UNIFIED MERGING OF HETEROGENEOUS PRETRAINED MODELS: FROM LAYER SELECTION TO SEMANTIC ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

As a novel technique for large pretrained models, model merging integrates multiple independently pretrained or fine-tuned models to construct a unified and efficient model without accessing the original training data or incurring an expensive computational cost. Existing model merging methods predominantly focus on parameter-level fusion within a fixed model architecture, facing inherent limitations in scalability and performance enhancement. While merging models at the architecture level shows promising potential, current approaches lack a clear and unified problem formulation, and suffer from exponential growth of the search space, and incompatibility of inter-layer feature representations. To address these challenges, we propose a unified architecture-parameter merging framework named UniMerge. Specifically, UniMerge formulates model merging as a Markov decision process and incorporates parameter-level merged candidates into an architecture-level search pool. To solve the formulated problem, UniMerge designs a state-enhanced embedding network to capture inter-layer parameter correlations and introduces a contrastive self-supervised learning-assisted actor-critic policy to improve exploration efficiency. To mitigate inter-layer incompatibility, we develop a glue layer based on dual knowledge distillation to ensure inter-layer semantic alignment and cross-layer feature representation. Extensive experiments on language and vision models demonstrate strong performance and search efficiency, while cross-scale experiments further show that UniMerge can stitch compatible models with different widths and depths, extending architecture-level merging from fine-tuning-induced heterogeneity to explicit structural heterogeneity.

1 INTRODUCTION

In recent years, large pretrained models have achieved remarkable progress in task domains Chang et al. (2024); Wu et al. (2025b). To address diverse tasks, researchers have developed numerous large pretrained or fine-tuned models Radford et al. (2021). However, due to privacy and usage restrictions, the original training data of these models is often inaccessible, posing significant challenges for building unified models across domains. Model merging Li et al. (2026), which aggregates knowledge by merging existing models without original data, has emerged as a promising paradigm for constructing multi-task models. Existing model merging methods can be broadly categorized into three classes Yang et al. (2024c): weighted-based Ilharco et al. (2023), subspace-based Yadav et al. (2023); Yu et al. (2024), and router-based method Huang et al. (2024a). Nevertheless, these methods operate on the parameter-level merging within isomorphic model architectures, limiting their extension to heterogeneous architecture models. Wan et al. (2024; 2025) attempt to convert heterogeneous models into a unified architecture via knowledge distillation and then perform parameter-level merging. However, these approaches incur high computational costs and still merge within a fixed architecture, thereby failing to overcome the performance bottlenecks of the single model architecture. Consequently, architecture-level merging on heterogeneous models has emerged as a promising solution.

Architecture-level model merging can effectively leverage the representation capacity and task generalization ability across different architectures. Theoretically, it has the potential to surpass the performance limits of any single architecture Kaplan et al. (2020). Franken merging Goddard et al. (2024) has demonstrated the promise of architecture-level merging through layer concatenation

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

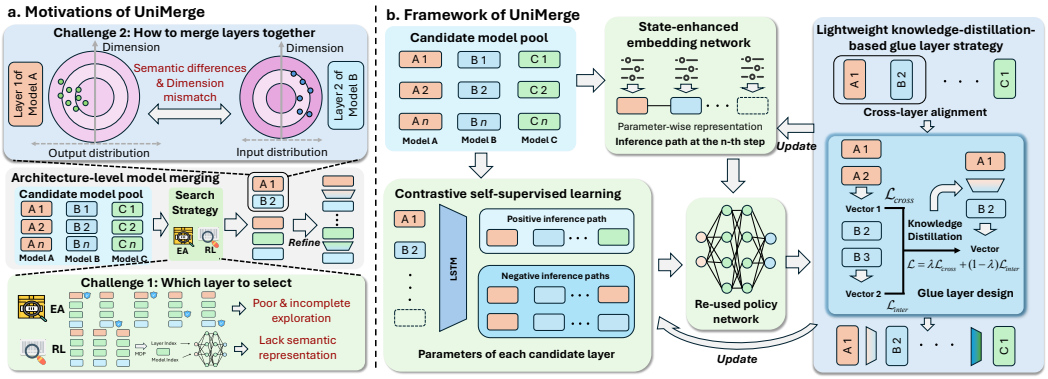


Figure 1: (a) Challenges of architecture-level model merging. i) ***Which layers to merge***: the exponential search space makes exhaustive exploration infeasible. ii) ***How to merge layers together***: adjacent layers from different models may exhibit dimensional, distributional, and semantic mismatches. (b) Framework of UniMerge. UniMerge formulates architecture-level merging as an MDP. To address ***which layers to select***, it uses a parameter-wise state-enhanced embedding network and a contrastive self-supervised actor-critic search strategy. To address ***how to merge layers together***, it introduces a knowledge-distillation-based glue layer to align inter-layer semantics and cross-layer feature representations.

strategies. Based on Franken merging, Kim et al. (2024) expanded 32-layer Mixtral-8×7B Jiang et al. (2024) into a more efficient 48-layer SOLAR 10.7B. Nevertheless, SOLAR 10.7B requires continued pretraining after merging. To address this issue, EvoMerge Akiba et al. (2025) and HM3 Zhou et al. (2025) attempt to assemble architectures by searching over candidate layers or inference paths.

Despite the potential of architecture-level merging, significant challenges remain in both definition and implementation. From definition, Li et al. (2026) consider architecture-level model merging as automatic model configuration. However, Li et al. (2025c;a) adjust parameters rather than performing genuine architecture-level combination, whereas EvoMerge and HM3 jointly optimize parameters and architecture. Thus, existing work lacks a clear, unified formalization of architecture-level model merging. From implementation, existing research requires constructing a candidate model pool where a search method is designed for an optimal inference path to obtain the final merged model. In this process, one of the critical issues is the exponential growth of the search space, as shown in challenge 1 of Fig. 1-(a). Assuming each candidate model comprises 32 layers, a pool of M models induces a search space with M^{32}. This exponential growth severely limits the efficiency of existing methods, where EvoMerge employs evolutionary search, potentially missing many promising search regions and leading to incomplete exploration. Additionally, HM3 uses a reinforcement learning (RL) algorithm to search for inference paths but its inadequate state representation hinders effective guidance. To this end, the first question is: which layers to merge in the exponential search space.

After layer selection, adjacent layers may come from different pretrained models, leading to heterogeneous hidden dimensions, normalization statistics, or even residing in different semantic spaces. Direct concatenation of adjacent layers can cause feature misalignment and representation drift, disrupting information flow and forward-propagation stability. Even within the same architecture, variations in training processes can lead to significant differences in statistical distributions He et al. (2025). As shown in challenge 2 of Fig. 1-(a), another key challenge is the layer incompatibility and misalignment He et al. (2025). To alleviate this, EvoMerge employs a hypernetwork to generate inter-layer scaling factors, while HM3 uses feedforward networks to mitigate distribution shifts. However, existing methods mainly address fine-tuning-induced heterogeneity under the same or similar backbone scale, while their connector designs remain heuristic and lack systematic validation for width- and depth-level heterogeneity. Thus, the second question is: how to merge layers together.

To address the above challenges, we propose a *Uni*fied architecture-parameter model *Merg*ing framework called UniMerge. In UniMerge, we define an architecture-parameter model merging problem, which incorporates the parameter-level merged models into the candidate pool for architecture-level merging, and then formulate an inference path search problem with the Markov decision process (MDP) property. To overcome layer selection in the exponential search space, UniMerge designs a parameter-wise embedding network to enhance the MDP's state representation and introduces a

contrastive self-supervised learning-assisted actor-critic method to solve the enhanced MDP, thereby improving the efficiency. To resolve cross-model layer alignment, UniMerge proposes a knowledge-distillation-based glue layer strategy, which minimizes feature distribution disparities via inter-layer semantic alignment-based and cross-layer feature representation-based knowledge distillation. Finally, UniMerge analyzes the effectiveness of the glue layer strategy through optimal transport. More importantly, UniMerge extends architecture-level merging from fine-tuning-induced heterogeneity to explicit structural heterogeneity by stitching compatible model components with different widths and depths. Thus, the main contributions are as follows:

- Different from previous studies that mainly focus on layer concatenation or search strategies, we provide a systematic formulation of the architecture-level model merging problem with the MDP property. Based on this formulation, we propose UniMerge to clarify and address two fundamental questions: which layers to merge and how to merge layers together.
- To answer ***which layers to merge***, UniMerge first designs a state-enhanced embedding network to capture structural and parametric correlations between candidate layers. A contrastive learning–assisted actor-critic algorithm is further developed to guide efficient layer selection in high-dimensional spaces.
- To answer ***how to merge layers together***, we propose a lightweight glue layer strategy based on dual knowledge distillation, which aligns both inter-layer semantics and feature representations. We then provide a theoretical analysis through optimal transport. This strategy further enables UniMerge to stitch compatible pretrained models with different widths and depths, extending heterogeneous merging to explicit structural heterogeneity.

2 Problem Formulation of Architecture-level Model Merging

We review existing studies on model merging from parameter and architecture perspectives, and introduce the background of knowledge distillation. Detailed related work is provided in **Appendix A**.

Before introducing UniMerge, we first present a unified problem formulation for architecture-level model merging. Given a set of tasks $\{\mathcal{T}_k\}_{k=1}^{K}$, each associated with a fine-tuned model $M_k \in \{M_1, \ldots, M_K\}$ and a corresponding dataset $\mathcal{D}_k = \{(\mathbf{x}_k^i, y_k^i)\}_{i=1}^{|\mathcal{D}_k|}$, each M_k consists of L_k stacked transformer decoder layers and can be expressed as: $M_k : y(\mathbf{x}; \mathbf{\Theta}_k) = f_{\text{out}} \circ f_{k,L_k}(\cdot; \boldsymbol{\theta}_{k,L_k}) \circ \cdots \circ f_{k,1}(f_{\text{input}}(\mathbf{x}); \boldsymbol{\theta}_{k,1})$, where $f_{\text{input}}(\mathbf{x})$ denotes the embedded input function. The model parameters of each M_k are $\mathbf{\Theta}_k = \{\boldsymbol{\theta}_{k,l}\}_{l=1}^{L_k}$. Each layer $f_{k,l}(\cdot; \boldsymbol{\theta}_{k,l}) : \mathbb{R}^{d_{k,l-1}} \to \mathbb{R}^{d_{k,l}}$, $l \in \{1, \ldots, L_k\}$, represents the internal representation of layer l with learnable parameters $\boldsymbol{\theta}_{k,l}$, with $d_{k,l}$ the output dimension of layer l. We denote the model architecture as $\mathcal{F}_k = \{f_{k,1}, \ldots, f_{k,L_k}\}$. For simplicity, $f_{k,l}(\mathbf{x}; \boldsymbol{\theta}_{k,l})$ is abbreviated as $f_{k,l}(\cdot; \boldsymbol{\theta}_{k,l})$. In this paper, we aim to merge models in architecture and parameter spaces by optimizing $\mathcal{F}_{\text{merge}} = (f_1, \mathcal{G}_1, \ldots, \mathcal{G}_{L_{\text{merge}}-1}, f_{L_{\text{merge}}})$ and $\mathbf{\Theta}_{\text{merge}} = (\{\boldsymbol{\theta}_l\}_{l=1}^{L_{\text{merge}}}, \{\boldsymbol{\theta}_l^{\text{glue}}\}_{l=1}^{L_{\text{merge}}-1})$. The optimization objective is to minimize the loss of the merged model over all test datasets $\mathcal{D}_k^{\text{Test}}, \forall k \in \{1, \ldots, K\}$, which is formulated as

$$\mathcal{P}_1 : \min_{\mathbf{\Theta}_{\text{merge}}, \mathcal{F}_{\text{merge}}} \quad \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_k^{\text{Test}}} \Big[\mathcal{L}_{\mathcal{T}_k}(\mathbf{\Theta}_{\text{merge}}, \mathcal{F}_{\text{merge}}) \Big] \tag{1a}$$

$$\text{s.t.} \quad \mathcal{C}_1 : \mathcal{G}_l(\cdot; \boldsymbol{\theta}_l^{\text{glue}}) : \mathbb{R}^{d_l^{\text{out}}} \to \mathbb{R}^{d_{l+1}^{\text{in}}}, \quad \forall l \in \{1, \ldots, L_{\text{merge}} - 1\} \tag{1b}$$

where $\mathcal{L}_{\mathcal{T}_k}$ is the loss function for $\mathcal{T}_k$ and $\mathcal{C}_1$ ensures dimension compatibility between adjacent layers after architecture merging. Note that L_{merge} is not necessarily equal to L_k. This differs from parameter-level merging, which fixes model structure L_k and only adjusts parameters. Due to the combinatorial complexity and the strong coupling of architecture and parameter spaces, directly optimizing $\mathcal{P}_1$ is computationally intractable.

Given the maturity of parameter-level merging methods that efficiently generate high-quality merged models, we reformulate $\mathcal{P}_1$ to a tractable formulation. Specifically, we leverage existing parameter-level merging methods to produce several high-performance merged models, and then construct a candidate model pool $\mathcal{M} = \{M_1, \ldots, M_K, M_{K+1}, \ldots, M_{N_{\text{total}}}\}$. Here, $\mathcal{M}$ includes each fine-tuned model M_k for $\mathcal{T}_k, k \in \{1, \ldots, K\}$, along with $N_{\text{total}} - K$ merged models via parameter-level merging methods (e.g., Task Arithmetic Ilharco et al. (2023), Ties Merging Yadav et al. (2023), etc.).

Each candidate model $M_m \in \mathcal{M}$ provides its parameter and architecture candidates $(\boldsymbol{\theta}_{m,l}, f_{m,l})$, $\forall l \in \{1, \ldots, L_m\}$ for each layer. Based on $\mathcal{M}$, the architecture-level merging can be performed. To this end, $\mathcal{P}_1$ is transformed into

$$\mathcal{P}_2 : \min_{\boldsymbol{\Theta}_{\text{merge}}, \mathcal{F}_{\text{merge}}} \quad \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_k^{\text{Test}}} \left[\mathcal{L}_{\mathcal{T}_k}(\boldsymbol{\Theta}_{\text{merge}}, \mathcal{F}_{\text{merge}}) \right] \tag{2a}$$

$$\text{s.t.} \quad \mathcal{C}_1 : \mathcal{G}_l(\cdot; \boldsymbol{\theta}_l^{\text{glue}}) : \mathbb{R}^{d_l^{\text{out}}} \to \mathbb{R}^{d_{l+1}^{\text{in}}}, \quad \forall l \in \{1, \ldots, L_{\text{merge}} - 1\} \tag{2b}$$

$$\mathcal{C}_2 : \boldsymbol{\Theta}_k = g_{\text{merge}}(\boldsymbol{\Theta}_1, \ldots, \boldsymbol{\Theta}_K), \quad \forall k \in \{K+1, \ldots, N_{\text{total}}\} \tag{2c}$$

where g_{merge} denotes a parameter-level merging method; $\mathcal{C}_1$ ensures dimension compatibility between adjacent layers after architecture merging; $\mathcal{C}_2$ indicates that the first K models in $\mathcal{M}$ are fine-tuned models, and the remainder are merged models via parameter-level merging.

From $\mathcal{P}_2$, one of the difficulties lies in selecting suitable models and their corresponding layers by capturing the characteristics and functions of each layer. In addition, $\mathcal{P}_2$ is a constrained optimization problem subject to $\mathcal{C}_1$. When a feasible solution of $\mathcal{P}_2$ is obtained, its corresponding merged model should satisfy dimensional alignment and maintain consistency of representation spaces. Therefore, there are two challenges for solving $\mathcal{P}_2$: ***which layers to merge*** and ***how to merge layers together***.

3 UNIMERGE

To address the above challenges, we propose a *Uni*fied architecture-parameter model *Merg*ing framework, termed UniMerge in Fig. 1-(b). For ***which layer to select***, UniMerge designs a **state-enhanced embedding representation** to compress layer parameters and encode inter-layer correlations, and then formulates layer selection as a **state-enhanced MDP** optimized by an actor-critic policy. Moreover, a **contrastive self-supervised learning mechanism** is introduced to improve the discriminability of candidate layers and the efficiency of path exploration. For ***how to merge layers together***, UniMerge develops **a lightweight knowledge-distillation-based glue layer** to align inter-layer semantics and feature representations, whose effectiveness is interpreted through optimal transport Villani (2009).

3.1 WHICH LAYER TO SELECT

State-Enhanced Embedding Representation For layer selection, a prerequisite is a robust representation of each layer in $\mathcal{M}$. As a core optimization objective during the construction of large pretrained models, parameter matrices can reliably reflect the structural, capacity, and functional attributes of the layers, which are independent of specific tasks Wu et al. (2025a; 2024). Therefore, we select the parameter matrix of each layer as the input state information, and extract parameter matrices of all layers from each model in $\mathcal{M}$ for low-rank compression. Specifically, for the l-th layer of the m-th model, the parameter matrix $\mathbf{W}_{m,l} \in \mathbb{R}^{d_{m,l}^{\text{in}} \times d_{m,l}^{\text{out}}}$ is approximated by truncated singular value decomposition (SVD) as a rank-r matrix:

$$\mathbf{W}_{m,l} \approx \tilde{\mathbf{W}}_{m,l} = \mathbf{U}_{m,l}^{(r)} \boldsymbol{\Sigma}_{m,l}^{(r)} \left(\mathbf{V}_{m,l}^{(r)} \right)^{\top}, \tag{3}$$

where $\mathbf{U}_{m,l}^{(r)} \in \mathbb{R}^{d_{m,l}^{\text{in}} \times r}$, $\boldsymbol{\Sigma}_{m,l}^{(r)} \in \mathbb{R}^{r \times r}$, and $\mathbf{V}_{m,l}^{(r)} \in \mathbb{R}^{d_{m,l}^{\text{out}} \times r}$ denote the top r singular values and corresponding singular vectors retained. We use the singular values in the diagonal matrix $\boldsymbol{\Sigma}_{m,l}^{(r)}$ as a compact feature vector input to the encoder of the state-enhance representation network. Then, the state embedding vector is generated by this representation network, which can be formulated as:

$$\mathbf{h}_{m,l} = f_{\text{embedding}}(m, l, \text{diag}(\boldsymbol{\Sigma}_{m,l}^{(r)}); \boldsymbol{\theta}_{\text{embedding}}) \in \mathbb{R}^{d_{m,l}}, \tag{4}$$

where $f_{\text{embedding}}(\cdot; \boldsymbol{\theta}_{\text{embedding}})$ is an encoder with self-attention mechanism, and $\boldsymbol{\theta}_{\text{embedding}}$ is the parameters of the representation network. The detailed process of $\mathbf{h}_{m,l}$ is provided in **Appendix B.1**.

Based on state-enhanced representation, we define the key components of the enhanced MDP as follows: (i) State space: At the t-th step, the state s_t is represented as the combination of the embeddings of previously selected layers and the current candidate layer set: $s_t = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{t-1}\} \in \mathcal{S}$,

where $\mathbf{h}_t = \mathbf{h}_{m_t,l_t} = f_{\text{embedding}}(m_t, l_t, \text{diag}(\mathbf{\Sigma}^{(r)}_{m_t,l_t}))$ denotes the state embedding of the selected layer at the t-th step. (ii) Action space: At the t-th step, the action is to select a layer (m, l) from $\mathcal{M}$: $\mathcal{A}_t = \{(m, l) \mid M_m \in \mathcal{M}, l \in \{1, \ldots, L_m\}\}$. (iii) Reward function: After running the MDP and obtaining a model architecture path $\mathcal{F} = \{f_{m_t,l_t}\}_{t=1}^{T}$, we define a composite reward for the path as a weighted combination of performance, diversity, and computational efficiency: $R(\mathcal{F}) = \alpha_1 R_{\text{perf}} + \alpha_2 R_{\text{div}} - \alpha_3 R_{\text{eff}}$, where R_{perf} is the average performance metric across multiple tasks; $R_{\text{div}} = \frac{1}{T}\sum_{i=2}^{T}[1 - \max_{1\le j<i} \text{sim}(\mathbf{h}_i, \mathbf{h}_j)]$ and $R_{\text{div}} = 0$ for $T = 1$; and $R_{\text{eff}} = \log\left(1 + \frac{\text{FLOPs}}{10^9}\right)$ penalizes computational overhead.

At the t-th step, the policy network parameterized by $\boldsymbol{\theta}_{\text{policy}}$ constructs a context representation $\mathbf{h}_c$ from the previously selected layer embeddings. Based on $\mathbf{h}_c$, $\boldsymbol{\theta}_{\text{policy}}$ selects one candidate layer at each step, thereby forming an inference path $\mathcal{F} = \{f_{m_t,l_t}\}_{t=1}^{T}$ with the corresponding embedding sequence $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T)$. The inference-path generation process is factorized as

$$\pi_{\boldsymbol{\theta}_{\text{policy}}}(\mathbf{H} \mid \mathcal{S}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}_{\text{policy}}}(\mathbf{h}_t \mid \mathcal{S}, \mathbf{h}_1, \ldots, \mathbf{h}_{t-1}), \tag{5}$$

where $\mathcal{S}$ denotes the candidate layer state space. Since Eq. (5) defines an autoregressive sequence model, we implement the policy network as a Transformer decoder Yuan et al. (2022). During training, the next layer is sampled from $p_{\boldsymbol{\theta}_{\text{policy}}}(\mathbf{h}_t \mid \mathcal{S}, \mathbf{h}_{1:t-1})$ to encourage exploration. The detailed policy network modeling is provided in **Appendix B.2**.

Contrastive Self-Supervised Learning Mechanism To enhance the semantic discriminability of candidate layers in the embedding space, we introduce a contrastive learning mechanism to construct positive and negative sample pairs. Specifically, UniMerge designed a contrastive self-supervised network comprising an LSTM and a set of linear transformation layers. At each step, the sequence of historical embeddings $[\mathbf{h}_1, \ldots, \mathbf{h}_{t-1}]$ is fed into the LSTM to obtain a contrastive context representation $\tilde{\mathbf{h}}_c$, which is further linearly mapped to predict the candidate layer embedding at the next step: $\tilde{\mathbf{h}}_t = \mathbf{W}_{\text{CL}}\tilde{\mathbf{h}}_c + \mathbf{b}_{\text{CL}}$.

In this contrastive learning network, $\mathbf{h}_t$ selected by the policy network is treated as the positive sample $\mathbf{h}^+$, while the unselected candidate layer embeddings form the negative sample set $\mathcal{H}^- = \{\mathbf{h}_j \mid j \neq t\}$. Inspired by Oord et al. (2018), instead of regressing $\mathbf{h}_t$ directly, we maximize the mutual information between $\mathbf{h}_t$ and $\tilde{\mathbf{h}}_t$ by maximizing their density ratio: $f_{\text{CL}}(\mathbf{h}_t, \tilde{\mathbf{h}}_t) \propto \frac{p(\mathbf{h}_t|\tilde{\mathbf{h}}_t)}{p(\mathbf{h}_t)}$. Here, $\propto$ denotes proportionality. Intuitively, $f_{\text{CL}}(\mathbf{h}_t, \tilde{\mathbf{h}}_t)$ measures the correlation between the positive sample $\mathbf{h}^+ = \mathbf{h}_t$ and $\tilde{\mathbf{h}}_t$. The training objective of the LSTM with the contrastive loss $\mathcal{L}_{\text{CL}}$ is that the predicted embedding $\tilde{\mathbf{h}}_t$ is closest to the positive sample $\mathbf{h}^+$ and farthest from negative samples $\mathcal{H}^-$. Considering the large pretrained model scenario, UniMerge adopts cosine similarity to construct the density ratio function: $f_{\text{CL}}(\mathbf{h}_t, \tilde{\mathbf{h}}_t) = \exp\left(\frac{\cos(\mathbf{h}_t, \tilde{\mathbf{h}}_t)}{\tau}\right)$, where $\cos(\mathbf{h}_t, \tilde{\mathbf{h}}_t)$ is the cosine similarity and τ is the temperature coefficient. Consequently, $\mathcal{L}_{\text{CL}}$ used to train both the contrastive self-supervised and the state-enhanced embedding network is formulated as:

$$\mathcal{L}_{\text{CL}} = -\mathbb{E}\left[\log \frac{f_{\text{CL}}(\mathbf{h}_t, \tilde{\mathbf{h}}_t)}{\sum_{\mathbf{h}_j \in \mathcal{N}} f_{\text{CL}}(\mathbf{h}_j, \tilde{\mathbf{h}}_t)}\right], \quad \mathcal{N} = \{\mathbf{h}_{m,l} \mid M_m \in \mathcal{M}, l \in \{1, \ldots, L_m\}\}. \tag{6}$$

Here, minimizing $\mathcal{L}_{\text{CL}}$ effectively maximizes the mutual information between $\mathbf{h}_t$ and $\tilde{\mathbf{h}}_t$. A detailed description from the mutual information perspective is provided in **Appendix B.3**.

Incorporating the contrastive self-supervised mechanism, UniMerge employs an actor-critic algorithm to solve the state-enhanced MDP and update the associated network parameters. Specifically, the policy network is implemented as a Transformer-based decoder responsible for generating inference path policies. The value network, designed as a sequence evaluation encoder, estimates the value of the current state. These two networks are optimized jointly during training. In particular, the policy network is updated by minimizing the clipped PPO loss Schulman et al. (2017); Cao et al. (2025):

$$\mathcal{L}_{\text{policy}} = \mathbb{E}_t\left[\min\left(r_t(\boldsymbol{\theta}_{\text{policy}})\hat{A}_t, \text{clip}(r_t(\boldsymbol{\theta}_{\text{policy}}), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right], \tag{7}$$

where $r_t(\boldsymbol{\theta}_{\text{policy}}) = \frac{\pi_{\boldsymbol{\theta}_{\text{policy}}}(\mathbf{h}_t|\mathbf{h}_c)}{\pi_{\boldsymbol{\theta}^{\text{old}}_{\text{policy}}}(\mathbf{h}_t|\mathbf{h}_c)}$, $\hat{A}_t = R(\mathcal{F}) - V_{\boldsymbol{\phi}}(\mathbf{h}_c)$, with $V_{\boldsymbol{\phi}}(\mathbf{h}_c)$ the value function estimated by the value network at state $\mathbf{h}_c$, and ϵ the policy clipping parameter. To effectively train the value

network, its loss is defined as: $\mathcal{L}_{\text{value}} = \mathbb{E}\left[\left(V_{\boldsymbol{\phi}}(\mathbf{h}_c) - R(\mathcal{F})\right)^2\right]$. Thus, the actor-critic loss is denoted as $\mathcal{L}_{\text{AC}} = \mathcal{L}_{\text{policy}} + c_1 \cdot \mathcal{L}_{\text{value}} - c_2 \cdot \mathcal{L}_{\text{entropy}}$, where $\mathcal{L}_{\text{entropy}}$ is an entropy regularization term encouraging exploration, and c_1, c_2 are hyperparameters to balance the importance of each loss term. Finally, the overall training objective is formulated as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{AC}} + c_3 \mathcal{L}_{\text{CL}}, \tag{8}$$

where c_3 controls the contribution of $\mathcal{L}_{\text{CL}}$.

3.2 How to Merge Layers Together

To answer ***how to merge layers together***, UniMerge designs a lightweight glue layer strategy between adjacent layers to achieve compatibility and semantic continuity in the hidden space, which serves two main purposes: (i) to map the output of the preceding layer into the target layer's expected representation space, alleviating structural and distributional discrepancies; (ii) to transfer and align knowledge by learning the semantic behavior of the target layer in the source model. Inspired by knowledge distillation, UniMerge extends this concept to the layer-level alignment scenario, encouraging the glue layer's output to closely approximate the real input and the processed output of the corresponding layer in the target model, preserving information flow across layer junctions.

Concretely, consider the i-th source model M_i with its j-th layer denoted by $Q_{i,j}$. For two adjacent layers Q_l^{merged} and Q_{l+1}^{merged} in the merged model M_{merged}, originating from source models $M_a, M_b \in \mathcal{M}$ respectively, let the output of layer l in M_{merged} be $\mathbf{z}_l = \mathbf{u}_a^{\text{source}}$, and $\mathbf{u}_b^{\text{source}}$ be the original input to the target layer in M_b. We denote the nonlinear transformation of the target layer as $\ell_b(\cdot)$ and the glue layer inserted between the two layers as $\mathcal{G}_l$. To ensure consistency in semantic behavior and feature representation along the merging path, we design a lightweight dual distillation loss as follows: (i) Inter-layer semantic alignment distillation loss: to ensure that the glue layer output after the target layer can reproduce the semantic expression of the original model:

$$\mathcal{L}_{\text{inter}} = \mathbb{E}\left[\left\|\ell_b\left(\mathcal{G}_l(\mathbf{z}_l)\right) - \ell_b\left(\mathbf{u}_b^{\text{source}}\right)\right\|_2^2\right], \tag{9}$$

(ii) Cross-layer feature representation alignment distillation loss: to directly minimize the representation distance between the glue layer output and the target layer's original input:

$$\mathcal{L}_{\text{cross}} = \mathbb{E}\left[\left\|\mathcal{G}_l(\mathbf{z}_l) - \mathbf{u}_b^{\text{source}}\right\|_2^2\right], \tag{10}$$

Thus, the final distillation loss is defined as:

$$\mathcal{L}_{\text{glue}} = \lambda \cdot \mathcal{L}_{\text{inter}} + (1 - \lambda) \cdot \mathcal{L}_{\text{cross}}, \tag{11}$$

where $\lambda \in [0, 1]$ balances semantic and feature alignment.

Theoretical Analysis of the Glue Layer Strategy Since the original training data of candidate models is generally inaccessible in the model merging, UniMerge trains the glue layer using synthetic Gaussian hidden representations rather than original training data or downstream task supervision. Specifically, we sample synthetic calibration representations from simple Gaussian priors with dimensions matching the corresponding layer interfaces. These synthetic representations are forwarded through the selected source layer and the target layer to obtain paired calibration representations $\{(\mathbf{z}_l^{(s)}, \mathbf{u}_b^{\text{source},(s)})\}_{s=1}^S$, where $\mathbf{z}_l^{(s)}$ denotes the output of the preceding selected layer and $\mathbf{u}_b^{\text{source},(s)}$ denotes the reference input representation of the target layer in its source model.

We interpret the glue layer as a data-free representation transport map. Let $\hat{P}_{z_l}$ be the empirical distribution induced by $\{\mathbf{z}_l^{(s)}\}_{s=1}^S$, and let $\hat{P}_{u_b}$ be the empirical distribution induced by $\{\mathbf{u}_b^{\text{source},(s)}\}_{s=1}^S$. The glue layer $\mathcal{G}_l$ pushes $\hat{P}_{z_l}$ forward to $\mathcal{G}_{l\#}\hat{P}_{z_l}$, where $\mathcal{G}_{l\#}\hat{P}_{z_l}$ denotes the push-forward distribution induced by $\mathcal{G}_l$. Following optimal transport theory Villani (2009); Peyré & Cuturi (2019), the cross-layer feature alignment loss can be viewed as an empirical transport cost between the transformed preceding-layer representation distribution and the target-layer input distribution.

Proposition 1 (Glue-layer transport bound)**.** *The cross-layer feature alignment loss upper bounds the squared Wasserstein-2 distance between the transformed preceding-layer representation distribution and the target-layer input representation distribution:*

$$W_2^2\left(\mathcal{G}_{l\#}\hat{P}_{z_l}, \hat{P}_{u_b}\right) \leq \mathcal{L}_{\text{cross}}. \tag{12}$$

Proposition 1 follows from the definition of the Wasserstein-2 distance as the minimum transport cost over all valid couplings. The paired synthetic calibration representations define one feasible empirical coupling, whose cost is exactly $\mathcal{L}_{\text{cross}}$. Therefore, minimizing $\mathcal{L}_{\text{cross}}$ reduces an empirical upper bound on the interface-level representation transport discrepancy.

Proposition 2 (Semantic push-forward transport bound)**.** *The inter-layer semantic alignment loss upper bounds the squared Wasserstein-2 distance between the empirical semantic push-forward distributions induced by the stitched target layer and the original target layer:*

$$W_2^2\left((\ell_b \circ \mathcal{G}_l)_{\#}\hat{P}_{z_l}, (\ell_b)_{\#}\hat{P}_{u_b}\right) \leq \mathcal{L}_{\text{inter}}. \tag{13}$$

where $(\ell_b \circ \mathcal{G}_l)_{\#}\hat{P}_{z_l}$ denotes the semantic push-forward distribution induced by first applying the glue layer and then the target-layer transformation, and $(\ell_b)_{\#}\hat{P}_{u_b}$ denotes the semantic distribution induced by the original target layer. Proposition 2 indicates that $\mathcal{L}_{\text{inter}}$ aligns the semantic behavior after the target-layer transformation $\ell_b(\cdot)$.

Corollary 1 (Stitching error control)**.** *Let T_{tail} denote the remaining network after the target layer. If T_{tail} is L_{tail}-Lipschitz continuous, then the output perturbation caused by layer stitching is bounded by the inter-layer semantic alignment loss:*

$$\mathbb{E}\left[\left\|T_{\text{tail}}\left(\ell_b(\mathcal{G}_l(\mathbf{z}_l))\right) - T_{\text{tail}}\left(\ell_b(\mathbf{u}_b^{\text{source}})\right)\right\|_2^2\right] \leq L_{\text{tail}}^2 \mathcal{L}_{\text{inter}}. \tag{14}$$

Corollary 1 shows that $\mathcal{L}_{\text{inter}}$ controls the downstream propagation of stitching errors. The proofs of Proposition 1, Proposition 2, and Corollary 1 are provided in **Appendix B.4**.

4 Experiments

Experimental Setup We compared UniMerge against the following baselines. The weight-based methods included Fisher Merging Matena & Raffel (2022), RegMean Merging Jin et al. (2023), Weight Averaging Wortsman et al. (2022), and Task Arithmetic Ilharco et al. (2023). The classical subspace-based methods comprised DARE Merging Yu et al. (2024), Ties Merging Yadav et al. (2023), and DARE-Ties Merging. The state-of-the-art subspace-based methods were PCB Merging Du et al. (2024) and Consensus Merging Wang et al. (2024). Finally, the architecture-level methods included Franken Merging Goddard et al. (2024), EvoMerge Akiba et al. (2025), and HM3 Zhou et al. (2025). Additionally, the SVD-related methods including KnOTS-Ties Stoica et al. (2025), RobustMerge Zeng et al. (2025), and TSV-Merge Gargiulo et al. (2025) under their reported ViT configurations.

For datasets and backbones, following HM3, we use GLUE for general understanding; WMT14, WMT16, and IWSLT17, denoted as WMT&IWS, together with XNLI for translation; GSM8K and MathQA for mathematical reasoning; and HumanEval and MBPP for code generation. Qwen2.5-1.5B, Llama-2-7B, and Llama-2-13B are considered. For vision tasks, following Task Arithmetic, we evaluated eight image classification datasets: SUN397, Cars196, RESISC45, EuroSAT, SVHN, GTSRB, MNIST, and DTD. Accuracy was used as the metric. We leveraged two variants of CLIP's Vision Transformers Radford et al. (2021): ViT-B/32 and ViT-L/14. Detailed descriptions and references for all datasets, evaluation metrics, and experimental settings are provided in **Appendix C**.

Merging Language Models. We evaluate UniMerge across three LLMs and four task domains. The results in Table 1 (Detailed results are provided in **Tables 3**, **4**, and **5** in **Appendix D.1**) show that UniMerge achieves the best performance across all evaluated LLMs, benefiting from state-enhanced representations and contrastive actor-critic search. Weight-based methods underperform for parameter aggregation and do not handle heterogeneity. Subspace-based methods improve performance by reducing parameter interference, but remain unstable. Architecture-level methods offer greater structural flexibility, but they often fail to identify robust inference paths in high-dimensional spaces.

Merging Vision Models The results in Table 1 show that UniMerge achieves the highest average accuracy among the merging methods on both ViT-B/32 and ViT-L/14. Detailed results are provided in Tables 6 and 7 of **Appendix D.2**. These results indicate that its state-enhanced layer selection and glue-layer alignment generalize to vision backbones.

Table 1: Average performance across language and vision model merging tasks.

Method	Llama-2-7B	Qwen2.5-1.5B	Llama-2-13B	ViT-B/32	ViT-L/14	Avg.
Pretrained	19.60	22.12	22.87	48.06	64.49	35.43
Individual	44.51	42.69	48.00	90.51	94.15	63.97
Fisher Merging	28.68	22.32	29.14	68.34	82.19	46.13
RegMean Merging	28.56	23.37	28.72	71.78	83.71	47.23
Weight Average	23.86	23.25	23.91	65.80	79.63	43.29
Task Arithmetic	27.30	28.23	29.32	69.16	84.48	47.70
DARE Merging	29.92	28.33	30.09	69.62	84.71	48.53
Ties Merging	31.21	27.84	31.39	72.89	85.99	49.86
DARE-Ties Merging	32.52	28.92	34.20	73.53	85.75	50.98
Consensus Merging	37.18	33.26	35.00	72.78	84.71	52.59
PCB Merging	37.25	34.29	39.04	73.81	86.25	54.13
Franken Merging	17.44	19.93	16.94	38.79	54.35	29.49
EvoMerge	25.15	23.81	19.09	59.45	69.48	39.40
HM3	42.13	38.82	41.92	71.70	84.45	55.80
UniMerge	**43.65**	**40.40**	**43.01**	**74.02**	**86.45**	**57.51**

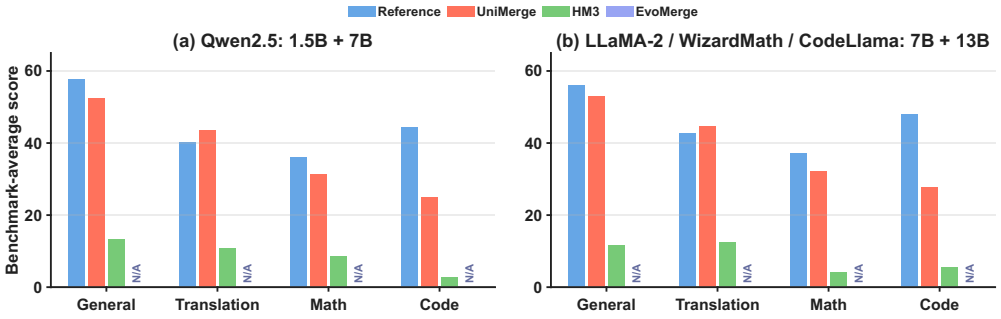


Figure 2: Cross-scale structural heterogeneity across four task scenarios. N/A denotes unavailable.

Extending Heterogeneity across Model Structures. We further evaluate UniMerge under cross-scale structural heterogeneity by stitching components from compatible models with different architectural scales. Qwen2.5-1.5B and Qwen2.5-7B pair introduces width heterogeneity, while the Llama-based 7B and 13B pairs for general, mathematical, and code tasks further introduce both width and depth heterogeneity. In Fig. 2, UniMerge maintains strong capability for general and translation tasks, and remains effective on math and code tasks, where existing methods largely fail. These results demonstrate that UniMerge extends architecture-level merging to explicit structural heterogeneity. Detailed results and analysis are provided in **Appendix D.3**.

Ablation Experiment We conduct ablation studies to examine the contributions of UniMerge's key components. In Table 2, removing SVD representation, State Encoder + CL, PPO search, or Glue training consistently degrades performance, with path search and cross-model alignment showing particularly important effects. The controlled comparison with HM3 further shows that replacing the connector or search framework alone provides limited gains, whereas incorporating more informative state representations leads to clearer improvements. These results indicate that UniMerge benefits from the joint effects of the designed components. Additional analyses of contrastive learning, Glue complementarity, and reward components are provided in **Appendix D.4**.

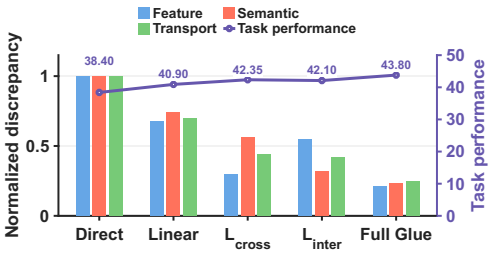


Figure 3: Fixed-path Glue analysis.

Table 2: Process-level ablation and controlled comparison. The left block reports the average performance of LLM backbones, while the right block reports the average over all backbones.

Variant	SVD Representation	State Encoder + CL	PPO Search	Glue Training	LLM Avg.	Variant	State	Search	Connector	Overall Avg.
w/o SVD Representation	×	✓	✓	✓	41.17	HM3	Index-only	HM3	Original	55.80
w/o State Encoder + CL	✓	×	✓	✓	40.57	HM3 + UniMerge Glue	Index-only	HM3	UniMerge Glue	56.05
Replace PPO Search	✓	✓	×	✓	39.17	UniMerge w/ Index State	Index-only	UniMerge	UniMerge Glue	56.54
w/o Glue Training	✓	✓	✓	×	38.33	UniMerge w/ NAS Metadata	Architecture metadata	UniMerge	UniMerge Glue	57.22
Full UniMerge	✓	✓	✓	✓	**43.77**	**Full UniMerge**	**Spectral/ SVD**	**UniMerge**	**UniMerge Glue**	**57.51**

Glue Layer Effectiveness and Diagnostics. Figure 3 evaluates different connector designs under fixed inference paths and jointly reports task performance and alignment diagnostics. The full Glue layer achieves the strongest task performance while simultaneously reducing feature mismatch, semantic discrepancy, and representation transport discrepancy, showing that its dual alignment objective improves cross-model layer compatibility independently of path selection. Beyond Fig. 3, we further evaluate the Glue layer as a plug-in connector for existing works, and provide detailed connector-level comparisons under synthetic calibration. The detailed analysis is provided in **Appendix D.5**.

Selected Path Analysis Fig. 4 shows the source-model selection frequency across merged blocks on the Llama-2-7B backbone. The 32-layer inference path is grouped into 16 blocks, and each heatmap entry reports the normalized frequency over 10 random seeds. Compared with HM3, which exhibits scattered source selections, UniMerge reveals a clear depth-wise pattern, selecting the base model in early blocks, task-specific models in middle blocks, and parameter-merged candidates in later blocks. This indicates that the state-enhanced representation and contrastive actor-critic search enable more stable and meaningful inference-path construction. The detailed analysis is provided in **Appendix D.6**.

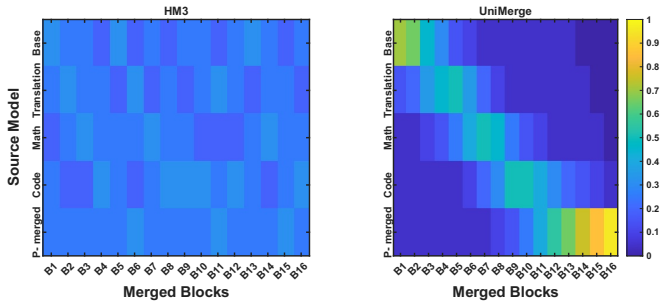


Figure 4: Selected inference path frequency, where P-Merged means parameter-level merged models and B1 is block 1.

Additional Experimental Results The search efficiency analysis in **Appendix D.7** shows that UniMerge achieves the best overall performance among existing methods under the same evaluation budget. The out-of-domain generalization analysis in **Appendix D.8** shows that UniMerge obtains the highest average performance on unseen tasks. The stability analysis in **Appendix D.9** further shows that UniMerge converges faster and reaches higher rewards across random seeds. Finally, **Appendix D.10** analyzes the sensitivity of the main representation, Glue, and search hyperparameters.

5 Conclusion

In this paper, we formulated heterogeneous architecture- and parameter-level model merging around two fundamental questions: which layers to select and how to merge them. To address the first, UniMerge formulates architecture-level merging as a state-enhanced MDP and employs a contrastive learning-assisted actor-critic algorithm for efficient layer selection. To address the second, it introduces a knowledge-distillation-based Glue layer to mitigate structural and distributional mismatches across layers. Experiments on language and vision models demonstrate strong performance, while cross-scale results further show that UniMerge can stitch compatible pretrained models with different widths and depths, extending architecture-level merging to structural heterogeneity. Future work will focus on improving efficiency and extending UniMerge to more challenging cross-family settings.

Reproducibility Statement All algorithmic details and experimental settings are provided in the paper and appendix. The source code will be released publicly after the review process to ensure full reproducibility.

REFERENCES

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, 7(2):195–204, 2025.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 2357–2367, 2019.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 37 (7):3896–3915, 2025.

Yuji Cao, Huan Zhao, Yuheng Cheng, Ting Shu, Yue Chen, Guolong Liu, Gaoqi Liang, Junhua Zhao, Jinyue Yan, and Yun Li. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Transactions on Neural Networks and Learning Systems*, 36(6):9737–9757, 2025.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuitho Sudoh, Koichiro Yoshino, and Christian Federmann. Overview of the iwslt 2017 evaluation campaign. In *Proceedings of the 14th International Workshop on Spoken Language Translation*, pp. 2–14, 2017.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

Haotian Chen, Yanyu Xu, Yonghui Xu, Yu Zhang, Yixin Zhang, Fang Wang, and Lizhen Cui. Toward effective model merging in semantic segmentation. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2025. doi: 10.1109/TNNLS.2025.3626606.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

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.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2475–2485, 2018.

Nico Daheim, Thomas Möllenhoff, Edoardo Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

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

Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim Kuan Goh, Ho-Kin Tang, Daojing He, et al. Parameter competition balancing for model merging. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 84746–84776, 2024.

Ronald A Fisher. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 222(594-604):309–368, 1922.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024.

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

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's mergekit: A toolkit for merging large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 477–485, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

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

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *Stat*, 1050:9, 2015.

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. EMR-MERGING: tuning-free high-performance model merging. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 122741–122769, 2024a.

Wenke Huang, Mang Ye, Zekun Shi, Guancheng Wan, He Li, Bo Du, and Qiang Yang. Federated learning for generalization, robustness, fairness: A survey and benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9387–9406, 2024b.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Gabriel Ilharco, Mitchell Wortsman, Josip Djolonga, Benoit Steiner, Pavel Izmailov, Max Welling, John Anderson, Simon Kornblith, Xiaohua Zhai, Mario Lucic, et al. Editing models with task arithmetic. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Simran Khanuja, Melvin Johnson, and Partha Talukdar. MergeDistill: Merging language models using pre-trained distillation. In *Findings of the Association for Computational Linguistics*, pp. 2874–2887, 2021.

Sanghoon Kim, Dahyun Kim, Chanjun Park, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 23–35, 2024.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *Proceedings of the IEEE Conference on Computer Vision Workshops*, pp. 554–561, 2013.

Yann LeCun. The mnist database of handwritten digits, 1998.

Bingdong Li, Zixiang Di, Yanting Yang, Hong Qian, Peng Yang, Hao Hao, Ke Tang, and Aimin Zhou. It's morphing time: Unleashing the potential of multiple llms via multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2025a. doi: 10.1109/TEVC.2025.3613937.

Hongkang Li, Yihua Zhang, Shuai Zhang, Pin-Yu Chen, Sijia Liu, and Meng Wang. When is task vector provably effective for model editing? A generalization analysis of nonlinear transformers. In *Proceedings of the 13th International Conference on Learning Representations*, 2025b.

Lu Li, Tianyu Zhang, Zhiqi Bu, Suyuchen Wang, Huan He, Jie Fu, Yonghui Wu, Jiang Bian, Yong Chen, and Yoshua Bengio. MAP: Low-compute model merging with amortized pareto fronts via quadratic approximation. In *Proceedings of the 13th International Conference on Learning Representations*, 2025c.

Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. Deep model fusion: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 37(5):2008–2024, 2026.

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: dynamic integration of modular expertise in model merging. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 78905–78935, 2024.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jian-Guang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, et al. WizardMath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. In *Proceedings of the 13th International Conference on Learning Representations*, 2025.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. WizardCoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations*, 2024.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 17703–17716, 2022.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: improved editing of pre-trained models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 66727–66754, 2023.

Gabriel Peyré and Marco Cuturi. *Computational optimal transport: With applications to data science*. Now Foundations and Trends, 2019.

Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 1267–1273, 2019.

Haiyun Qiu, Xingyu Wu, Ran Cheng, and Kay Chen Tan. Db-merging: Distribution-aware model merging for multi-task inference. In *Proceedings of the 2025 International Conference on Machine Intelligence and Nature-Inspired Computing*, pp. 179–180, 2025.

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 *Proceedings of the 38th International Conference on Machine Learning*, pp. 8748–8763, 2021.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, 2011.

Anna Rogers, Olga Kovaleva, Matthew Downey, and Anna Rumshisky. Getting closer to AI complete question answering: A set of prerequisite real tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8722–8731, 2020.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pp. 4463–4473, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Rico Sennrich, Alexandra Birch, and Barry Haddow. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation (WMT16)*, pp. 371–376, 2016.

Li Shen, Anke Tang, Enneng Yang, Guibing Guo, Yong Luo, Lefei Zhang, Xiaochun Cao, Bo Du, and Dacheng Tao. Efficient and effective weight-ensembling mixture of experts for multi-task model merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–13, 2025. doi: 10.1109/TPAMI.2025.3629605.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *Proceedings of the International Joint Conference on Neural Networks*, pp. 1453–1460, 2011.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with SVD to tie the KnOTS. In *International Conference on Learning Representations*, 2025.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pp. 4323–4332, 2019.

Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. Merging multi-task models via weight-ensembling mixture of experts. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 47778–47799, 2024.

Anke Tang, Li Shen, Yong Luo, Shuai Xie, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. Zero-shot sparse mixture of low-rank experts construction from pre-trained foundation models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(2):1145–1157, 2026.

Qwen Team. Qwen2.5: A party of foundation models, September 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Cédric Villani. *Optimal Transport: Old and New*. Springer, 2009.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. Knowledge fusion of large language models. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Fanqi Wan, Longguang Zhong, Ziyi Yang, Ruijun Chen, and Xiaojun Quan. Fusechat: Knowledge fusion of chat models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21629–21653, 2025.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jiménez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 50268–50287, 2024.

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

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 *Proceedings of the 39th International Conference on Machine Learning*, pp. 23965–23998, 2022.

Xingyu Wu, Yan Zhong, Jibin Wu, Bingbing Jiang, and Kay Chen Tan. Large language model-enhanced algorithm selection: Towards comprehensive algorithm representation. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence*, pp. 5235–5244, 2024.

Xingyu Wu, Jibin Wu, Yu Zhou, Liang Feng, and Kay Chen Tan. Towards robustness and explainability of automatic algorithm selection. In *Proceedings of the 42rd International Conference on Machine Learning*, 2025a.

Xingyu Wu, Sheng-Hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. Evolutionary computation in the era of large language model: Survey and roadmap. *IEEE Transactions on Evolutionary Computation*, 29(2):534–554, 2025b.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492, 2010.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *Proceedings of the 12th International Conference on Learning Representations*, 2024a.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024b.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: resolving interference when merging models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 7093–7115, 2023.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024a.

Chuanpeng Yang, Yao Zhu, Wang Lu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. Survey on knowledge distillation for large language models: Methods, evaluation, and application. *ACM Transactions on Intelligent Systems and Technology*, pp. 3699518, 2024b.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024c.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *Proceedings of the 12th International Conference on Learning Representations*, 2024d.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 57755–57775, 2024.

Zhongju Yuan, Genghui Li, Zhenkun Wang, Jianyong Sun, and Ran Cheng. Rl-csl: A combinatorial optimization method using reinforcement learning and contrastive self-supervised learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(4):1010–1024, 2022.

Netzer Yuval. Reading digits in natural images with unsupervised feature learning. In *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Fanhu Zeng, Haiyang Guo, Fei Zhu, Li Shen, and Hao Tang. RobustMerge: Parameter-efficient model merging for MLLMs with direction robustness. In *Advances in Neural Information Processing Systems*, volume 38, pp. 71071–71095, 2025.

Yu Zhou, Xingyu Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. HM3: Hierarchical multi-objective model merging for pretrained models. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*, 2025.

Didi Zhu, Zhongyi Sun, Zexi Li, Tao Shen, Ke Yan, Shouhong Ding, Chao Wu, and Kun Kuang. Model tailor: mitigating catastrophic forgetting in multi-modal large language models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 62581–62598, 2024.

APPENDIX OF UNIFIED MERGING OF HETEROGENEOUS PRETRAINED MODELS: FROM LAYER SELECTION TO SEMANTIC ALIGNMENT

APPENDIX CONTENTS

A Detailed Related Work **17**

A.1 Model Merging in Parameter Perspective 17

A.2 Model Merging in Architecture Perspective 17

A.3 Knowledge Distillation in Large Pretrained Models 18

B Detailed Methods **18**

B.1 Detailed Generation Process of $\boldsymbol{h}_{m,l}$ 18

B.2 Detailed Modeling of the Policy Network 19

B.3 Effectiveness Analysis of Contrastive Learning: Mutual Information Perspective . . 19

B.4 Theoretical Analysis of the Glue Layer 20

C Detailed Experimental Setup **23**

C.1 Language Models 24

C.2 Vision Models 26

C.3 Implementation Details 27

D Detailed Experimental Results **27**

D.1 Detailed Experiments of Merging Language Models 28

D.2 Detailed Experiments of Merging Vision Models 28

D.3 Extending Heterogeneous Merging to Different Model Widths and Depths 29

D.4 Detailed Ablation Analysis 31

D.5 Glue Layer Effectiveness and Fixed-Path Diagnostics 32

D.6 Selected Path Analysis 33

D.7 Search Efficiency Analysis 34

D.8 Out-of-Domain Generalization Analysis 35

D.9 Stability Analysis 35

D.10 Hyperparameter Sensitivity 36

E Discussion **36**

F Statement about usage of LLMs **37**

A Detailed Related Work

This section reviews existing studies on model merging from parameter and architecture perspectives, and introduces the background of knowledge distillation.

A.1 Model Merging in Parameter Perspective

Model merging Li et al. (2026), also known as model fusion, is a technique that integrates parameters from multiple pretrained models to build a unified model without original training data and high computational cost. It has become an important paradigm for model reuse Chen et al. (2025), and existing model merging methods can be broadly classified into three categories Yang et al. (2024c).

The first category is weighted-based methods. Originating from the model averaging Huang et al. (2024b), these methods compute a weighted average for the parameters of multiple task-specific models via $\mathbf{\Theta}_{\text{merge}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{\Theta}_k$. However, such simple and straightforward averaging often fails to achieve satisfactory performance. Subsequent works have aimed to improve the performance by introducing Fisher information Matena & Raffel (2022) or uncertainty estimation Daheim et al. (2024). In the large pretrained model scenario, Task Arithmetic Ilharco et al. (2023) introduced the task vector, which is defined as the parameter differences between finetuned models and the pretrained one, denoted as $\boldsymbol{\tau}_k = \mathbf{\Theta}_k - \mathbf{\Theta}_{\text{base}}$. Here, $\mathbf{\Theta}_{\text{base}}$ is the pretrained model and $\mathbf{\Theta}_k$ is the finetuned model for the k-th task $\mathcal{T}_k$. Further studies from tangent space Ortiz-Jimenez et al. (2023), and nonlinear transformer perspectives Li et al. (2025b) have verified the effectiveness of the task vector. Advanced weighted-based methods Yang et al. (2024d) mainly focus on designing more fine-grained weight coefficients to balance preserving the performance of each task while enhancing generalization. The second category is subspace-based methods Davari & Belilovsky (2024); Zhu et al. (2024). These methods exploit parameter redundancy by projecting parameters of large pretrained models into low-dimensional subspaces via sparsification or masking, and then merging parameters within those subspaces to alleviate multi-task interference. Representative subspace-based methods are Ties Merging Yadav et al. (2023) and DARE Yu et al. (2024), which employ parameter pruning, sign disambiguation, and rescaling mechanisms to reduce merging error in subspaces. The third category is router-based methods Tang et al. (2024); Shen et al. (2025), which combine mixture-of-experts (MoE) mechanisms Cai et al. (2025); Tang et al. (2026); Qiu et al. (2025) to dynamically select merging strategies based on input or task types during inference. For example, EMR Merging Huang et al. (2024a) and Twin Merging Lu et al. (2024) utilize routing networks to activate only the top-k experts, thereby avoiding the evaluation of all experts and improving the trade-off between efficiency and performance.

However, these categories merge pretrained models at the parameter level within the same model architecture, which limits the scalability and performance of the merged model, especially in heterogeneous scenarios. Recent works Wang et al. (2025); Wan et al. (2024; 2025) attempt to distill heterogeneous large language models (LLMs) into a unified target model and then perform parameter-level merging. Although this paradigm is feasible, it is constrained by the choice of the target model and remains a parameter-level merging approach. Moreover, finetuned models for different tasks under the same architecture are not always readily accessible, which restricts their practical deployment.

A.2 Model Merging in Architecture Perspective

Compared with parameter-level merging, architecture-level merging focuses on adjusting model structures to fully exploit the potential of different architectures in representation and generalization. From the perspective of the scaling law, the combination and extension of architectures may break the performance ceiling of single models. The work Kim et al. (2024) has supported this hypothesis by an extension from the 32-layer Mixtral-7B model to the 48-layer SOLAR 10.7B via layer concatenation. However, after layer concatenation, this method still requires additional continuous pretraining, which increases computational costs and conflicts with the training-free principle of model merging. Furthermore, some surveys Li et al. (2026) consider architecture-level merging as automatic model configuration, where search algorithms are designed to configure merging paths for large pretrained models. However, some methods, such as MAP Li et al. (2025c) and MM-MO Li et al. (2025a), focus primarily on configuring model parameters rather than optimizing model architecture. Other

approaches jointly optimize parameters and architecture. For example, EvoMerge Akiba et al. (2025) employs evolutionary search to discover cross-domain and high-performance merging strategies, which explores both parameter and data flow spaces for architecture path search. HM3 Zhou et al. (2025) formulates the architecture optimization problem as an MDP for inference path search and solves it using RL.

The above methods are still in the early stages of architecture path search and lack a clear and unified problem definition. They often perform insufficient exploration or simple layer concatenation, which neglects inter-layer information and dependencies, resulting in low search efficiency. Additionally, architecture-level merging faces another challenge of layer incompatibility or misalignment. Directly concatenating such layers can lead to severe feature misalignment and representation drift, thereby disrupting the stability of the information flow. For this problem, existing methods Akiba et al. (2025); Zhou et al. (2025) rely on heuristic designs, lacking systematic analysis of layer alignment and broad generalization validation. In this paper, we propose a unified framework across parameter and architecture spaces, which captures parameter relationships between layers to guide efficient architecture search. To address layer incompatibilities, we design a knowledge distillation-based approach, whose related works are introduced in the following subsection.

A.3 Knowledge Distillation in Large Pretrained Models

Knowledge distillation, proposed by Hinton et al. (2015), is an important approach for model compression, which aims to train a student model under the guidance of a teacher model. In the natural language processing field, knowledge distillation has been widely applied in text classification and generation, including training student models by mimicking the teacher's output probability distributions Xu et al. (2024b), as well as distilling feature or relational structure information from the teacher's intermediate layers Sun et al. (2019); Gu et al. (2024); Xu et al. (2024a). Generally, standard distillation strategies guide students to replicate the teacher's predictive behavior by minimizing differences in the output layer. However, recent studies have indicated that relying solely on output alignment may be insufficient for transferring the teacher's knowledge. Consequently, research has explored alignment of intermediate representations Khanuja et al. (2021) to facilitate the transfer of deep semantic information. These methods demonstrate significant effectiveness in scenarios with strong task semantic continuity Yang et al. (2024b). In this paper, we adopt distillation techniques to distill knowledge both within and across layers of the model.

B Detailed Methods

This section provides detailed descriptions and theoretical analyses of the main methodological components in UniMerge. Specifically, Appendix B.1 presents the detailed generation process of the state-enhanced layer embedding $\mathbf{h}_{m,l}$, including parameter compression and attention-based representation modeling. Appendix B.2 describes the policy network used for inference-path generation and layer selection. Appendix B.3 provides the mutual-information interpretation of the proposed contrastive learning mechanism. Appendix B.4 further presents the theoretical analysis of the glue layer strategy, including the transport-bound interpretation and stitching-error control.

B.1 Detailed Generation Process of $\boldsymbol{h}_{m,l}$

The detailed generation process of $\mathbf{h}_{m,l}$ is as follows. The embedding vector for the i-th layer is defined as $\mathbf{h}_i = [m, l, f_{\text{SVD}}(\mathbf{W}_{m,l})]$. An attention mechanism then aggregates information from other candidate layers to enhance this embedding. For the i-th layer, its key $\mathbf{k}_i$, query $\mathbf{q}_i$, and value $\mathbf{v}_i$ are computed as $\mathbf{k}_i = \mathbf{W}_K\mathbf{h}_i, \mathbf{q}_i = \mathbf{W}_Q\mathbf{h}_i, \mathbf{v}_i = \mathbf{W}_V\mathbf{h}_i$, where $\mathbf{W}_K \in \mathbb{R}^{d_K \times d_h}$, $\mathbf{W}_Q \in \mathbb{R}^{d_Q \times d_h}$, and $\mathbf{W}_V \in \mathbb{R}^{d_V \times d_h}$ are learnable projection matrices. The compatibility between the i-th and j-th layers is defined as:

$$\text{Comp}_{i,j} = \begin{cases} \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_K}}, & \text{if } i \text{ connects with } j, \\ -\infty, & \text{otherwise}, \end{cases} \tag{15}$$

where d_K is the dimension of the key vectors. Then, the attention weight of the i-th layer on the j-th layer is: $a_{i,j} = \frac{\exp(\text{Comp}_{i,j})}{\sum_{j'=1}^{n} \exp(\text{Comp}_{i,j'})}$. The embedding of $\mathbf{h}_i$ after the attention mechanism is

given by $\mathbf{h}'_i = \sum_{j=1}^{n} a_{i,j}\mathbf{v}_j$. Using multi-head attention (MHA) with B heads, the dimensions of the key and value are $d_K = d_V = d_h/B$. The multi-head embedding for the i-th candidate layer is computed as $\mathbf{h}''_i = \mathbf{W}_O[\mathbf{h}'_{i,1}, \ldots, \mathbf{h}'_{i,B}]$, where $\mathbf{h}'_{i,b}$ is the embedding of the i-th layer on the b-th head, and $\mathbf{W}_O \in \mathbb{R}^{d_h \times d_h}$ is the output projection matrix. By adding residual connections and layer normalization, the output embedding of the i-th layer after the MHA layer is $\mathbf{h}_i^{\mathrm{MHA}} = \mathrm{LN}(\mathbf{h}_i + \mathbf{h}''_i)$. Finally, the encoder output is further processed by a feed-forward network with residual connections to produce the structure-aware embedding for each layer:

$$\mathbf{h}_i^{\mathrm{FF}} = \mathrm{LN}(\mathbf{h}_i^{\mathrm{MHA}} + \mathrm{FF}(\mathbf{h}_i^{\mathrm{MHA}})), \tag{16}$$

where MHA denotes multi-head attention, FF denotes the feed-forward sub-layer, and LN represents layer normalization. Thus, for the l-th layer of the m-th model, its state-enhanced embedding vector is:

$$\mathbf{h}_{m,l} = \mathbf{h}_i^{\mathrm{FF}}. \tag{17}$$

B.2 Detailed Modeling of the Policy Network

At the t-th step, the policy network constructs the context representation $\mathbf{h}_c$ based on the previously selected embeddings, which is defined as:

$$\mathbf{h}_c = \begin{cases} [\bar{\mathbf{h}}_{t-1}, \mathbf{h}_1, \mathbf{h}_{t-1}], & t \geq 2, \\ [\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_f], & t = 1, \end{cases} \tag{18}$$

where $\bar{\mathbf{h}}_{t-1} = \frac{1}{t-1}\sum_{i=1}^{t-1}\mathbf{h}_i$ is the mean embedding of the historical path, $\mathbf{h}_1$ and $\mathbf{h}_{t-1}$ are the embeddings of the initial and most recently selected layers, respectively, and $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_f$ are learnable vectors initializing the context.

Based on $\mathbf{h}_c$, the policy network is constructed to select one candidate layer at each step, and then form a layer combination inference path $\mathcal{F} = \{f_{m_t,l_t}\}_{t=1}^{T}$ with corresponding embedding sequence $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T)$. The process of the inference path search via the policy network is defined as:

$$\pi_{\boldsymbol{\theta}_{\mathrm{policy}}}(\mathbf{H} \mid \mathcal{S}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}_{\mathrm{policy}}}(\mathbf{h}_t \mid \mathcal{S}, \mathbf{h}_1, \ldots, \mathbf{h}_{t-1}), \tag{19}$$

where $\boldsymbol{\theta}_{\mathrm{policy}}$ denotes the parameters of the policy network. Since (19) is a sequence model, the policy network is built as a transformer decoder Yuan et al. (2022). Based on the policy network, the action generation process conditions on $\mathbf{h}_c$, generates query vectors via $\mathbf{q}_c = \mathbf{W}_Q\mathbf{h}_c$, scores candidate layers via attention mechanisms, and samples the final candidate layer from the probability distribution. Here, the scoring function for $\mathbf{h}_i$ is defined as:

$$u_i^{(c)} = \begin{cases} \beta \cdot \tanh\left(\frac{\mathbf{q}_c^\top \mathbf{z}_i}{\sqrt{d}}\right), & \text{if } \mathbf{z}_i \text{ has not been selected,} \\ -\infty, & \text{otherwise.} \end{cases} \tag{20}$$

The selection probability for each layer is obtained by softmax normalization:

$$p_{\boldsymbol{\theta}_{\mathrm{policy}}}(\mathbf{h}_t = \mathbf{h}_i \mid \mathcal{S}, \mathbf{h}_{1:t-1}) = \frac{\exp(u_i^{(c)})}{\sum_j \exp(u_j^{(c)})}. \tag{21}$$

Finally, the candidate layer to be selected at the t-th step is determined either by greedy selection $\mathbf{h}_t = \arg\max_{i\in\{1,\ldots,n\}} p_{\boldsymbol{\theta}_{\mathrm{policy}}}(\mathbf{h}_t = \mathbf{h}_i \mid \mathcal{S}, \mathbf{h}_{1:t-1})$ or by sampling from this probability distribution. In this paper, UniMerge samples from the distribution to select the final layer.

B.3 Effectiveness Analysis of Contrastive Learning: Mutual Information Perspective

Here, minimizing $\mathcal{L}_{\mathrm{CL}}$ effectively maximizes the mutual information between $\mathbf{h}_t$ and $\tilde{\mathbf{h}}_t$, which can be provided as

$$\begin{aligned} I(\mathbf{h}_t; \tilde{\mathbf{h}}_t) &= \sum p(\mathbf{h}_t, \tilde{\mathbf{h}}_t) \log\left(\frac{p(\mathbf{h}_t \mid \tilde{\mathbf{h}}_t)}{p(\mathbf{h}_t)}\right) \\ &= \sum p(\mathbf{h}_t, \tilde{\mathbf{h}}_t) \log f_{\mathrm{CL}}(\mathbf{h}_t, \tilde{\mathbf{h}}_t), \end{aligned} \tag{22}$$

where $f_{\mathrm{CL}}(\mathbf{h}_t, \tilde{\mathbf{h}}_t)$ denotes the density ratio. $\mathcal{L}_{\mathrm{CL}}$ can be decomposed as follows:

$$\begin{aligned}
\mathcal{L}_{\mathrm{CL}} &= -\mathbb{E}\log\left[\frac{\frac{p(\mathbf{h}_t|\tilde{\mathbf{h}}_t)}{p(\mathbf{h}_t)}}{\frac{p(\mathbf{h}_t|\tilde{\mathbf{h}}_t)}{p(\mathbf{h}_t)} + \sum_{\mathbf{h}_j\in\mathcal{H}^-}\frac{p(\mathbf{h}_j|\tilde{\mathbf{h}}_t)}{p(\mathbf{h}_j)}}\right] \\
&= \mathbb{E}\log\left[1 + \frac{p(\mathbf{h}_t)}{p(\mathbf{h}_t \mid \tilde{\mathbf{h}}_t)}\sum_{\mathbf{h}_j\in\mathcal{H}^-}\frac{p(\mathbf{h}_j \mid \tilde{\mathbf{h}}_t)}{p(\mathbf{h}_j)}\right] \\
&\approx \mathbb{E}\log\left[1 + \frac{p(\mathbf{h}_t)}{p(\mathbf{h}_t \mid \tilde{\mathbf{h}}_t)}(N-1)\mathbb{E}_{\mathbf{h}_j}\left[\frac{p(\mathbf{h}_j \mid \tilde{\mathbf{h}}_t)}{p(\mathbf{h}_j)}\right]\right].
\end{aligned}$$

$$\begin{aligned}
\mathcal{L}_{\mathrm{CL}} &= \mathbb{E}\log\left[1 + \frac{p(\mathbf{h}_t)}{p(\mathbf{h}_t \mid \tilde{\mathbf{h}}_t)}(N-1)\right] \\
&\geq \mathbb{E}\log\left[\frac{p(\mathbf{h}_t)}{p(\mathbf{h}_t \mid \tilde{\mathbf{h}}_t)}N\right] \\
&= \mathbb{E}\left[-\log\frac{p(\mathbf{h}_t \mid \tilde{\mathbf{h}}_t)}{p(\mathbf{h}_t)} + \log N\right] \\
&= -\sum p(\mathbf{h}_t, \tilde{\mathbf{h}}_t)\log f_{\mathrm{CL}}(\mathbf{h}_t, \tilde{\mathbf{h}}_t) + \log N \\
&= -I(\mathbf{h}_t; \tilde{\mathbf{h}}_t) + \log N. \qquad (23)
\end{aligned}$$

Therefore, the mutual information satisfies

$$I(\mathbf{h}_t; \tilde{\mathbf{h}}_t) \geq \log N - \mathcal{L}_{\mathrm{CL}}. \qquad (24)$$

As N increases, $\mathcal{L}_{\mathrm{CL}}$ becomes a tighter and more accurate estimator.

B.4 Theoretical Analysis of the Glue Layer

The glue layer in UniMerge is designed to connect two adjacent layers that may originate from different source models. Its goal is not to learn a task-specific decision boundary, but to align the hidden-space interface and semantic behavior between the preceding selected layer and the target layer. However, in the model-merging setting, the original training data of the candidate models are generally unavailable due to privacy, storage, or licensing constraints. Therefore, UniMerge trains each glue layer using synthetic Gaussian hidden representations. This design is consistent with the model-merging assumption because it does not require original training samples or downstream task labels. Instead, the synthetic representations provide calibration signals that probe the local input-output behavior of the selected source and target layers.

Consider two adjacent layers Q_l^{merged} and $Q_{l+1}^{\mathrm{merged}}$ in the merged model. Suppose that Q_l^{merged} originates from source model M_a, while $Q_{l+1}^{\mathrm{merged}}$ originates from source model M_b. We use $\mathcal{G}_l$ to denote the glue layer inserted between them, and use $\ell_b(\cdot)$ to denote the nonlinear transformation of the target layer from M_b. For each calibration index $s = 1, \ldots, S$, UniMerge samples synthetic hidden representations from Gaussian priors and constructs the corresponding calibration representations as

$$\boldsymbol{\xi}_a^{(s)} \sim \mathcal{N}(\mathbf{0}, \sigma_a^2\mathbf{I}). \qquad (25)$$

$$\boldsymbol{\xi}_b^{(s)} \sim \mathcal{N}(\mathbf{0}, \sigma_b^2\mathbf{I}). \qquad (26)$$

$$\mathbf{z}_l^{(s)} = Q_l^{\mathrm{merged}}(\boldsymbol{\xi}_a^{(s)}). \qquad (27)$$

$$\mathbf{u}_b^{\mathrm{source},(s)} = \boldsymbol{\xi}_b^{(s)}. \qquad (28)$$

$$\mathbf{y}_b^{\mathrm{source},(s)} = \ell_b\left(\mathbf{u}_b^{\mathrm{source},(s)}\right). \tag{29}$$

where $\mathbf{z}_l^{(s)}$ denotes the output representation of the preceding selected layer, $\mathbf{u}_b^{\mathrm{source},(s)}$ denotes the reference input expected by the target layer, and $\mathbf{y}_b^{\mathrm{source},(s)}$ denotes the corresponding output after the target-layer transformation. The role of the glue layer is to transform $\mathbf{z}_l^{(s)}$ into the representation space expected by the target layer, while preserving the semantic behavior induced by $\ell_b(\cdot)$.

Synthetic empirical distributions. The synthetic calibration samples induce empirical representation distributions in the preceding-layer output space and the target-layer input space. Denote these empirical distributions by

$$\hat{P}_{z_l} = \frac{1}{S}\sum_{s=1}^{S}\delta_{\mathbf{z}_l^{(s)}}, \tag{30}$$

$$\hat{P}_{u_b} = \frac{1}{S}\sum_{s=1}^{S}\delta_{\mathbf{u}_b^{\mathrm{source},(s)}}. \tag{31}$$

where $\delta_{\mathbf{x}}$ is the Dirac measure at $\mathbf{x}$. For a measurable map $\mathcal{G}_l$, the notation $\mathcal{G}_{l\#}\hat{P}_{z_l}$ denotes the push-forward distribution of $\hat{P}_{z_l}$ induced by $\mathcal{G}_l$. That is, if $\mathbf{z}_l \sim \hat{P}_{z_l}$, then $\mathcal{G}_l(\mathbf{z}_l) \sim \mathcal{G}_{l\#}\hat{P}_{z_l}$. This push-forward notation allows us to view the glue layer as a representation transport map from the preceding-layer distribution to the target-layer input distribution.

Under the synthetic calibration set, the two glue-layer losses in the main paper can be written as

$$\mathcal{L}_{\mathrm{cross}} = \frac{1}{S}\sum_{s=1}^{S}\left\|\mathcal{G}_l(\mathbf{z}_l^{(s)}) - \mathbf{u}_b^{\mathrm{source},(s)}\right\|_2^2. \tag{32}$$

$$\mathcal{L}_{\mathrm{inter}} = \frac{1}{S}\sum_{s=1}^{S}\left\|\ell_b(\mathcal{G}_l(\mathbf{z}_l^{(s)})) - \ell_b(\mathbf{u}_b^{\mathrm{source},(s)})\right\|_2^2. \tag{33}$$

$$\mathcal{L}_{\mathrm{glue}} = \lambda\mathcal{L}_{\mathrm{inter}} + (1-\lambda)\mathcal{L}_{\mathrm{cross}}. \tag{34}$$

Therefore, $\mathcal{L}_{\mathrm{cross}}$ aligns the feature interface of the target layer, while $\mathcal{L}_{\mathrm{inter}}$ aligns the semantic behavior after the target-layer transformation.

Empirical coupling induced by synthetic calibration. We first show an auxiliary result: the paired synthetic calibration samples define a valid empirical coupling between the transformed preceding-layer distribution and the target-layer input distribution. This lemma will be used to prove Proposition 1 in the main paper.

Lemma 1 (Synthetic calibration induces an empirical coupling)**.** *The paired representations*

$$\left\{\left(\mathcal{G}_l(\mathbf{z}_l^{(s)}), \mathbf{u}_b^{\mathrm{source},(s)}\right)\right\}_{s=1}^{S} \tag{35}$$

define a valid empirical coupling between $\mathcal{G}_{l\#}\hat{P}_{z_l}$ and $\hat{P}_{u_b}$.

Proof. Define the empirical joint distribution

$$\hat{\gamma}_l = \frac{1}{S}\sum_{s=1}^{S}\delta_{\left(\mathcal{G}_l(\mathbf{z}_l^{(s)}), \mathbf{u}_b^{\mathrm{source},(s)}\right)}. \tag{36}$$

The first marginal of $\hat{\gamma}_l$ is

$$\frac{1}{S}\sum_{s=1}^{S}\delta_{\mathcal{G}_l(\mathbf{z}_l^{(s)})} = \mathcal{G}_{l\#}\hat{P}_{z_l}, \tag{37}$$

and the second marginal is

$$\frac{1}{S}\sum_{s=1}^{S}\delta_{\mathbf{u}_b^{\mathrm{source},(s)}} = \hat{P}_{u_b}. \tag{38}$$

Hence, $\hat{\gamma}_l$ is a valid coupling between $\mathcal{G}_{l\#}\hat{P}_{z_l}$ and $\hat{P}_{u_b}$. □

Proof of Proposition 1. We now prove the glue-layer transport bound stated in Proposition 1. The squared Wasserstein-2 distance between two distributions is defined as the minimum expected squared transport cost over all valid couplings. Since Lemma 1 shows that the synthetic calibration pairs provide one feasible empirical coupling, the cost induced by this coupling gives an upper bound on the optimal transport cost.

Proof. By the definition of the squared Wasserstein-2 distance Villani (2009); Peyré & Cuturi (2019), we have

$$\begin{aligned} W_2^2\left(\mathcal{G}_{l\#}\hat{P}_{z_l}, \hat{P}_{u_b}\right) &= \inf_{\pi\in\Pi(\mathcal{G}_{l\#}\hat{P}_{z_l}, \hat{P}_{u_b})} \mathbb{E}_{(\mathbf{x},\mathbf{u})\sim\pi}\left[\|\mathbf{x}-\mathbf{u}\|_2^2\right] \\ &\le \mathbb{E}_{(\mathbf{x},\mathbf{u})\sim\hat{\gamma}_l}\left[\|\mathbf{x}-\mathbf{u}\|_2^2\right] \\ &= \frac{1}{S}\sum_{s=1}^{S}\left\|\mathcal{G}_l(\mathbf{z}_l^{(s)}) - \mathbf{u}_b^{\mathrm{source},(s)}\right\|_2^2 \\ &= \mathcal{L}_{\mathrm{cross}}. \end{aligned} \tag{39}$$

The inequality holds because $\hat{\gamma}_l$ is one feasible coupling, whereas the Wasserstein distance takes the infimum over all feasible couplings. This completes the proof. □

This proposition shows that minimizing $\mathcal{L}_{\mathrm{cross}}$ reduces an empirical upper bound on the transport discrepancy between the transformed preceding-layer representation distribution and the target-layer input distribution. Thus, $\mathcal{L}_{\mathrm{cross}}$ can be interpreted as an interface-level representation transport objective.

Proof of Proposition 2. We now prove the semantic push-forward transport bound stated in Proposition 2. Feature-interface alignment alone may not fully preserve the nonlinear semantic behavior of the target layer. Therefore, UniMerge further introduces $\mathcal{L}_{\mathrm{inter}}$, which compares the outputs after applying the target-layer transformation $\ell_b(\cdot)$. This loss can also be interpreted through the push-forward distribution induced by the composed mapping $\ell_b \circ \mathcal{G}_l$.

Proof. The paired semantic representations

$$\left\{\left(\ell_b(\mathcal{G}_l(\mathbf{z}_l^{(s)})), \ell_b(\mathbf{u}_b^{\mathrm{source},(s)})\right)\right\}_{s=1}^{S} \tag{40}$$

define an empirical coupling between $(\ell_b\circ\mathcal{G}_l)_{\#}\hat{P}_{z_l}$ and $(\ell_b)_{\#}\hat{P}_{u_b}$. Using the definition of the squared Wasserstein-2 distance, we obtain

$$\begin{aligned} &W_2^2\left((\ell_b\circ\mathcal{G}_l)_{\#}\hat{P}_{z_l}, (\ell_b)_{\#}\hat{P}_{u_b}\right) \\ &\le \frac{1}{S}\sum_{s=1}^{S}\left\|\ell_b(\mathcal{G}_l(\mathbf{z}_l^{(s)})) - \ell_b(\mathbf{u}_b^{\mathrm{source},(s)})\right\|_2^2 = \mathcal{L}_{\mathrm{inter}}. \end{aligned} \tag{41}$$

This completes the proof. □

This proposition indicates that $\mathcal{L}_{\mathrm{inter}}$ aligns not only the target-layer input interface but also the semantic distribution after the nonlinear target-layer transformation. Consequently, the dual distillation objective combines interface-level compatibility and semantic-level continuity.

Proof of Corollary 1. We further prove the stitching error control result stated in Corollary 1, which shows that the inter-layer semantic loss controls the propagation of stitching errors to the remaining part of the merged model.

Proof. By the L_{tail}-Lipschitz continuity of T_{tail}, for any paired representations we have

$$\begin{aligned} &\left\|T_{\mathrm{tail}}\left(\ell_b(\mathcal{G}_l(\mathbf{z}_l))\right) - T_{\mathrm{tail}}\left(\ell_b(\mathbf{u}_b^{\mathrm{source}})\right)\right\|_2^2 \\ &\le L_{\mathrm{tail}}^2\left\|\ell_b(\mathcal{G}_l(\mathbf{z}_l)) - \ell_b(\mathbf{u}_b^{\mathrm{source}})\right\|_2^2. \end{aligned} \tag{42}$$

Taking expectation over the synthetic calibration distribution gives

$$\begin{aligned}&\mathbb{E}\left[\left\|T_{\text{tail}}\left(\ell_b(\mathcal{G}_l(\mathbf{z}_l))\right)-T_{\text{tail}}\left(\ell_b(\mathbf{u}_b^{\text{source}})\right)\right\|_2^2\right]\\&\quad\le L_{\text{tail}}^2\mathbb{E}\left[\left\|\ell_b(\mathcal{G}_l(\mathbf{z}_l))-\ell_b(\mathbf{u}_b^{\text{source}})\right\|_2^2\right]=L_{\text{tail}}^2\mathcal{L}_{\text{inter}}.\end{aligned}\tag{43}$$

This completes the proof. □

This corollary explains why reducing $\mathcal{L}_{\text{inter}}$ can improve the forward propagation stability of the merged model. If the semantic discrepancy immediately after the stitched target layer is small, then the discrepancy propagated through the remaining network is also controlled, up to the Lipschitz factor of the tail network.

Linear special case and SVD interpretation. We show that the proposed representation transport view recovers the classical SVD-based subspace alignment interpretation in the linear case. Assume the glue layer is linear:

$$\mathcal{G}_l(\mathbf{z})=\mathbf{W}_l\mathbf{z}.\tag{44}$$

Let

$$\mathbf{Z}_l=[\mathbf{z}_l^{(1)},\dots,\mathbf{z}_l^{(S)}],\qquad\mathbf{U}_b=[\mathbf{u}_b^{\text{source},(1)},\dots,\mathbf{u}_b^{\text{source},(S)}].\tag{45}$$

Then minimizing $\mathcal{L}_{\text{cross}}$ becomes a least-squares representation alignment problem:

$$\min_{\mathbf{W}_l}\|\mathbf{W}_l\mathbf{Z}_l-\mathbf{U}_b\|_F^2.\tag{46}$$

With ridge regularization, the objective becomes

$$\min_{\mathbf{W}_l}\|\mathbf{W}_l\mathbf{Z}_l-\mathbf{U}_b\|_F^2+\rho\|\mathbf{W}_l\|_F^2,\tag{47}$$

whose closed-form solution is

$$\mathbf{W}_l^*=\mathbf{U}_b\mathbf{Z}_l^\top\left(\mathbf{Z}_l\mathbf{Z}_l^\top+\rho\mathbf{I}\right)^{-1}.\tag{48}$$

If $\mathbf{W}_l$ is constrained to be orthogonal and the source and target representations have the same dimension, Eq. (46) reduces to the orthogonal Procrustes problem:

$$\min_{\mathbf{W}_l^\top\mathbf{W}_l=\mathbf{I}}\|\mathbf{W}_l\mathbf{Z}_l-\mathbf{U}_b\|_F^2.\tag{49}$$

Let the singular value decomposition of $\mathbf{U}_b\mathbf{Z}_l^\top$ be

$$\mathbf{U}_b\mathbf{Z}_l^\top=\mathbf{P}\mathbf{\Sigma}\mathbf{Q}^\top.\tag{50}$$

The optimal orthogonal mapping is then

$$\mathbf{W}_l^*=\mathbf{P}\mathbf{Q}^\top.\tag{51}$$

Therefore, in the linear case, the glue layer learns a transport map that rotates and rescales the preceding-layer representation subspace to align with the target-layer input subspace. This provides a geometric interpretation of the glue layer: it acts as a data-free representation transport module that simultaneously improves feature-interface compatibility and semantic continuity between adjacent heterogeneous layers.

C Detailed Experimental Setup

This subsection outlines the models used in our experiments, covering both language and vision domains, and details the hyperparameter configurations applied. The source code will be made publicly available after the review process.

C.1 LANGUAGE MODELS

The core objective of this study is to design and implement an efficient and multitask-adaptive model merging framework. To verify the generality and performance of the proposed merging method, we apply it to three popular large language model (LLM) series, namely the fine-tuned models based on Qwen2.5-1.5B Team (2024), Llama-2-7B Touvron et al. (2023) and Llama-2-13B. Specifically, for Llama-2-7B Touvron et al. (2023), we include three fine-tuned models: Llama-2-7B-Chat for text translation Touvron et al. (2023), WizardMath-7B for mathematical reasoning Luo et al. (2025), and CodeLlama-7B for code generation Roziere et al. (2023). Similarly, for Qwen2.5-1.5B Team (2024), we include three fine-tuned models: Qwen2.5-1.5B-Instruct for text translation Team (2024), Qwen2.5-Coder-1.5B for code generation Hui et al. (2024), and Qwen2.5-Math-1.5B for mathematical reasoning Yang et al. (2024a). As for Llama-2-13B, we include three fine-tuned models: WizardLM-13B for text translation Xu et al. (2024a), WizardMath-13B for mathematical reasoning Luo et al. (2025), and WizardCoder-Python-13B for code generation Luo et al. (2024). These models are publicly available on the Hugging Face platform, with related links as follows: Llama-2-7B-Chat[1], WizardMath-7B[2], CodeLlama-7B[3], Llama-2-7B[4], Qwen2.5-1.5B[5], Qwen2.5-Coder-1.5B[6], Qwen2.5-Math-1.5B[7], Qwen2.5-1.5B-Instruct[8], Llama-2-13B[9], WizardLM-13B[10], WizardMath-13B[11] and WizardCoder-Python-13B[12].

Utilizing these fine-tuned LLMs, our proposed UniMerge method is applied to integrate high-performing models across multiple tasks. To comprehensively evaluate the performance of the final merged LLMs, we focus on four key task domains: generative task, text translation, mathematical reasoning, and code generation. For efficient and standardized assessment, we employ two widely used evaluation frameworks: lm-evaluation-harness[13] Gao et al. (2024) for generative tasks, language translation, and mathematical reasoning, and bigcode-evaluation-harness[14] for code generation tasks.

To further demonstrate the effectiveness and superiority of our method compared to other model merging approaches, for language models, we utilize the `mergekit` package Goddard et al. (2024) to merge models via multiple merging strategies, including the following:

- **Individual Models**: single-task models trained independently prior to merging.
- **Traditional Multi-Task Learning (MTL)**: a shared model trained jointly on all task datasets.
- **Weight Averaging**: element-wise averaging of multiple model weights, proven effective for merging fine-tuned models originating from the same pretrained base Wortsman et al. (2022).
- **Fisher Merging** Matena & Raffel (2022): uses the Fisher information matrix to evaluate parameter importance and performs weighted merging accordingly Fisher (1922).
- **RegMean** Jin et al. (2023): proposes a closed-form solution to merge multiple linear model weights by weighted averaging. Given K linear model weights $\mathbf{W}_i$ where the i-th model output is $f_i(\mathbf{x}) = \mathbf{W}_i^\top \mathbf{x}$, the merging problem is:

$$\min_{\mathbf{W}} \sum_{i=1}^{K} \|\mathbf{W}^\top \mathbf{X}_i - \mathbf{W}_i^\top \mathbf{X}_i\|_F^2, \tag{52}$$

[1] https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
[2] https://huggingface.co/WizardLMTeam/WizardMath-7B
[3] https://huggingface.co/codellama/CodeLlama-7b-hf
[4] https://huggingface.co/meta-llama/Llama-2-7b-hf
[5] https://huggingface.co/Qwen/Qwen2.5-1.5B
[6] https://huggingface.co/Qwen/Qwen2.5-Coder-1.5B
[7] https://huggingface.co/Qwen/Qwen2.5-Math-1.5B
[8] https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct
[9] https://huggingface.co/meta-llama/Llama-2-13b-hf
[10] https://huggingface.co/WizardLMTeam/WizardLM-13B-V1.2
[11] https://huggingface.co/vanillaOVO/WizardMath-13B-V1.0
[12] https://huggingface.co/WizardLMTeam/WizardCoder-Python-13B-V1.0
[13] https://github.com/EleutherAI/lm-evaluation-harness
[14] https://github.com/bigcode-project/bigcode-evaluation-harness

where $\mathbf{W}$ is the merged model weight and $\mathbf{X}_i$ is the input for the i-th model. The closed-form solution is:

$$\mathbf{W} = \left(\sum_{i=1}^{K} \mathbf{X}_i^\top \mathbf{X}_i \right)^{-1} \left(\sum_{i=1}^{K} \mathbf{X}_i^\top \mathbf{X}_i \mathbf{W}_i \right). \tag{53}$$

This method requires computing input inner product matrices prior to merging.

- **Task Arithmetic** Ilharco et al. (2023): defines task vectors as the difference between fine-tuned model weights and pretrained weights. For a model $\boldsymbol{\theta}_i$ fine-tuned from pretrained model $\boldsymbol{\theta}_{\text{pre}}$, the task vector is $\boldsymbol{\tau}_i = \boldsymbol{\theta}_i - \boldsymbol{\theta}_{\text{pre}}$. The merged model weight is:

$$\boldsymbol{\theta}_{\text{M}} = \lambda \sum_{i=1}^{K} \boldsymbol{\tau}_i + \boldsymbol{\theta}_{\text{pre}}, \tag{54}$$

where λ is the merging coefficient.

- **Ties Merging** Yadav et al. (2023): applies pruning (Trim), sign election (Elect Sign), and merging steps to resolve conflicts among task vectors, removing redundant parameters and resolving sign conflicts to improve merging performance.
- **DARE** Yu et al. (2024) (Drop and Rescale): exploits the high redundancy of parameters in language models by randomly dropping a large fraction (90% or even 99%) of incremental parameters (task vectors) before merging, aiming to reduce interference and improve merging results.
- **PCB Merging** Du et al. (2024): PCB Merging dynamically balances parameter competition both within individual tasks and across multiple tasks by adjusting scaling coefficients at the parameter level. It efficiently merges models without additional training by dropping less important parameters and rescaling the rest.
- **Consensus Merging** Wang et al. (2024): Consensus Merging is a method that uses task-specific binary masks to identify and remove selfish and catastrophic weights that negatively impact the performance of the merged model, thereby reducing task interference and improving the overall accuracy and stability of multi-task model merging.
- **Franken Merging** Goddard et al. (2024): Franken Merging is a model merging approach based on a concatenation strategy, where the parameters of corresponding layers from different models are directly concatenated to form a new fused model without additional training or parameter tuning. This method preserves the independence of each model's layers during merging but often suffers from performance degradation due to the lack of fine-grained parameter coordination, especially when handling multi-task or heterogeneous models.
- **EvoMerge** Akiba et al. (2025): evolutionary algorithms for automated search of model merging strategies across parameter and data flow spaces.
- **HM3** Zhou et al. (2025): formulates architecture-level model merging as a Markov decision process (MDP), where a reinforcement learning policy searches for effective inference paths by selecting and connecting layers from multiple pretrained models. It adopts a hierarchical multi-objective optimization strategy to balance multiple merging objectives during architecture search.
- **KnOTS-Ties** Stoica et al. (2025): uses SVD to jointly align task-specific low-rank updates into a shared representation space, after which Ties Merging is applied to combine the aligned updates.
- **RobustMerge** Zeng et al. (2025): analyzes parameter-efficient task updates in low-rank space and improves direction robustness by adjusting singular-value-related components and normalizing task updates to reduce interference.
- **TSV-Merge** Gargiulo et al. (2025): decomposes layer-wise task matrices into Task Singular Vectors using SVD and reduces task interference through singular-component compression and cross-task whitening.

The candidate model pool includes not only fine-tuned models but also models generated by parameter merging methods such as Task Arithmetic, TIES, DARE Merging (DARE-Linear), and DARE-TIES.

These models can be efficiently generated using the mergekit package Goddard et al. (2024), which is available at mergekit[15]. Additionally, the Evolutionary Search method Akiba et al. (2025) is also integrated in the mergekit package. For other methods, we reproduced and adapted codes from the following GitHub repositories Weight Averaging[16] and RegMean Merging[17].

Next, we describe the datasets and evaluation metrics for the tasks of generative, text translation, mathematical reasoning, and code generation task domains.

Generative Task Domain In this task domain, we employ GLUE benchmark Wang et al. (2019), a comprehensive suite designed for training, evaluating, and analyzing natural language understanding systems. GLUE comprises nine sentence- and sentence-pair classification tasks drawn from well-established datasets, carefully selected to represent a diverse range of dataset sizes, textual genres, and levels of difficulty. The benchmark tasks included in GLUE are CoLA, SST-2, MRPC, STS-B, QQP, MNLI, QNLI, RTE and WNLI. Each task evaluates distinct aspects of language understanding, providing a broad and challenging evaluation framework.

Text Translation Task Domain To evaluate the multilingual translation capability of LLMs, we use a set of translation tasks from the lm-evaluation-harness package, including WMT14[18], WMT16[19] Sennrich et al. (2016), IWSLT2017 Cettolo et al. (2017), and XNLI Conneau et al. (2018). These tasks assess the models' translation accuracy and fluency across different language pairs. For the first three translation tasks, we use the "chrf" metric, which measures character-level n-gram precision and recall. For the last task, we adopt accuracy as the evaluation metric.

Mathematical Reasoning Task Domain The mathematical reasoning domain comprises two classical benchmarks: GSM8K and MathQA. MathQA Amini et al. (2019) is a large-scale dataset consisting of 37,000 multiple-choice math word problems in English, spanning diverse mathematical domains. It models operation programs corresponding to problems in the AQuA dataset. Performance on MathQA is evaluated using accuracy as the metric. GSM8K Cobbe et al. (2021) contains over 8,000 high-quality problems ranging from basic arithmetic to complex algebraic reasoning, designed to rigorously assess a model's reasoning and computational capabilities on structured mathematical tasks. The evaluation metric employed is flexible-match, which tolerates minor variations in the predicted answers.

Code Generation Task Domain The Code Generation domain includes two widely used benchmarks: MBPP and HumanEval. MBPP Austin et al. (2021) dataset consists of 974 programming tasks designed to be solvable by entry-level programmers and aims to evaluate models' ability to synthesize short Python programs from natural language descriptions. We use pass@1 as the evaluation metric to measure performance on MBPP. HumanEval Chen et al. (2021), proposed by OpenAI, contains 164 programming problems where models generate Python functions from natural language prompts. The pass@10 metric is also employed, indicating the probability that at least one out of up to 100 generated solutions passes all test cases.

C.2 Vision Models

To evaluate our method's performance on vision models, we adopt two variants of the CLIP Vision Transformer Radford et al. (2021) as pretrained models: ViT-B/32 and ViT-L/14. We test multi-task abilities across the following image classification datasets:

- SUN397 dataset Xiao et al. (2010): Contains 397 scene categories with a total of 108,754 images, each category has at least 100 images.
- Stanford Cars dataset Krause et al. (2013): Includes 196 car model categories with 16,185 images, evenly split between training and test sets.

[15] https://github.com/arcee-ai/mergekit
[16] https://github.com/mlfoundations/model-soups
[17] https://github.com/bloomberg/dataless-model-merging
[18] https://www.statmt.org/wmt14/translation-task.html
[19] https://www.statmt.org/wmt16/translation-task.html

- RESISC45 dataset Cheng et al. (2017): A satellite remote sensing scene classification dataset with 45 categories and 31,500 images, approximately 700 images per class.
- EuroSAT dataset Helber et al. (2019): Satellite image classification dataset covering 10 classes, containing 27,000 geo-located and labeled images.
- SVHN dataset Yuval (2011): Real-world digit recognition dataset from Google Street View, with 10 digit classes, 73,257 training samples, 26,032 test samples, plus 531,131 additional easy samples.
- GTSRB dataset Stallkamp et al. (2011): Traffic sign recognition dataset covering 43 classes with over 50,000 images.
- MNIST dataset LeCun (1998): Classic handwritten digit recognition benchmark with 10 balanced classes, 60,000 training and 10,000 test images.
- DTD dataset Cimpoi et al. (2014): Texture classification dataset containing 47 categories and 5,640 images, about 120 images per class.

The fine-tuned models on these datasets are obtained from Task Arithmetic Ilharco et al. (2023) or Ties Merging Yadav et al. (2023) methods.

C.3 Implementation Details

The language candidate pool contains one base model, three task experts, and four parameter-level merged candidates, with all candidate backbones kept frozen throughout the search. We use rank-16 truncated SVD features for the state representation and a PPO-based actor–critic search with a budget of 1000 candidate-path evaluations. The reward coefficients are set to $(\alpha_1, \alpha_2, \alpha_3) = (0.7, 0.2, 0.1)$. Glue layers are inserted at cross-model or dimension-changing interfaces and optimized using 4096 Gaussian calibration probes with $\lambda = 0.5$. Search and Glue optimization use only calibration/validation data, and the resulting inference path and Glue parameters are fixed before test evaluation. Complete network configurations, optimization hyperparameters, and training scripts are provided in the supplementary code.

Our hyperparameter selection process involves initially performing grid search and sensitivity analysis on smaller-scale models. The optimized hyperparameters are then directly transferred to larger-scale models without further tuning. Specifically, in this work, we conducted grid search and sensitivity analysis on the Qwen2.5-1.5B and ViT-B/32 models with smaller scales, and subsequently applied the same hyperparameter settings to Llama-2-7B, Llama-2-13B and ViT-L/14 models with larger scales. For UniMerge, the clipping range ϵ is set to 0.1, the discount factor γ to 0.990, β_A to 0.95, and the coefficients c_1 and c_2 to 1.0 and 0.15, respectively.

D Detailed Experimental Results

This section provides detailed experimental results and supplementary analyses to support the main findings. Specifically, Appendix D.1 reports the detailed results of merging language models, while Appendix D.2 provides the results of merging vision models. Appendix D.3 extends the heterogeneous merging setting from models differentiated primarily by task-specific fine-tuning to compatible cross-scale models with different widths and depths. Appendix D.4 presents the detailed ablation analysis of UniMerge, including the complementarity between contrastive learning and Glue alignment, as well as the contributions of the diversity and efficiency terms in the search reward. Appendix D.5 further evaluates the Glue layer through cross-method generality experiments and fixed-path connector comparisons, examining both downstream task performance and feature, semantic, and transport alignment. Appendix D.6 investigates the selected inference paths and analyzes the source-model selection patterns across different network depths. Appendix D.7 evaluates search efficiency under a matched candidate-evaluation budget and reports the practical computational overhead of UniMerge. Appendix D.8 evaluates out-of-domain generalization on unseen tasks, while Appendix D.9 examines convergence stability across different random seeds. Finally, Appendix D.10 analyzes the sensitivity of the main representation, Glue, and search hyperparameters.

Table 3: Performance comparison on Llama-2-7B LLMs

Method	Generative	Translation		Math		Code		Average
	GLUE	WMT&IWS	XNLI	GSM8K	MathQA	HumanEval	MBPP	
Pretrained	21.54	21.64	25.09	16.34	16.57	15.36	20.62	19.60
Individual	55.97	42.23	43.21	45.60	28.76	43.11	52.67	44.51
Fisher Merging	29.40	34.22	34.77	33.96	22.41	20.67	25.33	28.68
RegMean Merging	29.32	27.93	34.97	36.78	23.76	20.06	27.09	28.56
Weight Average	18.39	26.41	33.86	29.94	15.39	22.26	20.73	23.86
Task Arithmetic	28.16	31.30	32.92	37.83	17.30	21.36	22.20	27.30
DARE Merging	33.42	30.24	37.51	39.24	19.51	22.09	27.47	29.92
Ties Merging	38.62	34.15	35.72	29.73	22.40	26.35	31.48	31.21
DARE-Ties Merging	37.03	33.93	37.47	38.20	22.70	28.20	30.11	32.52
Consensus Merging	47.52	37.97	40.74	37.95	27.99	30.15	37.93	37.18
PCB Merging	49.11	39.29	33.35	39.25	28.14	32.53	39.05	37.25
Franken Merging	16.96	26.17	12.89	19.35	15.23	17.66	13.84	17.44
EvoMerge	21.33	37.51	27.07	25.51	22.64	25.17	16.84	25.15
HM3	51.04	41.68	40.24	45.62	28.08	43.62	44.62	42.13
UniMerge	51.72	47.21	42.20	45.41	28.54	43.67	46.81	**43.65**

D.1 Detailed Experiments of Merging Language Models

We first compared the merging performance of all methods on three LLMs across four task domains, and the results are summarized in Tables 3, 4, and 5. Across all three LLM backbones, UniMerge achieves the highest overall average performance among the merging methods. In contrast, weight-based methods, which allocate weights according to statistical information, failed to account for task heterogeneity. Subspace-based methods yielded slight improvements on mathematical reasoning tasks through principal-axis transformations, but exhibited unstable performance in code generation. Existing architecture-level merging approaches, which rely on heuristic or random searches, struggled to identify robust inference paths in the high-dimensional space, resulting in performance that was even below that of the original pretrained model in several task domains. This observation explains why SOLAR-10.7B required continued pre-training after Franken Merging to recover its performance. Although HM3 encodes states using only model and layer indices without semantic representations, this limited its search efficiency. Specifically, in Table 3, UniMerge achieved the highest average score across all domains, consistently outperforming previous merging baselines. It showed particularly clear advantages in translation and code generation, highlighting its strong cross-domain generalization capability. Table 4 further compares various merging methods on Qwen2.5-1.5B across the four task domains. UniMerge obtained the highest overall average score of 40.40, outperforming all baselines. Traditional parameter-level methods produced only moderate results, while advanced subspace-based methods such as Consensus Merging and PCB Merging improved performance but still lagged behind UniMerge. Although HM3 performed competitively, it was slightly outperformed by UniMerge. Finally, Table 5 presents the performance comparison on the Llama-2-13B series. UniMerge achieved the highest average score of 43.01, again surpassing all competing merging approaches. Weight-based methods exhibited relatively weak performance, particularly in code generation, while subspace-based approaches offered moderate gains but could not match UniMerge's consistent improvements. Architecture-level methods delivered lower performance, underscoring the inherent difficulty of achieving stable and effective model merging at the architectural scale.

D.2 Detailed Experiments of Merging Vision Models

We compared merging performance on ViT-B/32 and ViT-L/14 across vision tasks, and the results are reported in Tables 6 and 7. Across both backbones, UniMerge consistently shows comprehensive advantages. In contrast, weight-based methods do not explicitly model cross-task representation disparities, which leads to weak performance on complex distributions. Subspace-based methods utilize principal axis transformations and task importance modeling, yielding notable improvements on certain high-semantic datasets. Architecture-level methods attempt to optimize layer combinations but lack explicit structural and task alignment strategies. Table 6 presents results under the ViT-B/32 backbone on the same eight datasets. UniMerge again achieved the best average accuracy and led on

Table 4: Performance comparison on Qwen2.5-1.5B LLMs

Method	Generative	Translation		Math		Code		Average
	GLUE	WMT&IWS	XNLI	GSM8K	MathQA	HumanEval	MBPP	
Pretrained	28.30	21.64	23.08	16.34	28.67	15.36	21.45	22.12
Individual	57.76	39.01	41.39	32.61	39.61	36.12	52.34	42.69
Fisher Merging	32.94	27.36	18.65	17.31	17.38	20.14	22.46	22.32
RegMean Merging	32.48	27.43	23.20	17.67	21.14	19.84	21.84	23.37
Weight Average	31.17	26.96	20.87	17.72	26.71	19.68	19.61	23.25
Task Arithmetic	42.76	27.40	30.90	19.73	37.71	17.63	21.45	28.23
DARE Merging	41.95	27.54	32.36	19.40	35.33	18.08	23.62	28.33
Ties Merging	42.72	29.07	28.46	22.66	36.07	16.32	19.61	27.84
DARE Ties Merging	38.25	27.91	30.87	20.63	40.33	19.61	24.84	28.92
Consensus Merging	46.42	29.82	38.09	23.98	37.84	22.59	34.08	33.26
PCB Merging	47.25	30.05	38.31	24.29	36.90	21.87	41.33	34.29
Franken Merging	23.92	17.32	22.31	14.51	26.81	13.89	20.73	19.93
EvoMerge	29.77	21.36	23.87	17.40	35.68	15.33	23.27	23.81
HM3	48.22	32.26	41.73	28.05	35.39	34.31	51.80	38.82
UniMerge	54.98	34.07	37.20	30.14	40.13	34.61	51.64	**40.40**

Table 5: Performance Comparison on Llama-2-13B LLMs

Method	General	Translation		Math		Code		Average
	GLUE	WMT&IWS	XNLI	GSM8K	MathQA	HumanEval	MBPP	
Pretrained	19.27	27.02	32.49	14.75	22.67	21.32	22.56	22.87
Individual	67.05	43.29	47.11	55.74	31.33	51.82	39.63	48.00
Fisher Merging	34.71	28.94	34.31	28.63	22.16	28.75	26.50	29.14
RegMean Merging	35.37	31.01	33.72	25.60	20.48	26.70	28.17	28.72
Weight Average	26.06	20.99	29.90	25.70	18.27	25.35	21.14	23.91
Task Arithmetic	35.24	29.16	37.33	27.74	21.47	28.52	25.80	29.32
DARE Task Arithmetic	36.02	30.54	35.89	26.85	21.57	31.49	28.31	30.09
Ties Merging	37.67	33.23	39.11	28.02	21.53	28.74	31.40	31.39
DARE Ties Merging	48.17	33.56	40.40	38.83	23.81	30.18	24.43	34.20
Consensus Merging	45.15	32.54	43.03	36.24	24.21	38.82	25.03	35.00
PCB Merging	52.77	41.96	40.90	46.34	24.56	40.15	26.57	39.04
Franken Merging	12.49	19.62	28.80	12.51	14.78	18.19	12.22	16.94
EvoMerge	15.92	25.06	28.23	17.36	19.29	13.31	14.48	19.09
HM3	53.20	41.12	45.92	46.82	29.13	43.93	33.29	41.92
UniMerge	54.27	42.74	46.28	48.64	30.05	46.28	32.83	**43.01**

most individual datasets, showing consistent gains over PCB Merging, Ties Merging, and HM3. The pattern mirrors the ViT-L/14 setting, where parameter averaging is weak, subspace methods offer moderate improvements, and architecture-level methods remain less reliable for stable fusion. Table 7 reports results on eight widely used image classification datasets under the ViT-L/14 backbone, including SUN397, Cars, RESISC45, EuroSAT, SVHN, GTSRB, MNIST, and DTD. UniMerge achieved the highest average accuracy across all datasets, outperforming strong subspace methods such as PCB Merging and Ties Merging, as well as the architecture-level baseline HM3. Traditional parameter-level merging delivered moderate results, while simple averaging underperformed. Franken Merging and EvoMerge lagged substantially, indicating limited capability for vision model merging.

D.3 Extending Heterogeneous Merging to Different Model Widths and Depths

Existing architecture-level merging methods such as EvoMerge and HM3 are primarily evaluated by merging task-specialized models obtained from the same or similar backbone architecture. In these settings, candidate models differ mainly because of task-specific fine-tuning, while their architectural configurations remain largely unchanged. The resulting heterogeneity is therefore primarily functional and representational rather than explicitly structural. In contrast, we further evaluate UniMerge on cross-scale candidate models whose Transformer components differ in model width, model depth, or both.

Table 6: Performance comparison on ViT-B/32

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Average
Pretrained	62.30	59.70	60.70	45.50	31.40	32.60	48.50	43.80	48.06
Individual	75.30	77.70	96.10	99.70	97.50	98.70	99.70	79.40	90.51
Traditional MTL	73.90	74.40	93.90	98.20	95.80	98.90	99.50	77.90	89.06
Fisher Merging	68.60	69.20	70.70	66.40	72.90	51.10	87.90	59.90	68.34
RegMean Merging	65.30	63.50	75.60	78.60	78.10	67.40	93.70	52.00	71.78
Weight Averaging	65.30	63.40	71.40	71.70	64.20	52.80	87.50	50.10	65.80
Task Arithmetic	55.20	54.90	66.70	78.90	80.20	69.70	97.30	50.40	69.16
DARE Merging	56.77	53.03	67.37	77.79	82.48	67.99	99.58	51.90	69.62
Ties Merging	65.00	64.40	74.80	77.40	81.20	69.30	96.50	54.50	72.89
DARE Ties Merging	67.03	66.47	72.79	77.64	83.52	71.65	96.43	52.71	73.53
Consensus Merging	65.73	63.56	76.51	77.20	80.66	69.03	96.89	52.63	72.78
PCB Merging	63.58	62.03	76.80	78.63	84.51	72.57	97.55	54.81	73.81
KnOTS-Ties	63.14	61.34	62.73	49.52	51.64	45.29	68.49	42.93	55.64
RobustMerge	52.83	62.21	43.95	65.83	52.66	42.30	64.56	63.39	55.97
TSV-Merge	57.47	61.46	64.38	55.37	44.87	43.82	55.30	43.56	53.28
Franken Merging	41.09	51.89	37.90	43.97	31.40	29.43	48.98	25.63	38.79
EvoMerge	53.27	54.03	62.14	62.33	59.32	56.16	95.34	32.97	59.45
HM3	63.22	64.02	73.27	78.06	77.62	68.21	96.94	52.22	71.70
UniMerge	67.22	65.07	73.71	80.64	80.96	70.35	96.90	57.31	**74.02**

Table 7: Performance comparison on ViT-L/14

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Average
Pretrained	66.80	77.70	71.00	59.90	58.40	50.50	76.30	55.30	64.49
Individual	82.30	92.40	97.40	100.00	98.10	99.20	99.70	84.10	94.15
Traditional MTL	80.80	90.60	96.30	96.30	97.60	99.10	99.60	84.40	93.09
Fisher Merging	69.20	**88.60**	87.50	93.50	80.60	74.80	93.30	70.00	82.19
RegMean Merging	73.30	81.80	86.10	97.00	88.00	84.20	98.50	60.80	83.71
Weight Average	72.10	81.60	82.60	91.90	78.20	70.70	97.10	62.80	79.63
Task Arithmetic	73.90	82.10	86.60	94.10	87.90	86.70	98.90	65.60	84.48
DARE Merging	74.58	83.23	85.98	95.38	88.15	87.15	97.57	65.69	84.71
Ties Merging	76.50	85.00	89.30	95.70	90.30	83.30	99.00	68.80	85.99
DARE Ties Merging	75.29	83.54	88.34	94.41	90.71	84.39	99.95	69.39	85.75
Consensus Merging	73.39	84.26	89.23	94.42	88.43	81.16	98.88	67.91	84.71
PCB Merging	75.04	86.01	89.54	94.88	87.76	86.55	99.01	71.21	86.25
KnOTS-Ties	61.07	82.73	79.40	61.89	72.40	67.75	84.16	56.65	70.76
RobustMerge	65.62	80.63	57.52	71.54	71.21	62.37	83.40	77.68	71.25
TSV-Merge	61.01	82.25	79.05	63.30	70.04	66.53	85.68	55.74	70.45
Franken Merging	58.84	63.88	52.86	54.67	57.09	46.23	51.05	50.23	54.35
EvoMerge	61.95	75.05	58.11	71.24	76.32	66.36	96.77	50.04	69.48
HM3	72.85	86.17	88.00	89.50	90.48	83.43	98.41	66.72	84.45
UniMerge	76.57	86.68	89.44	94.63	91.43	87.31	97.31	68.26	**86.45**

We consider two representative types of structural heterogeneity. For the Qwen2.5 family, Qwen2.5-1.5B and Qwen2.5-7B both contain 28 Transformer layers, while their hidden dimensions increase from 1536 to 3584. This setting therefore mainly evaluates width heterogeneity. For the 7B and 13B model pairs, the backbone changes from 32 layers with a hidden dimension of 4096 to 40 layers with a hidden dimension of 5120, introducing heterogeneity in both width and depth. The corresponding cross-scale pairs are constructed for general understanding, translation, mathematical reasoning, and code generation. The architectural configurations are summarized in Table 8. Under these settings, UniMerge directly searches and stitches components from models with different architectural scales. The state-enhanced representation provides a unified representation of candidate layers for inference-path construction, while the Glue layer aligns cross-model and dimension-changing interfaces. The resulting model is therefore assembled from structurally different pretrained models rather than only from differently fine-tuned checkpoints sharing the same architecture.

The detailed results are reported in Table 9. For general understanding, UniMerge preserves most of the source-model capability under both cross-scale settings. The improvement is more evident on translation, where UniMerge achieves positive transfer over the smaller-model references for

Table 8: Architectural configurations used in the cross-scale structural heterogeneity experiments.

Backbone	Scale	Layers	Hidden Dim.
Qwen2.5	1.5B	28	1536
Qwen2.5	7B	28	3584
Llama-2	7B	32	4096
Llama-2	13B	40	5120

Table 9: Performance comparison under structural heterogeneity with different model widths and depths. Bold numbers indicate the best result for each setting.

General	Qwen2.5-1.5B + Qwen2.5-7B				Llama-2-7B + Llama-2-13B			
	Reference	**UniMerge**	**HM3**	**EvoMerge**	**Reference**	**UniMerge**	**HM3**	**EvoMerge**
GLUE	**57.76**	52.35	13.20	–	**55.97**	52.94	11.50	–
Translation	**Qwen2.5-1.5B + Qwen2.5-7B**				**Llama-2-7B + Llama-2-13B**			
	Reference	**UniMerge**	**HM3**	**EvoMerge**	**Reference**	**UniMerge**	**HM3**	**EvoMerge**
XNLI	41.39	**44.67**	13.97	–	43.21	**45.87**	14.71	–
WMT&IWS	39.01	**42.48**	7.41	–	42.23	**43.50**	9.92	–
Math	**Qwen2.5-Math-1.5B + Qwen2.5-Math-7B**				**WizardMath-7B + WizardMath-13B**			
	Reference	**UniMerge**	**HM3**	**EvoMerge**	**Reference**	**UniMerge**	**HM3**	**EvoMerge**
GSM8K	**32.61**	27.66	4.32	–	**45.60**	38.65	1.67	–
MathQA	**39.61**	34.94	12.91	–	**28.76**	25.44	6.30	–
Code	**Qwen2.5-Coder-1.5B + Qwen2.5-Coder-7B**				**CodeLlama-7B + WizardCoder-Python-13B**			
	Reference	**UniMerge**	**HM3**	**EvoMerge**	**Reference**	**UniMerge**	**HM3**	**EvoMerge**
MBPP	**52.34**	27.66	3.81	–	**52.67**	29.80	7.97	–
HumanEval	**36.12**	21.95	1.89	–	**43.11**	25.44	3.05	–

both Qwen2.5 and Llama model pairs. In contrast, HM3 suffers severe degradation under the same cross-scale settings, while valid EvoMerge results are unavailable.

Mathematical reasoning and code generation are more sensitive to structural mismatch. Nevertheless, UniMerge remains effective under both width-only and width-and-depth heterogeneity and substantially alleviates the degradation observed with HM3. This indicates that the proposed layer representation, architecture search, and Glue-based alignment remain effective even when structurally different task-specialized models are stitched together.

Overall, the Qwen2.5 experiments demonstrate that UniMerge can merge models with different hidden widths, while the 7B and 13B experiments further show that it can handle simultaneous differences in model width and depth. These results extend architecture-level merging from fine-tuning-induced functional heterogeneity toward explicit structural heterogeneity.

D.4 Detailed Ablation Analysis

We first examine the contribution of the main components in UniMerge through the process-level ablation reported in Table 2. This analysis is conducted on Qwen2.5-1.5B, Llama-2-7B, and Llama-2-13B, and therefore reports the average over the three LLM backbones. Here, we use a domain-balanced protocol over the three LLM backbones, where the four language task domains receive equal weight. This differs from the benchmark-level averaging used in the main comparison and is intended to prevent domains containing more benchmarks from receiving disproportionately larger weights. Removing SVD representation, State Encoder + CL, PPO search, or Glue training consistently degrades performance, showing that parameter-aware representation, effective path search, and inter-layer alignment all contribute to the final performance.

We additionally examine the interaction between contrastive learning and the Glue layer on the same three LLM backbones. Specifically, we compare the full UniMerge with three variants: **w/o CL**, which removes the contrastive self-supervised learning mechanism; **w/o Glue**, which directly connects the selected layers; and **w/o Both**, which removes both components. As shown in Fig. 5, removing either component leads to clear performance degradation, while removing both causes the largest drop. These results indicate that representation discrimination and inter-layer alignment provide complementary benefits for architecture-level model merging.

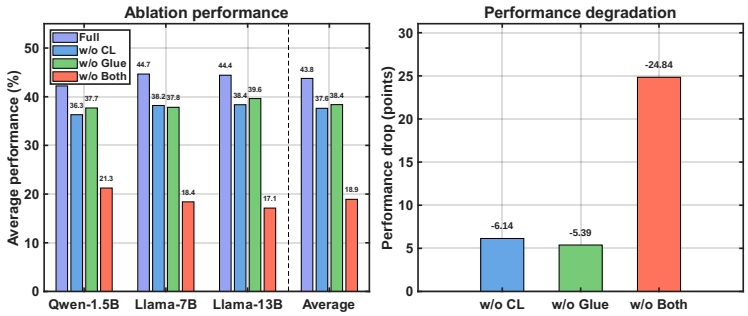


Figure 5: Ablation results of contrastive learning and Glue alignment.

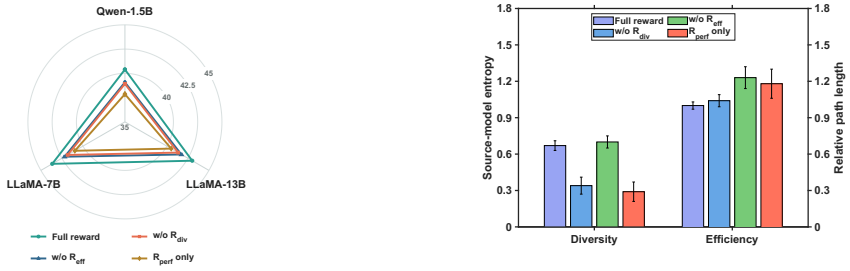
We further conduct a controlled comparison with HM3 in Table 2. Unlike the process-level ablation, this comparison reports the overall average across all five evaluated language and vision backbones to examine whether the observed gains generalize across model domains. The two averages therefore correspond to different evaluation scopes and are not intended for direct numerical comparison. Replacing the original HM3 connector with the proposed UniMerge Glue improves overall performance while retaining HM3's index-only state representation, demonstrating the benefit of inter-layer alignment. When both methods use index-only states and the UniMerge Glue, replacing HM3 search with UniMerge search provides a further gain. Introducing architecture metadata yields additional improvement, while the full spectral/SVD state achieves the strongest performance. Together, these results show that UniMerge benefits from the complementary effects of state representation, architecture search, and Glue-based inter-layer alignment.

Beyond the module-level ablations, we further examine the diversity and efficiency terms in the search reward. We separately remove each term while retaining the remaining components, and also consider a performance-only reward. Figure 6 shows that removing the diversity term decreases the equal-weight mean over the three language backbones by 1.66 points, removing the efficiency term decreases it by 1.36 points, and retaining only the performance term decreases it by 2.57 points. The search-behavior diagnostics further show that removing the diversity term reduces source-model entropy, whereas removing the efficiency term increases the relative path length. These observations confirm that the diversity term encourages broader use of candidate sources, while the efficiency term discourages unnecessarily long merging paths.

D.5 Glue Layer Effectiveness and Fixed-Path Diagnostics

We first examine whether the proposed Glue layer can also benefit architecture-level merging methods beyond UniMerge. Figure 7 compares Franken Merging, EvoMerge, and HM3 using their original connectors and the proposed Glue layer. Across translation, mathematical reasoning, and code generation, introducing the Glue layer consistently improves or preserves task performance. These results indicate that the proposed alignment mechanism addresses a general cross-model layer-connection problem rather than a property specific to the UniMerge search framework.

We next fix the inference paths and vary only the connector design to separate connector quality from path-selection effects. Table 10 compares direct connection, linear projection, feature-only alignment, semantic-only alignment, and the full Glue layer. The full Glue layer achieves the highest task performance together with the lowest feature, semantic, and transport discrepancies. Linear projection improves over direct connection but remains clearly inferior to learned alignment, showing that dimensional compatibility alone is insufficient for reliable cross-model layer stitching.



(a) Reward-term ablation.

(b) Search-behavior diagnostics.

Figure 6: Reward ablation and search-behavior diagnostics. (a) Each backbone score averages seven language benchmarks, with a radial range of 35–45. (b) Source-model entropy is shown on the left axis, while path length normalized to the full-reward setting is shown on the right axis.

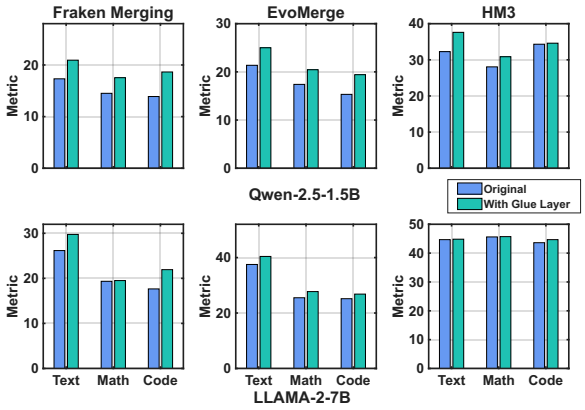


Figure 7: Cross-method generality of the Glue layer. Original and Glue-equipped variants of Franken Merging, EvoMerge, and HM3 are compared on translation, mathematical reasoning, and code generation for Qwen2.5-1.5B and Llama-2-7B.

The two alignment objectives also exhibit distinct effects. Cross-layer feature alignment produces a stronger reduction in interface-level feature mismatch, whereas inter-layer semantic alignment more effectively reduces the discrepancy after the target-layer transformation. Combining both objectives achieves the strongest alignment across all three diagnostic measures and the best downstream task performance. These results support the use of dual alignment for simultaneously maintaining target-input compatibility and target-layer semantic consistency.

Table 10: Fixed-path connector comparison.

Connector	Task Perf. ↑	Feature ↓	Semantic ↓	Transport ↓
Direct connection	38.40	1.00	1.00	1.00
Linear projection	40.90	0.68	0.74	0.70
$\mathcal{L}_{\text{cross}}$ only	42.35	0.30	0.56	0.44
$\mathcal{L}_{\text{inter}}$ only	42.10	0.55	0.32	0.42
Full Glue	**43.80**	**0.21**	**0.23**	**0.25**

D.6 Selected Path Analysis

To analyze the search behavior of architecture-level merging methods, we visualize the source-model selection frequency across merged blocks in Fig. 8. This analysis is conducted on the Llama-2-7B backbone, which contains 32 Transformer decoder layers. For compact visualization, the 32-layer

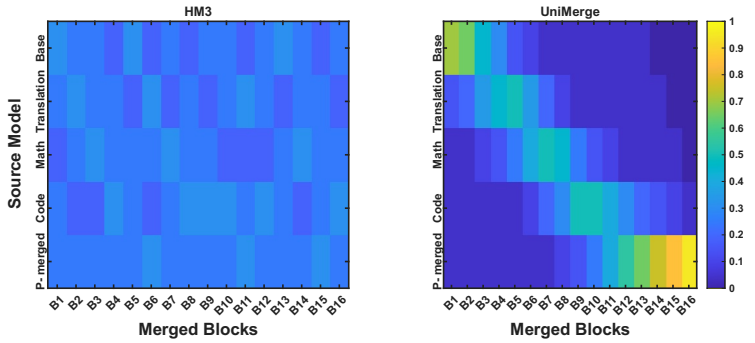


Figure 8: Selected inference path frequency.

inference path is divided into 16 merged blocks, each containing two consecutive layers. For each method, we repeat the search process under 10 random seeds. Thus, each block contains 20 layer selections in total, and each heatmap entry denotes the normalized frequency with which a given source model is selected in the corresponding block. As shown in Fig. 8, HM3 shows a relatively scattered selection pattern. The selection frequencies of different source models are close in most blocks, indicating that HM3 does not exhibit a clear depth-aware preference when constructing the merged inference path. This suggests that its search process is less stable and less interpretable, especially when multiple task-specialized and parameter-level merged models are included in the candidate pool. In contrast, UniMerge presents a more structured selection pattern. In the shallow blocks, UniMerge mainly selects the Base model, suggesting that general linguistic representations from the original Llama-2-7B backbone are preserved in early layers. In the middle blocks, the selection gradually shifts to task-specialized models, including Translation, Math, and Code, indicating that UniMerge introduces task-specific functional representations at intermediate depths. In the deeper blocks, parameter-level merged candidates are selected more frequently, suggesting they are useful for integrating multi-task knowledge near the model's output. Overall, the visualization shows that UniMerge learns a clear depth-aware merging strategy: general representations are preserved in early blocks, task-specific capabilities are introduced in middle blocks, and parameter-level merged models are exploited in later blocks. This confirms that the proposed state-enhanced representation can guide the policy network to discover more meaningful and stable layer-combination paths.

D.7 Search Efficiency Analysis

This subsection analyzes the search efficiency of UniMerge. We evaluate the search efficiency of UniMerge under a matched candidate-evaluation budget. Unlike full-model pretraining or fine-tuning, UniMerge keeps all candidate backbones frozen and optimizes only the search-related modules and Glue layers. We compare UniMerge with Franken Merging, EvoMerge, HM3, Grid Search, and Evolutionary Search using the same number of candidate-path evaluations. Grid Search is capped at the same evaluation budget, while Evolutionary Search adopts differential evolution; both use the same Glue configuration as UniMerge. As shown in Table 11, UniMerge achieves the highest overall performance across the three LLM backbones, indicating more effective exploration under a fixed search budget.

To further isolate search quality from downstream benchmark performance, we construct a small enumerable space containing $5^5 = 3125$ legal paths with an exact optimum of $R^\star = 0.842$. Under a budget of 300 evaluations, UniMerge reaches 0.837 ± 0.004, compared with 0.827 ± 0.007 for HM3, and requires only 43 ± 12 evaluations to reach $0.95R^\star$, versus 76 ± 21 for HM3. These results show that UniMerge can identify higher-quality solutions with fewer evaluations under a fixed search budget. We further measure the practical computational overhead of the complete UniMerge pipeline on a single RTX A6000 48GB GPU, including SVD preprocessing, state and contrastive representation learning, PPO-based path search, and Glue optimization. The end-to-end runtime is 1.68, 3.53, and 6.00 hours for Qwen2.5-1.5B, Llama-2-7B, and Llama-2-13B, respectively, with peak memory usage of 11.6, 26.8, and 39.5 GB. Candidate backbones remain frozen throughout the pipeline,

Table 11: Search efficiency analysis among UniMerge, two architecture-level merging methods and two search methods

Method	Qwen2.5-1.5B				Llama-2-7B				Llama-2-13B			
	GLUE	Translation	Math	Code	GLUE	Translation	Math	Code	GLUE	Translation	Math	Code
Grid Search	23.92	20.38	24.67	18.50	18.16	20.43	21.98	20.33	16.55	24.93	23.73	23.96
Evolutionary Search	31.19	26.03	26.34	24.46	26.59	29.41	26.50	27.51	33.93	29.83	21.80	28.39
Franken Merging	23.92	19.82	20.66	17.31	16.96	19.53	17.29	15.75	12.49	24.21	13.65	15.21
EvoMerge	29.77	22.62	26.54	19.30	21.33	32.29	24.07	21.00	15.92	26.64	18.32	13.90
HM3	48.22	**37.00**	31.72	43.06	51.04	40.96	36.85	44.12	53.20	43.52	37.98	38.61
UniMerge	**54.98**	35.64	**35.13**	**43.13**	**51.72**	**44.71**	**36.98**	**45.24**	**54.27**	**44.51**	**39.35**	**39.56**

Table 12: Performance comparison in the out-of-domain scenario

Method	Question Answering	Word Sense Disambiguation	Sentence Completion	Average
Fisher Merging	26.55	73.37	39.64	46.52
RegMean Merging	26.47	63.08	36.60	42.05
Weight Average	23.54	61.56	35.29	40.13
Task Arithmetic	27.31	66.45	37.26	43.67
DARE Merging	28.57	66.39	39.03	44.66
Ties Merging	32.77	72.30	40.09	48.39
DARE Ties Merging	31.18	58.08	40.85	43.37
Consensus Merging	28.67	76.12	44.51	49.77
PCB Merging	32.26	75.44	41.47	49.72
Franken Merging	18.11	21.32	18.68	19.37
EvoMerge	24.48	42.66	37.44	34.86
HM3	33.19	72.83	44.11	50.04
UniMerge	33.87	73.36	45.99	51.07

with PPO-based path search accounting for the major computational cost. These measurements characterize the practical overhead of UniMerge without requiring full-model fine-tuning.

D.8 Out-of-Domain Generalization Analysis

To evaluate the out-of-domain generalization capability of UniMerge, we reported average performance across a diverse set of tasks and datasets: SQuAD Rajpurkar et al. (2016), Social IQA Sap et al. (2019), and QuAIL Rogers et al. (2020) for question answering (QA) task; WiC Pilehvar & Camacho-Collados (2019) for word sense disambiguation (WSD) task; and COPA Roemmele et al. (2011) and H-SWAG Zellers et al. (2019) for sentence completion (SC) task, as presented in Table 12. Overall, UniMerge achieved the highest average score and outperformed all model merging approaches. This demonstrates UniMerge's superior capability in generating merged models with enhanced generalization across heterogeneous natural language processing challenges. In contrast, traditional parameter-level methods lagged behind significantly, particularly on sentence completion and word sense disambiguation tasks, highlighting their limited capacity to capture diverse linguistic nuances when transferred to novel domains. Similarly, Franken Merging and EvoMerge showed notably lower results, reflecting challenges in stable fusion under domain shift.

D.9 Stability Analysis

This subsection evaluates UniMerge's stability. We performed ten independent runs with different random seeds and compared UniMerge, HM3, and grid search across four task domains. The results are provided in Fig. 9. We can observe that UniMerge showed rapid performance improvement in the early stage and reached a high reward within about 400 episodes, while HM3 required more iterations to achieve similar results. These findings indicate higher sample efficiency and faster policy optimization. In contrast, grid search can find competitive solutions through systematic exploration but suffers from slow convergence and high computational cost.

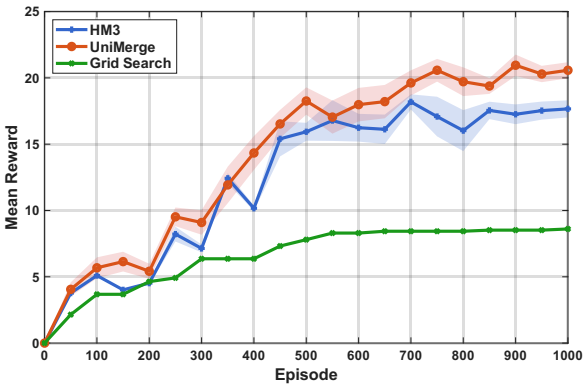
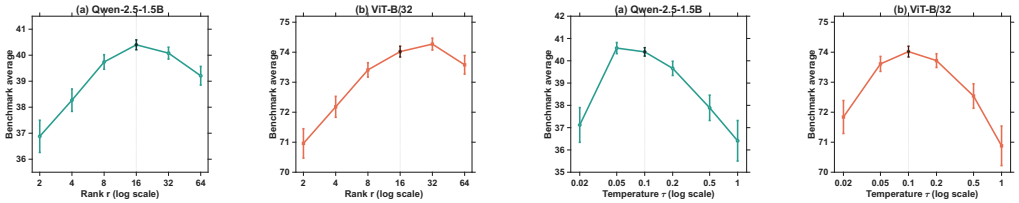


Figure 9: RL convergence comparison.



(a) SVD rank r. (b) Contrastive temperature τ.

Figure 10: Sensitivity of the state and contrastive representation hyperparameters on Qwen2.5-1.5B and ViT-B/32.

D.10 Hyperparameter Sensitivity

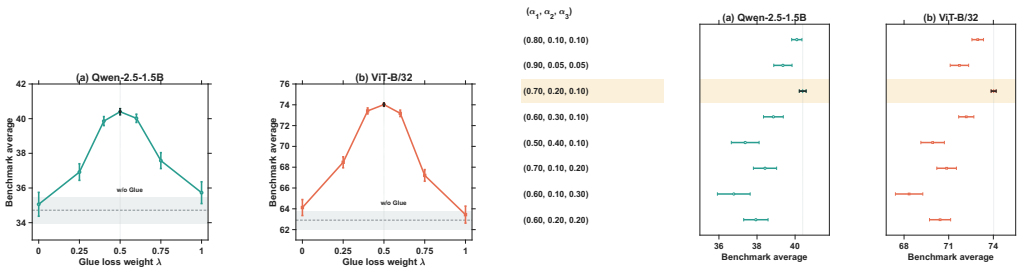
We conduct one-at-a-time hyperparameter sensitivity analyses on Qwen2.5-1.5B and ViT-B/32 while fixing the candidate pool, calibration data, and search budget of 1000 candidate-path evaluations. Each point reports the mean benchmark performance over three random seeds, with error bars indicating one standard deviation. The default settings are $r = 16$, $\tau = 0.1$, $\lambda = 0.5$, and $(\alpha_1, \alpha_2, \alpha_3) = (0.7, 0.2, 0.1)$.

For the SVD rank, performance improves as r increases from 2 to 16 and then largely saturates, as shown in Fig. 10(a). Rank 32 performs slightly better on ViT-B/32, while rank 16 performs better on Qwen2.5-1.5B; we therefore adopt $r = 16$ as a practical trade-off between representation size and performance. For the contrastive temperature, Fig. 10(b) shows that moderate values between 0.05 and 0.20 consistently outperform the extreme settings, while the preferred value varies slightly across backbones. We use $\tau = 0.1$ as a shared default.

For the Glue objective, Fig. 11(a) shows that intermediate values of λ, particularly around 0.4–0.6, outperform the single-loss endpoints. Here, $\lambda = 0$ retains only cross-layer feature alignment, whereas $\lambda = 1$ retains only inter-layer semantic alignment, supporting their complementary roles. Finally, Fig. 11(b) examines the reward coefficients. The default $(0.7, 0.2, 0.1)$ achieves the highest mean performance among the tested configurations on both backbones, while excessive emphasis on either diversity or efficiency degrades performance. These settings are selected on the smaller backbones and transferred to Llama-2-7B, Llama-2-13B, and ViT-L/14 without further tuning.

E Discussion

The significance of these experiments lies not merely in evaluating UniMerge on larger model pairs, but in broadening the scope of architecture-level heterogeneous merging. Existing methods such as HM3 and EvoMerge establish the feasibility of searching over model architectures or inference paths,



(a) Glue loss weight λ. (b) Reward coefficients $(\alpha_1, \alpha_2, \alpha_3)$.

Figure 11: Sensitivity of the Glue objective and search reward configuration.

whereas UniMerge further investigates a setting in which candidate models can differ in both task specialization and architectural scale.

Accordingly, the heterogeneous scenario evolves from *fine-tuning-induced functional heterogeneity* to *cross-scale structural heterogeneity* involving different widths and depths. The results show that the proposed state-enhanced layer representation, architecture search, and Glue-based alignment remain effective under this more challenging setting. Thus, UniMerge extends architecture-level model merging beyond differently fine-tuned models with a shared architectural scale toward structurally heterogeneous pretrained models.

Our current empirical scope focuses on compatible models within the same model family, such as Qwen2.5 variants with different widths and Llama-based models with different widths and depths. This setting preserves a certain degree of architectural compatibility while introducing explicit structural heterogeneity. A more challenging direction is cross-family merging, where models such as Qwen and Llama may differ simultaneously in hidden dimensions, layer organization, normalization schemes, and representation spaces. Future work will therefore investigate whether UniMerge can be extended from within-family structural heterogeneity to such strongly heterogeneous cross-family model merging.

F Statement about usage of LLMs

In this work, we did not use generative AI tools for tasks requiring disclosure, and the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for language editing, LaTeX and scientific-figure formatting, checking the arithmetic consistency of reported results, formatting references, and sourcing or searching for information. We reviewed all AI-assisted work. In particular, information and references suggested by AI tools were verified against the original sources, and arithmetic checks were independently verified against the reported experimental results. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.