L-1})$, which specifies the participating block subset at every position. The Cartesian product of these position-wise spaces defines the full composition space

$$\Omega_{\text{comp}} = \prod_{p=0}^{L-1} \mathcal{Z}_p. \tag{2}$$

Under this encoding, each S_p jointly specifies the identities of the participating blocks and their participation cardinality $r_p = |S_p|$, providing a unified representation of source selection and participation structure. When $K_p = 1$, the formulation reduces to conventional single-block selection, whereas $K_p > 1$ admits joint participation of multiple pretrained blocks at the same position.

3.2.2 INTERFACE ADAPTATION AND FUSION

The composition encoding specifies the participating blocks at each position, but heterogeneous sources may expose incompatible feature dimensions, spatial resolutions, and representation types. HeteroWeave resolves these mismatches with lightweight adapters deterministically specified by each source–target interface pair. Let α and β denote the source and target interfaces, respectively, and let $\tau_\beta \in \{\mathrm{cnn}, \mathrm{tok}\}$ denote the target representation type. For an input representation x, the adapter is defined as

$$\mathcal{A}_{\alpha\to\beta}(x) = \begin{cases} x, & \alpha = \beta, \\ \sigma(\mathrm{Conv}_W\left[\mathrm{BN}(\mathcal{R}_{\alpha\to\beta}(x))\right]), & \alpha \neq \beta,\ \tau_\beta = \mathrm{cnn}, \\ \sigma(\mathrm{Linear}_W\left[\mathrm{LN}(\mathcal{R}_{\alpha\to\beta}(x))\right]), & \alpha \neq \beta,\ \tau_\beta = \mathrm{tok}, \end{cases} \tag{3}$$

Here, $\alpha = \beta$ denotes matched representation type, feature dimension, and spatial or token layout. $\mathcal{R}_{\alpha\to\beta}$ performs CNN–token conversion and spatial alignment when needed, while Conv_W and Linear_W provide bias-free projections to the target dimension; σ denotes LeakyReLU. Together, normalization, projection, and structural conversion reconcile representation-scale, dimensional, and layout mismatches, respectively. Because the adapter is fully determined by the interface pair, it is instantiated rather than searched, leaving interface compatibility outside the architectural search space (Bansal et al., 2021; Pan et al., 2024; He et al., 2024).

Let $\mathcal{I}_p$ denote the common boundary interface at position p, and $\mathcal{I}_c^{\mathrm{in}}$ and $\mathcal{I}_c^{\mathrm{out}}$ the native input and output interfaces of block c, respectively. For each $c \in S_p$, the aligned branch output and fused representation are

$$u_{p,c} = \mathcal{A}_{\mathcal{I}_c^{\mathrm{out}}\to\mathcal{I}_{p+1}}\left[c\left(\mathcal{A}_{\mathcal{I}_p\to\mathcal{I}_c^{\mathrm{in}}}(h_p)\right)\right], \qquad c \in S_p, \tag{4}$$

$$h_{p+1} = \frac{1}{r_p}\sum_{c\in S_p} u_{p,c}, \qquad r_p = |S_p|. \tag{5}$$

Since all branch outputs share the interface $\mathcal{I}_{p+1}$, mean fusion is permutation-invariant over S_p, normalizes by r_p, and introduces no learnable fusion parameters. All instantiated interface modules are included in the parameter and FLOP objectives. Across all positions, this adaptation–execution–fusion process maps $\mathbf{z}$ to an executable architecture $A(\mathbf{z})$ while maintaining interface compatibility between heterogeneous blocks.

3.3 MULTIOBJECTIVE ARCHITECTURE OPTIMIZATION

3.3.1 PERFORMANCE AND RESOURCE OBJECTIVES

For a reassembled architecture $A = A(\mathbf{z})$, let $S(A)$ denote its task performance score obtained after full training, while $P(A)$ and $F(A)$ denote its parameter count and FLOPs, respectively. Let $\mathcal{R}$ denote the admissible region in the parameter–FLOPs space and $\mathcal{A}_\mathcal{R}$ the executable architectures within this region. To express all objectives in minimization form, we use $-S(A)$ for task performance and formulate reassembly as

$$\underset{A\in\mathcal{A}_\mathcal{R}}{\text{minimize}} \quad \mathbf{f}(A) = \left[-S(A), P(A), F(A)\right]. \tag{6}$$

The parameter count and feature-extraction FLOPs include the selected pretrained blocks together with the stem, interface transformations, and fusion operations. The shared task head is included in the parameter count but excluded from FLOPs:

$$P(A) = P_{\mathrm{stem}} + P_{\mathrm{head}} + \sum_{p=0}^{L-1}\sum_{c\in S_p} P(c) + P_{\mathrm{int}}(A) + P_{\mathrm{fuse}}(A), \tag{7}$$

$$F(A) = F_{\mathrm{stem}} + \sum_{p=0}^{L-1}\sum_{c\in S_p} F(c) + F_{\mathrm{int}}(A) + F_{\mathrm{fuse}}(A). \tag{8}$$

Here, $P_{\text{int}}(A)$ and $F_{\text{int}}(A)$ account for all interface modules, while $P_{\text{fuse}}(A)$ and $F_{\text{fuse}}(A)$ account for fusion-specific parameters and computation. For the default mean fusion, $P_{\text{fuse}}(A) = 0$.

While $P(A)$ and $F(A)$ are directly computable, obtaining $S(A)$ requires full training and evaluation, making exhaustive evaluation over the enlarged composition space impractical. We therefore use a low-cost proxy $\widehat{S}(A)$ during search to rank candidates before full training and evaluation.

3.3.2 Low-Cost Score Estimation

Activation-based proxies may favor larger compositions because increasing participation cardinality expands the activation space without necessarily yielding proportional performance gains. We introduce the Cardinality-Aware Layerwise Activation Score (CLAS) to moderate this cardinality-dependent score growth and retain comparable rankings as cardinality changes.

Given a fixed unlabeled sample set $\mathcal{X}$, let $\mathcal{L}(A)$ denote the monitored modules of architecture A and $a_\ell(A; \mathcal{X})$ the activation output of module ℓ. We encode its flattened activations as binary response patterns

$$B_\ell(A) = \mathbf{1}\left[\text{Flat}\left(a_\ell(A; \mathcal{X})\right) > 0\right]. \tag{9}$$

Let d_ℓ denote the flattened activation dimension. We define the proxy performance score as

$$\widehat{S}(A) = \text{CLAS}(A) = \sum_{\ell \in \mathcal{L}(A)} \sqrt{|\{B_\ell(A)_{:,j} : j = 1, \ldots, d_\ell\}|}. \tag{10}$$

The module-wise aggregation preserves local activation structure, while the square root assigns diminishing marginal credit as the number of unique patterns increases, thereby moderating score growth caused by activation-space expansion.

3.3.3 Pareto-Guided Evolutionary Search

Substituting the performance score for the unavailable full-training score gives the search-time objective vector

$$\widehat{\mathbf{f}}(A) = \left[-\widehat{S}(A), P(A), F(A)\right]. \tag{11}$$

HeteroWeave uses NSGA-III (Fan, 2014) to explore the variable-cardinality composition space under these objectives and obtain a nondominated candidate set $\widehat{\mathcal{P}}$ spanning different performance–resource trade-offs. Since no single architecture is optimal across all objectives, representative architectures are selected from $\widehat{\mathcal{P}}$ only after the Pareto set is fixed. We retain representatives from prespecified resource regions together with a balanced operating point, and determine their identities before full training. Detailed search and selection are provided in Appendices A.

4 Experiments (Detailed in Appendix B)

We evaluate whether variable-cardinality reassembly can improve the attainable performance–resource trade-off and generalize across downstream tasks with different prediction structures. We assess this cross-task applicability across four tasks, including image classification, vision–language retrieval, semantic segmentation, and object detection. We further examine whether the classification gains arise from variable-cardinality participation structure rather than increased capacity alone, whether CLAS can reliably identify promising architectures without full training, and how Pareto-guided search compares with random architecture selection in identifying performance–resource trade-offs. More experimental settings and analyses are provided in Appendix B.

4.1 Main Results

4.1.1 ImageNet Classification

We evaluate HeteroWeave on ImageNet-1K (Deng et al., 2009) with respect to both predictive performance and resource cost, setting the maximum participation cardinality to $K_p = 2$ at variable-cardinality positions, so that each admits either one or two participating blocks. Table 2 compares

HeteroWeave with model reuse baselines trained under the same protocol. Since SN-Net (Pan et al., 2023) diverged under this protocol, we report its published results, alongside standalone pretrained architectures at comparable resource scales as non-reassembled references. Figure 2 further examines training stability through convergence behavior and final accuracy across matched seeds, together with parameter and FLOP comparisons against DeRy.

Table 2: ImageNet-1K classification results under compact (10M/3G) and medium (30M/6G) regimes. *Shared protocol* uses the common 100 epoch training setting, while *Scale references* are taken from original sources. HeteroWeave-P and HeteroWeave-E are representatives favoring performance and efficiency, respectively, while HeteroWeave balances both.

	Regime	Method	Params.	FLOPs	Top-1	Top-5
Shared protocol	10M/3G	Model Stitching	6.67	1.45	72.60	91.18
		Side-Tuning	7.80	0.29	73.83	91.74
		DeRy	7.83	2.99	76.71	93.24
		HeteroWeave-P (Ours)	8.61	2.38	**76.74**	**93.26**
		HeteroWeave-E (Ours)	7.71	1.62	76.24	92.99
	30M/6G	Model Stitching	24.86	3.33	76.45	93.21
		Side-Tuning	23.28	3.73	73.33	91.25
		DeRy	24.89	4.48	78.38	94.30
		HeteroWeave (Ours)	**18.26**	3.95	**79.46**	**94.67**
Scale references	10M/3G	SN-Net (Pan et al., 2023)	10.00	2.10	72.60	–
		RegNetY-800MF (Radosavovic et al., 2020)	6.26	0.83	76.42	93.14
		MobileNetV3-L (Howard et al., 2019)	5.40	0.22	75.20	–
	30M/6G	SN-Net (Pan et al., 2023)	18.00	3.80	78.20	–
		RegNetY-3.2GF (Radosavovic et al., 2020)	19.44	3.18	78.95	94.58
		ResNet-50 (He et al., 2016)	25.56	4.12	76.80	93.30
		Swin-T (Liu et al., 2021)	28.29	4.36	78.30	94.60

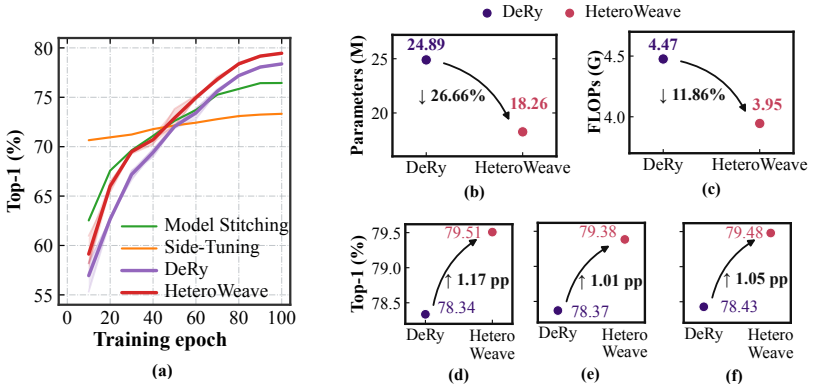

Figure 2: ImageNet-1K comparison under the shared 100 epoch protocol. (a) Training dynamics. (b)–(c) Resource comparison with DeRy. (d)–(f) Final Top-1 accuracy across three matched seeds.

Resource-efficient performance across scales. At the compact scale, HeteroWeave-P achieves the highest accuracy among the shared protocol methods with lower FLOPs than DeRy, while HeteroWeave-E further reduces resource consumption with only a modest accuracy decrease. Together, these representatives provide viable solutions for different performance and resource requirements. At the medium scale, the search-based reassembly methods, DeRy and HeteroWeave, achieve strong accuracy under the shared protocol, as shown in Table 2, indicating that explicit exploration of the reassembly space can provide additional opportunities to identify favorable architectures beyond predefined reuse configurations. HeteroWeave further improves over DeRy by 1.08 Top-1 points while using 26.6% fewer parameters and 11.8% fewer FLOPs. Since DeRy employs single-block serial reassembly, this additional improvement suggests that variable-cardinality composition can better exploit complementary information across pretrained sources through joint multi-block participation.

Convergence behavior and seed robustness. As shown in Figure 2(a), early performance does not predict the final ranking. HeteroWeave starts below Side-Tuning, but improves steadily and ultimately achieves the highest accuracy. Figure 2(b)–(f) further shows consistent parameter and FLOP reductions together with similar accuracy gains across all three matched seeds, indicating that the improved accuracy–resource trade-off is not specific to a single training run.

4.1.2 CROSS-TASK GENERALIZATION

Composition effects vary across tasks. HeteroWeave transfers beyond classification, improving retrieval and segmentation but only marginally affecting detection (Table 3). At fixed cardinality and resource cost, segmentation is more sensitive to block identity, likely because dense prediction relies more on spatially detailed, source-specific features, whereas retrieval varies less. Detection changes even less, suggesting that downstream task structure can attenuate participation effects.

Higher cardinality is not always better. Increasing cardinality does not yield monotonic performance gains. On segmentation, the two-block HeteroWeave-S4 outperforms the three-block S3 (76.08 vs. 74.83 mIoU) while using roughly half the added parameters and FLOPs. Retrieval exhibits the same tendency more mildly, with only a 0.19-point Mean Recall gain despite nearly doubling resource cost. These results indicate that the utility of additional blocks depends on both the task and the composition, making cardinality a variable to optimize rather than maximize.

Table 3: Cross-task evaluation of HeteroWeave on retrieval, segmentation, and detection. Scores are Mean Recall for retrieval, mIoU for segmentation, and AP50 for detection. ΔParams and ΔFLOPs are measured relative to the task baseline. Values are reported in percent as mean ± sample standard deviation over three seeds; the frozen CLIP reference is deterministic and has zero variance.

Task	Architecture	Cardinality	ΔParams.(M)	ΔFLOPs (G)	Score (%)
Retrieval	Frozen CLIP (baseline)	1	–	–	83.05 ± 0.00
	HeteroWeave-R1	2	7.09	1.46	87.34 ± 0.30
	HeteroWeave-R2	3	14.23	2.92	$\mathbf{87.53 \pm 0.20}$
	HeteroWeave-R3	3	14.23	2.92	87.37 ± 0.34
Segmentation	ResNet-50 (baseline)	1	–	–	73.74 ± 0.07
	HeteroWeave-S1	3	14.26	16.38	74.45 ± 0.12
	HeteroWeave-S2	3	14.26	16.38	74.47 ± 0.24
	HeteroWeave-S3	3	14.26	16.38	74.83 ± 0.27
	HeteroWeave-S4	2	7.13	8.19	$\mathbf{76.08 \pm 0.13}$
Detection	Faster R-CNN (baseline)	1	–	–	87.02 ± 0.88
	HeteroWeave-D1	2	7.10	14.35	87.05 ± 1.11
	HeteroWeave-D2	2	7.10	14.35	86.96 ± 0.78
	HeteroWeave-D3	2	14.97	7.93	$\mathbf{87.06 \pm 0.93}$

4.2 IN-DEPTH ANALYSIS

Composition matters beyond capacity. Table 4 controls for added capacity by comparing serial and parallel extensions of the same DeRy baseline with matched parameters and FLOPs. The 1.87-point advantage of parallel composition shows that the improvement depends on how pretrained blocks are organized to participate, rather than on added capacity alone.

Table 4: Capacity-matched control isolating the effect of composition from added capacity. Results are mean ± standard deviation over three seeds.

Setting	Capacity organization	Params. (M)	FLOPs (G)	Top-1 (%)
Baseline	DeRy serial path	24.89	4.48	71.80 ± 0.09
Capacity-matched	Serial extension	29.18	4.69	71.92 ± 0.14
Capacity-matched	**Parallel composition**	29.18	4.70	$\mathbf{73.79 \pm 0.25}$

CLAS provides a reliable ranking signal. Table 5 compares CLAS with established proxies (Abdelfattah et al., 2021; Krishnakumar et al., 2022; Mellor et al., 2021; Li et al., 2023; Peng et al., 2024). CLAS achieves the highest Spearman correlation in both topology pools, reaching 0.775 and 0.570, respectively. It outperforms all competing proxies and maintains a clear advantage on Pool 2, where several existing methods exhibit weak or negative correlations. These results show that its cardinality-aware formulation provides more reliable rankings for variable-cardinality architectures.

Pareto search retains distinct tradeoffs. Figure 3(a) shows that the three-objective search preserves distinct performance and resource trade-offs across the archive. Figure 3(b) further shows that HeteroWeave architecture outperforms randomly sampled candidates at a comparable computation level. Together, these results support Pareto-guided search for both diverse resource preferences and effective architecture selection.

Table 5: Spearman rank correlation between zero-cost proxies and short-training Top-1. Pool 1 and Pool 2 use one-epoch and five-epoch targets, respectively.

Proxy	Pool 1	Pool 2
CLAS (Ours)	**0.775**	**0.570**
ZiCo	0.727	0.517
GradNorm	0.731	0.382
SynFlow	0.723	0.351
NASWOT	0.619	0.268
SWAP	0.648	0.232
Fisher	0.518	-0.066
SNIP	0.471	-0.067
Jacov	0.338	-0.102

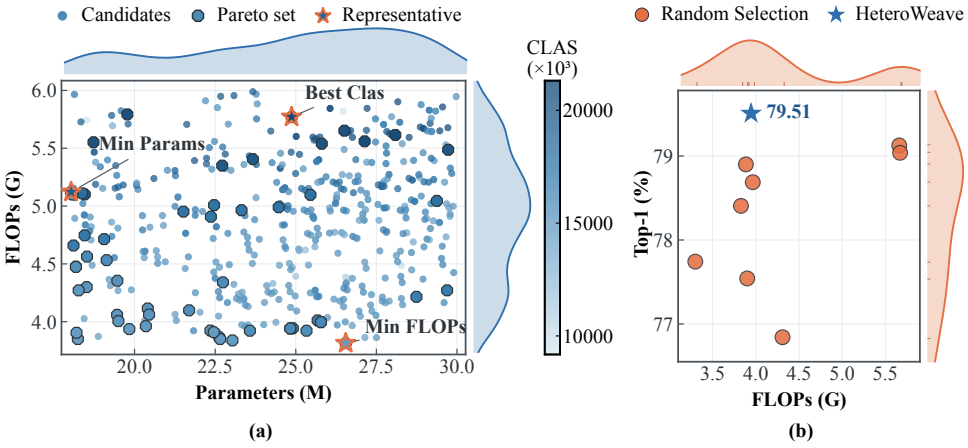

Figure 3: Search outcomes of HeteroWeave. (a) Three-objective archive with the strict Pareto set and representative solutions selected for performance and resource efficiency. (b) Top-1–FLOPs comparison between the selected HeteroWeave architecture and randomly sampled architectures; marginal densities show the corresponding distributions.

5 Conclusion

This work introduces HeteroWeave, a variable-cardinality reassembly framework that jointly determines block identity and participation cardinality under explicit performance, parameter, and computation objectives. As the composition space expands with variable cardinality, it is impractical to fully train every candidate. HeteroWeave therefore uses CLAS to rank candidates without full training and Pareto-guided search to retain nondominated architectures spanning different performance–resource trade-offs. Across the four evaluated tasks, reassembly effectiveness varies with participation structure, showing that component identity alone is insufficient to characterize a useful composition. Capacity-matched comparisons further show that increased capacity alone is insufficient to account for the observed gains, indicating that composition structure is itself a consequential architectural factor. Moreover, the nonmonotonic effect of cardinality further shows that increasing participation is not uniformly beneficial, supporting cardinality as an architectural variable to optimize rather than maximize. Taken together, these findings position pretrained model reassembly as a problem of designing participation structures under performance and resource objectives, rather than selecting blocks in isolation. A natural next step is to determine how such participation structures should adapt as the composition landscape changes across downstream tasks.

Reproducibility Statement

To support reproducibility, we provide detailed descriptions of the HeteroWeave search space, variable-cardinality reassembly mechanism, CLAS proxy, multi-objective search procedure, training protocols, and experimental settings in the main paper and appendix. The appendix further reports implementation details, hyperparameter settings, model configurations, and additional experimental analyses. Anonymous source code, including the implementations of architecture search, proxy evaluation, model training and evaluation, as well as the model architecture definitions and configurations corresponding to the architectures reported in the paper, is available at `https://anonymous.4open.science/r/HeteroWeave-28F5/`.

AI use statement

In this work, we used generative AI tools to support research ideation and execution, including providing feedback on the conceptual framing, research methodology, and experimental design, and assisting with the interpretation of experimental results. We did not use generative AI tools to generate synthetic datasets, formulate or prove mathematical claims, implement the proposed method, translate research materials, clean or reformat datasets, or conduct qualitative or thematic data analysis; other required-disclosure tasks were not applicable to this work.

Additionally, we used generative AI tools for literature search and summarization, brainstorming, drafting and editing parts of the manuscript, improving readability, and providing suggestions on manuscript structure, titles, and scientific presentation. All AI-assisted research suggestions, methodological feedback, experimental interpretations, and manuscript content were critically reviewed by the authors. Scientific and factual claims were independently checked against the experimental evidence and relevant literature. The authors take full responsibility for the final content of this work, including all text, claims, results, and artifacts produced with the aid of generative AI.

References

Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*, 2021.

Lucas Bandarkar, Benjamin Muller, Pritish Yuvraj, Rui Hou, Nayan Singhal, Hongjiang Lv, and Bing Liu. Layer swapping for zero-shot cross-lingual transfer in large language models. In *International Conference on Learning Representations*, volume 2025, pp. 51315–51338, 2025.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in neural information processing systems*, 34:225–236, 2021.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9630–9640. IEEE, 2021.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in neural information processing systems*, 35:16664–16678, 2022.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In *European conference on computer vision*, pp. 540–555. Springer, 2018.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

Zhun Fan. An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 2014.

Yuchu Fang, Wenzhong Li, Yao Zeng, Yang Zheng, Zheng Hu, and Sanglu Lu. Patchnas: repairing dnns in deployment with patched network architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(12):14811–14819, 2023.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pp. 178–178. IEEE, 2004.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4800–4809. IEEE, 2019.

Haoyu He, Zizheng Pan, Jing Liu, Jianfei Cai, and Bohan Zhuang. Efficient stitchable task adaptation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28555–28565. IEEE, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 15979–15988. IEEE, 2022.

Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899, 2013.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.

Raphael Husistein, Markus Reiher, and Marco Eckhoff. Near: a training-free pre-estimator of machine learning model performance. In *International Conference on Learning Representations*, volume 2025, pp. 68153–68173, 2025.

Pritam Kadasi, Sriman Reddy Kondam, Srivathsa Vamsi Chaturvedula, Rudranshu Sen, Agnish Saha, Soumavo Sikdar, Sayani Sarkar, Suhani Mittal, Rohit Jindal, and Mayank Singh. Model hubs and beyond: analyzing model popularity, performance, and documentation. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 19, pp. 965–993, 2025.

Arjun Krishnakumar, Colin White, Arber Zela, Renbo Tu, Mahmoud Safari, and Frank Hutter. Nasbench-suite-zero: Accelerating research on zero cost proxies. *Advances in Neural Information Processing Systems*, 35:28037–28051, 2022.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Junghyup Lee and Bumsub Ham. Az-nas: Assembling zero-cost proxies for network architecture search. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5893–5903. IEEE, 2024.

Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. Zico: Zero-shot nas via inverse coefficient of variation on gradients. *arXiv preprint arXiv:2301.11300*, 2023.

Mingzhuo Lin and Jianping Luo. Per-architecture training-free metric optimization for neural architecture search. *Advances in Neural Information Processing Systems*, 38:92152–92193, 2026.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 936–944. Ieee, 2017.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9992–10002. Ieee, 2021.

Zheda Mai, Ke Zhang, Fu-En Wang, Zixiao Ken Wang, Albert YC Chen, Lu Xia, Min Sun, Wei-Lun Chao, and Cheng-Hao Kuo. Revisiting model stitching in the foundation model era. *arXiv preprint arXiv:2603.12433*, 2026.

Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International conference on machine learning*, pp. 7588–7598. PMLR, 2021.

Zizheng Pan, Jianfei Cai, and Bohan Zhuang. Stitchable neural networks. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16102–16112. IEEE, 2023.

Zizheng Pan, Jing Liu, Haoyu He, Jianfei Cai, and Bohan Zhuang. Stitched vits are flexible vision backbones. In *European Conference on Computer Vision*, pp. 258–274. Springer, 2024.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Yameng Peng, Andy Song, Haytham Fayek, Vic Ciesielski, and Xiaojun Chang. Swap-nas: Sample-wise activation patterns for ultra-fast nas. In *International Conference on Learning Representations*, volume 2024, pp. 31100–31125, 2024.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: main volume*, pp. 487–503, 2021.

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.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. *arXiv preprint arXiv:2003.13678*, 2020.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Debopam Sanyal, Anantharaman Iyer, Alind Khare, Trisha Jain, Akshay Jajoo, Myungjin Lee, James Kerce, and Alexey Tumanov. Klas: Using similarity to stitch neural networks for improved accuracy-efficiency tradeoffs. In *International Conference on Learning Representations*, volume 2026, pp. 144515–144537, 2026.

Rhea Sanjay Sukthanker, Arber Zela, Benedikt Staffler, Samuel Dooley, Josif Grabocka, and Frank Hutter. Multi-objective differentiable neural architecture search. *arXiv preprint arXiv:2402.18213*, 2024.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2815–2823. IEEE, 2019.

Ondrej Tybl and Lukas Neumann. Training-free neural architecture search through variance of knowledge of deep network weights. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14881–14890. IEEE, 2025.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. Self-expansion of pre-trained models with mixture of adapters for continual learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10087–10098. IEEE, 2025.

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.

Xingyu Wu, Yu Zhou, and Kay Chen Tan. Building llms like lego: Two-dimensional architecture reassembly of large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44939–44956, 2026.

Han Xiao, Ziwei Wang, Zheng Zhu, Jie Zhou, and Jiwen Lu. Shapley-nas: Discovering operation contribution for neural architecture search. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11882–11891. IEEE, 2022.

Zhengqi Xu, Ke Yuan, Huiqiong Wang, Yong Wang, Mingli Song, and Jie Song. Training-free pretrained model merging. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5915–5925. IEEE, 2024.

Jiechao Yang, Yong Liu, Wei Wang, Haoran Wu, Zhiyuan Chen, and Xibo Ma. Patnas: A path-based training-free neural architecture search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(3):1484–1500, 2024.

Xingyi Yang, Daquan Zhou, Songhua Liu, Jingwen Ye, and Xinchao Wang. Deep model reassembly. *Advances in neural information processing systems*, 35:25739–25753, 2022.

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-tuning: a baseline for network adaptation via additive side networks. In *European conference on computer vision*, pp. 698–714. Springer, 2020.

Yiyang Zhao, Linnan Wang, and Tian Guo. Multi-objective neural architecture search by learning search space partitions. *Journal of Machine Learning Research*, 25(177):1–41, 2024.

Yaoyan Zheng, Huiqun Wang, Nan Zhou, and Di Huang. Implicit modeling for transferability estimation of vision foundation models. *Advances in Neural Information Processing Systems*, 38: 36718–36745, 2026.

A Additional Method Details

Composition and evaluation implementation. Using the candidate pools and cardinality constraints defined for each position in Section 3.2, the nominal size of the variable-cardinality composition space is

$$|\Omega_{\text{comp}}| = \prod_{p=0}^{L-1} \left[\sum_{r=1}^{K_p} \binom{N_p}{r} \right]. \tag{12}$$

Not every nominal encoding necessarily produces an executable reassembled architecture. Each sampled encoding is first repaired to satisfy $1 \le |S_p| \le K_p$ and then decoded. Let $V(\mathbf{z}) \in \{0, 1\}$ indicate whether all required interface adapters can be constructed and the complete architecture produces a valid forward pass. The executable composition space is

$$\Omega_{\text{exec}} = \{\mathbf{z} \in \Omega_{\text{comp}} : V(\mathbf{z}) = 1\}. \tag{13}$$

Candidates outside Ω_{exec} or the admissible resource region $\mathcal{R}$ are excluded before CLAS evaluation. Equivalent encodings are identified by their decoded phenotype and reuse cached objective values.

The default experiments use the mean fusion defined in Equation 5. In the gated-fusion ablation, it is replaced by

$$\mathbf{w}_p = \text{softmax}(\boldsymbol{\alpha}_p), \qquad h_{p+1} = \sum_i w_{p,i} u_p^{(i)}, \tag{14}$$

where $\boldsymbol{\alpha}_p$ is initialized to zero and optimized only during full training. Fusion is otherwise fixed during architecture search.

CLAS is evaluated using a fixed set of 32 unlabeled ImageNet-1K images shared across all candidate architectures. Candidate parameters remain fixed, and each candidate requires one forward pass. Every candidate uses the same input images, preprocessing, parameter initialization, and monitored module definitions. The monitored activations include those from the participating pretrained blocks and their interface adapters.

Evolutionary search and evaluated archive. HeteroWeave uses NSGA-III (Fan, 2014) with a population of 96. A 40-generation exploratory run is followed by a 120-generation search over the full variable-cardinality composition space. The two runs use separate populations but share the evaluated archive. NSGA-III uses 91 Das–Dennis reference directions, simulated binary crossover with probability 0.9 and distribution index $\eta = 15$, and polynomial mutation with distribution index $\eta = 20$. For every proposed candidate, the implementation repairs its cardinalities, decodes the participating block subsets, constructs the required interface adapters, and verifies the forward pass. Parameter count and FLOPs are then checked against the admissible resource region. If the decoded phenotype has already been evaluated, its cached objective values are reused. Otherwise, CLAS is computed and the valid candidate is added to the evaluated archive. After search, representative architectures are selected from the resulting Pareto set before full training and evaluation.

Representative architecture selection. Representative architectures are selected only after the evaluated archive and its strict Pareto set $\widehat{\mathcal{P}}$ have been fixed. Let $\{\mathcal{R}_b\}_{b=1}^{B}$ denote prespecified regions in the parameter–FLOP plane. For every nonempty region, the representative architecture is the Pareto candidate with the highest CLAS score:

$$A_b = \underset{A \in \widehat{\mathcal{P}},\ (P(A),F(A)) \in \mathcal{R}_b}{\arg\max} \widehat{S}(A). \tag{15}$$

The regional representatives provide operating points for different requirements on performance and resource cost without rerunning the search.

A balanced operating point is selected separately from the same fixed Pareto set. We first orient the three objectives so that larger values are preferred:

$$\phi_S(A) = \widehat{S}(A), \qquad \phi_P(A) = -P(A), \qquad \phi_F(A) = -F(A). \tag{16}$$

Each quantity is then min–max normalized over $\widehat{\mathcal{P}}$:

$$U_j(A) = \frac{\phi_j(A) - \phi_j^{\min}}{\phi_j^{\max} - \phi_j^{\min}}, \qquad j \in \{S, P, F\}. \tag{17}$$

If $\phi_j^{\max} = \phi_j^{\min}$, we set $U_j(A) = 0$ for that objective. The balanced operating point is

$$A_{\text{bal}} = \arg\max_{A \in \widehat{\mathcal{P}}} \left[U_S(A) + U_P(A) + U_F(A) \right]. \tag{18}$$

Duplicate selections are removed when a regional representative and the balanced operating point correspond to the same architecture. Ties are resolved according to the fixed archive order.

B Experimental Details and Additional Results

B.1 Experimental Protocols

Data, preprocessing, and measurement. ImageNet-1K (Deng et al., 2009) uses its standard training and validation sets. Flickr8K (Hodosh et al., 2013) uses the first 6,000 examples of its public training split for optimization and its 1,000-image validation split for evaluation, retaining all five captions for each validation image. The Oxford-IIIT Pets (Parkhi et al., 2012) experiments use 2,880 training images and a fixed 800-image validation split. PASCAL VOC 2007 trainval (Everingham et al., 2010) is divided into 4,011 training and 1,000 validation images. The Flickr8K test split, Pets test split, and VOC test set are not used for architecture or hyperparameter selection. The frozen-transfer evaluation additionally covers Caltech-101 (Fei-Fei et al., 2004), CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009), CUB-200-2011 (Wah et al., 2011), and the Describable Textures Dataset (Cimpoi et al., 2014), alongside Oxford-IIIT Pets.

Within each task, directly compared variants use the same data split, training schedule, checkpoint rule, and evaluation protocol. Repeated downstream comparisons use matched random seeds. Parameter count and FLOPs are compared within each task because input resolution and the task-specific head differ across settings. ImageNet training uses random resized cropping to 224×224, horizontal flipping, RandAugment, and ImageNet normalization. Validation resizes the shorter side to 256 and applies a 224×224 center crop. ImageNet FLOPs are measured at 224×224 along the feature-extraction path and exclude the shared classifier. Flickr8K uses the CLIP (Radford et al., 2021) ViT-B/16 processor at 224×224 and token sequences of at most 77 tokens. Pets uses 192×192 inputs during candidate screening and 256×256 inputs during final training. Detection follows Faster R-CNN (Ren et al., 2015) preprocessing with a short side of 384 pixels and a maximum long side of 640 pixels; FLOPs are measured using the same input convention.

ImageNet search and training. Each source model is partitioned into blocks aligned at four positions. Position 0 uses $K_0 = 1$, while Positions 1–3 use $K_p = 2$. Complete-system comparisons follow a common 100-epoch protocol. Training uses AdamW with an initial learning rate of 10^{-3}, weight decay 0.05, linear warmup, cosine decay, and an effective global batch size of 1,024. Label smoothing is set to 0.1. Mixup and CutMix are each applied with probability 0.5, using $\alpha = 0.1$ and $\alpha = 1.0$, respectively. The participating pretrained blocks, interface adapters, fusion parameters, and classifier are optimized jointly. Validation and checkpointing are performed every 10 epochs, and the final validation accuracy is reported. DeRy (Yang et al., 2022), Model Stitching (Bansal et al., 2021), and Side-Tuning (Zhang et al., 2020) are reproduced using this training schedule. SN-Net (Pan et al., 2023) and the complete-network references use results reported by their original sources and are identified separately in Table 2.

Vision–language retrieval. The candidate pool is constructed for aligned block 10 of the CLIP visual encoder, with maximum participation cardinality $K_p = 3$. The evaluated compositions include CLIP + MAE (He et al., 2022) and CLIP + MAE + DINO (Caron et al., 2021), together with the same-source control CLIP + MAE + MAE. Wider and parallel adapter configurations serve as adaptation-budget controls. CLIP + MAE uses bottleneck width 4 and residual scale 0.05. The cardinality-three compositions and the parallel-adapter control use width 16 and scale 0.025, whereas the wider-adapter control uses width 32 and scale 0.025. All compositions use sum fusion. Retrieval training uses AdamW for 1,000 optimizer steps with batch size 32, learning rate 3×10^{-3},

weight decay 10^{-4}, and seeds 0, 1, and 2. The objective combines symmetric CLIP contrastive loss with an image-feature alignment loss weighted by 0.1. The CLIP encoders and participating pretrained blocks remain frozen, and only the introduced interface adapters are optimized. The final training state is reported without validation-based checkpoint selection.

Semantic segmentation. The candidate pools are constructed for aligned blocks 2–4 of ResNet-50 (He et al., 2016) and permit maximum participation cardinality $K_p = 3$. A fixed pool of 96 candidate architectures is evaluated for 150 training steps to characterize the composition landscape. Final comparisons use 10,000 iterations with batch size 8 at 256×256 resolution over seeds 0, 1, and 2. The participating pretrained blocks remain frozen, while the segmentation head and lightweight interface adapters are optimized. The final checkpoint is reported. The segmentation comparisons include ResNet-50 + Random + Random, ResNet-50 + DINO + DINO, ResNet-50 + DINO + SwAV (Caron et al., 2020), and the cardinality-two ResNet-50 + DINO composition. The cardinality-three compositions use the aligned block 3 of ResNet-50, residual scale 0.025, bottleneck width 16, and normalized weighted-sum fusion. These settings hold the evaluated block and interface configuration fixed while varying the source and organization of the participating blocks.

Object detection. Object detection uses Faster R-CNN with a ResNet-50 backbone and FPN (Lin et al., 2017). The candidate pools are constructed for aligned blocks 2–4 of ResNet-50 and permit maximum participation cardinality $K_p = 4$. A fixed pool of 64 candidate architectures is screened for 300 training steps. Final comparisons use 5,000 steps with batch size 2 over seeds 0, 1, and 2. The pretrained ResNet blocks remain frozen, while FPN, RPN, ROI heads, and the lightweight interface adapters are optimized. The final checkpoint is reported. The final comparison evaluates cardinality-two ResNet-50 + BYOL (Grill et al., 2020) compositions at aligned blocks 3 and 4 of ResNet-50. The additional block is combined residually with the ResNet output using $\gamma \in \{0.001, 0.0025, 0.005\}$. This scale controls the residual contribution during optimization and does not alter the composition encoding or participation cardinality.

Analysis protocols. The capacity-matched experiment compares two extensions of the same serial DeRy architecture. The serial extension adds computation along the existing route, whereas the parallel composition introduces an aligned heterogeneous pretrained block at the same position. The two extensions have identical parameter counts and nearly identical FLOPs. All variants use the same 20-epoch ImageNet protocol and seeds 11, 23, and 47.

The CLAS ranking experiment uses two candidate pools constructed independently of the formal search. Pool 1 and Pool 2 contain 23 and 25 candidate architectures and use Top-1 accuracy after one and five epochs, respectively, as ranking targets. Ranking reliability is measured using Spearman correlation. Partial rank correlation with Top-1 is computed after controlling parameter and FLOP ranks. CLAS is evaluated using model-initialization seeds 11, 17, and 23, and ranking stability is measured by the mean pairwise Spearman correlation across these seeds.

For the cardinality analysis, segmentation and detection candidates are grouped by participation cardinality $r_p = |S_p|$. Segmentation uses the 150-step screening results, whereas detection uses the 300-step screening results. Individual candidate measurements are retained, and each group is summarized using its mean and sample standard deviation. These screening results characterize the composition landscape and are not used as final task results.

The Pareto analysis uses the frozen ImageNet evaluated archive. Nondominance is determined jointly from CLAS, parameter count, and FLOPs. For visualization, the strict Pareto set is projected onto the parameter–FLOP plane, with CLAS represented by color. The displayed architectures are selected from the frozen Pareto set before full training.

B.2 Evaluated Architectures

ImageNet architectures. Table 6 reports the source-model compositions of the three ImageNet architectures evaluated in Table 2. Positions 0–3 follow the four-position composition encoding defined in Section 3. HeteroWeave-P prioritizes predictive performance, HeteroWeave-E prioritizes resource efficiency, and HeteroWeave is the representative reported for the 30M-parameter and 6G-FLOP regime.

Table 6: Source-model compositions of the ImageNet representatives. Models joined by "+" contribute jointly at the same position rather than forming a serial connection. Participation cardinality counts the model-derived components used at that position.

Position	Participating pretrained model or models	Participation cardinality
HeteroWeave-P	*Compact search regime: 10M parameters / 3G FLOPs*	
0	ResNet-50	1
1	RegNetY-1.6GF + RegNetY-16GF	2
2	RegNetY-800MF	1
3	RegNetY-1.6GF	1
HeteroWeave-E	*Compact search regime: 10M parameters / 3G FLOPs*	
0	ResNet-50	1
1	RegNetY-1.6GF	1
2	RegNetY-800MF	1
3	RegNetY-1.6GF	1
HeteroWeave	*Medium search regime: 30M parameters / 6G FLOPs*	
0	Semi-weakly supervised ResNeXt-50	1
1	ResNet-50	1
2	ResNet-50 + Vision Transformer Small	2
3	Vision Transformer Small	1

Table 7: Architectures used in the cross-task evaluation of Table 3. Models joined by "+" contribute jointly, and participation cardinality includes the model retained from the task baseline. Architecture names match the main-results table. CLIP, MAE, DINO, SwAV, and BYOL identify pretrained sources; a repeated source denotes two components from that model, while Random + Random denotes two randomly initialized components with matched capacity.

Main-text name	Composition and setting	Cardinality	Role
Vision–language retrieval			
Frozen CLIP (baseline)	CLIP	1	Baseline
HeteroWeave-R1	CLIP + MAE	2	Cross-source
HeteroWeave-R2	CLIP + MAE + DINO	3	Cross-source
HeteroWeave-R3	CLIP + MAE + MAE	3	Same-source
Semantic segmentation			
ResNet-50 (baseline)	ResNet-50	1	Baseline
HeteroWeave-S1	ResNet-50 + Random + Random	3	Random control
HeteroWeave-S2	ResNet-50 + DINO + DINO	3	Same-source
HeteroWeave-S3	ResNet-50 + DINO + SwAV	3	Cross-source
HeteroWeave-S4	ResNet-50 + DINO	2	Lower cardinality
Object detection			
Faster R-CNN (baseline)	ResNet-50 backbone	1	Baseline
HeteroWeave-D1	ResNet-50 + BYOL; γ=0.001	2	Scale variant
HeteroWeave-D2	ResNet-50 + BYOL; γ=0.0025	2	Scale variant
HeteroWeave-D3	ResNet-50 + BYOL; γ=0.0025	2	Position comparison

Cross-task architectures. Tables 6 and 7 show that cardinality acts as a local structural decision rather than a global width multiplier. The ImageNet representatives increase cardinality at only one position, and the preferred position changes with the resource regime; the remaining positions preserve a single source. The cross-task architectures follow the same pattern. Retrieval and segmentation test whether an additional source supplies complementary information, whereas detection keeps cardinality fixed and varies how the added source is integrated. The outcomes explain why more components are not automatically preferable. HeteroWeave-S4 reaches 76.08 mIoU with half the added resources of HeteroWeave-S3, while a third retrieval component adds only 0.19 Mean

Recall over HeteroWeave-R1. Composition therefore becomes useful where source representations complement one another, not merely where the architecture can accommodate another component.

B.3 Additional Analyses

Table 8: Spearman correlations and ranking stability of CLAS. Top-1, parameter, and FLOP correlations measure association with the corresponding candidate property. Partial Top-1 correlation controls for parameter and FLOP ranks. Seed stability is the mean pairwise correlation between rankings obtained with initialization seeds 11, 17, and 23. Pool 1 contains 23 architectures and uses one-epoch Top-1 as its target; Pool 2 contains 25 architectures and uses five-epoch Top-1.

Pool	Top-1 correlation	Parameter correlation	FLOP correlation	Partial Top-1 correlation	Seed stability
Pool 1	**0.775**	-0.039	-0.464	0.745	0.949
Pool 2	**0.570**	0.051	-0.025	0.652	0.819

CLAS remains informative after accounting for resources. Table 8 separates predictive ordering from resource association and initialization noise. If CLAS mainly rewarded larger architectures, its ranking would track parameters or FLOPs and would weaken after those variables were controlled. Instead, parameter correlations remain near zero, and the partial Top-1 correlations remain strong at 0.745 and 0.652. The increase after conditioning in Pool 2 indicates that resource variation obscures part of the quality signal rather than generating it. The ranking also remains similar across three model initializations, with pairwise stability above 0.81 in both pools. Together, these tests show that CLAS is capturing activation structure that is predictive of subsequent optimization, rather than acting as an indirect measure of model size or a seed-specific activation magnitude.

Table 9: Matched-seed ImageNet-1K results under the shared 100-epoch protocol. Seeds 0, 11, and 23 form the result in Table 2; seed 37 is an additional evaluation. Aggregates use the sample standard deviation; accuracies are percentages and gains are percentage points.

Evaluation	Metric	DeRy	HeteroWeave	HeteroWeave gain (points)
Seed 0	Top-1	78.336	79.508	+1.172
	Top-5	94.356	94.726	+0.370
Seed 11	Top-1	78.370	79.384	+1.014
	Top-5	94.280	94.664	+0.384
Seed 23	Top-1	78.428	79.480	+1.052
	Top-5	94.272	94.614	+0.342
Additional seed 37	Top-1	78.242	**79.550**	**+1.308**
	Top-5	94.268	**94.770**	**+0.502**
Three-seed aggregate	Top-1	78.378 ± 0.047	$\mathbf{79.457 \pm 0.065}$	$+1.079 \pm 0.082$
	Top-5	94.303 ± 0.046	$\mathbf{94.668 \pm 0.056}$	$+0.365 \pm 0.021$
Four-seed aggregate	Top-1	78.344 ± 0.078	$\mathbf{79.481 \pm 0.070}$	$+1.137 \pm 0.133$
	Top-5	94.294 ± 0.041	$\mathbf{94.694 \pm 0.069}$	$+0.400 \pm 0.071$

Selection remains stable across seeds and random samples. The consistent matched-seed direction in Table 9 shows that the gain survives changes in initialization. This separates the advantage of the selected composition from seed-specific optimization noise. Table 10 further tests whether the result can be explained by favorable architecture sampling. Among independently random sampled architectures, the selected model outperforms both the mean and the strongest completed sample while using fewer resources. Three other samples diverged, showing that arbitrary composition can destabilize training rather than improve it. Because HeteroWeave was fixed beforehand, these comparisons test a prespecified search decision rather than a favorable outcome selected retrospectively.

Table 10: Results for the eight architectures obtained by independent random sampling and the selected HeteroWeave architecture. Resources are total model costs; accuracies are percentages; the mean uses the sample standard deviation.

Architecture	Parameters (M)	FLOPs (G)	Top-1	Top-5
Sample 1	18.508	3.828	78.406	94.150
Sample 2	18.038	3.299	77.742	93.812
Sample 3	27.941	3.900	77.542	93.850
Sample 4	24.752	3.885	78.900	94.454
Sample 5	13.133	5.661	79.126	94.432
Sample 6	14.708	4.309	76.842	93.740
Sample 7	27.327	3.965	78.688	94.392
Sample 8	23.055	5.671	79.038	94.606
Random samples mean	20.933 ± 5.643	4.315 ± 0.878	78.286 ± 0.825	94.180 ± 0.339
Best completed sample	13.133	5.661	79.126	94.432
Selected HeteroWeave	**18.258**	**3.945**	**79.508**	**94.726**

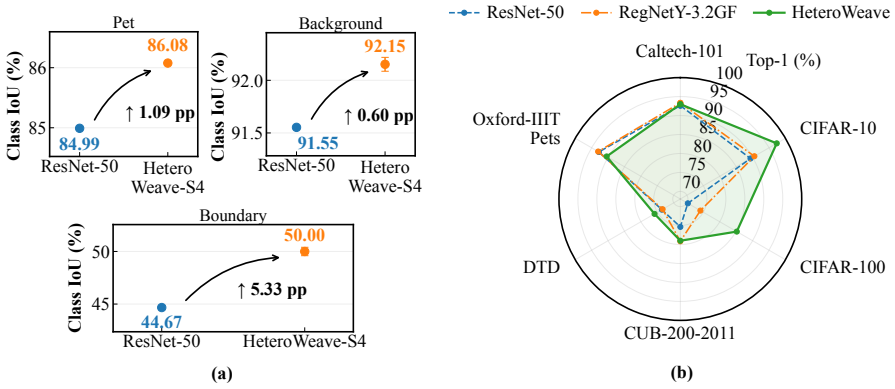

Figure 4: Task-level effects of pretrained block composition. (a) Class-wise IoU on Oxford-IIIT Pets for ResNet-50 and HeteroWeave-S4; arrows denote paired gains over three matched seeds. (b) Frozen-transfer Top-1 across six downstream datasets.

Table 11: Frozen-transfer Top-1 accuracy (%) under the shared 40,000-iteration protocol. HeteroWeave uses the ImageNet-selected composition in Table 6 and optimizes its task head and interfaces; complete-model references optimize only their task heads.

Dataset	ResNet-50	RegNetY-3.2GF	HeteroWeave
Caltech-101	92.62	**93.34**	92.98
CIFAR-10	89.36	90.54	**97.34**
CIFAR-100	70.28	74.11	**85.18**
CUB-200-2011	75.35	**79.15**	79.00
Describable Textures Dataset	73.62	73.40	**75.85**
Oxford-IIIT Pets	92.80	**93.02**	90.43
Mean	82.34	83.93	**86.80**

Task-level analyses localize where composition helps. Figure 4(a) reports mean class-wise IoU over three matched seeds, with error bars showing sample standard deviation. HeteroWeave-S4 improves boundary IoU by 5.33 points, substantially more than for pet interiors or background, suggesting that the additional pretrained source primarily benefits ambiguous spatial transitions. Figure 4(b) and Table 11 reveal a complementary cross-task pattern. The ImageNet-selected composition achieves the highest overall mean and leads on CIFAR-10, CIFAR-100, and textures, while complete-model references remain stronger on Caltech-101, CUB-200-2011, and Oxford-IIIT Pets. These results indicate that composition benefits are structured by the prediction problem rather than reflecting a task-independent increase in representation capacity.