applied sciences

MDPI

Article

Automatic Generation of Deep Learning Models Based on Heterogeneous Pre-Trained Model Stitching

Rongping Xie [1], Shanshan Wu [1], Lanqin Peng [2], Di Cui [3,*] and Yu Zhao [2,4,5,*]

[1] Nanjing Research Institute of Electronics Engineering, Nanjing 210007, China
[2] School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[3] School of Computer Science and Technology, Xidian University, Xi'an 710071, China
[4] Shenzhen Research Institute, Nanjing University of Aeronautics and Astronautics, Shenzhen 518109, China
[5] Key Laboratory of Safety-Critical Software Development and Verification, Ministry of Industry and Information Technology, Nanjing 210008, China
* Correspondence: cuidi@xidian.edu.cn (D.C.); zhao_yu@nuaa.edu.cn (Y.Z.)

Abstract

Deep learning models have achieved remarkable progress in computer vision tasks, but constructing efficient models still requires substantial expert experience and computational resources. Pre-trained model reuse provides a practical way to reduce model generation cost; however, existing methods still face challenges in feature alignment, structural integration, and resource-constrained search when stitching heterogeneous architectures such as convolutional neural networks (CNNs) and Vision Transformers (ViTs). To address these issues, this paper proposes Multi-Pretrained Model Stitching for Automatic Generation (MPMS-AG), an automatic generation framework for deep learning models based on heterogeneous pre-trained model stitching. MPMS-AG decomposes heterogeneous pre-trained models into reusable neural blocks and formulates model stitching as a sequential decision-making problem. Specifically, it uses hierarchical feature extraction and Radial Basis Function Centered Kernel Alignment (RBF-CKA) to quantify functional similarity between heterogeneous blocks, introduces adaptive block partitioning and hybrid clustering to reduce the search space, and adopts a Generalized Advantage Estimation (GAE)-based Actor–Critic strategy with a Single-Shot Network Pruning (SNIP)-based zero-shot proxy reward to search for stitching paths under parameter and floating point operations (FLOPs). Under the current CIFAR-10 experimental protocol, MPMS-AG generates trainable hybrid architectures with competitive classification performance across homogeneous stitching, heterogeneous cross-architecture stitching, and lightweight model stitching scenarios. The ablation results provide descriptive evidence that hybrid clustering and reinforcement learning-based search contribute to the observed performance. These findings support the feasibility of MPMS-AG for automatic generation of customized deep learning models from heterogeneous pre-trained model libraries within the evaluated setting.

Keywords: model stitching; pre-trained model reuse; heterogeneous architectures; RBF-CKA; reinforcement learning

Check for updates

Academic Editor: Christos Bouras

Received: 11 August 2026
Revised: 11 September 2026
Accepted: 14 September 2026
Published: 17 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Deep learning models have achieved remarkable progress in computer vision tasks, such as image classification and object detection. From classical convolutional neural

networks (CNNs), such as ResNet [1] and MobileNet, to Transformer-based visual architectures, such as Vision Transformer (ViT) and Swin Transformer, neural network structures have continuously evolved and significantly improved visual representation capability.

Deep learning has been increasingly applied to diverse engineering and intelligent-system applications, including data visualization and aerospace-oriented collaborative mission planning [2,3].

However, designing and deploying deep learning models still require substantial expert experience and computational resources [4]. In practical scenarios, developers often need to construct models that satisfy specific resource constraints, such as model parameters and floating point operations (FLOPs), while maintaining competitive accuracy. Existing pre-trained model families usually provide only a limited number of fixed architectures, making it difficult to flexibly obtain customized models under diverse deployment requirements.

To reduce the cost of model construction, reusing and recombining pre-trained models has become a promising direction [5]. With the rapid development of open-source model communities and model zoos, a large number of high-quality pre-trained models with different architectures and capacities have been accumulated [6]. Compared with training a new model from scratch, pre-trained model reuse can preserve the knowledge learned from large-scale datasets and significantly reduce computational overhead [7]. Existing studies have shown that pre-trained models can be stitched [8], transferred [9], or modularly recombined to construct new architectures. However, most existing stitching methods mainly focus on models from the same architecture family, where feature distributions and structural patterns are relatively consistent. When handling heterogeneous architectures, such as CNNs and ViTs, these methods often suffer from feature-space mismatch, dimension inconsistency, and semantic misalignment. In addition, directly searching over all layers or modules in a large pre-trained model library may lead to a huge search space, making it difficult to efficiently identify high-quality stitching paths.

To address these challenges, this paper proposes Multi-Pretrained Model Stitching for Automatic Generation (MPMS-AG), a framework for automatic generation of deep learning models based on heterogeneous pre-trained model stitching. Different from conventional methods that treat pre-trained models as indivisible units, MPMS-AG decomposes heterogeneous models into reusable neural blocks and formulates model stitching as a sequential decision-making problem. Specifically, MPMS-AG uses the hook mechanism to intercept intermediate model outputs and applies Radial Basis Function Centered Kernel Alignment (RBF-CKA) to quantify functional similarity between heterogeneous blocks. Then, adaptive block partitioning and a hybrid clustering algorithm integrating Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and K-Means are designed to decompose heterogeneous pre-trained models into functionally similar module clusters, thereby compressing the search space. Finally, MPMS-AG introduces an Actor–Critic reinforcement learning algorithm based on Generalized Advantage Estimation (GAE), together with a Single-Shot Network Pruning (SNIP)-based zero-shot proxy reward [10], to search for stitching paths under resource constraints on parameter count and FLOPs.

The key idea of MPMS-AG is to transform heterogeneous pre-trained model reuse into a structured and resource-aware search problem. By jointly considering feature alignment, adaptive model decomposition, hybrid clustering, and reinforcement learning-based sequential decision-making, MPMS-AG can automatically generate trainable hybrid architectures while reducing invalid sampling and expensive trial-and-error. In this way, the proposed framework provides a feasible approach for constructing customized deep learning models from heterogeneous pre-trained model libraries.

To evaluate MPMS-AG under the current experimental setting, we conduct this study around the following three research questions (RQs):

RQ1: How does MPMS-AG perform in the homogeneous convolutional network stitching scenario? This research question evaluates whether MPMS-AG can effectively stitch CNN-based models with homogeneous convolutional structures and achieve a better trade-off between classification accuracy and computational complexity.

RQ2: How does MPMS-AG perform in the heterogeneous cross-architecture stitching scenario? This research question investigates whether MPMS-AG can handle cross-architecture stitching between CNN and ViT models, especially under feature dimension mismatch, structural inconsistency, and semantic gaps.

RQ3: Can MPMS-AG generate efficient lightweight models under resource constraints? This research question evaluates the applicability of MPMS-AG in lightweight deployment scenarios by using MobileNetV3 series models as candidate pre-trained models.

In summary, the main contributions of this paper are as follows:

- We propose MPMS-AG, an automatic generation framework for deep learning models based on heterogeneous pre-trained model stitching. The framework models heterogeneous model stitching as a sequential decision-making problem and supports model generation under parameter and FLOPs constraints.
- We introduce a hierarchical feature extraction and RBF-CKA-based alignment mechanism to quantify functional similarity between heterogeneous neural blocks, thereby alleviating representation-space differences between CNN and Transformer architectures.
- We design an adaptive block partitioning strategy and a hybrid clustering algorithm integrating DBSCAN and K-Means to decompose heterogeneous pre-trained models into functionally coherent block clusters, effectively reducing the search-space dimension.
- We develop a GAE-based Actor–Critic reinforcement learning search strategy with a SNIP-based zero-shot proxy reward to search for stitching paths under resource constraints and explore the accuracy–efficiency trade-off of generated models.
- We conduct systematic experiments on homogeneous convolutional network stitching, heterogeneous cross-architecture stitching, and lightweight model stitching scenarios. The results show that MPMS-AG can generate trainable stitched models with competitive reported performance within the evaluated settings.

Paper Organization. The remainder of this paper is organized as follows. Section 2 discusses the related work on pre-trained model reuse and model stitching. Section 3 presents the details of the proposed MPMS-AG framework. Section 4 introduces the experimental settings. Section 5 reports the experimental results and analysis. Section 6 discusses the effectiveness of key components in MPMS-AG. Finally, Section 7 concludes this paper.

2. Related Work

This section introduces related work on pre-trained model recombination and model stitching. Pre-trained model recombination aims to break the limitations of traditional model compression and neural architecture search by stitching, transferring, and reusing components, layers, or functional modules of pre-trained models [9–11]. With the rapid development of model zoos and open-source model communities, a large number of pre-trained models with different architectures and capacities have been accumulated, making modular model reuse a promising direction for efficient model generation [5,12,13]. According to whether the stitched models share the same architecture family, existing methods can be divided into two categories: homogeneous model stitching and cross-architecture model stitching.

2.1. Homogeneous Model Stitching

Homogeneous model stitching mainly recombines components within the same model family or among models with similar structures. These methods usually rely on representation similarity to bridge models with different scales or training variants, and aim to generate efficient interpolation paths under different resource constraints [14,15].

A large number of studies have explored homogeneous model stitching. Pan et al. proposed Stitchable Neural Networks (SN-Net), which introduces a many-to-many model recombination paradigm [8]. By inserting stitching layers within a pre-trained model family, such as DeiT [16], SN-Net enables efficient performance interpolation among models with different scales and supports flexible accuracy–efficiency trade-offs. Pan et al. further proposed SN-Netv2, which introduces bidirectional stitching and resource-constrained sampling for ViT-based downstream adaptation [17]. Although these methods demonstrate the effectiveness of reusing pre-trained model families, their stitching spaces are mainly constructed from structurally compatible models or model families. Consequently, directly extending this paradigm to heterogeneous architectures such as CNNs and ViTs requires additional mechanisms for cross-architecture representation alignment and candidate-space construction.

Several studies further focus on efficiency optimization and parameter sharing. He et al. proposed the Efficient Stitchable Task Adaptation (ESTA) framework [18], which combines parameter-efficient fine-tuning with model stitching and enables low-rank update sharing across models. However, its performance gain relies on high representation compatibility between stitched models, making it difficult to directly extend to heterogeneous architectures. Anakewat et al. proposed Self-Stitching, showing that stitching can work in a simple setting by introducing convolutional stitching layers into a single feature extractor [19]. Nevertheless, since this method performs stitching only within a single feature extractor, its expressive ability and performance improvement are limited in more complex tasks.

Representation alignment is also an important factor affecting stitching quality. Athanasiadis et al. proposed FuLA, which aligns homogeneous models by minimizing the L2 distance between the latent functional representations of subsequent layers [20]. This method provides a more stable representation bridging strategy for homogeneous model stitching. However, FuLA is mainly designed for models with similar architectures, such as ResNet variants, and its applicability to heterogeneous architectures remains limited.

2.2. Cross-Architecture Model Stitching

Cross-architecture model stitching aims to break the structural barriers among different model families, such as CNNs and Transformers, and construct customized networks by reusing heterogeneous pre-trained components. Compared with homogeneous model stitching, cross-architecture stitching faces more severe challenges, including feature-space mismatch, dimension inconsistency, and semantic misalignment. Therefore, the core difficulty lies in designing effective adapters and reliable representation alignment mechanisms for heterogeneous components.

In terms of heterogeneous component mining and recombination, Yang et al. proposed Deep Model Reassembly (DeRy), which establishes a general model reuse paradigm [21]. DeRy decomposes pre-trained models from different sources into functionally independent neural blocks through set-cover optimization. To solve the compatibility problem among heterogeneous blocks, it uses representation similarity metrics such as centered kernel alignment (CKA) to construct functionally equivalent sets, so that components from different architectures can be regarded as interchangeable. Furthermore, DeRy formulates network customization as an integer programming problem and introduces a NASWOT-based

training-free proxy metric to search for heterogeneous combinations satisfying parameter and FLOPs constraints.

DeRy is closely related to MPMS-AG because both methods reuse heterogeneous pre-trained blocks and perform resource-constrained architecture search. However, DeRy constructs functionally equivalent block sets and uses integer programming with a NASWOT-based proxy, whereas MPMS-AG combines adaptive block partitioning and hybrid clustering for search-space compression and formulates stitching-path construction as a sequential decision-making problem solved by reinforcement learning. Thus, MPMS-AG differs from DeRy mainly in adaptive search-space construction and sequential resource-constrained stitching-path search.

To address cross-architecture feature incompatibility, Traft and Cheney investigated model stitching between disparate neural network layers using complex adapters and spatial rescaling [22]. In our experiments, we denote this complex-adapter-based method as CS-Adapter. It is selected as a representative baseline because it directly addresses feature adaptation between heterogeneous architectures. In contrast, MPMS-AG additionally constructs and searches stitching paths under explicit resource constraints.

Another line of research focuses on unit matching and parameter fusion in cross-architecture model merging. Xu et al. proposed MuDSC, a merging framework based on dual-space constraints [23]. Although this method mainly targets multi-task model merging, its core idea is useful for addressing structural differences in heterogeneous architectures. By combining similarity matrices from both activation space and weight space, MuDSC searches for optimal permutations under dual constraints and designs matching algorithms suitable for grouped structures, such as multi-head attention in ViTs or GroupNorm.

In summary, existing model stitching methods address different aspects of pre-trained model reuse, but important limitations remain. Homogeneous methods such as SN-Net and FuLA generally rely on relatively compatible architectures or representation spaces, which may limit their applicability to heterogeneous model pools. Cross-architecture approaches improve model reuse flexibility: DeRy constructs functionally equivalent block sets for resource-constrained architecture optimization, CS-Adapter focuses on cross-architecture feature adaptation, and MuDSC performs structural alignment through dual-space matching. However, jointly addressing heterogeneous representation alignment, search-space reduction, and automatic stitching-path search under explicit resource constraints remains challenging.

MPMS-AG addresses these challenges in a unified framework by combining RBF-CKA-based representation analysis, adaptive block partitioning and hybrid clustering for search-space reduction, and resource-aware sequential stitching-path search based on a GAE-based Actor–Critic strategy. Thus, MPMS-AG differs from existing approaches mainly in the joint integration of heterogeneous representation analysis, adaptive search-space construction, and resource-constrained sequential architecture search.

3. Automatic Generation of Deep Learning Models Based on Heterogeneous Pre-Trained Model Stitching

As shown in Figure 1, MPMS-AG is designed to automatically generate hybrid neural architectures by stitching heterogeneous pre-trained models under resource constraints. Compared with existing stitching methods that mainly rely on fixed adapters or predefined equivalent sets, MPMS-AG jointly performs semantic block construction, adaptive clustering, and resource-aware sequential search. The framework integrates structure-aware representation, clustering-based search-space reduction, and reinforcement learning-based

sequential decision-making. It consists of four main modules: hierarchical feature extraction, adaptive block partitioning, hybrid clustering, and adaptive model stitching.

In the hierarchical feature extraction module, MPMS-AG extracts intermediate feature maps from pre-trained models and uses Radial Basis Function Centered Kernel Alignment (RBF-CKA) to quantify functional similarity between heterogeneous layers. Based on the similarity matrix, the adaptive block partitioning module divides each network into functionally coherent blocks by preserving intra-block consistency and maximizing inter-block semantic differences. The hybrid clustering module then groups functionally similar blocks into candidate clusters, thereby reducing the search space from the layer level to the block-cluster level. Finally, the adaptive model stitching module formulates hybrid architecture construction as a sequential decision-making problem, where a reinforcement learning agent selects one block from each ordered cluster and connects heterogeneous blocks through adaptive stitching adapters.

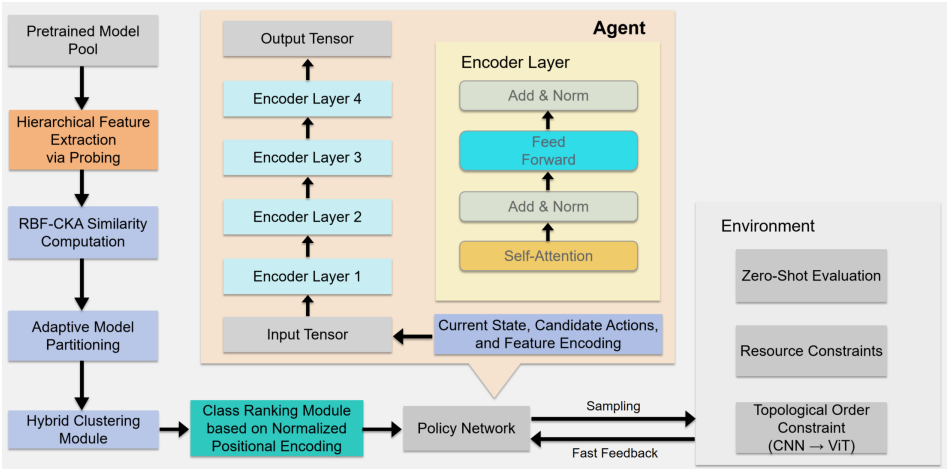

Figure 1. The framework of the proposed Multi-Pretrained Model Stitching for Automatic Generation (MPMS-AG). Abbreviations: RBF-CKA, Radial Basis Function Centered Kernel Alignment; CNN, convolutional neural network; ViT, Vision Transformer.

3.1. Hierarchical Feature Extraction

Accurately measuring the semantic compatibility between heterogeneous neural networks is challenging because CNNs and Transformers generate feature maps with different spatial dimensions and channel definitions. Directly comparing model weights cannot reflect the actual response of neural modules to input data. Therefore, MPMS-AG adopts a hook-based hierarchical feature extraction strategy to obtain unified feature representations from intermediate layers.

For CNN-based models, the extracted feature map has the shape of $B \times C \times H \times W$, where B denotes the batch size, C denotes the number of channels, and H and W denote the spatial dimensions. Adaptive average pooling is used to compress the spatial dimensions into 1×1, and the result is flattened into a channel-level vector. For Transformer-based architectures, the sequence dimension is pooled to obtain a unified representation with the shape of $B \times C$. The vectorization process is defined as follows:

$$v_l = \text{Flatten}(\text{AdaptivePool}(f_l(X))) \in \mathbb{R}^{B \times C_l}, \quad (1)$$

where $f_l(X)$ denotes the output feature of input X at the l-th layer, and C_l denotes the channel dimension of this layer.

The pooling operation removes fine-grained spatial information, but its purpose is to obtain unified sample-level representations for RBF-CKA-based functional similarity measurement rather than pixel-level spatial matching. Spatial-resolution compatibility

between stitched blocks is handled separately by the adaptive stitching adapter described in Section 4.1. Thus, representation similarity and structural compatibility are treated separately in MPMS-AG.

For the CIFAR-10 experiments, the official training set of 50,000 images is further divided into a training subset and a validation subset at a ratio of 8:2, resulting in 40,000 training images and 10,000 validation images. RBF-CKA representation extraction is performed on the validation subset with a batch size of 128 and without shuffling. The images are resized to 224×224 and normalized using the CIFAR-10 statistics (mean $= [125.307, 122.961, 113.8575]$, std $= [51.5865, 50.847, 51.255]$), without random augmentation.

The validation subset is used to compute the RBF-CKA similarity matrix, which guides adaptive block partitioning and hybrid clustering. The resulting candidate clusters define the search space for reinforcement-learning-based architecture search. The official CIFAR-10 test set is kept independent of this process and is not used for representation extraction, RBF-CKA computation, block partitioning, clustering, or search-space construction.

Hooks are placed at the outputs of reusable blocks. ResNet18 uses `layer1.0-layer1.1`, `layer2.0-layer2.1`, `layer3.0-layer3.1`, and `layer4.0-layer4.1`, while ResNet34/50 use `layer1.0-layer1.2`, `layer2.0-layer2.3`, `layer3.0-layer3.5`, and `layer4.0-layer4.2`. ViT-Base uses `blocks.0-blocks.11`. Swin-Tiny uses all blocks in Stages 0–3 (2, 2, 6, and 2 blocks) together with the downsampling modules of Stages 0–2, while Swin-Base follows the same rule with 2, 2, 18, and 2 blocks. MobileNetV3 hooks are likewise placed at all reusable block outputs defined by the corresponding source architecture.

After obtaining unified feature vectors, MPMS-AG uses RBF-CKA to compute the semantic similarity between heterogeneous layers. Given two feature matrices $X \in \mathbb{R}^{B \times C_1}$ and $Y \in \mathbb{R}^{B \times C_2}$, the RBF Gram matrix and CKA similarity are defined as:

$$K_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), \quad \sigma^2 = \text{median}(\|x_i - x_j\|^2), \tag{2}$$

$$S(X, Y) = \text{CKA}(K, L) = \frac{\text{HSIC}(K, L)}{\sqrt{\text{HSIC}(K, K) \cdot \text{HSIC}(L, L)}}. \tag{3}$$

where K and L are the RBF Gram matrices of X and Y, respectively, and $\text{HSIC}(\cdot)$ denotes the Hilbert–Schmidt Independence Criterion. In this way, MPMS-AG obtains a unified semantic similarity matrix for subsequent block partitioning and clustering. The feature extraction process of ResNet18 is illustrated in Figure 2.

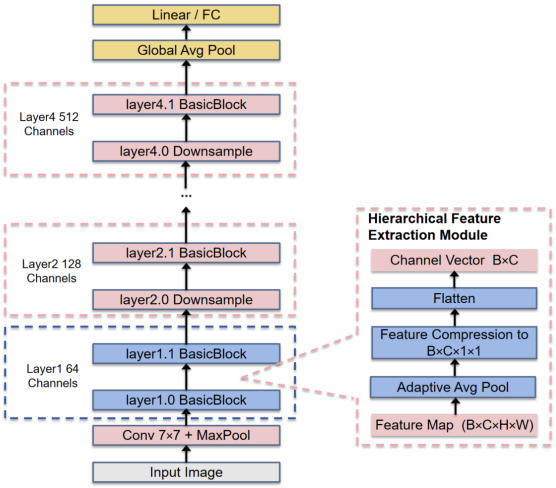

Figure 2. An illustration of feature extraction in ResNet18.

3.2. Adaptive Block Partitioning

After hierarchical feature extraction, MPMS-AG partitions each pre-trained model into reusable semantic blocks. If each layer is used as an independent search unit, the search space becomes excessively large and adjacent feature dependencies may be broken. In contrast, overly coarse partitioning reduces the flexibility of model recombination. Therefore, MPMS-AG determines block boundaries according to semantic differences between adjacent layers.

Suppose a pre-trained model contains N layers, denoted as $L = \{l_1, l_2, \ldots, l_N\}$. Based on the inter-layer similarity matrix S, the partition weight between two adjacent layers is defined as:

$$w_i = 1 - S(l_i, l_{i+1}), \quad i \in \{1, \ldots, N-1\}. \tag{4}$$

A larger w_i indicates a stronger semantic change between adjacent layers and a higher probability of being a suitable partition boundary.

To determine the optimal partitioning scheme $P = \{p_1, p_2, \ldots, p_{K-1}\}$, MPMS-AG maximizes the semantic differences at selected partition points while avoiding excessive fragmentation:

$$J(P, K) = \sum_{p \in P} w_p - \lambda \cdot \Psi(K), \tag{5}$$

where the normalized block-count penalty is defined as

$$\Psi(K) = \frac{K - K_{\min}}{\max(1, K_{\max} - K_{\min})},$$

with $K_{\min} = 2$, $K_{\max} = \lfloor N/2 \rfloor$, and $\lambda = 0.2$. MPMS-AG determines the final block number by searching $K \in [K_{\min}, K_{\max}]$. For each K, the $K-1$ largest partition weights define the initial partition set $P_{\text{init}}(K)$, and the solution maximizing $J(P_{\text{init}}(K), K)$ is selected before local boundary refinement:

$$b_{\text{new}} = \arg \max_{j \in \{b-1, b, b+1\}} (w_j). \tag{6}$$

Through this process, each pre-trained model is decomposed into functionally coherent blocks, which reduces the search space while preserving the local representation capability of the original model.

3.3. Hybrid Clustering

After adaptive block partitioning, heterogeneous models are transformed into a set of functional blocks $B = \{b_1, b_2, \ldots, b_M\}$. To further reduce the search space, MPMS-AG groups functionally similar blocks into candidate clusters. Since block distributions in the feature space may be spherical, irregular, or noisy, a single clustering algorithm may not be robust in all scenarios. Therefore, MPMS-AG combines DBSCAN and K-Means in a hybrid clustering strategy.

The input of the clustering module is the block-level similarity matrix $S \in [0, 1]^{M \times M}$. MPMS-AG first converts it into a distance matrix:

$$D_{ij} = 1 - S(b_i, b_j). \tag{7}$$

The distance matrix $\mathbf{D}$ is used for the DBSCAN branch, whereas K-Means does not directly take $\mathbf{D}$ as a precomputed pairwise-distance input. Instead, each block b_i is represented by the i-th row of the min–max-normalized and symmetrized similarity matrix, $\mathbf{x}_i = \widetilde{\mathbf{S}}_{i,:} \in \mathbb{R}^M$, which is further normalized to unit ℓ_2 norm before standard K-Means with Euclidean distance is applied.

DBSCAN is used to capture non-spherical structures and noise, while K-Means is used to identify compact clusters under different candidate values of K. For K-Means, MPMS-AG evaluates each candidate clustering result by considering compactness, separation, and stability:

$$J_{KM}(K) = \alpha \cdot \mathrm{Sil} + \beta \cdot \mathrm{Stab} + \gamma \cdot \mathrm{Inter}_{\mathrm{norm}} + \delta \cdot \mathrm{Intra}_{\mathrm{norm}}^{-1}, \tag{8}$$

where Sil denotes the silhouette coefficient, Stab denotes clustering stability measured by the Adjusted Rand Index (ARI), and $\mathrm{Inter}_{\mathrm{norm}}$ and $\mathrm{Intra}_{\mathrm{norm}}$ denote normalized inter-cluster and intra-cluster distances.

For DBSCAN, ε is searched over 20 uniformly spaced values between the 5th and 95th percentiles of the off-diagonal entries of $\mathbf{D}$, with $min\_samples \in \{2, 3, 4\}$. The target number of non-noise clusters ranges from 8 to $\lceil \sqrt{M} \rceil$, and candidate solutions are ranked by cluster-count deviation, noise ratio, and $Q(C)$.

For K-Means, K is searched from $|\Omega_{\mathrm{source}}| + 1$ to $\lceil \sqrt{M} \rceil$. For each K, three K-Means++ runs with seeds 0, 1, and 2 are performed using $max\_iter = 200$ and $tol = 10^{-4}$. In Equation (8), $\alpha = 0.4$, $\beta = 0.3$, $\gamma = 0.2$, and $\delta = 0.1$, and the stability term is computed by the mean pairwise ARI. The DBSCAN and K-Means results are finally compared using $Q(C)$, and the better solution is selected.

To select the final clustering scheme, MPMS-AG compares the DBSCAN and K-Means results using a unified quality function:

$$Q(C) = \mathrm{Sim}_{\mathrm{intra}} - \mathrm{Sim}_{\mathrm{inter}}, \tag{9}$$

where $\mathrm{Sim}_{\mathrm{intra}}$ and $\mathrm{Sim}_{\mathrm{inter}}$ represent the average intra-cluster and inter-cluster similarities, respectively. The clustering result with the higher $Q(C)$ is selected as the final candidate block cluster set.

The overall hybrid clustering workflow is shown in Figure 3.

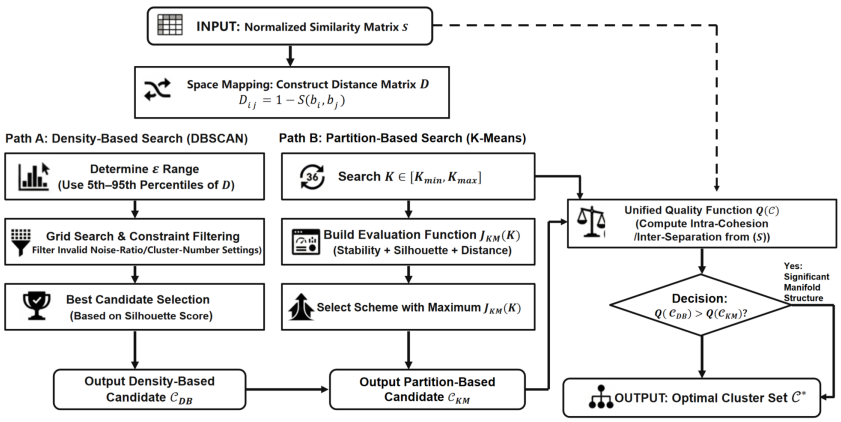

Figure 3. The workflow of the hybrid clustering module.

3.4. Normalized Relative Position Encoding

The candidate clusters generated by hybrid clustering need to follow a valid topological order from shallow feature extraction to deep semantic reasoning. However, heterogeneous source models usually have different depths, making absolute layer indices unsuitable for ordering. MPMS-AG therefore introduces normalized relative position encoding to map blocks from different architectures into a unified interval.

For a block b from model M, let $idx(b)$ denote its absolute index and N_M denote the total number of blocks in M. The normalized position is defined as:

$$P_{\text{norm}}(b) = \frac{idx(b)}{N_M} \in (0, 1]. \tag{10}$$

For each cluster G_i, its ranking score is computed by averaging the normalized positions of its blocks:

$$R(G_i) = \frac{1}{|G_i|} \sum_{b \in G_i} P_{\text{norm}}(b). \tag{11}$$

The cluster set is then sorted in ascending order of $R(G_i)$, forming an ordered cluster sequence $G_{\text{sorted}} = (G_1, G_2, \ldots, G_K)$. This sequence serves as the decision space of the reinforcement learning agent and reduces invalid architecture sampling.

3.5. Adaptive Model Stitching and Automatic Construction

After clustering and topological ordering, MPMS-AG formulates hybrid model construction as a sequential decision-making problem over the ordered cluster sequence G_{sorted}. At each step, the agent selects one candidate block from the current cluster and gradually constructs a complete hybrid backbone network. This process is modeled as a Markov Decision Process. The state s_t consists of the selected block sequence before step t, and the action space corresponds to the candidate blocks in cluster G_t.

To encode the current state and candidate blocks, MPMS-AG uses a Transformer encoder with a shared Actor–Critic architecture. The Actor $\pi_\theta(a_t|s_t)$ predicts the action distribution, and the Critic $V_\phi(s_t)$ estimates the expected future reward. To stabilize training, Generalized Advantage Estimation (GAE) is used:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \quad \hat{A}_t = \sum_{l=0}^{K-t-1} (\gamma\lambda)^l \delta_{t+l}, \tag{12}$$

where δ_t denotes the temporal-difference error, $\hat{A}_t$ denotes the advantage estimate, and γ and λ are the discount factor and GAE smoothing factor, respectively.

To enforce topological and resource constraints during action selection, MPMS-AG introduces a dynamic mask. For candidate block $b_{t,j}$ in cluster G_t, the mask value is defined as:

$$M_{t,j} = \begin{cases} 0, & \text{Type}(b_{t,j}) \text{ is valid} \wedge P(S_{t-1} \cup \{b_{t,j}\}) \leq P_{\max}, \\ -\infty, & \text{otherwise.} \end{cases} \tag{13}$$

During action selection, each candidate block is temporarily appended to the current partial architecture and checked for topology validity and resource usage. When both estimates are available, only candidates satisfying $P \leq P_{\max}$ and $F \leq F_{\max}$ are retained; otherwise, they are masked before sampling. If FLOPs cannot be reliably estimated, the current implementation falls back to parameter-based filtering.

Since fully training each candidate architecture is computationally expensive, MPMS-AG uses the Single-Shot Network Pruning (SNIP) metric as a zero-shot proxy reward. For a generated architecture M, the SNIP score is computed as:

$$\text{Score}_{SNIP}(M) = \sum_{w \in W} \left| \frac{\partial \mathcal{L}_{ce}(D_{\text{mini}}; W)}{\partial w} \odot w \right|. \tag{14}$$

The final reward combines the SNIP score with parameter and FLOPs constraints:

$$R = \frac{\text{Score}_{SNIP}(M)}{\eta} \times \sqrt{\left(\min\left(1, \max\left(\frac{P(M)}{P_{\max}}, \frac{F(M)}{F_{\max}}\right)\right)\right)^{-1}}. \tag{15}$$

where η is a normalization coefficient, and $P_{\max}$ and $F_{\max}$ denote the parameter and FLOPs budgets. Through semantic block construction, topology-aware ordering, dynamic masking, and zero-shot reward estimation, MPMS-AG efficiently searches for high-quality hybrid architectures without repeatedly training every candidate model from scratch.

4. Experimental Setting

This section describes the experimental settings used to evaluate MPMS-AG.

Specifically, the experimental setting consists of three parts: heterogeneous stitching and adaptive adapters, source checkpoints and training configurations, and implementation details and environment configuration. The first part describes the interface adaptation between heterogeneous blocks, the second reports the source checkpoints, corresponding training datasets, and source-model training configurations, and the third describes the dataset usage, fine-tuning budget, implementation framework, and hardware environment.

4.1. Heterogeneous Stitching and Adaptive Adapters

After the block sequence $\{b_1, \ldots, b_K\}$ is determined according to the model construction process, the interface incompatibility between adjacent blocks must be resolved. Let the output feature of the predecessor block b_i be O_i, whose dimension is $B \times C_{out} \times H_i \times W_i$, and let the expected input feature of the successor block b_{i+1} be I_{i+1}. The adapter mapping function $\Phi(\cdot)$ is performed in two steps.

1. Spatial resolution alignment.

In the implementation, the CNN-to-CNN adapter uses a discrete stride-selection rule based on the resolution ratio $r = H_i / H_{i+1}$. The convolution stride is determined as

$$s_i = \begin{cases} 2, & r \geq 2, \\ 1, & r < 2. \end{cases} \tag{16}$$

Accordingly, cases with $1 < r < 2$, $r < 1$, or other non-integer ratios do not result in non-integer convolution strides. A stride of 1 is used when $r < 2$, whereas a stride of 2 is used when $r \geq 2$. The current CNN-to-CNN adapter does not explicitly support arbitrary upsampling or independent height/width scaling. For Swin-based successors, when the target spatial grid is available, bilinear interpolation is additionally applied to match the required spatial resolution.

Therefore, the current CNN-to-CNN path does not guarantee exact spatial matching for arbitrary non-integer resolution ratios.

2. Feature dimension and structural remapping. After spatial alignment, the adapter performs architecture-specific channel projection and data structure transformation according to the architecture type of the successor block b_{i+1}, namely CNN or ViT. This process mainly includes the following three cases.

(1) CNN → CNN: homogeneous channel transformation. When two convolutional modules are connected, the main difference lies in the number of channels. The adapter uses a 1×1 convolutional layer to map the channel dimension from C_{out} to C_{next}, while keeping the 4D tensor structure unchanged:

$$I_{i+1} = \text{Conv}_{1\times1}(\tilde{O}_i) \in \mathbb{R}^{B \times C_{next} \times H' \times W'}. \tag{17}$$

(2) CNN → ViT: heterogeneous modality transformation. When convolutional features are fed into a Transformer module, the adapter needs to transform the 4D feature map into a 3D sequence. Specifically, the adapter first uses a 1×1 convolution to adjust the channel dimension to the embedding dimension D of ViT, and then flattens the spatial dimensions (H, W) into a sequence length L:

$$I_{i+1} = \text{Flatten}(\text{Conv}_{1\times1}(\widetilde{O}_i)) \in \mathbb{R}^{B\times L\times D}. \tag{18}$$

This operation realizes the semantic transition from local spatial features to global sequential features.

(3) ViT → ViT: homogeneous linear projection. When two Transformer modules are connected, both inputs and outputs are represented as 3D sequences. If their embedding dimensions are inconsistent, the adapter applies a fully connected layer, namely a linear layer, to each token in the sequence:

$$I_{i+1} = \widetilde{O}_i \cdot W_{proj} + b_{bias} \in \mathbb{R}^{B\times L\times D_{next}}. \tag{19}$$

The adaptive adapter module connects the block sequence selected by the search policy to an executable stitched network. Through channel projection, structural remapping, and the supported spatial-alignment operations described above, the adapter resolves interface incompatibilities between adjacent blocks within the current implementation scope. Consequently, block sequences satisfying these compatibility conditions can be instantiated as differentiable and executable computational graphs. The current adapter does not provide universal support for arbitrary spatial transformations, and its remaining limitations are discussed in Section 6.4.

4.2. Source Checkpoints and Training Information

MPMS-AG reuses fixed source checkpoints for representation extraction, search-space construction, and model stitching. To improve reproducibility, we explicitly report the source-model training configurations used to obtain these checkpoints, including the optimizer, initial learning rate, learning-rate scheduler, weight decay, batch size, training budget, and relevant optimizer-specific settings. The corresponding checkpoint references and training configurations are summarized in Table 1.

The source models include ResNet18, ResNet34, ResNet50, MobileNetV3-Large-1.0, MobileNetV3-Large-0.75, MobileNetV3-Small-1.0, ViT-Base, Swin-Tiny, and Swin-Base. The checkpoint files listed in Table 1 correspond to the CIFAR-10 source weights used in the MPMS-AG experiments. The source-model training configurations are reported separately from the post-construction fine-tuning configuration of the generated stitched architectures described in Section 4.3.

Table 1. Source checkpoints and training configurations used in the MPMS-AG experiments.

Source Model	Checkpoint	Optimizer	LR	Scheduler	WD	Batch/Ep.
ResNet18	`resnet18_cifar10.pth`	SGD ($m = 0.9$)	0.1	Multi-step	1×10^{-4}	128/200
ResNet34	`resnet34_cifar10.pth`	SGD ($m = 0.9$)	0.1	Multi-step	1×10^{-4}	128/200
ResNet50	`resnet50_cifar10.pth`	SGD ($m = 0.9$)	0.1	Multi-step	1×10^{-4}	128/200
MobileNetV3-Large-1.0	`mobilenetv3_large_100_cifar10.pth`	Adam	0.005	Cosine	1×10^{-5}	128/50
MobileNetV3-Large-0.75	`mobilenetv3_large_075_cifar10.pth`	Adam	0.005	Cosine	1×10^{-5}	128/50
MobileNetV3-Small-1.0	`mobilenetv3_small_100_cifar10.pth`	Adam	0.005	Cosine	1×10^{-5}	128/20
ViT-Base	`vit_base_patch16_224_cifar10.pth`	AdamW	5×10^{-4}	Cosine + WU	0.05	128/100
Swin-Tiny	`swin_tiny_cifar10.pth`	AdamW	5×10^{-4}	Cosine + WU	0.05	128/100
Swin-Base	`swin_base_cifar10.pth`	AdamW	5×10^{-4}	Cosine + WU	0.05	128/100

Note: All source checkpoints were trained on CIFAR-10. Adam and AdamW use $\beta_1 = 0.9$ and $\beta_2 = 0.999$. WU denotes a five-epoch linear warm-up for the Transformer-based source models. Abbreviations: LR, learning rate; WD, weight decay; Ep., epochs; SGD, stochastic gradient descent; ViT, Vision Transformer.

4.3. Implementation Details and Environment Configuration

All experiments are conducted on the CIFAR-10 dataset. CIFAR-10 contains 60,000 color images with a resolution of 32×32, including an official training set of 50,000 images and an official test set of 10,000 images. For the MPMS-AG experiments, the official training set is further divided into a training subset and a validation subset at a ratio of 8:2, resulting in 40,000 training images and 10,000 validation images. The official 10,000-image test set is kept completely independent and is not used during representation analysis, architecture search, architecture selection, parameter optimization, or checkpoint selection.

The three subsets serve distinct purposes throughout the MPMS-AG pipeline. RBF-CKA representation extraction is performed on the validation subset, and the resulting similarity matrix is used for adaptive block partitioning and hybrid clustering, thereby defining the candidate search space. The SNIP zero-shot proxy reward is computed using mini-batches sampled from the training subset, and the Actor–Critic reinforcement-learning agent is optimized according to these training-derived rewards. After the architecture-search process produces candidate stitched architectures, the validation subset is used to select the final architecture. The selected architecture is subsequently fine-tuned using the training subset, while validation performance is used for checkpoint selection. Only after both the final architecture and its checkpoint have been fixed is the official CIFAR-10 test set used to compute the final reported Top-1 accuracy. Table 2 summarizes the stage-wise data usage.

Table 2. Data usage at the main stages of the MPMS-AG pipeline.

Stage	Data	Role
RBF-CKA features	Validation subset (10,000)	Cross-model representation similarity
Partitioning & clustering	Validation-derived RBF-CKA	Candidate clusters and search-space construction
SNIP reward	Training subset (40,000)	Zero-shot proxy evaluation
RL agent training	Training subset (40,000)	Actor–Critic policy and value updates through SNIP rewards
Final architecture selection	Validation subset (10,000)	Selection of the final stitched architecture
Fine-tuning	Training subset (40,000)	Gradient-based parameter updates
Checkpoint selection	Validation subset (10,000)	Selection of the best fine-tuned checkpoint
Final Top-1 evaluation	Official test set (10,000)	Independent final performance evaluation

After model construction, the generated stitched architectures are fine-tuned for 50 epochs using the training subset under the reported experimental protocol. This post-construction fine-tuning budget is independent of the source-model training budgets reported in Table 1.

In terms of the experimental environment, the algorithm is implemented based on the PyTorch deep learning framework (version 1.10), and the OpenMMLab MMClassification toolkit (version 0.18.0) is used for modular management and data augmentation. All experiments, including inference evaluation during the architecture search process and final lightweight fine-tuning, are conducted on an NVIDIA GeForce RTX 4060 GPU. For each generated architecture, we report the Top-1 accuracy after lightweight fine-tuning under the same training protocol.

The architecture search was performed on the same single NVIDIA GeForce RTX 4060 GPU. Since the original runs did not separately log the wall-clock time, GPU-hours, or the number of evaluated candidate architectures for the search stage, these search-specific statistics cannot be reliably reconstructed retrospectively.

The source-checkpoint references and corresponding source-model training configurations are summarized in Table 1. These checkpoint weights provide the reusable source blocks used by MPMS-AG during representation extraction, search-space construction, and model stitching.

5. Experimental Results and Analysis

In this section, we evaluate MPMS-AG through the following three research questions (RQs). These RQs cover three experimental perspectives: homogeneous convolutional network stitching, heterogeneous cross-architecture stitching, and lightweight deep learning model stitching. Below, we elaborate on the approach, results, and observations of these three RQs to facilitate result interpretation and analysis.

The Top-1 results reported below are evaluated on the official CIFAR-10 test set under the data-usage protocol described in Section 4.3 and Table 2. The test set is kept independent of representation analysis, architecture search, architecture selection, fine-tuning, and checkpoint selection, and is used only for the final performance evaluation.

RQ1: How does MPMS-AG perform in the homogeneous convolutional network stitching scenario?

Approach: We first evaluate MPMS-AG under a homogeneous convolutional network stitching scenario. In this experiment, the candidate blocks are selected from CNN-based models with homogeneous convolutional structures. The purpose is to examine whether MPMS-AG can construct a stitched architecture with a better trade-off between classification accuracy and computational complexity. Following the experimental setting, the parameter budget is constrained to 23 M and the computational budget is constrained to 1.2 GFLOPs. We compare MPMS-AG with representative manually designed CNN models and existing stitching methods on the CIFAR-10 dataset. The evaluation metrics include the number of parameters, FLOPs, and Top-1 classification accuracy.

Results: As shown in Table 3, MPMS-AG achieves a Top-1 accuracy of 95.10% in the homogeneous stitching scenario. This value is 0.15 percentage points higher than the reported result of FuLA (94.95%). Since repeated independent runs and variance estimates are unavailable for this comparison, this difference is reported descriptively and is not interpreted as statistically significant. Meanwhile, MPMS-AG requires 21.50 M parameters and 0.92 GFLOPs, compared with 22.10 M parameters and 1.05 GFLOPs for FuLA. MPMS-AG also achieves higher reported Top-1 accuracy than DeRy (94.60%) and SN-Net (94.35%) under the reported experimental setting.

Table 3. Experimental results of different methods under the homogeneous stitching scenario on CIFAR-10.

Method	Parameters	FLOPs (G)	Top-1 Accuracy
ResNet18	11.69 M	0.56	93.20%
ResNet34	21.80 M	1.16	94.00%
ResNet50	25.56 M	1.30	94.80%
DeRy	14.20 M	0.78	94.60%
SN-Net	19.50 M	0.95	94.35%
FuLA	22.10 M	1.05	94.95%
Ours (MPMS-AG)	21.50 M	0.92	95.10%

Observation: The results suggest that MPMS-AG can identify competitive stitched architectures within the homogeneous convolutional search space under the specified resource budget. Under the reported experimental setting, MPMS-AG achieves the highest Top-1 accuracy among the compared methods while using fewer parameters and FLOPs than FuLA. However, because repeated independent runs and variance estimates are unavailable, the small accuracy difference between MPMS-AG and FuLA should be interpreted cautiously and should not be regarded as evidence of statistically significant superiority.

RQ2: How does MPMS-AG perform in the heterogeneous cross-architecture stitching scenario?

Approach: To further evaluate the cross-architecture stitching capability of MPMS-AG, we conduct experiments in a heterogeneous setting where the candidate blocks come from both CNN and ViT architectures. This experiment aims to verify whether MPMS-AG can effectively handle feature dimension mismatch, structural inconsistency, and semantic gaps between different neural architectures. Following the experimental setting, the parameter budget is constrained to 60 M. We compare MPMS-AG with representative heterogeneous stitching and adaptation methods, including DeRy, MuDSC, and CS-Adapter, on the CIFAR-10 dataset. The evaluation metrics include the number of parameters, FLOPs, and Top-1 classification accuracy.

Results: As shown in Table 4, MPMS-AG achieves competitive performance in the heterogeneous stitching scenario. Under the medium-scale setting with a parameter budget of 60 M, the architecture generated by MPMS-AG achieves a Top-1 accuracy of 98.70%. Compared with CS-Adapter and MuDSC, which achieve 97.50% and 97.10% Top-1 accuracy, respectively, the reported Top-1 accuracy of MPMS-AG is 1.20 and 1.60 percentage points higher, respectively. In addition, MPMS-AG requires 4.70 GFLOPs under the reported experimental setting.

Table 4. Experimental results of different methods under the heterogeneous stitching scenario on CIFAR-10.

Method	Parameters	FLOPs (G)	Top-1 Accuracy
ViT-Base	86.60 M	17.60	98.50%
Swin-Tiny	28.30 M	4.50	96.00%
Swin-Base	88.00 M	15.40	97.80%
ResNet18	11.69 M	0.56	93.20%
ResNet34	21.80 M	1.16	94.00%
ResNet50	25.56 M	1.30	94.80%
MobileNetV3 Large 100	5.40 M	10 M	92.27%
MobileNetV3 Large 75	4.00 M	8 M	91.15%
MobileNetV3 Small 100	2.52 M	3.15 M	90.34%
DeRy	22.50 M	0.85	96.20%
MuDSC	55.00 M	4.20	97.10%
CS-Adapter	48.00 M	4.50	97.50%
Ours (MPMS-AG)	**58.50 M**	**4.70**	**98.70%**

After re-examining the final heterogeneous architecture used in RQ2, we report its four-stage stitching sequence in Table 5. The architecture is constructed sequentially from selected blocks of MobileNetV3-Large-0.75, ResNet50, MobileNetV3-Large-0.75, and Swin-Tiny. Adjacent stages are connected through three adaptive inter-stage adapters (A1–A3), following the adapter mechanism described in Section 4.1.

Table 5. Architecture and parameter information of the heterogeneous model in RQ2.

Stage	Source	Selected Blocks	Inter-Stage Adapter
1	MobileNetV3-Large-0.75	`blocks.0.0--blocks.1.1`	Adaptive adapter (A1)
2	ResNet50	`layer3.2--layer3.5`	Adaptive adapter (A2)
3	MobileNetV3-Large-0.75	`blocks.2.2--blocks.3.3`	Adaptive adapter (A3)
4	Swin-Tiny	`stages.2.downsample-stages.3.blocks.1`	– (final stage)
		Reported total model parameters	58.50 M

Note: A1–A3 denote the inter-stage adapters connecting adjacent selected stages. The symbol "–" indicates that no inter-stage adapter is required after the final stage.

The reported 58.50 M value is the verified total parameter count of the final heterogeneous model. However, the available records do not support a reliable stage-wise

decomposition of this total across the individual source blocks and inter-stage adapters. Therefore, estimated or unverifiable parameter distributions are not reported.

To further investigate the training dynamics of MPMS-AG, we record and compare the Top-1 accuracy curves of MPMS-AG and the baseline methods over 50 training epochs, as shown in Figure 4. The training curves provide two additional observations.

First, MPMS-AG shows rapid accuracy improvement during the early training stage. In the homogeneous scenario, by reusing pre-trained weights, MPMS-AG improves its accuracy from 12.3% to 75.6% at the second epoch. In the heterogeneous scenario, with the adaptive adapter mechanism, MPMS-AG reaches 83.7% accuracy at the fifth epoch. These observations indicate that the generated stitched models can achieve relatively fast early-stage convergence under the reported training setting.

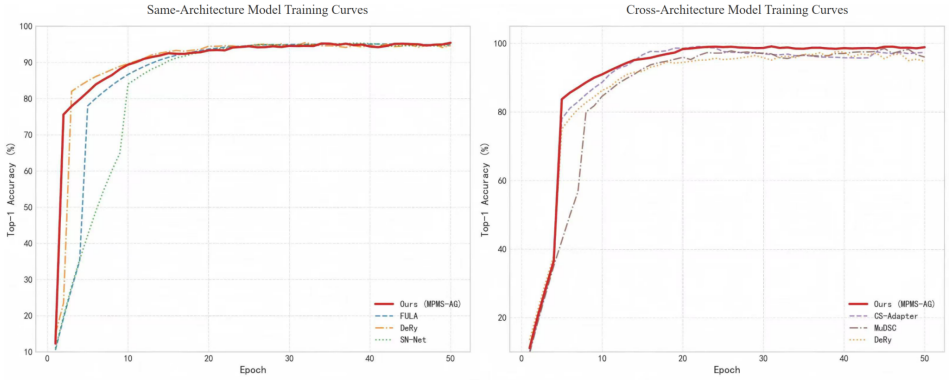

Figure 4. Top-1 accuracy curves over training epochs under homogeneous and cross-architecture stitching scenarios.

Furthermore, the training curves show continued accuracy improvement and relatively stable convergence during the middle and late training stages. In the homogeneous experiment, MPMS-AG reaches a Top-1 accuracy of 95.10% at the 43rd epoch. In the heterogeneous experiment, MPMS-AG reaches a Top-1 accuracy of 98.70%. Compared with the reported baseline curves, MPMS-AG exhibits a relatively smooth convergence trajectory under the current experimental setting. These observations are descriptive and are not interpreted as evidence of statistically significant superiority.

Observation: The results suggest that MPMS-AG can construct trainable architectures from heterogeneous CNN and ViT blocks under the current experimental setting. The training curves indicate relatively rapid accuracy improvement in the early training stage and stable convergence during subsequent training. Compared with CS-Adapter, MPMS-AG achieves a higher reported Top-1 accuracy (98.70% versus 97.50%), with reported FLOPs of 4.70 GFLOPs and 4.50 GFLOPs, respectively. These results provide evidence of the applicability of MPMS-AG to heterogeneous cross-architecture model stitching, while the performance differences are interpreted descriptively because repeated independent runs and statistical significance tests are unavailable.

RQ3: Can MPMS-AG generate efficient lightweight models under resource constraints?

Approach: After validating the effectiveness of MPMS-AG in homogeneous and heterogeneous large-scale stitching tasks, we further investigate its performance in ultra-lightweight scenarios. For deployment on mobile and embedded edge devices, lightweight deep learning models require high inference efficiency while preserving feature representation capability under significantly compressed parameter scales. Therefore, this experiment selects the MobileNetV3 series, which adopts a typical inverted residual structure, as the candidate model family. Specifically, MobileNetV3-Large 100, MobileNetV3-Large 75, and

MobileNetV3-Small 100 are used as candidate pre-trained feature donors, and the parameter budget is constrained to no more than 1.8 M. Since MobileNetV3 adopts a modular block-based architecture, its core components are distributed from `blocks.0` to `blocks.6`. To construct a fine-grained stitching search space, we define the component indices of each model, as shown in Table 6.

Table 6. Search space index definition of the MobileNetV3 candidate models.

Model	Search Space Block Index List
MobileNetV3 Large 100	`blocks.0.0`, `blocks.1.0`, `blocks.1.1`, `blocks.2.0`, `blocks.2.1`, `blocks.2.2`, `blocks.3.0`, `blocks.3.1`, `blocks.3.2`, `blocks.3.3`, `blocks.4.0`, `blocks.4.1`, `blocks.5.0`, `blocks.5.1`, `blocks.5.2`, `blocks.6.0`
MobileNetV3 Large 075	`blocks.0.0`, `blocks.1.0`, `blocks.1.1`, `blocks.2.0`, `blocks.2.1`, `blocks.2.2`, `blocks.3.0`, `blocks.3.1`, `blocks.3.2`, `blocks.3.3`, `blocks.4.0`, `blocks.4.1`, `blocks.5.0`, `blocks.5.1`, `blocks.5.2`, `blocks.6.0`
MobileNetV3 Small 100	`blocks.0.0`, `blocks.1.0`, `blocks.1.1`, `blocks.2.0`, `blocks.2.1`, `blocks.2.2`, `blocks.3.0`, `blocks.3.1`, `blocks.4.0`, `blocks.4.1`, `blocks.4.2`, `blocks.5.0`

During the search stage, MPMS-AG automatically performs sequential decision-making in the above fine-grained search space according to the feature representation capability and computational cost of each block. The final lightweight model topology consists of three stages, and the selected components and their source distributions are summarized in Table 7. The index range of each stage is defined locally within its corresponding source model rather than as a global index range in the final stitched architecture.

Table 7. Details of the lightweight model stitching path.

Stage	Source Model	Local Index Range	Stitched Blocks
Stage-0	MobileNetV3 Large 075	0–2	`blocks.0.0-blocks.1.1`
Stage-1	MobileNetV3 Small 100	2–7	`blocks.1.1-blocks.3.1`
Stage-2	MobileNetV3 Small 100	8–9	`blocks.4.0-blocks.4.1`

Accordingly, the same local index value 2 in Stage-0 and Stage-1 refers to different blocks from MobileNetV3 Large 075 and MobileNetV3 Small 100, respectively. Likewise, the identical block identifier `blocks.1.1` refers to distinct source-specific components and does not indicate repeated reuse of the same block.

The stitching path shows that MPMS-AG extracts robust shallow features from MobileNetV3 Large 075 and combines them with the middle- and deep-level semantic modules of MobileNetV3 Small 100. In this way, MPMS-AG achieves cross-model component recombination under an extremely low parameter budget.

Results: As shown in Table 8, the lightweight model generated by MPMS-AG contains only 1.68 M parameters and requires 2.38 M FLOPs. Compared with MobileNetV3 Small 100, which contains 2.52 M parameters and requires 3.15 M FLOPs, MPMS-AG reduces the number of parameters by approximately 33.3% and reduces FLOPs by approximately 24.4%. More importantly, despite the significant decrease in resource consumption, the stitched model achieves a Top-1 accuracy of 90.65%, which is 0.31% higher than the original MobileNetV3 Small 100.

Table 8. Comparison results of lightweight deep learning model stitching on CIFAR-10.

Dataset	Model	Parameters (M)	FLOPs	Top-1 (%)
CIFAR-10	MobileNetV3 Large 100	5.40	10 M	92.27
CIFAR-10	MobileNetV3 Large 75	4.00	8 M	91.15
CIFAR-10	MobileNetV3 Small 100	2.52	3.15 M	90.34
CIFAR-10	Ours (MPMS-AG)	1.68	2.38 M	90.65

Observation: The results indicate that MPMS-AG can preserve competitive feature representation capability under extremely limited computational resources. Through precise selection and cross-model recombination of hierarchical components, MPMS-AG achieves a competitive accuracy–efficiency trade-off compared with the manually designed lightweight baseline. Compared with the much larger MobileNetV3 Large series, the model generated by MPMS-AG still maintains competitive feature extraction capability after removing a large proportion of redundant parameters. Therefore, the lightweight model stitching experiment indicates the applicability of MPMS-AG under the specified resource-constrained setting and provides a potential approach for efficient automatic model deployment on embedded edge devices.

6. Discussion

In MPMS-AG, hybrid clustering and reinforcement learning-based search are two key components that affect search-space quality and final model performance. The hybrid clustering module determines whether heterogeneous neural blocks can be grouped into semantically coherent candidate clusters, while the search strategy determines whether the agent can identify an effective stitching path under resource constraints. Therefore, we first analyze the effectiveness of these two components through ablation experiments. Beyond component-level effectiveness, we further discuss the scalability of MPMS-AG to larger model libraries, its potential applicability to other computer vision tasks, and the current limitations of the proposed framework.

6.1. Effectiveness of the Hybrid Clustering Module

The hybrid clustering module is designed to address the uncertainty of feature-space distributions among heterogeneous models. Since CNNs and Transformers differ significantly in feature mapping mechanisms, a single clustering algorithm, such as K-Means or DBSCAN alone, may fail to maintain stable search-space quality across different architecture combinations. Therefore, we evaluate the adaptive clustering capability of the hybrid clustering module under different feature distributions.

We compare the complete MPMS-AG with two ablation variants: Ablation-K, which uses only K-Means clustering, and Ablation-D, which uses only DBSCAN clustering. We evaluate Ablation-K, Ablation-D, and the proposed hybrid clustering strategy on pure CNN, pure Transformer, and heterogeneous mixed model pools. The clustering quality score $Q(C)$ defined in Section 3 is used as the main metric, which jointly considers intra-cluster cohesion and inter-cluster separation. A higher $Q(C)$ indicates clearer semantic discrimination in the candidate action space. The experimental results are shown in Table 9.

As shown in Table 9 and Figure 5, the proposed hybrid clustering strategy achieves the highest $Q(C)$ scores across all three scenarios and shows more stable score distributions. In the pure CNN scenario, the hybrid strategy obtains an average $Q(C)$ score of 0.71 with a standard deviation of 0.03, slightly outperforming the single K-Means and DBSCAN variants. In the pure Transformer scenario, the hybrid strategy also achieves the highest average $Q(C)$ score of 0.70 and the best Top-1 accuracy of 97.80%, indicating that it can adapt to different feature-space structures.

Table 9. Comparison of clustering stability and search performance under random model combinations (↑: higher is better; ↓: lower is better).

Combination Type	Clustering Strategy	Avg. $Q(C)$ ↑	Std. ↓	Top-1 Acc.
Pure CNN (10 runs)	Ablation-K	0.69	0.04	94.28%
	Ablation-D	0.58	0.11	93.85%
	Ours	**0.71**	**0.03**	**94.86%**
Pure Transformer (10 runs)	Ablation-K	0.48	0.14	96.15%
	Ablation-D	0.67	0.05	97.42%
	Ours	**0.70**	**0.04**	**97.80%**
Heterogeneous Mixed (20 runs)	Ablation-K	0.42	0.18	95.72%
	Ablation-D	0.55	0.13	96.48%
	Ours	**0.68**	**0.06**	**98.43%**

Note: Bold values indicate the best-performing result for each metric within each model-combination group.

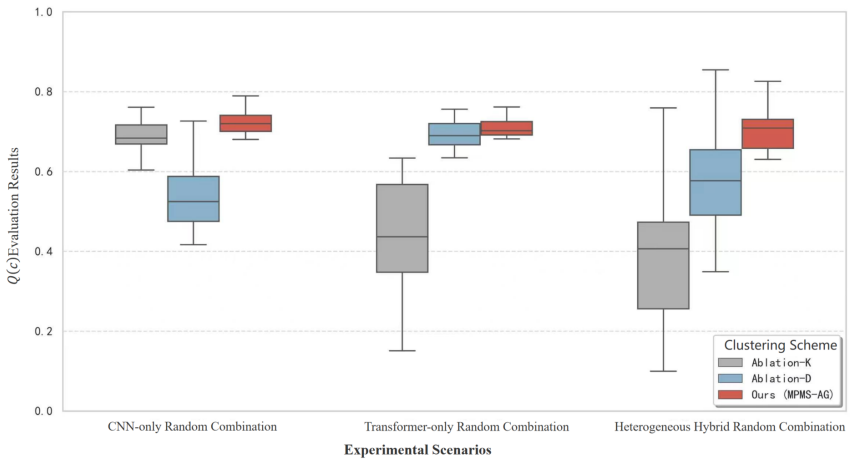

Figure 5. Boxplot comparison of clustering stability under random model combinations.

The advantage of the hybrid clustering module becomes more significant in the heterogeneous mixed scenario. In this setting, the standard deviations of Ablation-K and Ablation-D increase to 0.18 and 0.13, respectively, indicating that a single clustering algorithm is unstable when facing complex heterogeneous feature distributions. In contrast, the proposed strategy maintains a lower standard deviation of 0.06 and achieves the highest average clustering quality score of 0.68. As shown in Figure 5, MPMS-AG has a shorter box range and a higher score distribution, which visually confirms its low-variance and high-quality clustering behavior. Compared with the two single-clustering variants, MPMS-AG improves the final Top-1 accuracy by about 2.0–2.7 percentage points in the heterogeneous mixed scenario.

These results demonstrate that the hybrid clustering module is not merely a simple combination of two clustering algorithms. Instead, it provides an adaptive mechanism that selects a more suitable clustering path according to the intrinsic distribution of candidate neural blocks. This mechanism improves the semantic purity of candidate clusters and provides a more reliable action space for subsequent model stitching.

6.2. Comparison of Search Strategies

To further evaluate the reinforcement learning-based sequential decision-making mechanism under multi-constraint architecture search, we compare the Actor–Critic-based search strategy with a traditional evolutionary algorithm. In the ablation setting, the genetic algorithm (GA) is used to replace the original reinforcement learning module, while the candidate clusters generated by the clustering module remain unchanged. To ensure experimental consistency and comparability, all experiments are conducted under the

homogeneous stitching scenario using ResNet18, ResNet34, and ResNet50 as the candidate model pool. The objective is to search for a high-performing hybrid architecture on CIFAR-10 under the resource constraints of parameters no more than 23 M and FLOPs no more than 1.2 GFLOPs. For GA, real-valued encoding is used to represent module indices, and the fitness function combines the SNIP score with resource penalty terms. The population size is set to 50, and the maximum number of iterations is set to 200, ensuring that the number of sampled architectures is comparable to that of the RL-based search process.

As shown in Table 10, the reinforcement learning-based search strategy achieves a Top-1 accuracy of 95.10%, which is 0.90 percentage points higher than that of the genetic algorithm (94.20%). Meanwhile, RL-A2C requires 21.50 M parameters and 0.92 GFLOPs, compared with 22.20 M parameters and 1.05 GFLOPs for GA. Under the reported experimental setting, RL-A2C achieves a better reported accuracy–efficiency trade-off than GA.

Table 10. Ablation results of different search strategies based on the ResNet model pool.

Search Strategy	Top-1 Accuracy	Parameters (M)	FLOPs (G)
Genetic Algorithm (GA)	94.20%	22.20	1.05
Reinforcement Learning (RL-A2C)	**95.10%**	**21.50**	**0.92**

The observed performance difference may be related to the different optimization mechanisms of the two strategies. The genetic algorithm relies on population evolution, crossover, and mutation to explore the search space. Although it can perform global exploration, it does not explicitly model the sequential dependency among stitched blocks. Consequently, its search process may generate architectures that satisfy the resource constraints but do not fully preserve the hierarchical functional continuity of neural representations.

By contrast, the RL-A2C strategy formulates model stitching as a sequential decision-making problem. At each decision step, the agent selects a candidate block based on the previously selected sequence and the current resource state. This enables the search process to consider both local block compatibility and global architecture constraints. In addition, the value network provides an estimation of the long-term reward of partially constructed architectures, which can assist the agent in distinguishing candidate stitching paths during sequential architecture construction.

These results provide descriptive evidence for the utility of the reinforcement learning-based search strategy in MPMS-AG under the current constrained stitching setting. By modeling architecture construction as a sequential decision-making process, RL-A2C provides a mechanism for incorporating previous block selections and resource states during the search process.

6.3. Scalability and Applicability

The random model-combination experiments in Section 6.1 provide preliminary evidence of the stability of MPMS-AG under different candidate model compositions. As shown in Table 6, the proposed hybrid clustering strategy maintains relatively consistent clustering quality across pure CNN, pure Transformer, and heterogeneous mixed model combinations. In particular, under the heterogeneous mixed setting, MPMS-AG achieves an average clustering quality score of 0.68 with a standard deviation of 0.06. These results suggest that the proposed clustering mechanism remains relatively stable when the architectural composition of the evaluated model pool changes.

Nevertheless, the current experiments are conducted on a moderate-scale candidate model library, and the scalability of MPMS-AG to substantially larger model zoos has not yet been experimentally validated. As the number of source models increases, the number of reusable blocks and representation comparisons also increases, leading to higher

costs for RBF-CKA similarity analysis, clustering, and architecture search. Adaptive block partitioning, hybrid clustering, and dynamic masking alleviate this issue by compressing and pruning the candidate space, but they cannot completely eliminate the additional computational overhead. Future work will investigate more efficient candidate pre-filtering and representation analysis strategies for larger model repositories.

The current implementation of MPMS-AG is primarily evaluated on image classification tasks. Its representation alignment, functional block clustering, and resource-aware architecture search mechanisms may potentially be extended to backbone construction for other computer vision tasks, such as object detection and semantic segmentation. However, these tasks involve additional requirements, including multi-scale feature representations, spatial-resolution preservation, and task-specific prediction heads and objectives. Therefore, corresponding modifications to the stitching adapters and architecture evaluation functions would be required. The applicability of MPMS-AG to these downstream tasks remains to be experimentally investigated.

6.4. Limitations

Several limitations of MPMS-AG should be acknowledged.

First, the current evaluation is limited to CIFAR-10 and a moderate-scale candidate model library. Although MPMS-AG is evaluated under homogeneous, heterogeneous, and lightweight stitching scenarios, its cross-dataset generalizability remains to be verified on more challenging benchmarks and larger pre-trained model repositories.

Second, although the final heterogeneous architecture and its total parameter count can be recovered, the available records do not support a reliable stage-wise decomposition of the reported 58.50 M parameters across individual source blocks and inter-stage adapters.

Third, RBF-CKA representation analysis introduces additional preprocessing costs as the number of reusable blocks increases. Fourth, the SNIP zero-shot proxy may not always be consistent with final performance after fine-tuning. Fifth, heterogeneous CNN–Transformer stitching requires adaptive feature transformation modules, which may introduce additional parameters and inference overhead.

Finally, the current CNN-to-CNN adapter uses discrete stride selection and does not explicitly support arbitrary upsampling, exact non-integer resizing, or anisotropic spatial resizing. Specifically, it uses stride 1 when $r < 2$ and stride 2 when $r \geq 2$.

For Swin-based successors, bilinear interpolation can additionally be applied when the target spatial grid is available.

Addressing these limitations and extending the evaluation to more diverse benchmarks will be important directions for future work.

7. Conclusions

This paper proposes MPMS-AG, an automatic generation framework for deep learning models based on heterogeneous pre-trained model stitching. MPMS-AG addresses the difficulties of feature alignment and structural integration between heterogeneous models, such as CNNs and ViTs, by decomposing heterogeneous pre-trained models into reusable neural blocks and formulating model stitching as a sequential decision-making problem. Specifically, MPMS-AG uses hierarchical feature extraction and Radial Basis Function Centered Kernel Alignment (RBF-CKA) to quantify functional similarity between heterogeneous layers, introduces adaptive block partitioning and hybrid clustering to construct functionally similar candidate block clusters and reduce the search space, and adopts a Generalized Advantage Estimation (GAE)-based reinforcement learning strategy to search for stitching sequences under parameter and FLOPs constraints.

Under the current CIFAR-10 experimental protocol, MPMS-AG generates trainable hybrid architectures with competitive classification performance across homogeneous, heterogeneous cross-architecture, and lightweight stitching scenarios. The ablation results provide descriptive evidence that hybrid clustering and reinforcement learning-based search contribute to the observed performance. The official CIFAR-10 test set is kept independent from representation analysis, architecture search, architecture selection, fine-tuning, and checkpoint selection, and is used only for the final performance evaluation. Future work will further investigate larger-scale datasets and downstream vision tasks.

Author Contributions: R.X. and S.W.: conceptualization; methodology; software; validation; formal analysis; investigation; resources; data curation; writing—original draft preparation; writing—review and editing; visualization. D.C.: methodology; validation; formal analysis; writing—review and editing. L.P. and Y.Z.: software; validation; data curation; writing—review and editing; visualization; supervision; project administration; funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the Shenzhen Science and Technology Program under grant No. JCYJ20240813152507009, the Basic and Applied Basic Research Foundation of Guangdong Province under grant No. 2026A1515010518, the Fundamental Research Funds for the Central Universities under grant No. NJ2025054, the Foundation of the Key Laboratory of Safety-Critical Software (Nanjing University of Aeronautics and Astronautics), the Collaborative Innovation Center of Novel Software Technology and Industrialization and the High Performance Computing Platform of Nanjing University of Aeronautics and Astronautics. This research is also supported by the 2024 Open Fund Projects of the Key Laboratory of Information System Requirements under grant No. LHZZ202404.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kaiming, H.; Xiangyu, Z.; Shaoqing, R.; Jian, S. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; Volume 34, pp. 770–778.
2. Tao, H.; Tong, Y.; Zhang, Z. Design and implementation of data visualization analysis systembased on UE4. *Command. Inf. Syst. Technol.* **2025**, *16*, 90–94.
3. Wei, Q.; Xia, M.; Yuan, Y. UAV collaborative mission planning based on multi-agent deep reinforcement learning. *Command. Inf. Syst. Technol.* **2026**, *17*, 52–59.
4. Liu, H.; Simonyan, K.; Yang, Y. Darts: Differentiable architecture search. *arXiv* **2018**, arXiv:1806.09055.
5. Zhou, D.; Kang, B.; Jin, X.; Yang, L.; Lian, X.; Jiang, Z.; Hou, Q.; Feng, J. Deepvit: Towards deeper vision transformer. *arXiv* **2021**, arXiv:2103.11886.
6. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online, 16–20 November 2020; pp. 38–45.
7. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
8. Pan, Z.; Cai, J.; Zhuang, B. Stitchable Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 16102–16112.
9. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning filters for efficient convnets. *arXiv* **2016**, arXiv:1608.08710.
10. Lee, N.; Ajanthan, T.; Torr, P.H. Snip: Single-shot network pruning based on connection sensitivity. *arXiv* **2018**, arXiv:1810.02340.
11. Elsken, T.; Metzen, J.H.; Hutter, F. Neural architecture search: A survey. *J. Mach. Learn. Res.* **2019**, *20*, 1–21.
12. Howard, J.; Ruder, S. Universal language model fine-tuning for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Melbourne, Australia, 15–20 July 2018; pp. 328–339.

13. Xuhong, L.; Grandvalet, Y.; Davoine, F. Explicit inductive bias for transfer learning with convolutional networks. In *Proceedings of the International Conference on Machine Learning*; PMLR: New York, NY, USA, 2018; pp. 2825–2834.
14. Bansal, Y.; Nakkiran, P.; Barak, B. Revisiting model stitching to compare neural representations. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 225–236.
15. Lenc, K.; Vedaldi, A. Understanding image representations by measuring their equivariance and equivalence. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 991–999.
16. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. Training data-efficient image transformers & distillation through attention. In *Proceedings of the International Conference on Machine Learning*; PMLR: New York, NY, USA, 2021; pp. 10347–10357.
17. Pan, Z.; Liu, J.; He, H.; Cai, J.; Zhuang, B. Stitched ViTs are Flexible Vision Backbones. In Proceedings of the European Conference on Computer Vision, Milan, Italy, 29 September–4 October 2024; pp. 258–274.
18. He, H.; Pan, Z.; Liu, J.; Cai, J.; Zhuang, B. Efficient stitchable task adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 16–22 June 2024; pp. 28555–28565.
19. Anakewat, T.; Mukuta, Y.; Westfechtel, T.; Harada, T. Self-Stitching: Widely Applicable and Efficient Transfer Learning Using Stitching Layer. In *Proceedings of the NeurIPS 2024 Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability, Vancouver, BC, Canada, 14 December 2024*; OpenReview: Amherst, MA, USA, 2024.
20. Athanasiadis, I.; Karmush, A.; Felsberg, M. Model Stitching by Functional Latent Alignment. *arXiv* **2025**, arXiv:2505.20142.
21. Yang, X.; Zhou, D.; Liu, S.; Ye, J.; Wang, X. Deep model reassembly. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 25739–25753. [CrossRef]
22. Traft, N.; Cheney, N. Stitching Disparate Neural Network Layers with Complex Adapters and Spatial Rescaling. In *Proceedings of the AutoML 2025 Non-Archival Content Track, New York, NY, USA, 8–11 September 2025*; OpenReview: Amherst, MA, USA, 2025.
23. Xu, Z.; Yuan, K.; Wang, H.; Wang, Y.; Song, M.; Song, J. Training-free pretrained model merging. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 16–22 June 2024; pp. 5915–5925.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.