Building LLMs Like LEGO: Two-dimensional Architecture Reassembly of Large Language Models

Xingyu Wu[1], Yu Zhou[1], Kay Chen Tan[1]
[1]The Hong Kong Polytechnic University
xingy.wu@polyu.edu.hk, zy-yu.zhou@connect.polyu.hk, kctan@polyu.edu.hk
Correspondence: Yu Zhou

Abstract

Pretrained large language models (LLMs) are typically reused as indivisible artifacts, adapted, merged, or ensembled as a whole. In this study, we show that LLMs can instead be structurally recomposed as modular building blocks to create new architectures without access to original training data. We introduce architecture-level reassembly as a new reuse paradigm, in which Transformer blocks from heterogeneous models are treated as reusable components. This idea is formalized through a two-dimensional reassembly space that supports both vertical recombination across depth and horizontal composition within layers. To make this space tractable, we propose a chromosome-based architectural encoding and perform a bi-level multi-objective evolutionary optimization over vertical structure and horizontal composition. To resolve representation incompatibility across heterogeneous blocks, we introduce lightweight glue layers trained via data-free knowledge distillation, enabling valid information flow without modifying pretrained parameters. Our results demonstrate that architecture-level reassembly unlocks a new dimension of flexibility in model reuse, pointing toward a modular and evolutionary view of LLM design.

1 Introduction

The rapid proliferation of pretrained large language models (LLMs) has fundamentally reshaped how natural language processing systems are developed. Rather than training models from scratch, practitioners increasingly rely on reusing (Parmar et al., 2024), adapting (Chen, 2024), and combining (Kim et al., 2024) pretrained models that differ in architecture, training data, and emergent capabilities. The field now possesses a rich and heterogeneous ecosystem of pretrained LLMs (Wolf et al., 2019). However, this abundance also raises a central yet underexplored question: when multiple heterogeneous LLMs are available, how should they be reused in a principled and systematic manner?

Existing approaches to LLM reuse usually treat LLMs as monolithic artifacts. As detailed in Appendix A, continued pretraining and fine-tuning adapt a single model by further optimizing its parameters, implicitly assuming a fixed architecture and a linear evolution of capabilities (Houlsby et al., 2019; Hu et al., 2022). Ensemble methods and routing-based approaches combine multiple models at the output level, but do not exploit their internal structure (Guha et al., 2024; Jitkrittum et al., 2025; Lu et al., 2024). Parameter-level model merging offers a data-free alternative by averaging or interpolating weights, yet typically assumes architectural alignment and operates under restrictive structural constraints (Kim et al., 2024; Dang et al., 2025). Consequently, current paradigms largely limit reuse to either parameter adaptation within a fixed architecture or coarse-grained combination of entire models.

Recent studies have begun to challenge the monolithic view of pretrained models by demonstrating that deep neural networks can be decomposed into blocks and vertically recombined to form new architectures (Yang et al., 2022). More recent work explores the layer-wise reuse of LLMs (Goddard et al., 2024; Akiba et al., 2025). These efforts provide early evidence that structural reuse beyond whole-model selection is feasible. However, existing approaches remain limited to one-dimensional, depth-wise recombination, typically assuming that each layer originates from a single source model and that inter-layer compatibility holds by construction. Such assumptions rule out more expressive forms of reuse, particularly horizontal composition where multiple models jointly contribute to a single layer, and leave unaddressed the core difficulty of aligning heterogeneous intermediate representations.

In this study, we take a step toward a more general and compositional perspective on reusing LLMs. Rather than improving performance

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 44939–44956

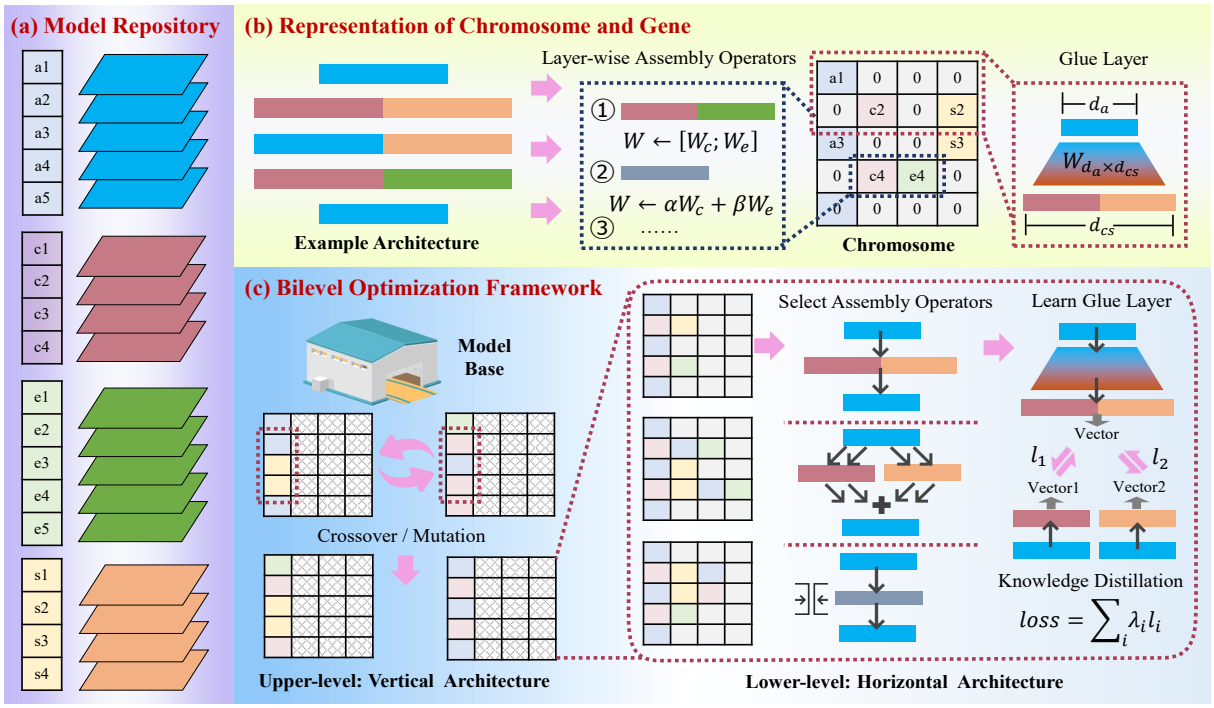


Figure 1: Overview of the LEGO-LLM, a novel paradigm for LLM reuse. (a) a model repository consisting of heterogeneous LLMs that serve as reusable building blocks. (b) chromosome and gene representations that encode layer ordering across depth, layer-wise assembly operators, and glue layers for both vertical and horizontal reassembly. (c) a bilevel optimization framework that jointly optimizes vertical structure and horizontal composition, where glue layers are learned via knowledge distillation.

through continued pretraining (Parmar et al., 2024) or parameter-level adaptation (Yang et al., 2024), we ask whether LLMs can be structurally recomposed, much like LEGO bricks, to form new model architectures without access to the original training data. To this end, we formulate architecture-level reassembly as a systematic reuse paradigm, in which Transformer blocks from heterogeneous source models are treated as reusable components. Based on this formulation, we introduce a two-dimensional architecture reassembly space that supports both vertical recombination across depth and horizontal composition within layers.

This formulation introduces two fundamental challenges. The resulting architecture reassembly space is combinatorially large, involving heterogeneous depth mappings across models, multiple layer-wise assembly operators, and complex cross-model interactions. Such a space precludes exhaustive exploration and renders naive extensions of existing architecture search methods impractical. On the other hand, reassembled architectures often differ substantially in hidden dimensionality, representation geometry, and normalization conventions, making direct block recombination ill-defined and prone to representation mismatch. These challenges suggest that architecture reassembly cannot be reduced to a straightforward extension of model merging (Zhou et al., 2025b; Qiu et al., 2026) or architecture search (Zhou et al., 2025a), but instead requires new mechanisms for scalable exploration and representation alignment.

To address these challenges, we propose a unified framework for two-dimensional architecture reassembly of LLMs, namely LEGO-LLM. As shown in Figure 1, we introduce a chromosome-based architectural representation that explicitly encodes, at each depth, which LLM blocks are selected, how they are assembled, and how representations are aligned across layers. Architecture exploration is performed via bi-level multi-objective evolutionary optimization, where vertical structure and horizontal composition are jointly optimized. To enable valid information flow between heterogeneous blocks, we further introduce lightweight glue layers trained via data-free knowledge distillation, which align intermediate representations without modifying internal block structures. These components make it possible to build new and functional LLM architectures by recomposing existing LLMs, demonstrating the potential of architecture-level reassembly as a novel paradigm for model reuse.

2 Architecture Reassembly Space and Chromosome Encoding

Given a repository of heterogeneous LLMs, our goal is to automatically construct a new model by reorganizing, combining, and integrating their internal layers, without relying on task-specific fine-tuning or access to original training data. Let $\mathcal{M} = \{M_1, M_2, \ldots, M_N\}$ denote a repository of N LLMs. Each model M_i is composed of a stack of Transformer blocks and can be written as $M_i = \{E_i, B_i^1, B_i^2, \ldots, B_i^{L_i}, H_i\}$, where E_i denotes the input embedding layer, B_i^ℓ is the ℓ-th Transformer block (consisting of self-attention and feed-forward sublayers) (Vaswani et al., 2017), and H_i is the output head. In this work, architectural reassembly is restricted to the Transformer blocks $\{B_i^\ell\}$. The embedding and output head are assumed to be either shared across models or inherited from a selected backbone model, ensuring compatibility at the input and output interfaces. Transformer blocks from different models are indexed by their depth, and we assume that blocks at similar depth indices are semantically comparable, a common empirical assumption in studies of deep Transformer representations.

Our objective is to construct a reassembled model $M^\star = \{E^\star, \tilde{B}^1, \tilde{B}^2, \ldots, \tilde{B}^L, H^\star\}$, where each $\tilde{B}^k$ is a composite block obtained by reorganizing and combining Transformer blocks selected from the repository $\mathcal{M}$. Unlike conventional model selection (Wu et al., 2025) or parameter fine-tuning, the reassembled model is not constrained to inherit a single linear stack of layers from any individual source model. Instead, at each depth k, the composite block $\tilde{B}^k$ may be constructed from one or multiple source blocks drawn from different models, using a predefined set of layer-level assembly operators. Formally, the architecture reassembly problem is defined as:

$$\max_{\mathcal{A},\Theta} \mathbf{F}(M^\star(\mathcal{A}, \Theta)), \tag{1}$$

where $\mathcal{A}$ denotes the architectural configuration, specifying which source blocks are selected at each depth and how they are assembled; Θ represents auxiliary parameters introduced during reassembly such as glue layers; $\mathbf{F}(\cdot)$ is a vector-valued fitness function measuring model performance over one or multiple downstream tasks. This formulation naturally leads to a multi-objective optimization problem, where different objectives may correspond to different tasks, datasets, or evaluation criteria.

At each depth k, the composite block $\tilde{B}^k$ is formed by applying a layer-wise assembly operator to a set of selected source blocks. In this study, we define a finite set of admissible assembly operators: $\mathcal{O} = \{\mathcal{O}^{\text{sub}}, \mathcal{O}^{\text{merge}}, \mathcal{O}^{\text{concat}}, \mathcal{O}^{\text{ens}}\}$, including

- Vertical substitution (stitching) $\mathcal{O}^{\text{sub}}$: A block from one model replaces the corresponding block from another model, possibly creating mismatches with adjacent layers.
- Parameter-level merging $\mathcal{O}^{\text{merge}}$: Blocks with compatible architectures are merged at the parameter level using existing model merging techniques.
- Horizontal concatenation (layer widening) $\mathcal{O}^{\text{concat}}$: Multiple blocks are combined along the feature dimension to form a wider composite representation.
- Output-level integration (ensemble) $\mathcal{O}^{\text{ens}}$: Outputs of multiple blocks are aggregated through a predefined integration function.

The choice of operator at each depth is discrete, while the internal behavior of each operator may be parameterized. Importantly, the specific instantiation of an operator is decoupled from the architectural decision of whether to use it, allowing the framework to flexibly incorporate different merging or integration techniques.

Reassembling blocks from different source models generally leads to representation mismatches between consecutive layers, due to differences in hidden dimensionality, feature distributions, or architectural conventions. To ensure valid information flow in the reassembled model, we introduce glue layers between consecutive composite blocks. A glue layer is defined as a lightweight parametric function that maps the output representation of $\tilde{B}^k$ to a compatible input representation for $\tilde{B}^{k+1}$. Conceptually, glue layers operate only at the block input/output level and do not interfere with the internal structure of Transformer blocks. When no mismatch exists, a glue layer can degenerate to an identity mapping. The parameters of glue layers are not optimized through task supervision. Instead, they are trained using a data-free knowledge distillation (Gou et al., 2021) mechanism, aligning the behavior of the reassembled architecture with that of the original source models. This design preserves the data-free nature of the overall framework.

Based on these settings, we formalize the architecture reassembly space through a structured chromosome representation, which simultaneously specifies admissible reassembled architectures and serves as the genotype for evolutionary optimization. We consider reassembled models with a depth L, where architectural reorganization is restricted to the level of Transformer blocks. Each candidate architecture is represented by a chromosome

$$\mathcal{C} = (\mathbf{C}, \mathbf{o}, \mathcal{G}), \tag{2}$$

where $\mathbf{C} \in \{0,1\}^{L \times N}$ is a binary selection matrix, $\mathbf{o} = (o_1, o_2, \ldots, o_L)$ is a vector of layer-wise assembly operators, and $\mathcal{G} = \{G_1, G_2, \ldots, G_{L-1}\}$ denotes a set of glue layers inserted between consecutive blocks.

The selection matrix $\mathbf{C}$ specifies how Transformer blocks are drawn from the model repository. Each entry $c_{k,i}$ is defined as

$$c_{k,i} = \begin{cases} 1, & \text{if block at depth } k \text{ is selected from } M_i, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where $k \in \{1, \ldots, L\}$ and $i \in \{1, \ldots, N\}$. The k-th row $\mathbf{C}_{k,:}$ therefore determines which source models contribute Transformer blocks to the composite block at depth k. This row-wise interpretation enables horizontal reassembly by combining blocks from multiple models at the same depth, while variations across rows naturally encode vertical substitutions and depth-wise recombination.

Given the selection matrix $\mathbf{C}$, a composite block is constructed at each depth using a layer-wise assembly operator. Based on the aforementioned operator set $\mathcal{O}$, the operator choice at depth k is denoted by $o_k \in \mathcal{O}$, and the resulting composite block $\tilde{B}^k$ is expressed as

$$\tilde{B}^k = \mathcal{O}_{o_k}\big(\{B_i^{d_{k,i}} \mid c_{k,i} = 1\}\big), \tag{4}$$

where $B_i^{d_{k,i}}$ denotes the Transformer block selected from model M_i at depth index $d_{k,i}$. The mapping from the reassembled depth k to the source depth $d_{k,i}$ is unconstrained, except that $d_{k,i}$ must correspond to a valid block in M_i. As a result, non-contiguous depth mappings and layer skipping are explicitly allowed. Architectures that induce incompatible or ineffective layer transitions are not ruled out a priori, but are implicitly eliminated through evolutionary selection.

While the assembly operators are discrete at the architectural level, their internal behaviors may involve continuous parameters. For example, parameter-level merging can rely on existing model merging algorithms, and ensemble-style operators may include learnable aggregation weights. This separation allows the chromosome to encode high-level structural decisions without committing to a specific instantiation of the underlying operator.

Due to heterogeneity across source models, including differences in hidden dimensionality, representation distributions, and normalization conventions, the outputs of consecutive composite blocks may not be directly compatible. To ensure valid information flow, we introduce a glue layer G_k between each pair of adjacent composite blocks. The forward propagation between depths k and $k+1$ is given by

$$\mathbf{h}_{k+1} = \tilde{B}^{k+1}\big(G_k(\mathbf{h}_k)\big), \tag{5}$$

where $\mathbf{h}_k$ denotes the output representation of $\tilde{B}^k$. Each glue layer G_k is implemented as a lightweight multi-layer perceptron acting on block-level representations. For architectural uniformity, a glue layer is associated with every inter-block connection and initialized as an identity mapping whenever the input and output dimensions are already compatible. Importantly, glue layers operate only on block inputs and outputs and do not modify the internal structure of Transformer blocks.

An initial population of chromosomes is constructed by encoding the original models. For a given model M_i, the corresponding chromosome is initialized as

$$c_{k,j} = \begin{cases} 1, & \text{if } j = i, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall k \in \{1, \ldots, L\}, \tag{6}$$

with all assembly operators set to vertical substitution and all glue layers initialized as identity mappings. This initialization guarantees that every chromosome corresponds to a valid model.

The unified chromosome formulation defines a highly expressive architecture reassembly space, subsuming standard model selection, parameter merging, and ensemble learning as special cases. At the same time, it enables novel hybrid architectures through arbitrary combinations of horizontal and vertical reassembly. In the following section, we introduce a bi-level multi-objective evolutionary optimization procedure that operates directly on the proposed chromosome representation.

3 Bi-level Multi-objective Evolutionary Optimization

The architecture reassembly problem defined in Section 2 induces a discrete, hierarchical, and multi-objective optimization problem over a structured chromosome space. In this section, we describe the evolutionary strategy to approximate its solution. For the purpose of optimization, we explicitly decompose the chromosome $\mathcal{C}$ into two hierarchically coupled components:

$$\mathcal{C} = (\mathcal{C}^{\mathrm{v}}, \mathcal{C}^{\mathrm{h}}), \tag{7}$$

where $\mathcal{C}^{\mathrm{v}}$ denotes the vertical chromosome, which specifies the depth-wise arrangement of blocks across source models, including layer ordering, depth mappings, and vertical substitutions; $\mathcal{C}^{\mathrm{h}}$ denotes the horizontal chromosome, which specifies, for each depth, the set of participating source blocks, the corresponding assembly operator o_k, and the associated glue-layer topology. Importantly, $\mathcal{C}^{\mathrm{h}}$ is conditionally defined on $\mathcal{C}^{\mathrm{v}}$. Let $\Omega(\mathcal{C}^{\mathrm{v}})$ denote the set of all horizontal chromosomes that are structurally compatible with a given vertical configuration. The feasible chromosome space is therefore $\Omega = \left\{(\mathcal{C}^{\mathrm{v}}, \mathcal{C}^{\mathrm{h}}) \mid \mathcal{C}^{\mathrm{v}} \in \Omega^{\mathrm{v}},\ \mathcal{C}^{\mathrm{h}} \in \Omega(\mathcal{C}^{\mathrm{v}})\right\}$. This asymmetric dependency induces an intrinsic bi-level structure: vertical decisions constrain the admissible space of horizontal recombination, but not vice versa.

Let $\mathcal{T} = \{T_1, \ldots, T_m\}$ be a set of evaluation tasks. For any feasible chromosome $\mathcal{C} \in \Omega$, we define a vector-valued fitness function

$$\mathbf{F}(\mathcal{C}) = \big(f_1(\mathcal{C}), \ldots, f_m(\mathcal{C})\big), \tag{8}$$

where $f_j(\mathcal{C})$ measures the task performance of the instantiated architecture on T_j. The architecture reassembly problem is then formulated as the following bi-level multi-objective optimization:

$$\begin{aligned} \max_{\mathcal{C}^{\mathrm{v}} \in \Omega^{\mathrm{v}}} \quad & \mathbf{F}\Big(\mathcal{C}^{\mathrm{v}}, \mathcal{C}^{\mathrm{h}*}\Big) \\ \text{s.t.} \quad & \mathcal{C}^{\mathrm{h}*} = \arg \max_{\mathcal{C}^{\mathrm{h}} \in \Omega(\mathcal{C}^{\mathrm{v}})} \mathbf{F}\Big(\mathcal{C}^{\mathrm{v}}, \mathcal{C}^{\mathrm{h}}\Big). \end{aligned} \tag{9}$$

In this formulation, the upper-level optimization explores alternative vertical architectures, while the lower-level optimization searches for Pareto-optimal horizontal compositions conditioned on a fixed vertical structure. The objective at both levels is vector-valued, reflecting task-dependent trade-offs rather than a single scalar criterion.

For any chromosome $\mathcal{C}$ generated during evolution, the corresponding executable model is obtained via a deterministic instantiation process. Specifically, pretrained Transformer blocks are assembled according to $(\mathbf{C}, \mathbf{o})$, and glue layers are inserted wherever heterogeneous block transitions occur. Let $\mathcal{I}(\mathcal{C})$ denote the instantiated model. The parameters of glue layers in $\mathcal{G}$ are obtained by applying the data-free knowledge distillation procedure described in Section 4. Crucially, glue-layer optimization is not part of the bi-level problem in Eq. (9), but rather a deterministic realization step. Fitness values $f_j(\mathcal{C})$ are computed by evaluating $\mathcal{I}(\mathcal{C})$ on small validation sets for each task T_j, with all pretrained blocks frozen. This evaluation protocol ensures that $\mathbf{F}(\mathcal{C})$ reflects architectural merit rather than training intensity.

The optimization problem in Eq. (9) is discrete, combinatorial, and non-differentiable. We therefore adopt a population-based evolutionary algorithm (Deb et al., 2002) to approximate its Pareto-optimal solution set. Evolution proceeds hierarchically. At each generation, vertical chromosomes $\mathcal{C}^{\mathrm{v}}$ are first evolved via mutation and crossover, generating candidate depth-wise architectures. For each vertical candidate, a conditional population of horizontal chromosomes $\mathcal{C}^{\mathrm{h}} \in \Omega(\mathcal{C}^{\mathrm{v}})$ is then explored to identify non-dominated compositions. Selection is performed based on Pareto dominance with respect to $\mathbf{F}$. By iteratively alternating between vertical exploration and conditional horizontal refinement, the algorithm can effectively navigate the two-dimensional reassembly space.

4 Data-free Glue Layer Alignment via Knowledge Distillation

A major challenge in architecture-level reassembly arises from representational incompatibilities between Transformer blocks originating from different source models. Even when blocks are semantically comparable, differences in hidden dimensionality, feature distributions, or internal architectural conventions may prevent direct composition. To address this issue without relying on task-specific data, we propose a data-free glue layer alignment mechanism based on knowledge distillation (Gou et al., 2021).

Let $\tilde{B}^k$ and $\tilde{B}^{k+1}$ denote two consecutive composite blocks in the reassembled architecture, connected through a glue layer G_k. The goal of glue layer alignment is to ensure that the information

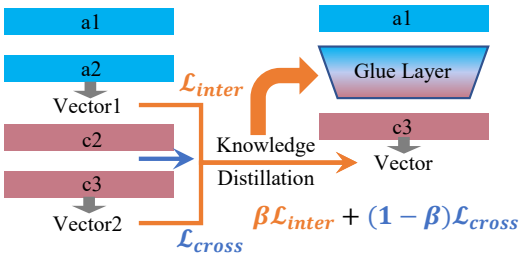


Figure 2: The design of the glue layer between two different layers.

passed from $\tilde{B}^k$ to $\tilde{B}^{k+1}$ in the reassembled model remains consistent with the behaviors of the original source models from which these blocks are drawn. Let $L_{i,j}$ denote the j-th Transformer block of source model M_i. Suppose that $\tilde{B}^k$ originates from model M_i at depth j, and $\tilde{B}^{k+1}$ originates from model $M_{i'}$ at depth j'. Let $\mathbf{z}_k$ denote the output representation of $\tilde{B}^k$ in the reassembled model, and let $\mathbf{z}_{k+1}$ denote the output of $\tilde{B}^{k+1}$ after applying the glue layer.

To preserve the data-free nature of the framework, glue layer alignment does not rely on any downstream task data or original training data. Instead, we generate a set of synthetic input representations $\mathbf{x}$ drawn from a simple prior distribution, such as a multivariate Gaussian[1]. These inputs are fed independently into both the reassembled model and the corresponding source models to obtain supervisory signals at the block level. Formally, for a sampled input $\mathbf{x}$, the forward pass in the reassembled model yields

$$\mathbf{z}_{k+1} = \tilde{B}^{k+1}\big(G_k(\tilde{B}^k(\mathbf{x}))\big). \tag{10}$$

In parallel, the same input $\mathbf{x}$ is forwarded through the relevant blocks of the source models to obtain reference representations. As shown in Figure 2, the proposed glue layer alignment is implemented through a dual distillation mechanism that supervises representation compatibility at two complementary levels. Specifically:

Inter-layer Distillation: The first component of the alignment objective enforces consistency between the behavior of a block in the reassembled model and the behavior of the same block in its original source model. This objective encourages the reassembled architecture to preserve the functional characteristics of individual pretrained blocks. Let $\mathbf{z}^{\text{src}}_{i',j'}$ denote the output of block $L_{i',j'}$ in its original source model when processing input $\mathbf{x}$. The inter-layer distillation loss is defined as

$$\mathcal{L}_{\text{inter}} = \left\|\mathbf{z}_{k+1} - \mathbf{z}^{\text{src}}_{i',j'}\right\|_2^2 + \alpha \left\|\mathbf{z}_{k+1} - \mathbf{z}^{\text{src}}_{i,j+1}\right\|_2^2. \tag{11}$$

This term ensures that, despite being embedded in a reassembled architecture, the composite block $\tilde{B}^{k+1}$ produces representations that remain close to its pretrained behavior, while also respecting the representation geometry induced by its original successor block. By anchoring the block output to both its standalone behavior and its original inter-layer context, the inter-layer distillation stabilizes block semantics under architectural recomposition.

Cross-layer Distillation: While inter-layer distillation preserves block-level behavior, it does not guarantee compatibility between consecutive blocks originating from different models. To address this issue, we introduce a cross-layer distillation objective that aligns the interface between $\tilde{B}^k$ and $\tilde{B}^{k+1}$. Let $\mathbf{z}^{\text{src}}_{i,j}$ denote the output of block $L_{i,j}$ in its original source model. The cross-layer distillation loss is defined as

$$\mathcal{L}_{\text{cross}} = \left\|G_k(\mathbf{z}_k) - \mathbf{z}^{\text{src}}_{i',j'-1}\right\|_2^2, \tag{12}$$

where $\mathbf{z}^{\text{src}}_{i',j'-1}$ represents the input that block $L_{i',j'}$ would have received from its preceding block in the original model. This objective encourages the glue layer to transform the output of $\tilde{B}^k$ into a representation that is compatible with the expected input distribution of $\tilde{B}^{k+1}$.

The final glue layer alignment loss is defined as a weighted combination of the inter-layer and cross-layer distillation terms:

$$\mathcal{L}_{\text{align}} = \beta\,\mathcal{L}_{\text{inter}} + (1-\beta)\,\mathcal{L}_{\text{cross}}, \tag{13}$$

where $\beta \in [0,1]$ controls the trade-off between preserving block-level behavior and enforcing cross-block compatibility. The value of β, together with the architectural configuration encoded in $\mathcal{C}$, is determined implicitly by the evolutionary optimization process described in Section 3. For each candidate architecture, glue layers are optimized independently by minimizing $\mathcal{L}_{\text{align}}$, after which the resulting model is evaluated for fitness.

The proposed data-free glue layer alignment mechanism enables flexible composition of heterogeneous Transformer blocks without requiring access to original training data or downstream task supervision. By leveraging block-level knowledge

[1]Under common conditions, deep representations exhibit approximately Gaussian behavior or converge to stable signal propagation regimes (Kedia et al., 2024; Si et al., 2025).

Table 1: Performance comparison between source models and the reassembled model in the toy example.

	Sum		English		Chinese		French		Spanish		Arabic		Turkish	
Model	ACC	std	ACC	std	ACC	std	ACC	std	ACC	std	ACC	std	ACC	std
LLaMA-2-7B	0.4109	0.0178	0.5221	0.0183	0.3748	0.0177	0.4632	0.0183	0.3949	0.0179	0.3467	0.0174	0.3641	0.0176
Open-LLaMA-7B	0.4096	0.0154	0.5241	0.0158	0.3745	0.0153	0.4679	0.0158	**0.4197**	0.0156	0.3323	0.0149	0.3394	0.0150
LEGO-LLM	**0.4276**	0.0302	**0.5534**	0.0304	**0.3813**	0.0297	**0.4693**	0.0298	0.3993	0.0307	**0.3735**	0.0300	**0.3886**	0.0306

distillation from pretrained source models, glue layers act as adaptive interfaces that reconcile representational discrepancies introduced by architectural reassembly. This design allows architectural exploration to focus on structural innovation, while compatibility issues are resolved through lightweight, localized optimization.

5 Experiments (Detailed in Appendix C)

We evaluate the LEGO-LLM through a series of experiments designed to reveal both its structural behavior and empirical effectiveness. Experiments are conducted in cross-lingual LLM settings, where heterogeneous objectives and language-specific inductive biases naturally align with our multi-objective evolutionary formulation. We begin with a controlled toy example to provide an illustration of the resulting architectural patterns, and then scale to a multilingual repository of language-specific LLMs evaluated on a range of monolingual and cross-lingual benchmarks. The detailed experimental settings are provided in **Appendix** C.1. The implementation of LEGO-LLM is available at `https://github.com/wuxingyu-ai/LEGO-LLM`.

(1) Experiments on Toy Example: We first introduce a controlled toy example to illustrate the behavior of the LEGO-LLM in a minimal yet interpretable setting. This experiment considers two widely used and architecturally compatible 7B-scale LLMs, LLaMA-2-7B-Chat and Open-LLaMA-7B, which are commonly adopted in prior heterogeneous model reuse studies (Wan et al., 2024; Zeng et al., 2025). As shown in Table 1, we evaluate the reassembled architectures on monolingual tasks using XNLI (Conneau et al., 2018), and the reassembled model achieves the best overall performance, outperforming both source models on the aggregated metric and consistently improving accuracy across most individual languages, including English, Chinese, Arabic, and Turkish. To further interpret these gains, Figure 3 visualizes the final chromosome of the reassembled model, revealing heterogeneous and non-aligned reuse patterns: blocks from different depths are interleaved, horizontal composition operators such as $\mathcal{O}^{\text{merge}}$ and $\mathcal{O}^{\text{ens}}$ are frequently selected, and single-source layers remain at specific depths. These observations highlight a fundamental distinction from parameter-space merging, demonstrating that architecture-level reassembly can discover flexible, non-monotonic structural compositions that effectively integrate complementary capabilities even in a constrained two-model setting.

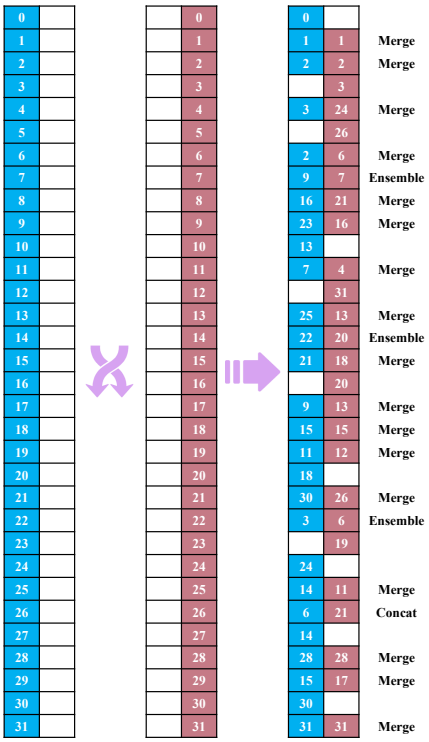


Figure 3: Architecture distribution of the reassembled model in the toy example. Each layer illustrates the selected source model (by colors), original layer indices, and the horizontal assembly operator.

(2) In the following, we scale the framework to a more realistic setting by reassembling a deeper LLM from **6 language-specific fine-tuned models**, and systematically compare the resulting architectures with pretrained baselines, multilingual fine-tuning, and parameter-space reuse methods.

Results on Monolingual Tasks: We evaluate the reassembled architectures on monolingual tasks using MELA (Zhang et al., 2023) and XNLI (Con-

Table 2: Performance Comparison on Mela Dataset

Mela	English	Chinese	French	Spanish	Arabic	Turkish
	Mcc ± Std	Mcc ± Std	Mcc ± Std	Mcc ± Std	Mcc ± Std	Mcc ± Std
Base LLM	0.3279±0.0419	0.1043±0.034	0.0731±0.0388	0.0334±0.0366	0.1693±0.0386	-
Arabic LLM	0.4900±0.0402	0.2423±0.0329	0.1723±0.0286	0.2174±0.0339	0.1535±0.0334	-
Chinese LLM	0.1031±0.0478	0.2307±0.0344	0.0731±0.0389	0.1034±0.0350	0.0485±0.0096	-
French LLM	0.4370±0.0336	0.3070±0.0267	0.1829±0.0212	0.2384±0.0265	0.0875±0.0242	-
English LLM	0.3968±0.0385	0.1428±0.0482	0.0733±0.0436	0.0371±0.0401	0.1829±0.0286	-
Spanish LLM	0.2126±0.0213	0.0245±0.0156	0.0103±0.0065	0.1108±0.0101	0.0000±0.0000	-
Turkish LLM	**0.5242±0.0411**	0.2503±0.0335	0.1966±0.0325	0.2021±0.0357	0.1222±0.0306	-
Multilingual LLM	0.4828±0.0408	**0.3321±0.0314**	0.1705±0.0303	0.2328±0.0345	0.1002±0.0307	-
Dare-Ties	0.4527±0.0387	0.3048±0.0320	0.1949±0.0291	0.2270±0.0333	0.1129±0.0301	-
Ties	0.5008±0.0397	0.3073±0.0329	0.1980±0.0290	0.2380±0.0342	0.1322±0.0310	-
LEGO-LLM	0.4952±0.0147	0.3138±0.0560	**0.2050±0.0114**	**0.2426±0.0486**	⋆ **0.3435±0.0136** ⋆	-

Table 3: Performance Comparison on Xnli Dataset

Xnli	English	Chinese	French	Spanish	Arabic	Turkish
	Acc ± Std	Acc ± Std	Acc ± Std	Acc ± Std	Acc ± Std	Acc ± Std
Base LLM	0.5333±0.0095	0.5522±0.0100	0.5225±0.0100	0.5028±0.0100	0.4851±0.0100	0.3811±0.0097
Arabic LLM	0.5221±0.0095	0.5590±0.0100	0.5112±0.0100	0.5321±0.0100	0.4313±0.0099	0.3502±0.0096
Chinese LLM	0.5060±0.0095	0.5562±0.0100	0.5092±0.0100	0.5514±0.0100	0.4582±0.0100	0.3671±0.0097
French LLM	0.3936±0.0095	0.4337±0.0099	0.5100±0.0100	0.5104±0.0100	0.4755±0.0100	0.3855±0.0098
English LLM	0.5522±0.0095	0.5333±0.0100	0.5233±0.0100	0.5297±0.0100	0.4707±0.0100	0.3972±0.0098
Spanish LLM	0.5317±0.0095	0.5317±0.0100	0.5241±0.0100	0.5181±0.0100	0.4803±0.0100	0.3735±0.0097
Turkish LLM	0.4900±0.0095	0.5462±0.0100	0.5261±0.0100	0.5390±0.0100	0.4900±0.0100	0.3739±0.0097
Multilingual LLM	0.4940±0.0095	0.5570±0.0100	0.5080±0.0100	0.5124±0.0100	0.4578±0.0100	0.4028±0.0098
Dare-Ties	0.3277±0.0095	0.3277±0.0094	0.3353±0.0095	0.3357±0.0095	0.3281±0.0094	0.3265±0.0094
Ties	0.5185±0.0095	0.5185±0.0100	0.5116±0.0100	0.5337±0.0100	0.4530±0.0100	0.3863±0.0098
LEGO-LLM	**0.5546±0.0213**	**0.5630±0.0124**	**0.5301±0.0224**	**0.5541±0.0224**	**0.4998±0.0223**	**0.4065±0.0217**

neau et al., 2018). As shown in Table 2 and Table 3, LEGO-LLM achieves strong and consistent performance across all languages, effectively integrating the strengths of multiple language-specific LLMs without access to multilingual training data. On XNLI, the reassembled model attains competitive accuracy across all evaluated languages and matches or exceeds the multilingual fine-tuned baseline in several cases, indicating that cross-lingual knowledge can be structurally reused through architectural recomposition. Moreover, LEGO-LLM consistently outperforms parameter-space baselines such as Dare-TIES and TIES, highlighting the advantage of structural recomposition over parameter interpolation. Notably, the improved performance on lower-resource languages such as Arabic in MELA reflects the effect of fairness-aware optimization, which discourages architectures that sacrifice individual language performance, resulting in balanced and robust monolingual performance.

(3) While monolingual evaluations assess language-specific competence, cross-lingual tasks are essential for revealing whether reassembled architectures can structurally integrate and coordinate complementary capabilities inherited from different source models.

Results on Cross-Lingual Tasks: We further assess cross-lingual generalization on LogQA (Huang et al., 2023), MELA (Zhang et al., 2023), MMLU (Hendrycks et al., 2021), and XNLI (Conneau et al., 2018), with results summarized in **Appendix** C.4. The LEGO-LLM outperforms the multilingual fine-tuned baseline on three of the four datasets, with particularly strong gains on XNLI, and achieve comparable performance on LogQA. Compared to monolingual LLMs, which exhibit substantial degradation on cross-lingual tasks, the reassembled architectures maintain consistently strong results across all datasets, demonstrating effective cross-language transfer enabled by vertical and horizontal block recomposition. LEGO-LLM also consistently outperforms parameter-space reuse methods, reinforcing that enabling architectural diversity yields a more expressive and effective form of model reuse for cross-lingual generalization.

We also evaluate cross-cultural understanding

Table 4: Experimental Results of Time and GPU Memory Consumption

Dataset	Time(h)			GPU Memory(GB)		
	EA	FT	PM	EA	FT	PM
Mela	1.25	2.00	0.095	20.5	41.0	29.0
Xnli	1.88	2.50	0.095	22.0	41.5	29.0
SeaEval	5.00	4.00	0.095	25.0	42.5	29.5
Flores	6.25	6.66	0.095	26.5	43.0	29.5

and translation on SeaEval (Wang et al., 2024a) and Flores, to provide a stringent test of whether reassembled models can go beyond any single source model by integrating complementary language competencies through architectural composition. Results are reported in **Appendix** C.5. These results demonstrate that LEGO-LLM enables robust cross-cultural and cross-lingual transfer in both understanding and translation tasks.

(4) The evolutionary optimization operates in a parameter-frozen regime. The only trainable components are lightweight glue modules with negligible parameter overhead. Each candidate architecture is evaluated via forward-only inference without full retraining. Hence, the cost is similar to the standard fine-tuning, which requires full backpropagation through the backbone. Moreover, this process is naturally parallelizable across individuals in the population, making large-scale architecture search feasible under distributed settings. In Table 4, we provide a comparison among fine-tuning (FT), evolutionary architecture reassembly (EA), and parameter merging (PM).

The results show that the total time cost of architecture reassembly is comparable to fine-tuning, while higher than parameter merging. Note that architecture reassembly targets a different reuse regime compared to parameter merging. Parameter merging aims to preserve and average capabilities across source tasks, typically seeking to approximate each original model. In contrast, architecture reassembly explores structural recomposition to potentially surpass individual source models on specific tasks. For example, in Table 2 (Arabic task), the reassembled model significantly outperforms all source models, an effect unlikely to emerge from standard parameter merging. This difference in objective partly explains the additional computational overhead. That said, we do not claim that Architecture Reassembly replaces fine-tuning or merge-based reuse. Rather, it explores a complementary axis of model reuse at architectural level.

Table 5: Performance Comparison of Different Layers

	Mela	XNLI	SeaEval	Flores
No glue layer	0.2423	0.4089	0.5068	0.2694
Real tokens	0.3024	0.4868	0.5631	0.3272
Our Glue Layer	0.3200	0.5163	0.5940	0.3460

(5) We conduct the ablation study and compare: (i) No glue layer (dimension mismatch handled via a single MLP projection), (ii) Glue layer trained on synthetic Gaussian data (our default), (iii) Glue layer trained on Real tokens (50% XNLI samples). The results are provided in Table 5. The glue layer substantially improves performance compared to direct projection, confirming that structural alignment is critical. Using limited real tokens does not outperform Gaussian training due to data scarcity: 50% of XNLI samples is still relatively small for stabilizing cross-model representation alignment, whereas synthetic Gaussian inputs can be generated at unlimited scale, enabling more robust estimation of alignment statistics. This suggests that the glue layer primarily learns distributional alignment rather than task-specific adaptation.

6 Conclusion

This work introduces architecture-level reassembly as a new paradigm for reusing LLMs. Instead of treating LLMs as indivisible artifacts, we demonstrate that Transformer blocks can be recomposed as modular components to construct new, functional architectures without access to training data. We formalize a two-dimensional reassembly space that supports both vertical recombination across depth and horizontal composition within layers, and propose a chromosome-based representation with bi-level multi-objective evolutionary optimization to efficiently explore this space. To enable valid information flow between heterogeneous blocks, we introduce lightweight glue layers trained via data-free knowledge distillation. Our findings suggest a shift toward a modular and evolutionary view of LLM design. For future work, we plan to extend the framework to fully heterogeneous LLMs with divergent engineering configurations, including harmonization mechanisms for normalization conventions, positional encoding schemes, attention masking rules, residual connection behaviors, and hidden dimension mismatches.

Limitations

Architecture reassembly also introduces new challenges. Because the final model emerges from an evolutionary search over discrete architectural choices, its performance can exhibit higher variance compared to conventionally fine-tuned models. This variability reflects sensitivity to block selection and composition strategies rather than instability in parameter optimization. In addition, evolutionary search may incur nontrivial computational overhead when the search space or iteration budget becomes large. However, this limitation is partially mitigated by the inherent parallelism of evolutionary algorithms and by the fact that the search relies primarily on inference rather than gradient-based training.

Acknowledgments

This work was supported in part by the Research Grants Council of the Hong Kong SAR (Grant No. C5052-23G, PolyU15229824, SRFS2526-5S04), and The Hong Kong Polytechnic University (Project IDs: P0051130, P0060651, P0058445).

References

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

Pin-Yu Chen. 2024. Model reprogramming: Resource-efficient cross-domain machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 22584–22591.

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

Xingyu Dang, Christina Baek, Kaiyue Wen, Zico Kolter, and Aditi Raghunathan. 2025. Weight ensembling improves reasoning in language models. *arXiv preprint arXiv:2504.10478*.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630.

Shangbin Feng, Wenxuan Ding, Alisa Liu, Zifeng Wang, Weijia Shi, Yike Wang, Zejiang Shen, Xiaochuang Han, Hunter Lang, Chen-Yu Lee, and 1 others. 2025. When one llm drools, multi-llm collaboration rules. *arXiv preprint arXiv:2502.04506*.

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

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Neel Guha, Mayee F Chen, Trevor Chow, Ishan S Khare, and Christopher Ré. 2024. Smoothie: label free language model routing. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pages 127645–127672.

Lan-Zhe Guo, Zhi Zhou, Yu-Feng Li, and Zhi-Hua Zhou. 2023. Identifying useful learnwares for heterogeneous label spaces. In *Proceedings of the International Conference on Machine Learning*, pages 12122–12131. PMLR.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *Proceedings of the 9th International Conference on Learning Representations*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *Proceedings of the 36th International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations*.

Shaohan Huang, Yi Liu, Carol Fung, Jiaxing Qi, Hailong Yang, and Zhongzhi Luan. 2023. Logqa: Question answering in unstructured logs. *arXiv preprint arXiv:2303.11715*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Zifeng Wang, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. 2025. Universal llm routing with correctness-based representation. In *Proceedings of the First Workshop*

on Scalable Optimization for Efficient and Adaptive Foundation Models.

Akhil Kedia, Mohd Abbas Zaidi, Sushil Khyalia, Jungho Jung, Harshith Goka, and Haejun Lee. 2024. Transformers get stable: An end-to-end signal propagation theory for language models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 23449–23531. PMLR.

Sanghoon Kim, Dahyun Kim, Chanjun Park, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, and 1 others. 2024. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 23–35.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1964–1974.

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

Jupinder Parmar, Sanjev Satheesh, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Reuse, don't retrain: A recipe for continued pretraining of language models. *arXiv preprint arXiv:2407.07263*.

Jonas Pfeiffer, Sebastian Ruder, Ivan Vulic, and Edoardo M Ponti. 2023. Modular deep learning. *Transation on Machine Learning Research*.

Haiyun Qiu, Xingyu Wu, Liang Feng, and Kay Chen Tan. 2026. Fine-grained model merging via modular expert recombination. *arXiv preprint arXiv:2602.06552*.

Chongjie Si, Jingjing Jiang, and Wei Shen. 2025. Unveiling the mystery of weight in large foundation models: Gaussian distribution never fades. *arXiv preprint arXiv:2501.10661*.

Sidak Pal Singh and Martin Jaggi. 2020. Model fusion via optimal transport. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 22045–22055.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Proceedings of the 31st Annual Conference on Advances in neural information processing systems*, 30.

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

Bin Wang, Zhengyuan Liu, Xin Huang, Fangkai Jiao, Yang Ding, Aiti Aw, and Nancy Chen. 2024a. Seaeval for multilingual foundation models: From cross-lingual alignment to cultural reasoning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 370–390.

Junlin Wang, WANG Jue, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024b. Mixture-of-agents enhances large language model capabilities. In *Proceedings of the 13th International Conference on Learning Representations*.

Qiu-Feng Wang, Xin Geng, Shu-Xia Lin, Shi-Yu Xia, Lei Qi, and Ning Xu. 2022. Learngene: From open-world to your learning task. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8557–8565.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and 1 others. 2022. 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*, pages 23965–23998. PMLR.

Xingyu Wu, Jibin Wu, Yu Zhou, Liang Feng, and Kay Chen Tan. 2025. Towards robustness and explainability of automatic algorithm selection. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 67864–67887. PMLR.

Xingyu Wu, Sheng-hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. 2024. Evolutionary computation in the era of large language model: Survey and roadmap. *IEEE Transactions on Evolutionary Computation*, 29(2):534–554.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2024. Ties-merging: resolving interference when merging models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 7093–7115.

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

Xingyi Yang, Daquan Zhou, Songhua Liu, Jingwen Ye, and Xinchao Wang. 2022. Deep model reassembly. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 25739–25753.

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

Runjia Zeng, James Chenhao Liang, Cheng Han, Zhiwen Cao, Jiahao Liu, Xiaojun Quan, Yingjie Victor Chen, Lifu Huang, Tong Geng, Qifan Wang, and 1 others. 2025. Probabilistic token alignment for large language model fusion. *arXiv preprint arXiv:2509.17276*.

Ziyin Zhang, Yikang Liu, Weifang Huang, Junyu Mao, Rui Wang, and Hai Hu. 2023. Mela: Multilingual evaluation of linguistic acceptability. *arXiv preprint arXiv:2311.09033*.

Xun Zhou, Xingyu Wu, Liang Feng, Zhichao Lu, and Kay Chen Tan. 2025a. Design principle transfer in neural architecture search via large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23000–23008.

Yu Zhou, Xingyu Wu, Jibin Wu, Liang Feng, and KC Tan. 2025b. Hm3: Hierarchical multi-objective model merging for pretrained models. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*.

Appendix

A Related Work

The most established paradigm for reusing pretrained language models treats a pretrained model as a monolithic artifact and focuses on adapting its parameters to downstream tasks. Typical approaches include continued pretraining, fine-tuning, and parameter-efficient adaptation methods such as adapter tuning (Houlsby et al., 2019) and low-rank adaptation (Hu et al., 2022). These methods assume a fixed architecture and reuse pretrained models by modifying or augmenting their parameters, thereby enabling efficient transfer without training from scratch. Beyond adapting a single model, recent studies have explored parameter-level model merging (Yang et al., 2024) as a data-free reuse strategy. Representative approaches include weight averaging (Wortsman et al., 2022), Fisher-weighted merging (Matena and Raffel, 2022), and optimal-transport-based fusion (Singh and Jaggi, 2020), which combine multiple fine-tuned models into a single model without additional training. While effective in certain scenarios, these methods generally rely on strong structural assumptions, requiring architectural alignment across source models and enforcing a one-to-one correspondence between parameters or layers. As a result, they do not support heterogeneous architectures or flexible structural recomposition.

Another line of work investigates reusing multiple models through ensemble or routing-based collaboration. In these approaches, different models are combined at inference time, either by aggregating their outputs or by dynamically selecting which model should handle a given input. Recent advances extend classical mixture-of-experts frameworks (Jacobs et al., 1991) to large language models by learning routers that assign queries to specialized pretrained models (Guha et al., 2024; Jitkrittum et al., 2025; Lu et al., 2024). In parallel, multi-LLM collaboration frameworks explore how multiple pretrained language models can jointly solve complex tasks, either sequentially or in parallel, to improve robustness and reasoning ability (Feng et al., 2025; Wang et al., 2024b). Although such approaches effectively exploit model diversity and complementarity, they treat individual models as black-box experts. The collaboration occurs at the decision or output level, without exposing or reusing the internal representations or architectural components of the models. Consequently, these methods do not address the problem of structural reuse, nor do they enable the construction of new architectures by recomposing LLM components.

To enable finer-grained reuse beyond whole models, recent studies have begun to investigate model assembly and layer-wise reuse (Pfeiffer et al., 2023). Deep Model Reassembly demonstrates that pretrained neural networks can be decomposed into blocks and vertically recombined across depths to construct new models (Yang et al., 2022). Related efforts further explore modularizing networks into reusable components or learnable genes, enabling more flexible transfer across tasks and architectures (Wang et al., 2022). These methods provide important evidence that LLMs are not indivisible and that block-level reuse is feasible (Wu et al., 2024). However, existing approaches typically impose strong structural assumptions. In particular, each layer is assumed to originate from a single source model, and compatibility between consecutive layers is implicitly guaranteed by construction. As a result, reuse is largely confined to vertical recombination along network depth, and more general forms of

composition, such as combining multiple heterogeneous blocks within the same layer, remain unexplored. Moreover, representation mismatch across heterogeneous blocks is not treated as a first-class problem, further restricting the scope of feasible reassembly (Guo et al., 2023). To the best of our knowledge, no prior work has systematically studied architecture-level reassembly of heterogeneous models that jointly supports vertical and horizontal composition without task data.

B Measurement of Fairness across Different Languages

In the context of our multilingual large model reassembly experiments, we introduce fairness across different languages as an extra objective due to its paramount importance. To quantitatively measure this fairness, we leverage the concept of semantic entropy as introduced in (Farquhar et al., 2024). Semantic entropy provides a means to assess the uncertainty in the meaning of the generated text by the model. In our approach, for a given language l from the set of multilingual set $\mathcal{L} = \{l_1, l_2, \cdots, l_n\}$, we first conduct experiments on the sub-dataset $\mathcal{D}_l$ specific to language l. Let the model M generate responses r_i^l for input samples $x_i^l \in \mathcal{D}_l$, where $i = 1, 2, \cdots, N_l$ and N_l is the number of samples in $\mathcal{D}_l$.

To calculate the semantic entropy $H_s(r^l)$ of the responses in language l, we sample multiple possible answers $\{a_{ij}^l\}$ for each response r_i^l. These answers are then clustered into groups $\mathcal{G}_{ik}^l$ based on a bidirectional entailment relationship, where $k = 1, 2, \cdots, K_{il}$ and K_{il} is the number of clusters for the i-th response in language l. The probability p_{ik}^l of a cluster $\mathcal{G}_{ik}^l$ is calculated as the proportion of answers belonging to that cluster among all sampled answers for the corresponding response, i.e.,

$$p_{ik}^l = \frac{|\{a_{ij}^l \in \mathcal{G}_{ik}^l\}|}{\sum_j |\{a_{ij}^l\}|} \tag{14}$$

The semantic entropy $H_s(r^l)$ is then computed using the Shannon entropy formula adapted to this clustered structure:

$$H_s(r^l) = -\sum_{i=1}^{N_l} \sum_{k=1}^{K_{il}} p_{ik}^l \log(p_{ik}^l) \tag{15}$$

Once we have calculated the semantic entropy for each language $l \in \mathcal{L}$, we define fairness as the variance of these semantic entropies across all languages. Let μ be the mean of the semantic entropies, calculated as:

$$\mu = \frac{1}{n} \sum_{l \in \mathcal{L}} H_s(r^l) \tag{16}$$

The variance σ^2, which serves as our fairness metric F, is computed as:

$$F = \sigma^2 = \frac{1}{n} \sum_{l \in \mathcal{L}} (H_s(r^l) - \mu)^2 \tag{17}$$

A lower value of the fairness metric F indicates that the model's performance, as measured by semantic entropy, is more balanced across different languages. In other words, the model is less likely to exhibit significant disparities in the uncertainty of generated text among the various languages it handles, suggesting a more equitable treatment of different languages during the reassembly process.

C Experiment Details

We conduct a comprehensive set of experiments to evaluate the proposed architecture-level reassembly framework from both mechanistic and empirical perspectives. Rather than focusing solely on large-scale performance comparisons, our experimental design aims to answer a sequence of increasingly complex questions: how reassembled architectures emerge from heterogeneous pretrained blocks, whether such architectures can effectively integrate complementary capabilities, and how the proposed framework compares with parameter-space reuse methods in realistic cross-lingual settings.

We evaluate the proposed architecture-level reassembly framework in a cross-lingual LLM setting. Cross-lingual tasks provide a particularly suitable testbed for our method for two reasons. First, multilingual competence inherently involves multiple heterogeneous objectives across languages, which naturally aligns with our multi-objective evolutionary formulation. Second, language-specific LLMs exhibit complementary strengths and representational biases, making cross-lingual scenarios ideal for assessing whether reusable Transformer blocks can be structurally recomposed to integrate diverse capabilities within a single architecture.

Specifically, we begin with a controlled toy example involving two widely used pretrained LLMs to provide an interpretable illustration of the reassembly process and its resulting architectural patterns. We then scale up to a multilingual repository of language-specific LLMs and evaluate the

framework on a diverse suite of monolingual and cross-lingual benchmarks, including natural language understanding, reasoning, cultural understanding, and translation tasks. In addition to task accuracy, we incorporate fairness across languages as an explicit optimization objective to demonstrate the flexibility of our multi-objective formulation. Across all experiments, we compare architecture-level reassembly against both pretrained baselines and parameter-space reuse methods, enabling a systematic assessment of the benefits of structural recomposition.

C.1 Experimental Setup

All experiments are conducted using two NVIDIA A6000 GPUs. We consider a repository of language-specific LLMs derived from the LLaMA-3.1 8B backbone[2], each fine-tuned on a single language. These models include English (Meta-Llama-3-8B-Instruct[3]), Chinese (Llama-3-8B-Instruct-Chinese[4]), French (EnnoAi-Pro-French-Llama-3-8B-v0.4[5]), Turkish (Turkish-Llama-8B-Instruct-v0.1[6]), Arabic (Arabic-Orpo-Llama-3-8B-Instruct[7]), and Spanish (fine-tuned via LoRA on `josecannete/large_spanish_corpus`, using 20% of the original data).

Using these models as a block repository, we apply the proposed architecture reassembly framework to construct new multilingual architectures. The maximum depth of reassembled models is set to 36 Transformer blocks. Architectural search is performed using a bi-level multi-objective evolutionary algorithm based on NSGA-II (Deb et al., 2002). The population size is set to 30. Glue layers are trained using the data-free distillation procedure described in Section 4. Synthetic input sequences of length 128 are sampled from a Gaussian prior. Unless otherwise specified, 1,000 samples are used for distillation, with a batch size of 32. All pretrained Transformer blocks remain frozen throughout the process. At each generation, parent and offspring populations are jointly ranked using Pareto non-dominated sorting and crowding distance, after which the top 30 individuals are retained. Although fairness is explicitly optimized during evolution, final model selection from the Pareto front prioritizes individuals with higher average task accuracy, while still maintaining balanced cross-lingual performance.

We compare three categories of models:

- Pretrained and monolingual fine-tuned LLMs;
- A multilingual fine-tuned baseline, suzume-llama-3-8B-multilingual[8];
- parameter-space reuse baselines constructed using Dare-TIES (Yu et al., 2024) and TIES (Yadav et al., 2024).

These baselines allow us to isolate the contribution of architecture-level reassembly beyond parameter-level reuse. Beyond task accuracy, we explicitly incorporate fairness across languages as an additional optimization objective. This choice is motivated by the flexibility of our framework: since architecture reassembly is formulated as a multi-objective problem, it naturally supports the inclusion of auxiliary criteria without modifying the underlying optimization mechanism. Fairness serves as a concrete example demonstrating that the proposed framework can balance heterogeneous, non-traditional objectives alongside performance.

To quantify fairness, we adopt semantic entropy as proposed in (Farquhar et al., 2024). As detailed in Appendix B, semantic entropy measures uncertainty in the meaning of generated responses rather than surface-level variability, making it particularly suitable for cross-lingual evaluation. For each language $l \in \mathcal{L}$, we compute the semantic entropy of model outputs on the corresponding dataset $\mathcal{D}_l$. Fairness is then defined as the variance of semantic entropy values across languages, with lower variance indicating more balanced behavior. This metric is used jointly with task performance during evolutionary search, encouraging architectures that avoid disproportionately high uncertainty in low-resource or underrepresented languages.

C.2 A Toy Example of Architecture-Level Reassembly

Before presenting large-scale cross-lingual results, we first introduce a controlled toy example to illustrate the behavior of the proposed architecture-level

[2] https://huggingface.co/meta-llama/Meta-Llama-3-8B

[3] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

[4] https://huggingface.co/Rookie/Llama-3-8B-Instruct-Chinese

[5] https://huggingface.co/Enno-Ai/EnnoAi-Pro-French-Llama-3-8B-v0.4

[6] https://huggingface.co/ytu-ce-cosmos/Turkish-Llama-8b-Instruct-v0.1

[7] https://huggingface.co/MohamedRashad/Arabic-Orpo-Llama-3-8B-Instruct

[8] https://huggingface.co/lightblue/suzume-llama-3-8B-multilingual

reassembly framework in a minimal yet representative setting. The goal of this experiment is to provide an interpretable demonstration of how heterogeneous pretrained blocks are selectively reused, combined, and reorganized through the evolutionary reassembly process. This experiment is designed to answer two fundamental questions: (1) whether architecture-level reassembly can consistently outperform its source models even in a highly constrained two-model setting, and (2) how the resulting architecture differs structurally from conventional layer-wise alignment or parameter-space merging. By restricting the model pool to two widely used and architecturally compatible LLMs, this toy example enables direct inspection of the learned reassembly patterns.

We consider two pretrained 7B-scale models as the block repository:

- LLaMA-2-7B-Chat (`meta-llama/Llama-2-7b-chat-hf`)
- Open-LLaMA-7B (`openlm-research/open_llama_7b`)

These two models are commonly adopted in existing heterogeneous model merging and fusion studies (Wan et al., 2024; Zeng et al., 2025), making them a natural baseline for comparison. The experimental protocol follows the same setup as described in the previous subsection, except that the maximum depth of the reassembled architecture is fixed to 32 Transformer blocks, matching the original depth of both source models for ease of analysis and visualization. All pretrained blocks remain frozen, and glue layers are trained using the same data-free distillation procedure.

We evaluate the reassembled models on monolingual tasks using XNLI (Conneau et al., 2018), a cross-lingual sentence representations dataset. Table 1 reports the performance of the two source models and the reassembled model across multiple languages. The reassembled architecture achieves the highest overall average accuracy, outperforming both LLaMA-2-7B and Open-LLaMA-7B on the aggregated metric. Notably, the reassembled model improves performance consistently across most individual languages, including English, Chinese, Arabic, and Turkish. While the performance gains on Spanish are moderate, the overall trend demonstrates that architecture-level reassembly can effectively integrate complementary strengths from heterogeneous pretrained models, even without access to multilingual fine-tuning data. The increased standard deviation of the reassembled model reflects the diversity of architectures explored during evolution, rather than instability in training.

To further understand how performance gains arise, we visualize the final chromosome of the reassembled model alongside the original layer structures of the two source models. Figure 3 illustrates the vertical ordering of layers, the horizontal selection of source blocks, and the assembly operators used at each depth. Several observations can be made. First, the reassembled architecture does not follow a monotonic or depth-aligned selection strategy; instead, blocks from different depths of both source models are interleaved throughout the network. Second, horizontal composition is frequent, with operators such as $\mathcal{O}^{\text{merge}}$ and $\mathcal{O}^{\text{ens}}$ applied at many layers, indicating that direct block reuse alone is often insufficient and that controlled combination of representations is beneficial. Finally, single-source layers remain present at specific depths, suggesting that certain blocks are consistently preferred without modification.

Together, these observations highlight a key distinction between architecture-level reassembly and conventional model merging. Rather than enforcing layer-wise correspondence or parameter interpolation, the proposed framework discovers heterogeneous, non-aligned reuse patterns that are difficult to capture using parameter-space methods. This toy example thus provides an interpretable and concrete illustration of the flexibility and expressiveness of architecture-level LLM reuse.

C.3 Results on Monolingual Tasks

We first evaluate the reassembled models on monolingual tasks using Mela (Zhang et al., 2023), a cross-lingual reasoning dataset, and XNLI (Conneau et al., 2018), a cross-lingual sentence representations dataset. Tables 2 and 3 report the results. Overall, the reassembled architectures demonstrate robust performance across all languages, effectively inheriting and integrating the strengths of multiple language-specific models.

Compared to monolingual fine-tuned LLMs, the reassembled models consistently achieve stronger cross-language generalization, even without access to multilingual training data. On XNLI, the reassembled model attains competitive accuracy across all evaluated languages, matching or exceeding the multilingual fine-tuned baseline in sev-

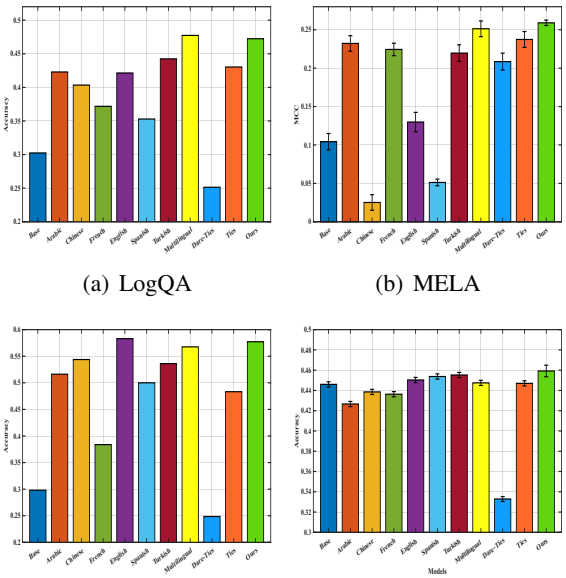
eral cases. This suggests that architecture-level reassembly enables effective reuse of cross-lingual knowledge embedded in pretrained blocks, without relying on additional multilingual supervision.

Notably, the reassembled model outperforms parameter-space baselines such as Dare-TIES (Yu et al., 2024) and TIES (Yadav et al., 2024) on both datasets. This performance gap highlights the importance of structural recomposition beyond parameter interpolation, as the ability to selectively recombine blocks at different depths allows the model to exploit architectural diversity that parameter-level reuse cannot capture. Furthermore, the strong performance on Arabic tasks with traditionally lower resource availability in MELA demonstrates the practical effect of fairness-aware optimization, as the evolutionary process explicitly discourages architectures that degrade performance on individual languages. Furthermore, the reassembled models not only integrate the strengths of language-specific LLMs but also achieve competitive results compared to multilingual fine-tuned models. The additional benefits of architecture-level reassembly ensure balanced and robust performance across diverse linguistic tasks.

C.4 Results on Cross-Lingual Tasks

We further assess cross-lingual capabilities on four diverse datasets: LogQA (Huang et al., 2023), MELA (Zhang et al., 2023), MMLU (Hendrycks et al., 2021), and XNLI (Conneau et al., 2018). These datasets span a range of cross-lingual tasks, including reasoning, inference, and comprehension. The results are summarized in Figure 4. Across three of the four datasets, the reassembled models outperform the multilingual fine-tuned baseline, with particularly pronounced gains on XNLI. These results indicate that the proposed framework effectively integrates knowledge across languages at the architectural level, enabling strong transfer in cross-lingual inference tasks. On the LogQA dataset, the performance of the reassembled models was on par with the multilingual fine-tuned LLM, reflecting their ability to maintain competitive accuracy across reasoning tasks.

Compared to language-specific models, the reassembled architectures exhibit substantially improved cross-lingual generalization. Except for the English LLM on MMLU, which achieves comparable performance, all monolingual models show significant performance degradation on cross-lingual tasks. In contrast, the reassembled models maintain consistently strong results across datasets, reflecting the benefit of combining heterogeneous blocks through vertical and horizontal recomposition. Again, architecture-level reassembly consistently outperforms parameter-space reuse methods. This observation reinforces the central claim of this work: enabling architectural diversity through structured reassembly provides a qualitatively different and more expressive form of model reuse than operating solely in parameter space.



(a) LogQA (b) MELA

(c) MMLU (d) XNLI

Figure 4: The performance comparison on four multilingual datasets.

Together, the experimental results demonstrate that the reassembled models not only inherit the strengths of base and fine-tuned models but also excel in their ability to generalize across languages. By leveraging architecture-level optimization and multi-objective fairness considerations, the reassembled models achieve a balanced, robust performance that is well-suited for diverse cross-lingual tasks. These findings further validate the effectiveness of our approach in addressing the challenges of multilingual tasks.

C.5 Cross-Lingual Culture Understanding and Translation

Finally, we evaluate the models on cross-cultural understanding and translation tasks using the SeaEval (Wang et al., 2024a) and Flores datasets. The SeaEval dataset contains tasks related to multicultural and multilingual understanding across various regions, while the Flores dataset focuses on transla-

Table 6: Performance Comparison on Multicultural Dataset SeaEval

Model	cn_eval	ph_eval	sg_eval	us_eval
Base LLM	0.3456	0.3800	0.2900	0.2600
Arabic LLM	0.3619	0.5600	0.5146	0.6222
Chinese LLM	0.4476	0.4500	0.6408	0.5701
French LLM	0.3143	0.4600	0.4660	0.5327
English LLM	0.5321	0.4200	0.6000	0.6238
Spanish LLM	0.4600	0.5200	0.5600	0.5500
Turkish LLM	0.4476	0.5900	0.5825	0.6916
Multilingual LLM	0.4381	0.5600	0.5825	0.7196
Dare-ties	0.3100	0.3100	0.2600	0.3000
Ties	0.3500	0.5300	0.5400	0.6400
LEGO-LLM	0.4800	0.5600	0.6412	0.6949

Table 7: Performance Comparison on Translation Dataset Flores

Model	ind2eng	vie2eng	zho2eng	zsm2eng
Base LLM	0.2759	0.2895	0.1655	0.1678
Arabic LLM	0.2853	0.2853	0.1943	0.2698
Chinese LLM	0.1054	0.0729	0.0641	0.0813
French LLM	0.3163	0.2654	0.2055	0.3093
English LLM	0.4059	0.3283	0.2504	0.3672
Spanish LLM	0.3607	0.3037	0.2155	0.3420
Turkish LLM	0.3202	0.2675	0.2003	0.3129
Multilingual LLM	0.3591	0.2914	0.2199	0.3660
Dare_ties	0.0003	0.0003	0.0001	0.0003
Ties	0.2533	0.2302	0.1827	0.2233
LEGO-LLM	0.4003	0.3110	0.2832	0.3893

tion tasks between different languages. The results for both datasets are summarized in the Tables 6 and 7.

The SeaEval dataset includes tasks that test the model's ability to understand and process content from diverse cultural perspectives. This dataset evaluates performance on four distinct regions: China (cn_eval), Philippines (ph_eval), Singapore (sg_eval), and United States (us_eval). On SeaEval, the reassembled model achieves competitive performance with the multilingual fine-tuned baseline, particularly on the US and Singapore subsets. While absolute performance varies across regions, the reassembled model exhibits more balanced behavior overall, consistent with the fairness objective incorporated during evolution. Compared to monolingual LLMs, the reassembled architecture demonstrates clear advantages across most regions, indicating improved cultural generalization.

The Flores dataset evaluates translation tasks between several languages to English, including Indonesian (ind2eng), Vietnamese (vie2eng), Chinese (zho2eng), and Malay (zsm2eng). On Flores, the reassembled model performs competitively across all translation directions. Although the multilingual baseline slightly outperforms it on certain language pairs, the reassembled model achieves stronger results on Indonesian and Malay. Parameter-space baselines perform substantially worse, especially Dare-Ties, underscoring the limitations of parameter-only reuse for complex multilingual tasks. These results further demonstrate that architecture-level reassembly enables robust and flexible cross-lingual transfer, even in challenging translation settings.

D Discussion of Architecture Reassembly and Parameter-only Reuse: Complementarity, Trade-offs, and Practical Value

Architecture reassembly represents a fundamentally different paradigm from parameter-only reuse methods, including fine-tuning, parameter-efficient adaptation, and parameter-space merging. While parameter-based approaches operate within a fixed architectural scaffold, architecture reassembly treats the network structure itself as a first-class optimization variable, enabling both vertical (depth-wise) and horizontal (block-wise) reorganization of pretrained components.

One major advantage of architecture reassembly lies in its structural flexibility. By explicitly optimizing model depth, the framework allows the construction of LLMs that better match target computational budgets, deployment constraints, or latency requirements. In contrast, fine-tuning and parameter-efficient methods must adapt performance within an immutable architecture, often leading to suboptimal trade-offs between efficiency and capability. Importantly, it also provides a practical alternative for researchers and practitioners with limited computational resources who aim to develop their own large-scale models. Moreover, architecture reassembly relaxes the performance ceiling imposed by a static layer ordering: by reselecting, reordering, or composing pretrained blocks, it can uncover functional configurations that are inaccessible to parameter updates alone.

Beyond vertical customization, architecture reassembly introduces a horizontal dimension of model reuse that is largely absent from parameter-only methods. Horizontal operators, such as block averaging or representation tying, enable controlled integration of heterogeneous pretrained representations at the same depth. This capability allows the reassembled model to combine complementary

strengths from different source models rather than committing to a single lineage throughout the network.

Another key distinction concerns data dependency and resource efficiency. Architecture reassembly leverages pretrained models as reusable building blocks and relies primarily on inference-time evaluation and lightweight glue-layer training, often through data-free distillation. This significantly reduces reliance on large, high-quality task datasets, which are typically required for effective fine-tuning. As a result, the reassembly process avoids repeated full-model training and instead focuses computational resources on architectural exploration, making it particularly attractive in data-scarce or cross-lingual settings.

Nevertheless, architecture reassembly also introduces new challenges. Because the final model emerges from an evolutionary search over discrete architectural choices, its performance can exhibit higher variance compared to conventionally fine-tuned models. This variability reflects sensitivity to block selection and composition strategies rather than instability in parameter optimization. In addition, evolutionary search may incur nontrivial computational overhead when the search space or iteration budget becomes large. However, this limitation is partially mitigated by the inherent parallelism of evolutionary algorithms and by the fact that the search relies primarily on inference rather than gradient-based training.

Architecture reassembly trades the stability and simplicity of parameter-only reuse for structural expressiveness and compositional flexibility. Rather than replacing fine-tuning or parameter merging, it should be viewed as a complementary paradigm, particularly well suited for scenarios where heterogeneous pretrained models must be integrated, architectural constraints matter, or data availability is limited. This work demonstrates that, under these conditions, optimizing how pretrained components are assembled can be as impactful as optimizing their parameters.