LEGO: Sizing Rules for Budget-Aware VLM Dense-to-MoE Conversion

Qishen Yin[1,*] **ZiangWu**[2,*] **Juntong Wu**[1] **Peng Jin**[1]
Hao Li[1] **Tanghui Jia**[1] **Liuhan Chen**[1] **Bin Zhu**[1] **Li Yuan**[1,3,4,†]
[1]Peking University, Shenzhen Graduate School
[2]School of Software and Microelectronics, Peking University
[3]Rabbitpre Intelligence [4]Peng Cheng Laboratory
[*]Equal contribution. [†]Correspondence to: Li Yuan <yuanli-ece@pku.edu.cn>.

Abstract

We propose LEGO, a budget-aware Dense-to-MoE methodology that converts pretrained vision-language models into efficient FFN-MoE variants under a target architectural budget (P_{total}, P_{active}). Our study is motivated by a counter-intuitive recovery pattern: fixed activation ratios that work well for some dense backbones can degrade sharply on others under the same recovery recipe, indicating that raw sparsity alone is an unreliable design rule. Through controlled recovery experiments across VLM families and scales, we identify the activated-FFN-to-hidden ratio D_I/D_H as a backbone-comparable structural factor and summarize empirical sizing rules for avoiding severe bottlenecks while limiting diminishing returns. LEGO operationalizes these rules through a discrete budget-aware search over Split, Upcycle, and Hybrid constructions, and introduces a moment-matching scaling factor to reduce the initialization scale shift caused by sparse FFN aggregation. Finally, we use a two-stage multimodal recovery recipe that first stabilizes the sparse LLM with the vision tower frozen, then performs joint fine-tuning to recover the full VLM. Under matched data and training protocols, LEGO improves Dense-to-MoE recovery quality and turns costly blind architecture search into a small set of rule-guided candidates. Implementation code is supplied.

1 Introduction

Vision-Language Models (VLMs) are becoming a core component for multimodal perception and human–computer interaction, but one-size-fits-all scaling is impractical: edge devices, private servers, and cloud clusters impose different memory and throughput constraints. Mixture-of-Experts (MoE) provides a natural capacity–compute trade-off by activating only a subset of experts per token [Shazeer et al., 2017, Jiang et al., 2024, Fedus et al., 2022a, Zoph et al., 2022, Zhou et al., 2022]. Given the high cost and instability of training MoE from scratch, Dense-to-MoE (Dense2MoE) converts a pretrained dense VLM by MoE-ifying only the LLM FFNs and performing lightweight recovery [Zhu et al., 2024, Qu et al., 2024, Lin et al., 2024a, Li et al., 2024a, Jin et al., 2024].

Despite growing progress, a key bottleneck for Dense2MoE is the lack of decision-making rules for structural design. Existing methods [Zhu et al., 2024, Li et al., 2024a, He et al., 2025] primarily focus on *how* to convert a dense model, leaving the *target structure* to heuristics such as fixed sparsity (e.g., a fixed $1/4$ activation ratio) that do not reliably transfer across model families and scales. In this work, we use (P_{total}, P_{active}) as the *Architectural Budget* under an LLM-only convention (ViT listed separately), and interpret P_{active} as a *per-token* quantity under Top-K routing.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

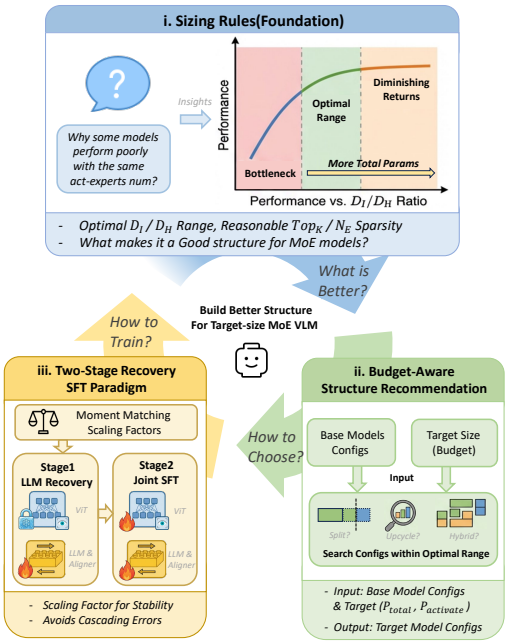


Figure 1: LEGO overview: sizing rules, budget-aware structure recommendation, and two-stage recovery-SFT paradigm.

Algorithm 1: LEGO decision procedure.

Input: Targets (P_T, P_A), slack ϵ, ordered bases $\mathcal{B}$, ordered tuples Ω_t, converted layers.
Output: Base, tuple, operator, realized budgets; or infeasible.

$\mathcal{C} \leftarrow \emptyset$; write $K = Top_K$.
for each $b \in \mathcal{B}$, $(N_S, N_E, K) \in \Omega_t$ **do**
  Skip nonpositive counts, $K > N_E$, or FFN widths not divisible by N_S.
  $r \leftarrow (K/N_S)(D_{\text{dense}}/D_H)$.
  Compute $(\hat{P}_T, \hat{P}_A)$ using Eq. 3.
  Skip if $r \notin [1, 12]$, or either realized budget exceeds $(1+\epsilon)$ times its target.
  $e_T \leftarrow |\hat{P}_T - P_T|/P_T$;
  $e_A \leftarrow |\hat{P}_A - P_A|/P_A$.
  $\kappa \leftarrow (\mathbf{1}[r \notin [1.5, 8]],$
  $\max(e_T, e_A), e_T + e_A,$
  $\mathbf{1}[K \notin [2, 8]], |K-4|, i_b, i_\Omega)$.
  Add $(\kappa, b, N_S, N_E, K, \hat{P}_T, \hat{P}_A)$ to $\mathcal{C}$.
Return infeasible if $\mathcal{C}$ is empty; otherwise return its lexicographic minimum with operator from Appendix I.

With the same recovery recipe, several backbones remain stable at a $1/2$ activation ratio, but the 14B backbone has a much larger relative drop at $1/4$ than smaller models (Fig. 2). This is not complete causal isolation; fixed ratios can yield different converted FFN structures across backbones.

The controlled sweeps, held-out observations, and training-duration checks below motivate a backbone-comparable structural measure beyond raw activation ratio.

The underlying structural factor is that fixing Top_K/N_E does not preserve the activated dimensionality of the FFN. Let D_I denote the activated FFN dimensionality and D_H denote the attention dimensionality (i.e., hidden size). For equal-width Split/Hybrid constructions,

$$D_I = Top_K \cdot \frac{D_{\text{dense}}}{N_S}, \quad thus \quad \frac{D_I}{D_H} = \frac{Top_K}{N_S} \cdot \frac{D_{\text{dense}}}{D_H}. \tag{1}$$

Different backbones have different D_{dense}/D_H, so the same activation ratio can correspond to very different D_I/D_H. In our recovery setting, organizing results by this ratio makes cross-family trends substantially more consistent. We therefore summarize two empirical *Sizing Rules*. First, recovery performance varies with D_I/D_H: very small values create an information bottleneck, while larger values eventually yield diminishing returns under a fixed recovery budget. Second, conditioned on D_I/D_H being in the recommended range, D_{total}/D_I provides a useful sparsity guideline for choosing N_E and Top_K. These rules offer practical guidance rather than universal laws.

Building on these findings, we propose LEGO (ruLes-guided Experts Generation and Orchestration), a Dense2MoE methodology and training framework with three components (Fig. 1): Sizing Rules, budget-aware structure recommendation/search, and a two-stage recovery paradigm tailored to multimodal recovery (Fig. 9). LEGO performs a budget-aware discrete search over Split/Upcycle/Hybrid guided by the Sizing Rules. For the Split+Upcycle path, we identify an initialization scale shift [He et al., 2025] and derive a moment-matching Scaling Factor to improve early-stage recovery stability.

Finally, we study recovery dynamics that arise when a pretrained VLM is structurally perturbed by Dense2MoE conversion. Synchronously updating the vision tower (e.g., the ViT) while the LLM is not yet recovered can trigger cascading errors and reduce recovery quality under the same training setup (Fig. 7). LEGO therefore adopts a two-stage paradigm: first freeze the ViT to recover the LLM, then perform full-parameter joint fine-tuning. The sizing signal and conversion/scaling components

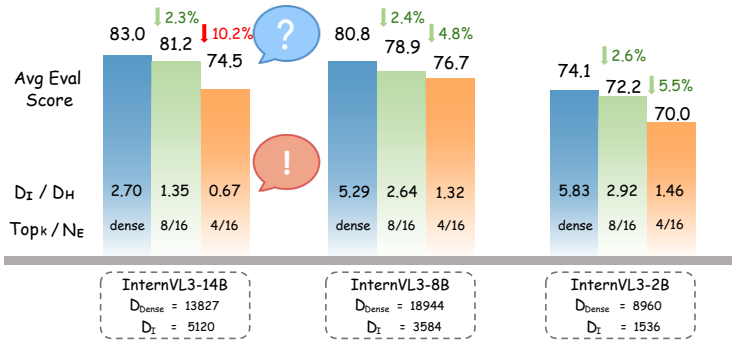


Figure 2: Motivating observation: fixed activation ratios yield different relative drops across model scales under the same recovery budget; drops are relative to trained-Dense controls.

operate on the LLM side; text-only controls support qualitative transfer beyond VLMs. Multimodal evidence concerns when to unfreeze the ViT. In summary, our contributions are:

(i) We summarize empirical Sizing Rules around backbone-comparable D_I/D_H for Dense2MoE recovery, with recommended ranges and a secondary sparsity guideline (Figs. 4 and 6).

(ii) We propose LEGO, a budget-aware Split/Upcycle/Hybrid search with a moment-matching Scaling Factor to stabilize initialization under matched total and active budgets (Fig. 3).

(iii) We validate a multimodal recovery recipe in which two-stage recovery improves early recovery when the converted LLM is still stabilizing after Dense2MoE conversion (Figs. 7 and 9).

2 Related work

Vision-Language Models (VLMs). VLMs integrate visual encoders with Large Language Models via lightweight projectors [Liu et al., 2023a, 2024a]; ViT backbones [Dosovitskiy, 2020] and recent Qwen/InternVL families continue to push the frontier [Bai et al., 2025, Wang et al., 2025b]. Their monolithic scaling creates deployment bottlenecks, especially in heterogeneous or resource-constrained settings. Prior work therefore explores compression and dynamic inference, including quantization [Lin et al., 2024b], pruning [Ma et al., 2023, Chen et al., 2024, Shinde et al., 2025], and distillation [Shu et al., 2024, Feng et al., 2025]. Because these techniques often trade off capacity or require task-specific tuning, sparse architectural alternatives remain attractive; MoE has also been explored for vision transformers [Riquelme et al., 2021].

Mixture-of-Experts. MoE [Shazeer et al., 2017] decouples total capacity from per-token computation via sparse activation. Its scalability was established by GShard [Lepikhin et al., 2020] and Switch Transformers [Fedus et al., 2022b]; recent work studies matched-resource comparisons [Li et al., 2025], Mixtral/DeepSeekMoE-style routing and granularity [Jiang et al., 2024, Dai et al., 2024, Krajewski et al., 2024], expert-choice routing [Zhou et al., 2022], and auxiliary-loss-free balancing [Wang et al., 2024]. BEAM learns token-adaptive expert activation through trainable binary masks [Wu et al., 2026a], while ReBA separates image/text balancing and uses equally weighted image-level routing instances to reduce sensitivity to the visual/text token mix [Wu et al., 2026b]. In multimodal settings, MoE-based VLMs such as DeepSeek-VL2 [Wu et al., 2024], MoIIE [Wang et al., 2025a], and Qwen3-VL [Bai et al., 2025] improve the capacity–compute trade-off. However, training these sparse models from scratch remains costly and recipe-sensitive, which motivates efficient adaptation from dense foundations under limited data and compute. Complementary to LLM-side sparsity, Task-Aware Mechanism uses a hybrid-gated MoE vision tower to adapt video frame count and resolution to the user query and video length [Yin et al., 2026].

Dense-to-MoE Conversion and Sizing Rules. Dense-to-MoE conversion offers a practical alternative to cold-start MoE training [Zhu et al., 2024, Qu et al., 2024], typically through parameter partitioning (Split) [Zuo et al., 2022, Zhang et al., 2022] or sparse upcycling [Komatsuzaki et al., 2022, He et al., 2025]. Random partitioning is simple [Zhu et al., 2024], while dense-sparse training [Pan et al., 2024] and pruning-based conversion [Gao et al., 2025, Pei et al., 2025] can better preserve performance in some settings. Recent MoE designs

explore fine-grained segmentation, shared expert isolation [Dai et al., 2024], heterogeneous expert sizes [Sun et al., 2024], and scaling laws for upcycling/granularity [Krajewski et al., 2024, Liew et al., 2025, Zhao et al., 2025a]. Multimodal MoE works such as MoE-LLaVA and compact VLMs further show that staged training and component freezing help stabilize VLM adaptation [Lin et al., 2024a, Shu et al., 2024]. These studies often describe behavior through macro choices such as expert counts or activation ratios. In contrast, we use the backbone-comparable ratio D_I/D_H as an empirical recovery predictor to guide Hybrid constructions under target (P_{total}, P_{active}) budgets. The two-stage recovery recipe reduces cascading errors in the studied VLM setting.

3 Method

3.1 Preliminaries and problem formulation

Scope. We focus on the most common setting of FFN-MoE: we convert **only the FFNs in the LLM** to MoE, while keeping attention / embedding / normalization dense and always-on. For VLMs, we do not apply MoE to the vision tower (e.g., ViT) or aligner by default.

Architectural Budget. We use $(P_{\text{total}}, P_{\text{active}})$ as the Architectural Budget, where P_{total} denotes total parameters and P_{active} denotes activated parameters in a forward pass. In our Top-K routing setup, we interpret P_{active} as a *per-token* quantity and decompose it as $P_{\text{active}} = P_{\text{fixed}} + P_{\text{FFN,act}}$, where P_{fixed} covers always-on LLM components (e.g., attention/embedding/norm) and $P_{\text{FFN,act}}$ is controlled by routed FFNs. Unless otherwise specified, we report LLM-only budgets and list the fixed ViT separately.

Configuration and structural quantities. Let D_H denote *hidden size* (i.e. *attention dimension*), and D_{dense} denote the *intermediate size* of dense FFN. For MoE-FFN models, we use (N_S, N_E, Top_K) to denote its discrete configuration, and define D_I as the activated FFN dimension and D_{total} as the total FFN dimension. In equal-width splitting,

$$\frac{D_I}{D_{\text{dense}}} = \frac{Top_K}{N_S}, \quad \frac{D_{\text{total}}}{D_{\text{dense}}} = \frac{N_E}{N_S}, \quad \frac{D_{\text{total}}}{D_I} = \frac{N_E}{Top_K}. \tag{2}$$

Here D_I counts executed routed width; repeated-shard hits under upcycling are handled in the scaling factor through N_E. Thus D_I/D_H is a cross-family structural quantity, while $D_{\text{total}}/D_I = N_E/Top_K$ quantifies the configuration's sparsity for each activated FFN size.

3.2 LEGO: budget-aware Dense2MoE conversion

This section describes how LEGO maps $(P_{\text{total}}, P_{\text{active}})$ to a discrete, feasible FFN-MoE configuration. The workflow is illustrated in Fig. 3, and the decision logic is summarized in Algorithm 1.

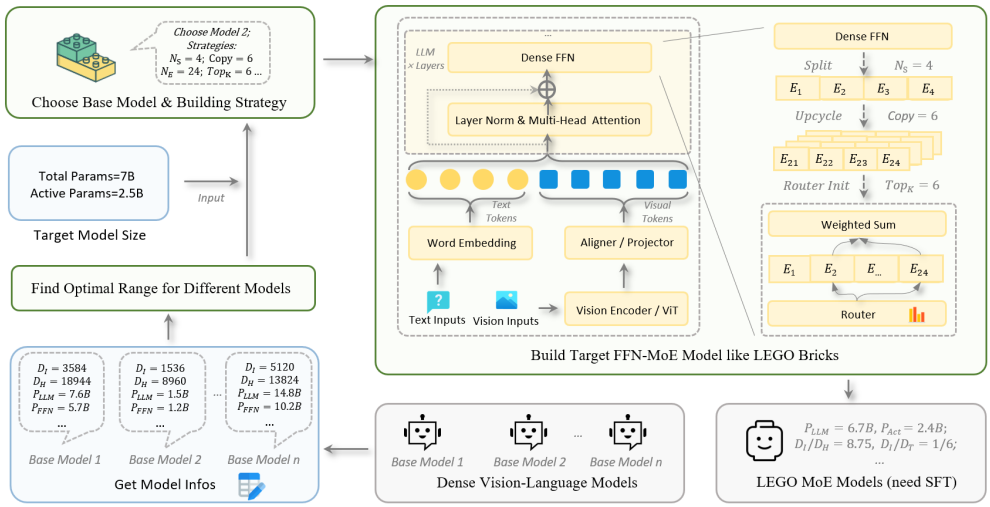


Figure 3: LEGO framework. Under a target budget, LEGO selects a base VLM, searches Split/Upcycle/Hybrid FFN-MoE structures under the sizing rules, and applies two-stage recovery.

3.2.1 Operators (split / upcycle / hybrid)

We can unify FFN-MoE constructions as compositions of:

Equal-width experts. Throughout this work, we adopt equal-width splitting where all shards and replicas share an identical intermediate dimensionality. This design ensures that the core structural quantities D_I and D_{total} are uniquely and fully determined by the specific configuration triplet (N_S, N_E, Top_K) used in this scaling analysis.

Split. We evenly partition the dense FFN's intermediate dimension into N_S distinct expert shards.

Upcycle. We replicate shards to obtain $N_E \geq N_S$ experts, thereby increasing both the total capacity P_{total} and D_{total} while maintaining a constant shard dimensionality.

Hybrid (Split+Upcycle). Combining Split and Upcycle enables precise budget matching and independent control over D_I/D_H and D_{total}/D_I. We only convert the FFN to MoE while keeping attention and ViT fixed; parameter growth is thus strictly restricted to the routed FFN components.

3.2.2 Decision rule and algorithm

LEGO receives positive budgets (P_T, P_A), ordered bases $\mathcal{B}$, a finite ordered target-specific tuple list Ω_t, and converted-layer scope. Algorithm 1 filters structurally valid candidates by $r = D_I/D_H \in [1, 12]$ and one-sided upper-budget slack ϵ (nominal default 0.05). It minimizes the displayed key lexicographically: recommended-width tier, worst relative budget error, combined error, then the complete deterministic stability tie-break. The i_b and i_Ω entries are declared input positions; benchmark scores do not enter. The $K/N_E \in [1/8, 1/2]$ range is empirical guidance, not a hard filter.

For converted layers $\mathcal{M}_b$, one expert shard with $q_l(N_S)$ parameters, router size $G_l(N_E)$, and always-active non-expert/non-router LLM parameters P_{fixed}, we use

$$\begin{aligned} \hat{P}_T &= P_{\text{fixed}} + \sum_{l \in \mathcal{M}_b} \big[N_E q_l(N_S) + G_l(N_E) \big], \\ \hat{P}_A &= P_{\text{fixed}} + \sum_{l \in \mathcal{M}_b} K q_l(N_S). \end{aligned} \tag{3}$$

For bias-free gated MLPs and linear routers, $q_l = 3H_l D_l / N_S$ and $G_l = H_l N_E$. Routers count toward total parameters and inference cost, but not the active-parameter reporting proxy. Neither LLM-only count includes ViT or projector; reported configurations differ from the full selection specification.

3.3 Scaling factor: moment matching

Split+Upcycle may look like "split and replicate," yet it often introduces a noticeable output-scale shift at initialization, leading to higher initial loss and harder convergence. Related observations have also been reported in prior work (e.g., [He et al., 2025]). We therefore introduce a scalar Scaling Factor C at the MoE-FFN aggregation output for scale alignment.

Applicability. Since Split and Upcycle are special cases of Hybrid, we enable this Scaling Factor by default for balanced Split/Hybrid constructions with $N_E = qN_S$, where each shard has exactly q replicas. For pure Upcycle, $N_S = 1$ and $C = 1$, so the factor is the identity.

Default factor. We use **Top-K-then-softmax**: we first select Top-K experts and then apply softmax only over the selected logits. The routed FFN output is

$$y_{\text{moe}}(x) = C \sum_{i \in \text{Top}_K(x)} p_i(x) E_i(x), \qquad \sum_{i \in \text{Top}_K(x)} p_i(x) = 1. \tag{4}$$

Under balanced replication, shard symmetry, approximate router–output independence, and near-uniform selected weights (Appendix F), our default factor is

$$C = C_{\text{mean}}^{1/2} C_{\text{var}}^{1/2}, \quad C_{\text{mean}} = N_S, \quad C_{\text{var}} = \begin{cases} 1, & N_S = 1, \\ \dfrac{N_S}{\sqrt{1 + \frac{N_S - 1}{Top_K} \cdot \frac{N_E - Top_K}{N_E - 1}}}, & N_S > 1. \end{cases} \tag{5}$$

This geometric-mean choice balances mean shift and training stability, recovers $C = 1$ for pure Upcycle ($N_S = 1$), and reduces early loss in the displayed balanced-configuration ablations. The explicit N_E dependence captures the increased probability of repeated shard hits after upcycling, which in turn changes effective shard coverage under Top-K routing.

3.4 Training settings

To mitigate cascading errors in multimodal recovery, Stage 1 trains the LLM and aligner while freezing the vision tower, and Stage 2 jointly fine-tunes all modules with a smaller vision-side learning rate. We use a standard load-balancing regularizer, $\mathcal{L} = \mathcal{L}_{\text{ori}} + \alpha_b \mathcal{L}_b$ with $\mathcal{L}_b = N_E \sum_e f_e \bar{p}_e$, where f_e is the routed-token fraction and $\bar{p}_e$ is the average routing probability.

Default layer strategy. The first two and last LLM layers remain dense for stability and simplicity; the other FFNs become MoE. This is the default unless stated otherwise (e.g., in ablations). Layer-strategy ablations indicate effects mainly on early stability and output quality.

Load-balancing trade-off. The coefficient α_b balances *training stability* and *expert specialization*. Larger values encourage uniform routing and tend to stabilize optimization; smaller values allow stronger specialization with slightly less stability. Fig. 10 compares $\alpha_b = 5 \times 10^{-4}$ (slightly more imbalance, aux loss ≈ 1.15, often correlated with stronger expert differentiation) and $\alpha_b = 10^{-3}$ (more uniform routing, aux loss ≈ 1.08, typically more stable). Under the tested recovery recipe, we recommend the 5×10^{-4}–10^{-3} range according to the stability–specialization priority.

4 Experiments

4.1 Setup

Unless otherwise specified, all experiments follow the default settings in Method: we convert only the LLM FFNs to MoE (FFN-MoE). Other always-on LLM parameters are also counted in P_{active}.

Table 1: Budgeted construction. Nine-benchmark means over three test runs; percentages use each block's displayed Dense reference, including cross-source comparisons. IVL3 denotes InternVL3; Qwen2VL-8B denotes Qwen2-VL-7B (about 8B total VLM parameters). Shading and bold mark LEGO, not the highest score.

Construction			Structure and quality		
Method	**Source**	**Configuration**	D_I/D_H	**Avg**	**vs. Dense**
Target: 8B/5B *LLM total/active*					
Upcycle	IVL3-2B	1s8E4A	23.30	76.3	↓ 5.7%
Split	IVL3-8B	16s16E8A	2.64	77.8	↓ 3.8%
LEGO	IVL3-8B	16s16E8A	2.64	**77.8**	↓ 3.8%
Dense	IVL3-8B	Dense-AfterTrain	5.29	80.9	–
Target: 8B/3B *LLM total/active*					
Upcycle	IVL3-2B	1s8E2A	11.67	75.8	↑ 1.7%
Split	IVL3-8B	16s16E2A	0.66	69.4	↓ 6.9%
LEGO	IVL3-2B	2s16E4A	11.67	**76.2**	↑ 2.3%
Dense	IVL3-2B	Dense-AfterTrain	5.83	74.5	–
Target: 17B/5B *LLM total/active*					
Upcycle	IVL3-2B	1s16E4A	23.30	76.9	↓ 6.8%
Split	IVL3-14B	32s32E2A	0.17	58.2	↓ 29.5%
LEGO	IVL3-8B	8s24E4A	2.64	**80.7**	↓ 2.2%
Dense	IVL3-14B	Dense-AfterTrain	2.70	82.5	–
Target: 8B/3B (Qwen2-VL) *LLM total/active*					
Upcycle	Qwen2-VL-2B	1s8E2A	11.53	75.3	↑ 3.0%
Split	Qwen2-VL-7B	16s16E2A	0.65	71.2	↓ 2.6%
LEGO	Qwen2-VL-2B	2s16E4A	11.53	**75.9**	↑ 3.8%
Dense	Qwen2-VL-2B	Dense-AfterTrain	5.76	73.1	–

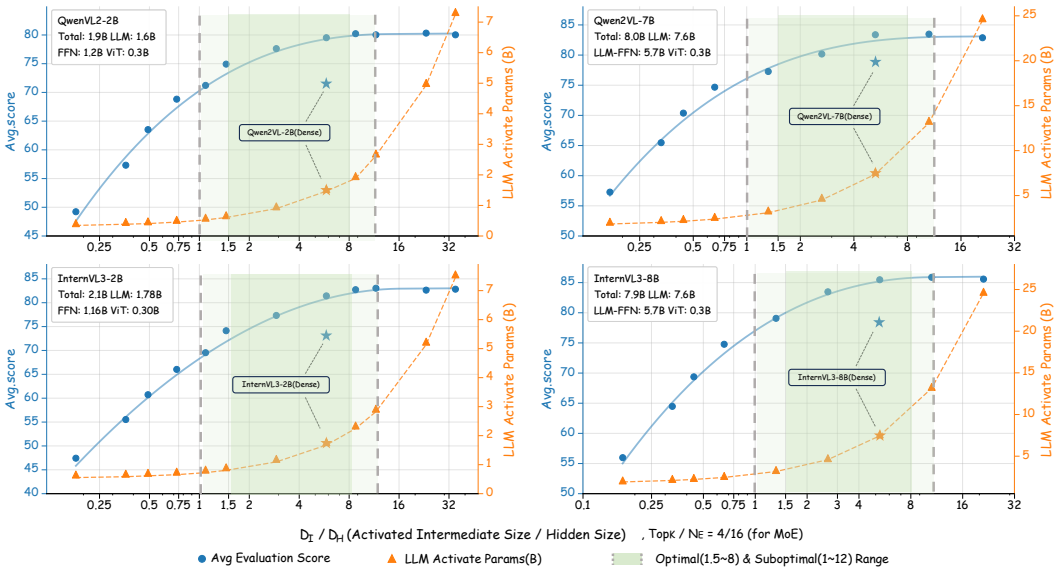


Figure 4: Sizing rule. Score (blue) and P_{active} (orange) versus D_I/D_H at $Top_K/N_E = 4/16$. Green and grey bands mark the empirical recommended (1.5–8) and feasible (1–12) ranges.

Reporting convention and budget definition. We report (P_{total}, P_{active}) under the LLM-only convention and interpret P_{active} as a per-token quantity under Top-K routing.

Training budget and recipe. To rule out data confounders, all MoE models and Dense controls are trained with the same two-stage recovery recipe. Figure 8 reports loss trajectories with different data amounts and training durations; it does not independently isolate data sufficiency at fixed updates.

Evaluation and metrics. We evaluate on nine public benchmarks, grouped into three categories: VQA/OCR, Document/Chart understanding, and Multi-image/Comprehensive tasks. The original nine-benchmark construction scores are reported as means of three independent test runs. The additional targeted evaluations are descriptive point estimates without repeated-run uncertainty; their four-task means are distinct from the nine-task means. *Avg* is the arithmetic mean over the nine benchmarks; means and percentages are recomputed from the displayed task scores. All per-task scores remain in Appendix J. To quantify "recovery quality", we additionally report the *Avg* change relative to the corresponding Dense control (AfterTrain) in Table 1 (↑ / ↓). Each percentage uses that block's displayed Dense row.

Baselines and comparability. To make the role of LEGO ("structural decision-making in an operator space") directly comparable, Table 1 only compares three constructions: (i) *Upcycle-only*: use the reported representative in the Upcycle-only subspace; (ii) *Split-only*: use the reported representative in the Split-only subspace; (iii) *LEGO*: search discrete feasible configurations in the Split/Upcycle/Hybrid operator space and prioritize structures guided by the Sizing Rules (see Algorithm 1). For each target budget, we also provide a Dense control (AfterTrain): these dense backbones are also trained with the same data mixture and the same two-stage recipe as LEGO, to minimize differences caused by data and training recipes (Table 4).

4.2 Main result 1: validation of the sizing rules

Empirical recommended range of D_I/D_H. Fig. 4 shows systematic sweeps across model families (InternVL3 and Qwen2-VL) and scales at a constant activation rate. Recovery performance as a function of D_I/D_H follows a consistent non-monotonic pattern in our recovery setting. In particular, when $D_I/D_H < 1$, we observe a clear information bottleneck; when D_I/D_H becomes larger, the gain gradually saturates and exhibits diminishing returns. We also annotate the growth of activated parameters along D_I/D_H, demonstrating that "larger activated parameters $\not\Rightarrow$ better recovery" in this recovery setting. Exact reconstruction of the empirical bands requires the missing threshold records.

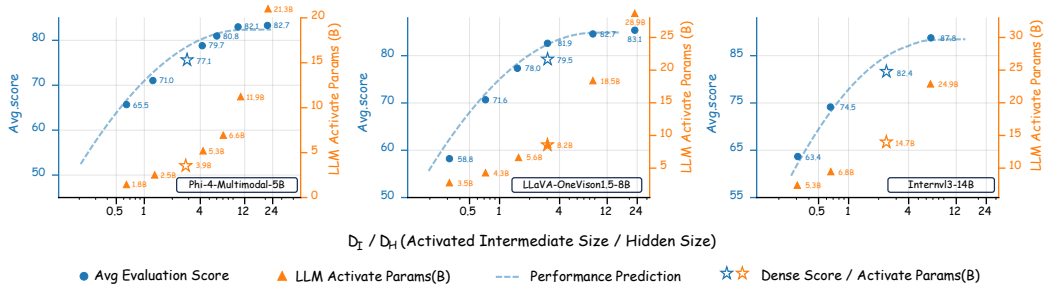


Figure 5: Hold-out validation of the sizing rules. Dashed curves are empirical predictions fitted on reference backbones without using the held-out blue points. Blue dots show measured hold-out backbones/scales; stars mark Dense baselines and orange triangles denote activated parameters.

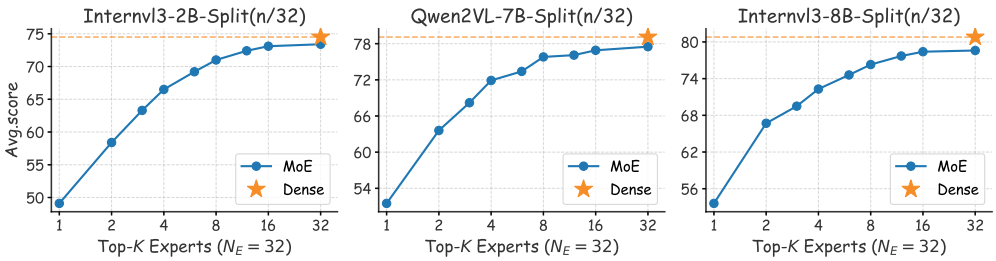


Figure 6: Sparsity sweep. At fixed expert-pool size and recommended D_I/D_H, recovery initially improves as more experts are activated, then saturates near the Dense reference.

To robustly visualize the trend over a wide D_I/D_H range, we fit a smoothing curve after the log transform $t = \log(D_I/D_H)$. The cubic smoothing spline $f(t)$ is obtained by solving

$$\min_f \sum_{i=1}^{n} \big(y_i - f(t_i)\big)^2 + s \int \big(f''(t)\big)^2 \, dt. \tag{6}$$

where $s \geq 0$ controls the smoothness. We plot $f(\log x)$ on a dense $x = D_I/D_H$ grid.

Sparsity regime and boundary effects. Conditioned on D_I/D_H being in the recommended range, Fig. 6 further examines activation sparsity Top_K/N_E (equivalently, $D_{total}/D_I = N_E/Top_K$ under equal-width splitting). The strongest evidence is that overly sparse settings are fragile, while increasing density eventually gives diminishing returns rather than proportional gains. Under our recovery budget, activation ratios within $1/8 \sim 1/2$ are typically more robust, corresponding to $D_{total}/D_I \in [2, 8]$. Since this sparsity guideline is validated on fewer representative settings than the D_I/D_H sweeps, we treat it as an empirical recommendation for fast Dense2MoE recovery.

Held-out backbone observations. Figure 5 reports Phi-4-Multimodal, LLaVA-OneVision 1.5, and InternVL3-14B (Table 8). These measurements follow the qualitative bottleneck-and-recovery trend across additional backbones. Dashed curves summarize reference-backbone fits; the original fitting and hold-out records are needed to assess calibrated accuracy. This supports testing transfer under comparable recovery settings, not a universal optimum.

4.3 Main result 2: effectiveness of LEGO

Table-level comparison across representative target budgets. Table 1 compares Upcycle-only, Split-only, and LEGO on three representative target budgets, reporting recovery quality measured by Avg relative to the corresponding Dense-AfterTrain control. The data and recovery recipe are matched, but some rows change both source backbone and structure. The table evaluates joint budgeted construction, rather than isolating a same-source causal effect.

We observe two complementary patterns. (i) When a target budget can be reasonably met by Split-only or Upcycle-only, LEGO often selects a simple configuration with similar structure and performance (e.g., Target: $(8B, 5B)$); when a hybrid configuration better matches the budget, LEGO uses it without forcing a pure path (e.g., Target: $(8B, 3B)$). This matches LEGO's goal: choose a rule-consistent option in the declared operator space, rather than forcing additional construction complexity. (ii) In the nominal $(17B, 5B)$ block, the reported Hybrid construction has a higher average than the pure-path representatives and places D_I/D_H in the recommended range. This is a comparison of the submitted operating points; exact budget feasibility must be checked against the original conversion records before interpreting it as evidence of a reachable-budget advantage.

These point estimates support rule-guided joint base-and-structure selection in the tested settings. They do not isolate the benefit of the ranking rule from candidate-space coverage, or establish global optimality. The 17B/5B block retains the submitted configurations; its exact realized-budget and historical selection records still require verification from the original runs.

4.4 Ablation studies

Appendices E–J give further experimental evidence, complete per-task scores, and scope limits.

External conversion comparisons. We compare LEGO, ToMoE [Gao et al., 2025], and CMoE [Pei et al., 2025] under nested 4%/49%/100% recovery data and the same two-stage protocol. For the shared InternVL3-8B source and $(8\mathrm{B}, 5\mathrm{B})$ target, full-data four-task means are 81.5, 79.2, and 81.4, respectively: LEGO and CMoE are essentially tied. At $(8\mathrm{B}, 3\mathrm{B})$, the common {2B,8B,14B} source pool yields 81.6, 76.3, and 77.8. Each method uses its native feasible construction, jointly varying source and structure. Neither a shared exhaustive search nor exact historical Algorithm 1 replay is established. Results are limited to these budgets and protocol.

Table 2: Four-task means at each recovery-data fraction for two approximately matched LLM budgets. Sources and configurations follow each method's native procedure.

Target	Method (source/configuration)	Total/active	D_I/D_H	4%	49%	100%
$(8\mathrm{B}, 5\mathrm{B})$	LEGO (8B/16s16E8A)	7.65/4.79B	2.64	70.1	78.3	81.5
	ToMoE (8B/1 mask, p=0.5)	7.63/4.68B	2.64	73.4	77.3	79.2
	CMoE (8B/16E8A, 2 shared)	7.65/4.79B	2.64	72.9	79.5	81.4
$(8\mathrm{B}, 3\mathrm{B})$	LEGO (2B/2s16E4A)	7.84/2.84B	11.67	77.4	78.8	81.6
	ToMoE (8B/1 mask, p=0.2)	7.63/2.91B	1.06	68.4	74.1	76.3
	CMoE (8B/16E2A, 1 shared)	7.65/2.66B	0.66	69.6	75.7	77.8
	Split (8B/16s16E2A)	7.65/2.66B	0.66	64.5	72.4	75.1

Measured deployment efficiency. On A100 at batch size one over approximately 48k full-VLM queries, the same-source InternVL3-8B comparison reduces mean latency from 0.801 to 0.702 s/query and raises generated-token throughput from 80.02 to 96.29 tok/s; nine-task quality changes from 80.9 to 77.8 and peak GPU memory from 17.6 to 17.8 GiB. Expert weights remain resident. These workload-level measurements complement P_{active}; they are not fixed-length kernel benchmarks.

Table 3: Batch-1 InternVL3 quality and efficiency on A100. Quality is the nine-benchmark average; total/active parameter counts are LLM-only, whereas the four system metrics cover the full VLM.

Operating point	Avg	Total/active	s/query	Output tok/s	Peak GiB	TFLOPs/query
2B Dense	74.5	1.78/1.78B	0.487	201.76	6.3	13.3
2B/16s16E2A	71.3	1.79/0.78B	0.413	223.51	6.4	8.2
8B Dense	80.9	7.63/7.63B	0.801	80.02	17.6	38.0
2B/2s16E4A	76.2	7.84/2.84B	0.614	125.16	18.0	19.6
8B/16s16E8A	77.8	7.65/4.79B	0.702	96.29	17.8	25.3

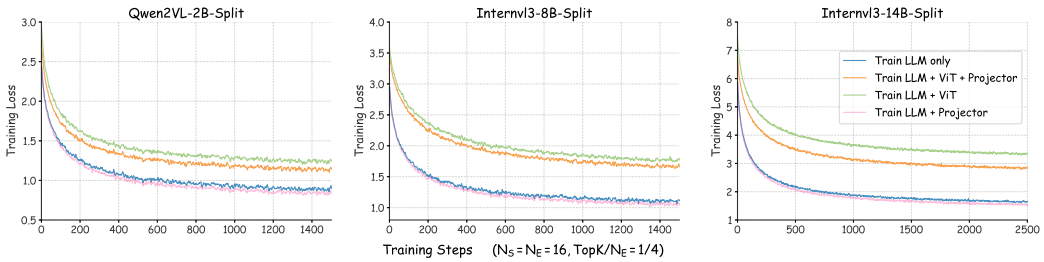


Figure 7: Early recovery dynamics. Recovery losses for different Stage-1 training targets across three model scales after Dense2MoE conversion.

Routing compatibility and text-only transfer. At mean active budgets within 5% of fixed Top-K references, post-recovery Top-P routing retains similar four-task quality. Text-only Qwen2.5-7B and Llama-3.1-8B controls show the same qualitative low-width bottleneck and recovery trend, but do not establish a modality-specific shift in the recommended range.

Early two-stage recovery behavior. Fig. 7 compares early training dynamics with or without updating the vision side (e.g., ViT) in the early recovery stage. We find that jointly updating the vision side while the LLM is not yet recovered can trigger cascading errors, worsening the early loss trajectory under the same setup. This supports delayed ViT unfreezing; the figure alone does not establish a final benchmark gain under the tested two-stage recovery protocol.

Sparsity and granularity trends. We further verify that, under our recovery budget, the number of activated experts and sparsity show sparsity-side fragility and diminishing returns (Fig. 6). This recommendation applies to fast Dense2MoE recovery under our budget, not necessarily to optimal from-scratch MoE pre-training designs.

Stage-1 training targets (freezing the vision side). We compare early recovery trajectories under different choices of ViT freezing and aligner training in Stage 1 (Fig. 7). The overall trends across the three displayed trajectories are consistent with "recover the LLM first, then jointly fine-tune".

Effect of the Scaling Factor (Moment Matching). Fig. 12 reports the ablation on the Scaling Factor. Enabling it reduces initialization loss and improves early convergence in the two displayed balanced constructions, which is consistent with the Distribution Shift analysis in Method.

Load-balancing behavior. Fig. 10 compares expert loads under two load-balancing coefficients. The larger coefficient produces more uniform routing in these diagnostics.

5 Conclusion

As VLM deployments increasingly demand model variants under heterogeneous resource constraints, MoE has become an attractive mechanism for trading capacity against inference cost. However, prior Dense-to-MoE work largely emphasizes *how* to convert dense models, leaving open the question of *what* structures should be built under a given budget. We present LEGO, a methodology for budget-aware FFN-MoE conversion of VLMs. LEGO summarizes empirical sizing rules around D_I/D_H, uses them to guide a discrete Split/Upcycle/Hybrid search, and stabilizes recovery with a moment-matching scaling factor and a two-stage multimodal recovery recipe. The resulting message is practical: a one-time controlled sweep can provide reusable structural guidance, so applying Dense2MoE to a new target budget requires far less blind search.

Limitations and broader impacts. The sizing ranges are empirical, and cross-source comparisons combine backbone and structure choices. Targeted additional evaluations lack repeated-run uncertainty; measured efficiency is specific to the tested A100 workload. Cheaper VLM adaptation may aid local, accessible visual understanding while lowering barriers to surveillance or misleading multimodal content. Responsible model release and transparent reporting remain necessary.

Acknowledgments and Disclosure of Funding

This work was supported in part by the Guangdong Grants (Grant No. 2023ZT10X075), the National Natural Science Foundation of China (Grant Nos. 62332002 and 62425101), and the Shenzhen Science and Technology Program (Grant No. KQTD20240729102051063). The authors declare no other competing interests.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 11 2025.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, et al. Llama-nemotron: Efficient reasoning models, 2025. URL https://arxiv.org/abs/2505.00949.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. Streamlining redundant layers to compress large language models. *arXiv preprint arXiv:2403.19135*, 2024.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Haodong Duan, Xinyu Fang, Junming Yang, Xiangyu Zhao, Yuxuan Qiao, Mo Li, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models, 2024. URL https://arxiv.org/abs/2407.11691.

William Fedus, Jeff Dean, and Barret Zoph. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*, 2022a.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022b.

Qianhan Feng, Wenshuo Li, Tong Lin, and Xinghao Chen. Align-kd: Distilling cross-modal alignment knowledge for mobile vision-language large model enhancement. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4178–4188, 2025.

Shangqian Gao, Ting Hua, Reza Shirkavand, Chi-Heng Lin, Zhen Tang, Zhengao Li, Longge Yuan, Fangyi Li, Zeyu Zhang, Alireza Ganjdanesh, et al. Tomoe: Converting dense large language models to mixture-of-experts through dynamic structural pruning. *arXiv preprint arXiv:2501.15316*, 2025.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, 2017. doi: 10.1109/CVPR.2017.670. URL https://arxiv.org/abs/1612.00837.

Jiawei Guo, Tianyu Zheng, Yizhi Li, Yuelin Bai, Bo Li, Yubo Wang, et al. MAmmoTH-VL: Eliciting multimodal reasoning with instruction tuning at scale. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.680. URL https://aclanthology.org/2025.acl-long.680/.

Ethan He, Abhinav Khattar, Ryan Prenger, Vijay Korthikanti, Zijie Yan, Tong Liu, Shiqing Fan, Ashwath Aithal, Mohammad Shoeybi, and Bryan Catanzaro. Upcycling large language models into mixture of experts, 2025. URL https://arxiv.org/abs/2410.07524.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder task needs more experts: Dynamic routing in MoE models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 12883–12895, 2024. doi: 10.18653/v1/2024.acl-long.696.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, et al. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. Moe++: Accelerating mixture-of-experts methods with zero-computation experts, 2024. URL https://arxiv.org/abs/2410.07348.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *Computer Vision – ECCV 2016*, pages 235–251, 2016. doi: 10.1007/978-3-319-46493-0_15.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*, 2022.

Jakub Krajewski, Jan Ludziejewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, et al. Scaling laws for fine-grained mixture of experts. *arXiv preprint arXiv:2402.07871*, 2024.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024. URL https://arxiv.org/abs/2411.15124.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, et al. Seed-bench-2: Benchmarking multimodal large language models, 2023. URL https://arxiv.org/abs/2311.17092.

Houyi Li, Ka Man Lo, Ziqi Wang, Zili Wang, Wenzhen Zheng, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Can mixture-of-experts surpass dense llms under strictly equal resources? *arXiv preprint arXiv:2506.12119*, 2025.

Jiachen Li, Xinyao Wang, Sijie Zhu, Chia-Wen Kuo, Lu Xu, Fan Chen, et al. Scaling multimodal llm with co-upcycled mixture-of-experts, 2024a. URL https://arxiv.org/abs/2405.05949.

Lei Li, Yuqi Wang, Runxin Xu, Peiyi Wang, Xiachong Feng, Lingpeng Kong, et al. Multimodal arxiv: A dataset for improving scientific comprehension of large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*, pages 14369–14387, 2024b. doi: 10.18653/v1/2024.acl-long.775. URL https://aclanthology.org/2024.acl-long.775/.

Seng Pei Liew, Takuya Kato, and Sho Takase. Scaling laws for upcycling mixture-of-experts language models. *arXiv preprint arXiv:2502.03009*, 2025.

Bin Lin, Zhenyu Tang, Yang Ye, Jinfa Huang, Junwu Zhang, Yatian Pang, et al. MoE-LLaVA: Mixture of experts for large vision-language models, 2024a. URL https://arxiv.org/abs/2401.15947.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pages 87–100, 2024b. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023a.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024a.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bohao Li, Songyang Zhang, Wangbo Zhao, et al. Mmbench: Is your multi-modal model an all-around player?, 2023b. URL https://arxiv.org/abs/2307.06281. MMBench v1.1 (English).

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, et al. Ocrbench: On the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024b. doi: 10.1007/s11432-024-4235-6.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning, 2022. URL https://arxiv.org/abs/2203.10244.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. Docvqa: A dataset for vqa on document images. *arXiv preprint arXiv:2007.00398*, 2020. URL https://arxiv.org/abs/2007.00398.

Minesh Mathew, Viraj Bagal, Rubèn Pérez Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V. Jawahar. Infographicvqa. *CoRR*, abs/2104.12756, 2021. URL https://arxiv.org/abs/2104.12756.

nz. arXiv-OCR dataset (v0.2). https://huggingface.co/datasets/nz/arxiv-ocr-v0.2, 2024. Accessed 2026-01-22.

Bowen Pan, Yikang Shen, Haokun Liu, Mayank Mishra, Gaoyuan Zhang, Aude Oliva, Colin Raffel, and Rameswar Panda. Dense training, sparse inference: Rethinking training of mixture-of-experts language models. *arXiv preprint arXiv:2404.05567*, 2024.

Zehua Pei, Lancheng Zou, Hui-Ling Zhen, Xianzhi Yu, Wulong Liu, Sinno Jialin Pan, Mingxuan Yuan, and Bei Yu. CMoE: Fast carving of mixture-of-experts for efficient LLM inference. *arXiv preprint arXiv:2502.04416v1*, 2025. URL https://arxiv.org/abs/2502.04416v1.

Xiaoye Qu, Daize Dong, Xuyang Hu, Tong Zhu, Weigao Sun, and Yu Cheng. LLaMA-MoE v2: Exploring sparsity of LLaMA from perspective of mixture-of-experts with post-training, 2024. URL https://arxiv.org/abs/2411.15708.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Gaurav Shinde, Anuradha Ravi, Emon Dey, Shadman Sakib, Milind Rampure, and Nirmalya Roy. A survey on efficient vision-language models. *arXiv preprint arXiv:2504.09724*, 2025. URL https://arxiv.org/abs/2504.09724.

Fangxun Shu, Yue Liao, Le Zhuo, Chenning Xu, Lei Zhang, Guanghao Zhang, Haonan Shi, Long Chen, Tao Zhong, Wanggui He, et al. LLaVA-mod: Making LLaVA tiny via MoE knowledge distillation. *arXiv preprint arXiv:2408.15881*, 2024.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, et al. Towards vqa models that can read. *arXiv preprint arXiv:1904.08920*, 2019. URL https://arxiv.org/abs/1904.08920. CVPR 2019.

Manxi Sun, Wei Liu, Jian Luan, Pengzhi Gao, and Bin Wang. Mixture of diverse size experts. *arXiv preprint arXiv:2409.12210*, 2024.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. In *Advances in Neural Information Processing Systems (NeurIPS 2024)*, 2024. doi: 10.48550/arXiv.2406.16860. URL https://arxiv.org/abs/2406.16860. Oral.

Dianyi Wang, Siyuan Wang, Zejun Li, Yikun Wang, Yitong Li, Duyu Tang, Xiaoyu Shen, Xuanjing Huang, and Zhongyu Wei. Moiie: Mixture of intra-and inter-modality experts for large vision language models. *arXiv preprint arXiv:2508.09779*, 2025a.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025b.

Luis Wiedmann, Orr Zohar, Amir Mahla, Xiaohan Wang, Rui Li, Thibaud Frere, et al. Finevision: Open data is all you need, 2025. URL https://arxiv.org/abs/2510.17269.

Juntong Wu, Jialiang Cheng, Qishen Yin, Yue Dai, Yuliang Yan, Fuyu Lv, Ou Dan, and Li Yuan. BEAM: Binary expert activation masking for dynamic routing in MoE. *arXiv preprint arXiv:2605.14438*, 2026a. URL https://arxiv.org/abs/2605.14438.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. DeepSeek-VL2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*, 2024.

Ziang Wu, Peng Jin, Qishen Yin, Munan Ning, Hao Li, Peizhen Zhang, and Li Yuan. Relax within, balance across: Geometry-guided load balancing for vision-language mixture-of-experts. *arXiv preprint arXiv:2608.00574*, 2026b. URL https://arxiv.org/abs/2608.00574.

Guowei Xu, Peng Jin, Ziang Wu, Hao Li, Yibing Song, Lichao Sun, et al. LLaVA-CoT: Let vision language models reason step-by-step. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2025)*, 2025. doi: 10.48550/arXiv.2411.10440. URL https://arxiv.org/abs/2411.10440.

Qishen Yin, Tanghui Jia, Peng Jin, Hao Li, Juntong Wu, Guanlin Lu, and Li Yuan. Task-aware mechanism: Hybrid MoE vision tower towards holistic video understanding. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/61723.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567, 2024. doi: 10.48550/arXiv.2311.16502. URL https://arxiv.org/abs/2311.16502.

Yi Zhang, Bolin Ni, Xin-Sheng Chen, Heng-Rui Zhang, Yongming Rao, Houwen Peng, et al. Bee: A high-quality corpus and full-stack suite to unlock advanced fully open mllms, 2025a. URL https://arxiv.org/abs/2510.13795.

Yi-Fan Zhang, Huanyu Zhang, Haochen Tian, Chaoyou Fu, Shuangqing Zhang, Junfei Wu, et al. Mme-realworld: Could your multimodal llm challenge high-resolution real-world scenarios that are difficult for humans? In *Proceedings of the 2025 International Conference on Learning Representations (ICLR 2025)*, 2025b. doi: 10.48550/arXiv.2408.13257. URL https://arxiv.org/abs/2408.13257.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, 2022.

Guoliang Zhao, Yuhan Fu, Shuaipeng Li, Xingwu Sun, Ruobing Xie, An Wang, Weidong Han, Zhen Yang, Weixuan Sun, Yudong Zhang, et al. Towards a comprehensive scaling law of mixture-of-experts. *arXiv preprint arXiv:2509.23678*, 2025a.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, et al. Swift: a scalable lightweight infrastructure for fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 29733–29735, 2025b.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, et al. LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training, 2024. URL https://arxiv.org/abs/2406.16554.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

Simiao Zuo, Qingru Zhang, Chen Liang, Pengcheng He, Tuo Zhao, and Weizhu Chen. Moebert: from bert to mixture-of-experts via importance-guided adaptation. *arXiv preprint arXiv:2204.07675*, 2022.

Limitations and broader impacts

Scope and limitations. LEGO is designed for fast recovery of FFN-only Dense-to-MoE VLMs with equal-width experts and fixed Top-K routing. The recommended D_I/D_H and sparsity ranges are empirical guidelines derived under our recovery recipe rather than universal architectural laws. We use (P_{total}, P_{active}) as hardware-agnostic budget proxies; real latency and throughput also depend on kernels, batching, routing overhead, memory bandwidth, and load imbalance. Our main results average three runs across nine benchmarks, but broader uncertainty estimates and explicit

distribution-shift benchmarks remain useful future work. The additional Top-P experiment tests post-recovery compatibility only. We leave heterogeneous expert widths, dynamic-routing-aware design, depth-aware layer selection, attention MoE, and vision-tower MoE to future studies. The A100 measurements are workload-specific; targeted additional comparisons lack repeated-run uncertainty.

Broader impacts. Reducing the search cost for Dense-to-MoE conversion can make capable VLM variants easier to deploy under constrained memory and compute budgets, which may benefit accessibility, education, and local/private visual understanding. The same reduction in deployment barriers can also increase the availability of strong multimodal systems for harmful use, including surveillance, misleading multimodal content, or unmonitored local deployment. We therefore view transparent reporting of model scope, data, compute, limitations, and release practices as part of responsible use. LEGO does not introduce new data or safety-sensitive capabilities by itself, but it can make existing VLM capabilities cheaper to adapt and deploy.

Appendix

Guide to the appendix. The following sections collect definitions, experimental details, supporting results, and the records needed to interpret the reported configurations.

A. Notation, structural quantities, and parameter-counting conventions.

B. Recovery data, dense controls, and held-out model specifications.

C. Benchmark descriptions and the evaluation protocol.

D. Training settings, implementation, compute costs, and staged recovery.

E. Design choices, empirical scope, limitations, and budget coverage.

F. Scaling-factor assumptions, derivation, and boundary cases.

G. Original supplementary baselines and scaling-factor sensitivity.

H. ToMoE/CMoE comparisons, measured inference cost, Top-P routing, and text-only controls.

I. Deterministic selection, operator mapping, and historical configuration records.

J. Complete per-task scores behind the main construction comparison.

A Abbreviations and notation

This appendix provides a compact reference for the abbreviations and symbols used throughout the paper. Our method and analyses rely on a small set of shared quantities (e.g., (N_S, N_E, Top_K), D_I/D_H, and (P_{total}, P_{active})); the table below is intended to reduce context switching while reading the main text. Unless otherwise specified, (P_{total}, P_{active}) follows the paper's reporting convention (LLM-only, with the vision tower treated as a constant and listed separately). In particular, throughout the paper we interpret P_{active} as a **per-token** quantity under Top-K routing, and write $P_{active} = P_{\text{fixed}} + P_{\text{FFN,act}}$, where P_{fixed} covers always-on LLM components (e.g., attention/embedding/norm) and $P_{\text{FFN,act}}$ is contributed by routed FFNs.

Notation summary. N_S is the number of splits; the dense FFN is evenly split into N_S shards. N_E is the number of experts after replication or dropping from these shards. Top_K is the number of activated experts per token, with $1 \leq Top_K \leq N_E$. The shorthand $xSyEzA$ means splitting the dense FFN into x shards, obtaining y experts, and activating z experts per token. D_I is the activated FFN intermediate dimensionality per token, D_H is the hidden size, D_{total} is the total FFN intermediate dimensionality summed over experts, and D_{dense} is the dense FFN intermediate size. For a dense model, $D_I = D_{total} = D_{dense}$. P_{total} denotes total parameters and P_{active} denotes activated parameters per token under the reporting convention used in this paper.

In addition to Top-K, some MoE systems use **Top-P** routing, where the number of activated experts is dynamic. Concretely, experts are sorted by routing probability, and we select the smallest set whose cumulative weight exceeds a threshold p (e.g., activate 3 experts if the top-3 routing probabilities sum to > 0.9). Compared to Top-K, Top-P can reduce computation on "easy" tokens by activating fewer experts, while allocating more experts when the router is uncertain. However, this dynamic expert count also introduces additional variability in the forward pass, which can

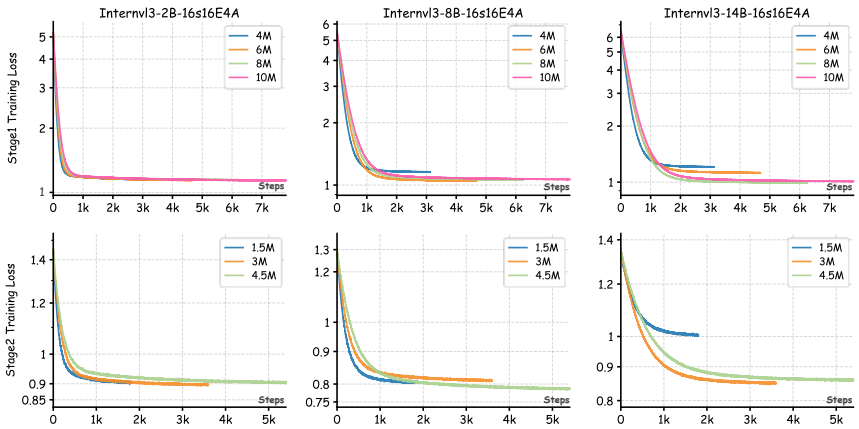


Figure 8: Recovery loss under different data amounts and schedules. The 14B runs differ in duration; the plot cannot isolate data-size effects at fixed updates or establish benchmark saturation.

Table 4: Dense checkpoint controls: perception and OCR. Base and After use the original checkpoint and the same recovery pipeline, respectively, for each model.

Model	State	AI2D$_{wo M}$	TextVQA	OCRBench
InternVL3-2B	Base	87.1	77.0	83.3
	After	87.3	77.2	83.8
InternVL3-8B	Base	92.2	80.1	88.0
	After	92.7	80.5	88.6
InternVL3-14B	Base	93.2	80.4	87.5
	After	93.9	80.9	88.9
Qwen2-VL-2B	Base	84.6	79.2	80.2
	After	85.3	80.1	81.6
Qwen2-VL-7B	Base	92.1	84.1	85.9
	After	92.6	84.6	86.4
Phi-4-Multimodal	Base	83.0	76.9	84.4
	After	83.9	80.1	82.3
LLaVA-OV-1.5	Base	94.1	83.2	82.9
	After	92.3	83.6	82.1

make optimization and system behavior less stable in some settings (e.g., fluctuating token-wise compute and load balance). Accordingly, by default, we use Top-K routing with the same fixed number of experts activated for each individual token throughout the paper.

B Training dataset specifications and extended results

B.1 Data scaling and quality analysis

Dense controls follow the converted models' recovery pipeline, helping characterize extra training without isolating backbone or architecture effects. Table 4 reports before/after component scores.

Data and training-duration sensitivity. Figure 8 reports loss curves for different recovery-data quantities and schedules, but runs end at different steps. It is not a fixed-update data-size ablation and cannot by itself rule out under-training or establish benchmark-level saturation.

Nine-task means. Recomputed Base/After means are InternVL3-2B: 74.8/74.5, InternVL3-8B: 81.0/80.9, InternVL3-14B: 83.0/82.5, Qwen2-VL-2B: 72.6/73.1,
Qwen2-VL-7B: 79.3/79.1, Phi-4-Multimodal: 77.6/77.1, LLaVA-OV-1.5: 80.7/79.2.

Table 5: Dense checkpoint controls: document and chart tasks. Base and After use the original checkpoint and the same recovery pipeline, respectively, for each model.

Model	State	ChartQA	DocVQA	InfoVQA
InternVL3-2B	Base	80.3	88.3	66.2
	After	80.1	89.4	64.8
InternVL3-8B	Base	86.6	92.1	76.8
	After	86.1	92.6	77.2
InternVL3-14B	Base	87.3	93.8	84.6
	After	86.6	93.6	84.0
Qwen2-VL-2B	Base	73.4	89.5	65.5
	After	73.5	90.1	66.7
Qwen2-VL-7B	Base	83.0	94.2	76.5
	After	82.4	94.8	75.1
Phi-4-Multimodal	Base	84.5	93.1	74.8
	After	84.0	92.6	75.7
LLaVA-OV-1.5	Base	86.5	94.7	78.4
	After	85.4	92.4	76.3

Table 6: Dense checkpoint controls: comprehensive tasks. Base and After use the original checkpoint and the same recovery pipeline, respectively, for each model.

Model	State	MMMU	SeedBench	MMB1.1$_{en}$
InternVL3-2B	Base	48.4	64.6	78.3
	After	47.3	62.9	78.0
InternVL3-8B	Base	62.7	69.2	81.7
	After	61.6	67.5	80.9
InternVL3-14B	Base	67.1	70.3	83.1
	After	65.2	67.9	81.6
Qwen2-VL-2B	Base	47.3	61.8	72.2
	After	46.7	61.2	72.8
Qwen2-VL-7B	Base	52.1	64.8	80.7
	After	51.5	63.9	80.4
Phi-4-Multimodal	Base	56.0	68.5	77.2
	After	53.4	66.8	75.1
LLaVA-OV-1.5	Base	55.4	69.2	82.3
	After	52.8	66.4	81.3

B.2 Detailed descriptions of training sources

We curate a heterogeneous training mixture comprising approximately 20M samples to facilitate the transition from dense models to LEGO-MoE. The data is organized into two thematic stages to progressively align the model's sparse architecture with complex multimodal tasks.

Stage 1: foundational capability recovery. This stage aims to restore basic perception and optical character recognition (OCR) abilities that MoE initialization may disturb. High-quality datasets such as `Llama-Nemotron-VLM` and `MAmmoTH-VL` provide robust visual-text alignment.

Stage 2: advanced reasoning and interaction. This stage adds multi-image inputs and complex scientific reasoning (e.g., `ArxivQA` and `Honey-Data-1M`). This mixture ensures that LEGO-MoE can use its expanded parameter space for sophisticated long-context multimodal conversations.

Table 7: Summary of training data sources and scales.

Stage	Dataset Source	Primary Task	Samples
Stage 1	MAmmoTH-VL-Instruct-12M-si [Guo et al., 2025]	General VQA	10M
	Cambrian-737k [Tong et al., 2024]	Multi-modal Instruction	737k
	LLaVA-CoT-100k [Xu et al., 2025]	Chain-of-Thought	99k
	Llama-Nemotron-VLM-Dataset-v1 [Bercovich et al., 2025]	Complex Reasoning	2.9M
	Arxiv-OCR-v0.2 [nz, 2024]	Document OCR	160k
	Stage 1 Total		**≈13.896M**
Stage 2	MAmmoTH-VL-Instruct-12M-ov [Guo et al., 2025]	Comprehensive VL	2M
	FineVisionShuffle [Wiedmann et al., 2025]	Visual Detail Recognition	3.5M
	Honey-Data-1M [Zhang et al., 2025a]	Instruction Following	1M
	MME-RealWorld [Zhang et al., 2025b]	Real-world Perception	29k
	VQAv2 [Goyal et al., 2017]	Traditional VQA	440k
	ArxivQA [Li et al., 2024b]	Scientific Reasoning	100k
	Stage 2 Total		**≈7.069M**
Grand Total			**≈20.965M**

Table 8: Model architecture specifications

LLM backbone						
Base Model/Method	**Name**	**Size**	**FFN Size**	D_H	D_I	D_I / D_H
Phi4-Multimodal	Phi4-Mini	3.8B	2.5B	3072	8192	2.67
LLaVA-OneVision1.5-8B	Qwen3-8B	8.2B	5.1B	4096	12288	3.00
InternVL3-14B	Qwen2.5-14B	14.7B	10.2B	5120	13824	2.70

Vision tower		
Base Model/Method	**Name**	**Size**
Phi4-Multimodal	SigLip-400M	400M
LLaVA-OneVision1.5-8B	RICE-ViT-L-patch14 - 560	310M
InternVL3-14B	InternViT - 300M - 448px - V2	304M

B.3 Hold-out model specifications

In the hold-out validation (Fig. 5), we evaluate cross-family predictability and forward validation across three representative model families. Table 8 lists their backbone specifications, LLM and vision-tower sizes, and backbone-dependent ratios used in our analysis (e.g., D_I/D_H).

C Evaluation benchmarks and protocol

C.1 Evaluation suites and dataset statistics

Diverse benchmarks test the robustness of LEGO-MoE initialization and recovery.

Foundational perception and OCR

- **AI2D** [Kembhavi et al., 2016] is a diagram-based science question-answering dataset. We report the AI2D variant denoted w/o in the original evaluation records; its exact evaluator setting must be verified before release.
- **TextVQA** [Singh et al., 2019] requires the model to read and reason about text embedded in natural scene images. It serves as a benchmark for evaluating the integration of high-resolution OCR with scene-level semantic understanding.
- **OCRBench** [Liu et al., 2024b] is a comprehensive evaluation suite consisting of 29 datasets across various text-related tasks. It provides a multi-dimensional score to measure the fine-grained OCR robustness of large multimodal models.

Structured document and data analysis

- **ChartQA** [Masry et al., 2022] involves visual reasoning over complex charts and plots. It requires the model to synthesize visual data with numerical information, featuring a mix of human-annotated and machine-generated queries.
- **DocVQA** [Mathew et al., 2020] focuses on understanding document images with complex layouts. The model must navigate hierarchical structures to extract and reason about specific textual entities within a professional document context.
- **InfoVQA** [Mathew et al., 2021] evaluates information-rich visuals such as infographics, posters, and advertisements, where visual design and text are intertwined.

Integrated intelligence and domain knowledge

- **MMMU** [Yue et al., 2024] is a massive multi-discipline benchmark designed for expert-level evaluation. It spans college-level subjects (e.g., Art, Science, Business) and demands advanced reasoning beyond simple visual recognition.
- **SeedBench** [Li et al., 2023] uses a large set of multiple-choice questions to assess image and video understanding, spatial awareness, and logical reasoning.
- **MMB1.1**$_{en}$ (MMBench v1.1 English) [Liu et al., 2023b] provides a systematic evaluation framework that uses a Circular Evaluation strategy to eliminate bias in model responses, ensuring a robust measure of the model's holistic multimodal capabilities.

C.2 Evaluation pipeline via VLMEvalKit

VLMEvalKit [Duan et al., 2024] standardizes evaluation across diverse model architectures by separating data preparation, model inference, and metric calculation, thereby avoiding potential biases from inconsistent prompt templates or idiosyncratic post-processing heuristics.

Standardized evaluation protocol. The framework provides a unified interface that ensures all models, including the original dense baselines and the initialized LEGO-MoE models, receive identical interleaved multimodal messages and operate under consistent decoding parameters. For benchmarks involving Multiple-Choice Questions (MCQ) or Yes/No formats, the framework's robust answer extraction mechanism is utilized. In cases where a model's raw output fails to adhere to strict formatting instructions, a semantic matching strategy is applied to capture the intended response, ensuring the results reflect actual comprehension rather than mere instruction-following proficiency.

Circular evaluation and consistency. To mitigate positional bias and the influence of random guessing, the CircularEval strategy is enabled for key MCQ benchmarks. Under this protocol, the options for a given question are circularly shifted N times; a model's response is marked as correct only if it consistently identifies the right answer across all variations. This strict criterion provides a reliable measure of the model's stability and true reasoning capabilities.

Distributed inference and efficiency. The large-scale evaluation suite, comprising nine benchmarks, is handled via a distributed task-partitioning strategy. Leveraging the framework's native support for multi-GPU and multi-process inference, the evaluation workload is decomposed

into independent subsets and executed in parallel across multiple compute nodes. This engineering approach accelerates the evaluation process while maintaining absolute consistency in post-processing and scoring heuristics across the entire experimental suite.

D Implementation and training details

D.1 Hyperparameter configurations

Unless stated otherwise, experiments use the two-stage recovery recipe from the main text. Table 9 lists its default settings: Stage 1 recovers the LLM with the vision tower frozen; Stage 2 uses full-parameter joint SFT with a smaller vision-side learning rate.

Table 9: Hyperparameters for the two-stage recovery paradigm (Stage 1 recovery; Stage 2 joint SFT).

Hyperparameter	**Stage 1**	**Stage 2**
Learning rate	5×10^{-5}	1×10^{-5}
Vision tower LR	–	1×10^{-6}
LR schedule	Cosine	Cosine
Batch size per GPU	8	4
Gradient accumulation	2	4
GPU configuration	$24\times A100$	$24\times A100$
ZeRO	ZeRO-1 offload	ZeRO-2 offload
Optimizer	AdamW	AdamW
Projector	Train	Train
Vision encoder	Freeze	Train
LLM	Train	Train
Max sequence length	4096	8192
Warmup ratio	0.05	0.05
Total steps	8k–12k	4k–6k

D.2 The SWIFT framework and architecture adaptation

The experimental infrastructure relies on SWIFT [Zhao et al., 2025b] for robust full-parameter fine-tuning in large-scale multimodal settings. We implement the transition from dense FFNs to sparse MoE layers through deep modifications to Transformers for models such as Qwen2-VL and InternVL3; SWIFT orchestrates these changes for high-performance execution.

Seamless architectural scaling and open-sourcing. We chose SWIFT for its modular design: the same LEGO-MoE training logic spans 2B–14B backbones without re-engineering the data pipeline or distributed configuration. We modify Transformers modeling scripts so specialized routing logic and MoE layers remain mathematically consistent with the original attention mechanisms. We will release all modified modeling files and training recipes upon publication.

Distributed efficiency and memory management. LEGO's multi-stage recovery requires substantial memory and I/O. SWIFT's mixed-precision training and optimized memory allocation proved critical to stability in high-parameter MoE recovery. Its unified interface significantly reduced cross-architecture overhead, letting us focus on MoE router design and expert recovery.

D.3 Compute cost of deriving the sizing rules

Deriving the sizing rules required a one-time controlled sweep, while applying LEGO to a new target budget requires only the discrete search and one standard recovery run. In our 24-A100 setup, a typical 8B-A3B recovery run takes about 600–800 GPU hours. The full rule-derivation sweep used about 35k GPU hours, excluding dense controls and several ablations, as summarized in Table 10. This cost is reported for transparency; the intended practical value of the rules is to amortize this one-time research cost over later budget-specific model conversions.

Table 10: Approximate compute used to derive and validate the sizing rules.

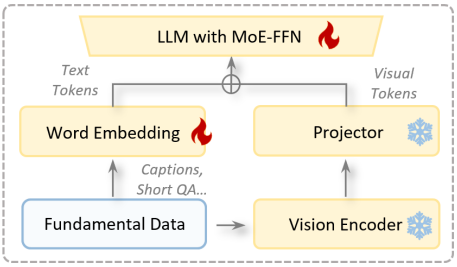
Experiment group	Runs / scale	Approx. GPU hours
Systematic sweeps (Figs. 2, 4)	2B and 8B backbones	24k
Hold-out / forward validation (Fig. 5)	5B/8B/14B backbones	11k
Total one-time rule derivation	–	35k

D.4 Two-stage recovery schedule

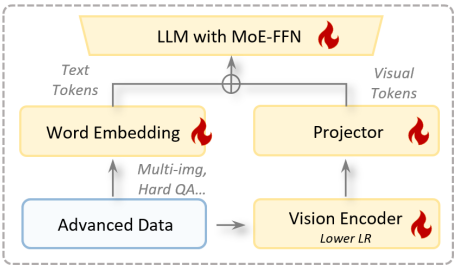
Motivation (cascading errors). In Dense2MoE recovery, Split/Hybrid structurally perturbs routed FFNs at initialization. Updating the vision tower (e.g., ViT) before the LLM recovers can trigger cascading errors: unstable gradients may disrupt aligned visual representations and slow or derail recovery. Fig. 7 shows early recovery trajectories under these Stage-1 target choices.

Paradigm (two-stage recovery). We therefore adopt a two-stage paradigm:

- **Stage 1 (recovery).** Freeze ViT and train the *entire LLM* plus Aligner by default to restore language-side capacity and routing stability; Experiments reports the ablations.
- **Stage 2 (joint SFT).** Appendix D.1 lists the schedule. Jointly fine-tune all parameters for a higher upper bound, using a smaller vision-side learning rate for stability.



(a) Training Stage1: LLM Recovery Training.

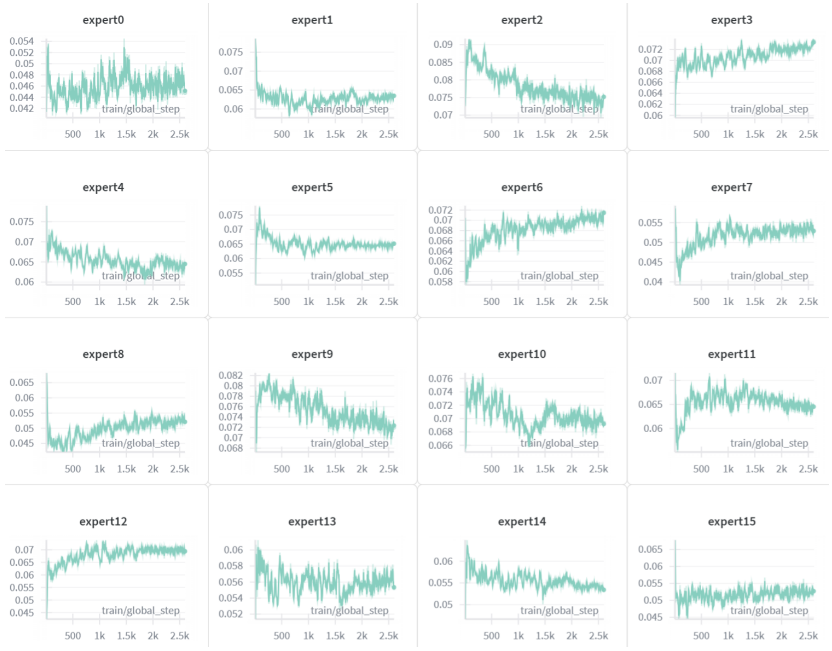


(b) Training Stage2: Full-Parameter Tuning with Differential LRs.

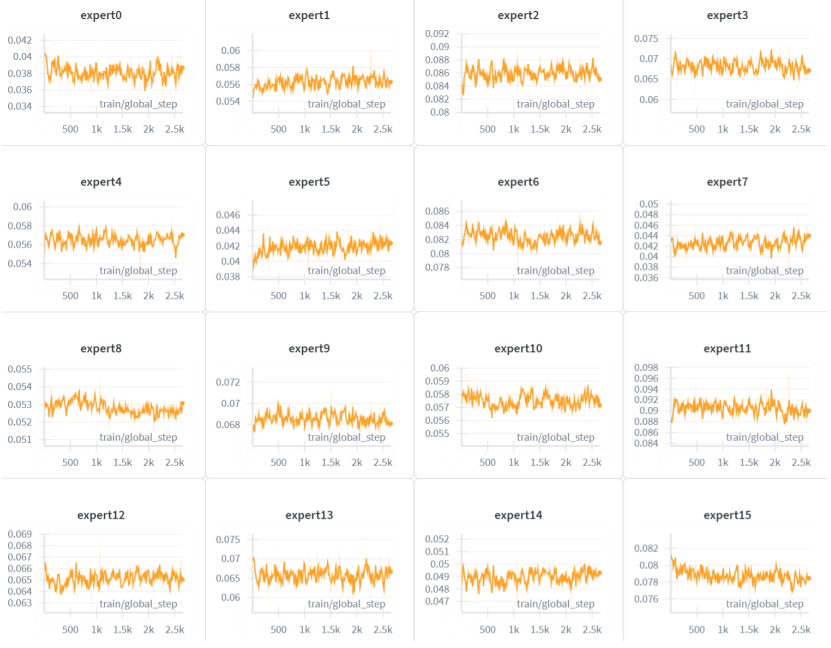
Figure 9: Two-stage recovery workflow used in our experiments. Stage 1 focuses on LLM recovery with the vision side frozen; Stage 2 performs full-parameter tuning with differential learning rates for better stability and a higher upper bound on recovery quality.

D.5 Expert load balancing and distribution analysis

Figure 10 shows the routing distributions for the two coefficients discussed in Method.



(a) $\alpha_b = 5 \times 10^{-4}$ (aux loss ≈ 1.15).



(b) $\alpha_b = 10^{-3}$ (aux loss ≈ 1.08).

Figure 10: Expert load distributions under different load-balancing coefficients. Aux loss is normalized so that 1.0 corresponds to perfectly uniform routing; larger values indicate imbalance.

E Design considerations and discussion

E.1 Design considerations

This section summarizes design choices that keep the study controlled and LEGO actionable under a target (P_{total}, P_{active}) budget. Several are orthogonal to identifying backbone-comparable structural quantities for *fast Dense2MoE recovery*, but address common reviewer and practitioner questions.

Why FFN-only MoE and LLM-only budgeting? We focus on the most common and practical Dense2MoE setting: converting only the LLM FFNs to MoE while keeping attention and the vision tower dense. This isolates the dominant controllable source of (P_{total}, P_{active}) changes in VLMs and avoids conflating language-side conversion with vision-side architectural redesign. Correspondingly, we report budgets under an LLM-only convention (with the ViT treated as a constant and listed separately) to make the structural quantities (D_I/D_H, D_{total}/D_I) comparable across backbones (Appendix A). We treat P_{active} as an *architectural proxy*, not a direct substitute for measured latency.

Why Top-K routing and equal-width splitting? Unless otherwise specified, we use Top-K routing with a fixed number of activated experts per token. This keeps per-token computation stable and makes (P_{total}, P_{active}) well-defined and easy to match in a discrete search space. We also adopt equal-width splitting so that the structural quantities D_I and D_{total} are fully determined by the discrete configuration (N_S, N_E, Top_K) (Sec. 3.1). More expressive designs (e.g., heterogeneous expert widths, dynamic Top-P routing, or expert-choice routing) are promising, but their added degrees of freedom and system variability can obscure our structural comparisons.

Why we do not treat depth / layer selection as a primary axis. A natural question is whether the observed recovery behaviors are mainly driven by *depth* (or which layers are converted) rather than by D_I/D_H. We deliberately do not make depth a primary variable for three reasons. First, changing depth typically requires architectural surgery (layer dropping, re-indexing, and re-tuning) and interacts strongly with the always-on parameter budget P_{fixed}, making (P_{total}, P_{active}) matching less clean across backbones. Second, adding depth as a decision variable expands the search space combinatorially and makes a controlled sweep substantially more expensive. Third, our goal is to provide a transferable rule for the most widely used Dense2MoE operators (Split/Upcycle/Hybrid) that practitioners can apply without re-architecting the backbone. That said, we do incorporate a minimal, stability-motivated layer strategy—keeping the first two and the last layer dense—as a default (Appendix D.4). Layer-wise conversion and depth-aware decisions are complementary directions; LEGO's operator-space formulation is compatible with adding such operators, but we keep them out of the main decision space to maintain clean budget matching and controlled sweeps.

Interpretation of the empirical ranges. The recommended $[1.5, 8]$ and hard-feasible $[1, 12]$ bands summarize observed sweep trends under the studied recovery recipe. The original figures use "Optimal/Suboptimal" as visual labels; these denote empirical guidance, not universal optima. A normalized retention set can be written as $\mathcal{S}_b(\tau) = \{r : y_b(r) \geq \tau y_b^\star\}$, but the exact historical threshold values and boundary extraction records are unavailable in the present draft. We therefore do not claim that this notation alone reproduces the numerical bands or calibrates held-out predictions.

E.2 Potential limitations and constraints

We summarize LEGO's limitations to prevent extrapolation beyond the studied setting.

Scope of architectures and routing. We establish and validate the Sizing Rules in the most common Dense2MoE setting considered in this paper: FFN-only MoEfication with equal-width experts and Top-K routing (topK-then-softmax), under a fixed two-stage recovery recipe. Other design axes (e.g., heterogeneous expert widths, attention MoE, vision-tower MoE, or dynamic-routing-aware sizing) introduce additional degrees of freedom and are outside the current sizing study. The Top-P experiment tests compatibility after fixed-Top-K recovery only.

Recipe sensitivity. The Sizing Rules are empirical and are derived under a specific *fast recovery* regime (two-stage recovery with a fixed data mixture and training budget). The loss curves in Appendix B vary both data amount and training duration, so they do not independently establish data

sufficiency at fixed updates. Accordingly, the reported ranges should be interpreted as robust within this recovery recipe, rather than as an architecture-agnostic law independent of training regime.

Generality of sparsity recommendations. Compared to the broad sweeps for D_I/D_H, the sparsity guideline based on activation ratio Top_K/N_E (equivalently $D_{total}/D_I = N_E/Top_K$) is validated on fewer representative backbones in our main analysis (Fig. 6). We thus present the activation-ratio range $1/8 \sim 1/2$ (equivalently $D_{total}/D_I \in [2, 8]$) as an empirical guideline for our recovery setting, rather than a universal recommendation for all MoE training regimes or from-scratch pretraining.

Budget proxies vs. real systems. We frame LEGO in terms of an Architectural Budget (P_{total}, P_{active}) because total parameters and activated parameters are the most hardware-agnostic "common denominator" for comparing MoE structures across backbones and deployments. In contrast, measured latency/throughput depends heavily on the specific hardware stack and implementation details (kernel fusion, memory bandwidth, routing overhead, load imbalance, and batching), making any single measurement less representative as a transferable sizing guideline. Crucially, our work is not about inference acceleration or MoE system/hardware optimization. LEGO does not introduce special execution logic beyond standard FFN-MoE layers; therefore, it is orthogonal to existing inference optimizations and can benefit from them without changing its structural conclusions. Under this scope, (P_{total}, P_{active}), together with (N_E, Top_K), provides a practical and consistent basis to describe the capacity–compute trade-off and to support our structural analysis.

Baselines and comparison scope. Our main comparisons are intentionally focused on the Split/Upcycle/Hybrid operator space to examine joint base-and-structure choices under the same recovery recipe. Recent Dense2MoE conversion methods (e.g., pruning-based conversion pipelines) and multimodal MoE architectures that MoE-ify additional components are complementary lines of work, but often involve different conversion operators, different assumptions (e.g., dense-training/sparse-inference), or additional training signals, which complicates strictly budget-matched comparisons under our definition. Within the comparable setting defined in Sec. 4, the comparison characterizes evaluated constructions in the Split/Upcycle/Hybrid operator space.

Statistical reporting. The original construction table reports means across three independent test runs. Additional targeted evaluations are point estimates without repeated-run uncertainty. Recorded evaluation repetitions should not be read as independent training seeds. We do not infer significance from small score differences; broader uncertainty estimates would strengthen comparisons.

E.3 Backbone coverage under a fixed sparsity ratio

Dense2MoE often fixes $Top_K/N_E = 1/4$ across scales, but this does *not* fix D_I/D_H: with equal-width splitting, D_I depends on N_S, Top_K, and the backbone-dependent D_{dense}/D_H ratio. Different backbones therefore cover different regions of the (P_{total}, P_{active}) plane at the same sparsity.

Fig. 11 visualizes this effect for $Top_K/N_E = 1/4$. For each candidate dense backbone, we enumerate feasible discrete configurations under this fixed sparsity ratio and highlight those whose resulting D_I/D_H falls into the Optimal/Suboptimal Range. Intuitively, if a target budget lies inside the highlighted (green) region, then the corresponding backbone can serve as a good base model to build a structurally well-matched MoE under sparsity $1/4$; otherwise, one should consider changing the base model, adjusting sparsity, or using a Hybrid construction to steer D_I/D_H back to the recommended range. This perspective provides a concrete explanation of why "fixed activation ratios" can fail systematically across model scales.

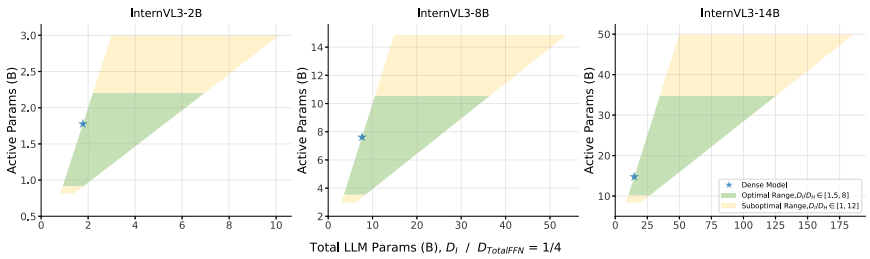


Figure 11: Backbone coverage at fixed sparsity ($Top_K/N_E = 1/4$). Colored regions mark feasible budgets with D_I/D_H in the recommended range; other targets typically require a different base model, sparsity setting, or Hybrid conversion to avoid structural mismatch.

F Scaling factor derivation details

This appendix provides a detailed derivation of the Scaling Factor used in Sec. 3 (Scaling Factor: Moment Matching), with explicit assumptions and boundary cases. We focus on the setting used throughout the paper: **equal-width splitting** with **topK-then-softmax** routing.

The closed form below assumes balanced replication $N_E = qN_S$, $q \in \mathbb{N}^+$, with exactly q copies of every shard, as well as the initialization approximations stated below. It does not cover arbitrary unequal replication or Split+Drop cases.

F.1 Setup and notation

Split+Upcycle construction. We split the dense FFN intermediate dimension into s equal-width shards, and replicate shards to obtain E experts in total. Each expert is a copy of one shard (possibly with multiple replicas per shard). We denote Top-K as K for brevity.

Why the derivation is largely activation-agnostic. Our scaling derivation is written at the *FFN branch output* level and uses only the fact that, after splitting the intermediate dimension into disjoint shards, the dense FFN output can be expressed as a *sum of shard outputs*. This decomposition holds for common FFN parameterizations (e.g., ReLU/GELU MLP, and SwiGLU as used by Qwen2-style MLPs) because the final projection is linear in the intermediate features. Concretely, for SwiGLU,

$$\mathrm{MLP}(x) = W_d\big(\phi(W_g x) \odot (W_u x)\big),$$

splitting the intermediate dimension corresponds to slicing columns of W_g, W_u and rows of W_d consistently; the overall output remains a sum over shard-level contributions. The following derivation therefore does not assume a specific activation function $\phi(\cdot)$, beyond standard initialization symmetry and moment assumptions stated explicitly below.

F.2 Routing order: topK-then-softmax vs. softmax-then-topK

We adopt **topK-then-softmax**: for an input x, we first select the Top-K experts by router logits, and then apply softmax only over these K logits to obtain weights $\{p_i(x)\}$ such that

$$\sum_{i \in \mathrm{Top}_K(x)} p_i(x) = 1.$$

The routed FFN output (before scaling) is

$$z(x) = \sum_{i \in \mathrm{Top}_K(x)} p_i(x)\, E_i(x), \quad y_{\text{moe}}(x) = C \cdot z(x).$$

Under **softmax-then-topK**, softmax over all E experts precedes Top-K, discarding the remaining probability mass. Diffuse probabilities at random initialization typically add a global shrinkage factor (roughly K/E) and complicate scale alignment; we therefore use topK-then-softmax.

F.3 Mean matching: $C_{\text{mean}} = s$

After splitting, denote by $y_j(x)$ the shard output of shard $j \in \{1, \dots, s\}$. The dense FFN output can be written as

$$y_{\text{dense}}(x) = \sum_{j=1}^{s} y_j(x).$$

Since experts are replicas of shards, the routed aggregation can be rewritten at the shard level as

$$z(x) = \sum_{j=1}^{s} w_j(x)\, y_j(x), \quad w_j(x) \geq 0, \ \sum_{j=1}^{s} w_j(x) = 1,$$

where $w_j(x)$ is the total routing weight assigned to replicas of shard j.

Symmetry assumption. At initialization, we assume symmetry across shards: $\mathbb{E}[w_j(x)] \approx 1/s$. Then

$$\mathbb{E}[y_{\text{moe}}(x)] = C \cdot \mathbb{E}[z(x)] \approx C \cdot \frac{1}{s} \sum_{j=1}^{s} y_j(x) = \frac{C}{s}\, y_{\text{dense}}(x).$$

Thus mean matching yields

$$C_{\text{mean}} = s.$$

F.4 Variance matching and the derivation of C_{var}

Mean matching alone may amplify routing-induced noise when $K < s$ or when replication causes repeated shard hits. We therefore also match the second moment (RMS scale) at initialization.

Moment assumptions. We use standard approximations during initialization: (i) shard outputs $\{y_j(x)\}$ are independent and identically distributed with zero mean and variance σ^2; (ii) shard outputs are approximately independent of routing weights $w_j(x)$; (iii) within the selected Top-K, routing weights are approximately uniform, so each selected expert carries weight $\approx 1/K$.

From expert selection to shard weights. Let n_j be the number of selected experts among the K that correspond to shard j. Under assumption (iii), the shard-level weight satisfies

$$w_j \approx \frac{n_j}{K}.$$

Moreover, when the router is close to uniform at initialization, selecting K experts can be approximated as sampling K items uniformly *without replacement* from E experts, where experts are partitioned into s groups (shards), each with E/s replicas. Then n_j follows a hypergeometric distribution with population size E, number of successes E/s, and draws K, giving

$$\mathbb{E}[n_j] = \frac{K}{s}, \qquad \mathrm{Var}(n_j) = K \cdot \frac{1}{s}\Big(1 - \frac{1}{s}\Big) \cdot \frac{E-K}{E-1}.$$

Therefore,

$$\mathbb{E}[n_j^2] = \mathrm{Var}(n_j) + \mathbb{E}[n_j]^2 = K \cdot \frac{s-1}{s^2} \cdot \frac{E-K}{E-1} \ + \ \frac{K^2}{s^2}.$$

Summing over shards and dividing by K^2, we obtain

$$\mathbb{E}\Big[\sum_{j=1}^{s} w_j^2\Big] \approx \mathbb{E}\Big[\sum_{j=1}^{s} \Big(\frac{n_j}{K}\Big)^2\Big] = \frac{1}{s}\Big(1 + \frac{s-1}{K} \cdot \frac{E-K}{E-1}\Big).$$

Matching the RMS scale. Under assumption (i), the dense output has variance

$$\mathrm{Var}(y_{\text{dense}}) = \mathrm{Var}\Big(\sum_{j=1}^{s} y_j\Big) = s\sigma^2.$$

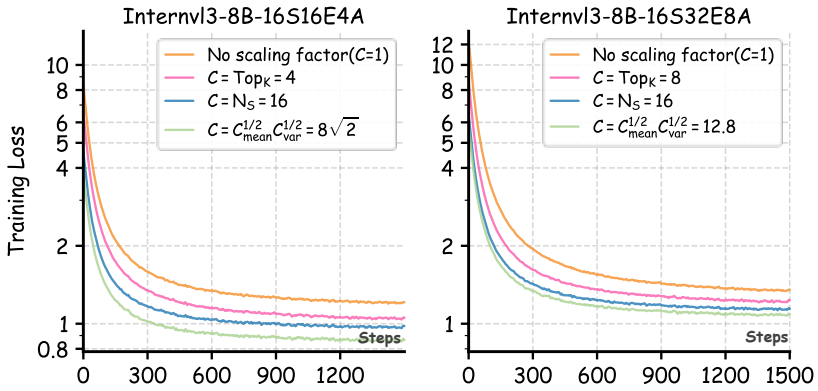


Figure 12: Scaling factor ablation: it reduces initial loss, improves early convergence, and stabilizes recovery in the two displayed balanced configurations.

The MoE output has variance (conditioning on weights and using independence)

$$\mathrm{Var}(y_{\text{moe}}) \approx C^2 \cdot \sigma^2 \cdot \sum_{j=1}^{s} w_j^2.$$

Taking expectation and matching $\mathbb{E}[\mathrm{Var}(y_{\text{moe}})] \approx \mathrm{Var}(y_{\text{dense}})$ yields

$$C_{\text{var}} = \sqrt{\frac{s}{\mathbb{E}[\sum_j w_j^2]}} = \frac{s}{\sqrt{1 + \frac{s-1}{K} \cdot \frac{E-K}{E-1}}}.$$

Boundary cases. This expression has the correct limits:

- **Pure Upcycle (no split):** $s = 1 \Rightarrow C_{\text{mean}} = C_{\text{var}} = 1$.
- **Pure Split (no replication):** $s = E \Rightarrow C_{\text{var}} = \sqrt{E \cdot K}$.

F.5 Why we use a geometric-mean compromise

When $K < s$ or when replication increases repeated shard hits, a single scalar cannot simultaneously satisfy mean matching ($C_{\text{mean}} = s$) and variance matching ($C_{\text{var}} < s$). We therefore use a robust compromise

$$C = \sqrt{C_{\text{mean}} \cdot C_{\text{var}}},$$

which lies in $[C_{\text{var}}, C_{\text{mean}}]$ and preserves the correct boundary cases (e.g., $C = 1$ for $s = 1$). In practice, it reduces early-stage loss and improves convergence stability for Split+Upcycle and Hybrid.

G Supplementary baselines and sensitivity

G.1 Pruning-based Dense2MoE comparison

Table 11 retains the original six-benchmark LEGO/ToMoE comparison. Its averages should not be compared with the four-benchmark ToMoE/CMoE results in Appendix H.1. Because ToMoE changes the conversion operator space, these runs provide supplementary evidence; the main paper focuses on rule-guided choices within the Split/Upcycle/Hybrid operator space.

Table 11: Supplementary ToMoE comparison. Scores are six-benchmark averages, with drops measured against the corresponding Dense-after-train control for each budget.

Target budget	Method	4% data	49% data	100% data
$(8B, 5B)$	ToMoE	72.3 (-10.4)	76.2 (-6.5)	77.6 (-5.1)
$(8B, 5B)$	LEGO	69.4 (-13.3)	77.8 (-4.9)	80.0 (-2.7)
$(2B, 0.8B)$	ToMoE	60.5 (-16.1)	67.6 (-9.0)	68.7 (-7.9)
$(2B, 0.8B)$	LEGO	61.1 (-15.5)	69.8 (-6.8)	71.9 (-4.7)

Table 12: Sensitivity to λ in the geometric-mean scaling factor.

Recovery data	0	0.25	0.5	0.75	1
4%	64.5	67.9	68.5	67.2	65.3
24%	71.1	73.8	74.6	74.3	73.2
49%	75.2	77.6	77.8	76.8	74.6
100%	78.2	79.3	80.0	79.1	77.8

G.2 Sensitivity to the scaling-factor exponent

Table 12 reports the recovery average for $C = C_{\text{mean}}^{\lambda} C_{\text{var}}^{1-\lambda}$ on InternVL3-8B-A5B. The extremes are weaker early in recovery, while $\lambda \in \{0.25, 0.5, 0.75\}$ is relatively stable; across data fractions, $\lambda = 0.5$ is best or tied for best among the tested exponents.

H Additional controlled evaluations

This appendix reports targeted evaluations of construction, routing, and deployment. Unless indicated otherwise, parameter counts refer to the LLM alone, whereas system measurements cover the full vision–language model (VLM). Each four-task average is the arithmetic mean of the displayed TextVQA, OCRBench, ChartQA, and MMBench 1.1 English (MMB1.1-en) scores. These targeted comparisons are descriptive; repeated-run uncertainty estimates are not available for the new results.

H.1 Matched conversion and budgeted construction

We compare a reported LEGO construction with ToMoE [Gao et al., 2025], which learns token-dependent expert subnetworks by dynamic structural pruning, and CMoE [Pei et al., 2025], which uses activation statistics to carve shared and routed experts. ToMoE uses one active mask; p is its pruning setting. All methods use the same nested 4%/49%/100% recovery-data fractions, two-stage recovery, and evaluation protocol. CMoE additionally uses 64 Cambrian-737k calibration samples; its shared experts are counted as active. The comparison covers two target budgets rather than an exhaustive sweep or a controlled test of Algorithm 1's ranking criterion. Table 2 summarizes the recovery trajectories; Table 13 reports the task-level scores at the full-data endpoint.

Table 13: Per-task scores at Table 2's 100% endpoint; Avg is the displayed-score arithmetic mean.

Target	Method	TextVQA	OCRBench	ChartQA	MMB1.1-en	Avg
(8B, 5B)	LEGO	80.6	87.9	83.0	74.3	81.5
	ToMoE	78.7	87.0	79.3	71.6	79.2
	CMoE	79.9	88.4	82.5	74.9	81.4
(8B, 3B)	LEGO	80.3	85.7	81.5	78.7	81.6
	ToMoE	79.1	85.1	76.6	64.3	76.3
	CMoE	81.3	85.6	79.5	64.9	77.8
	Split	76.2	84.1	73.4	66.5	75.1

With the 8B source fixed, ToMoE leads at 4% data, CMoE at 49%, and LEGO and CMoE are nearly tied at 100% (81.5 versus 81.4). Under the tighter active budget, LEGO's 2B Hybrid construction has the highest average at all three data fractions, but it also changes the source backbone. These results support conditional competitiveness for the tested budgets, data, and recovery protocol. The 81.6 versus 81.5 comparison across the two target blocks is an essentially tied result on this four-task subset; the corresponding nine-benchmark averages in the main paper are 76.2 and 77.8, respectively.

H.2 Measured inference cost

We measured batch-size-1 full-VLM inference on an A100 GPU over approximately 48,000 queries from the nine benchmarks in the main construction table. Latency is the mean end-to-end time per query over this workload, and throughput counts actual generated output tokens; neither is a

fixed-length microbenchmark. LEGO uses its native fixed Top-K routing and grouped-GEMM MoE kernels. Expert weights remain GPU-resident. Table 3 reports LLM-only parameter counts together with full-VLM latency, throughput, peak GPU memory, and computation. Here, "Dense-AfterSFT" labels the same trained Dense checkpoint as "Dense-AfterTrain" in the main construction comparison.

Recorded measurement definitions and missing metadata. The source record counts generated tokens using `outputs.sequences[:, input_length:]` and computes TFLOPs/query with `torch.utils.flop_counter`. It records A100 hardware, batch size one, approximately 48k queries, full-VLM inference, mean end-to-end seconds per query, and grouped-GEMM kernels for the MoE rows. The supplied record does not specify the A100 variant, numeric precision, software/kernel versions, warmup and detailed timing boundaries, decoding/image settings, input/output length distributions, or the denominator and order used to aggregate throughput. Table 3 therefore contains author-reported workload summaries with limited replayability. Its throughput ratios do not isolate decoder or kernel speed. Without the aggregation definition, mean latency times throughput does not give a verified estimate of mean output length for the measured workload.

As descriptive comparisons of the reported workload summaries, approximately total-parameter-matched points (within 5%) reduce TFLOPs/query by 33.4–48.4% and mean latency by 12.4–23.3%, while increasing output throughput by 10.8–56.4%. Peak memory increases by 0.1–0.4 GiB, consistent with resident expert weights and routing/dispatch buffers. The 8B Dense quality mean is recomputed from the displayed main-table task scores (80.9 rather than the earlier reported 80.8); the system measurements are unchanged. The quality averages fall by 3.1–4.7 points relative to the displayed Dense controls. The 2B pair and the 8B Dense versus $(8\mathrm{B}, 5\mathrm{B})$ LEGO pair have the same source backbone; the 2B-sourced $(8\mathrm{B}, 3\mathrm{B})$ row is a matched-total deployment point, not a same-source effect. These results are specific to the tested hardware, kernels, batch size, model family, and workload.

H.3 Post-recovery token-adaptive routing

To test compatibility with dynamic expert activation, we applied the cumulative Top-P selection rule of Huang et al. [2024] at inference to recovered InternVL3 checkpoints, with selected-weight renormalization in this adaptation. Router logits are softmaxed over all experts, experts are sorted by probability, and the smallest prefix whose cumulative probability strictly exceeds threshold p is selected; selected weights are then renormalized. The structure, recovered weights, a reported fixed reference aggregation setting, and benchmark protocol remain fixed within the Top-P test. Its exact scaling implementation is not mapped to the analytic expression in Method. Thresholds were chosen on an independent calibration split using router activation statistics, without benchmark accuracy, to keep mean active parameters within $\pm 5\%$ of each fixed Top-K reference. The $(8\mathrm{B}, 5\mathrm{B})$ LEGO and Split rows are the same checkpoint, so they appear once. Tables 14 and 15 separate routing-budget statistics from benchmark quality.

Table 14: Top-P routing budget; mean active parameters and expert counts are token averages.

Target	Checkpoint	Routing	Total	Mean active	Mean experts
$(8\mathrm{B}, 5\mathrm{B})$	LEGO = Split, 8B/16s16E8A	Top-8	7.65B	4.79B	8.0
	Same checkpoint	Top-P, p=0.80	7.65B	≈4.69B	≈7.8
$(8\mathrm{B}, 3\mathrm{B})$	LEGO Hybrid, 2B/2s16E4A	Top-4	7.84B	2.84B	4.0
	Same checkpoint	Top-P, p=0.65	7.84B	≈2.93B	≈4.1
	Split, 8B/16s16E2A	Top-2	7.65B	2.66B	2.0
	Same checkpoint	Top-P, p=0.45	7.65B	≈2.77B	≈2.2

The displayed results show that these recovered configurations remain effective with token-adaptive expert counts at closely matched mean active budgets. The fixed references and Top-P measurements have different public result lineages, so their small average differences should not be interpreted as controlled same-run gains. The $(8\mathrm{B}, 3\mathrm{B})$ LEGO and Split rows also use different source backbones. This experiment tests an inference-time adapter on InternVL3; it does not establish a Top-P-aware sizing rule or cross-family transfer of the routing thresholds. The supplied record does not identify the calibration split, its sample count, or its overlap-checking procedure; these routing thresholds remain specific to the reported checkpoints.

Table 15: Four-task quality of paper-reported fixed Top-K and post-recovery Top-P routing.

Target	Checkpoint/routing	TextVQA	OCRBench	ChartQA	MMB1.1-en	Avg
(8B, 5B)	LEGO, Top-8	80.6	87.9	83.0	74.3	81.5
	LEGO, Top-P	82.3	85.2	84.6	75.1	81.8
(8B, 3B)	LEGO, Top-4	80.3	85.7	81.5	78.7	81.6
	LEGO, Top-P	81.5	84.8	83.4	77.1	81.7
	Split, Top-2	76.2	84.1	73.4	66.5	75.1
	Split, Top-P	76.0	82.5	75.3	70.2	76.0

H.4 Text-only sizing controls

We tested Qwen2.5-7B-Instruct and Llama-3.1-8B-Instruct after recovery using the Tulu 3 SFT mixture [Lambert et al., 2024], evaluating MMLU-Redux (MMLU-R), GSM8K, HumanEval, and IFEval. Table 16 reports the four task scores and their arithmetic average. For both backbones and each task, the three displayed MoE configurations have the same low-to-high ordering as activated width D_I/D_H increases. The widest configurations recover to approximately the corresponding Dense-AfterSFT control. Thus the qualitative bottleneck-and-recovery sizing signal transfers to these text-only models; these data do not establish that the numerical VLM recommendation $[1.5, 8]$ is an LLM optimum or indicate a modality-specific shift in that range. The multimodal-specific evidence concerns the early recovery schedule across the converted LLM, projector, and vision tower (Fig. 7).

Table 16: Text-only controls under the Tulu 3 SFT mixture [Lambert et al., 2024]. Averages are arithmetic means of the four displayed task scores for each model row.

Backbone/configuration	D_I/D_H	MMLU-R	GSM8K	HumanEval	IFEval	Avg
Llama/16s16E4A	0.875	48.7	63.4	51.2	61.6	56.2
Llama/8s8E4A	1.750	61.3	76.8	65.7	70.4	68.6
Llama/Dense-AfterSFT	3.500	68.5	82.6	74.7	74.3	75.0
Llama/1s4E2A	7.000	69.1	83.9	75.0	74.5	75.6
Qwen/16s16E4A	1.321	65.3	78.1	74.3	65.4	70.8
Qwen/8s8E4A	2.643	71.2	83.5	78.7	70.2	75.9
Qwen/Dense-AfterSFT	5.286	73.6	87.4	83.0	73.9	79.5
Qwen/1s4E2A	10.571	74.1	88.5	83.3	73.8	79.9

I Selection specification and reported configurations

Algorithm 1 specifies filtering followed by lexicographic sorting, not a weighted score. Candidate inputs are declared before benchmark evaluation. The last four key entries are the complete stab tie-break: prefer $K \in [2, 8]$, then smaller $|K - 4|$, then earlier base and tuple input positions. If no candidate survives, return `infeasible`; do not silently relax the budget or width filters. Slack is one-sided: undersized candidates can survive, but the ranking penalizes them for their absolute relative budget errors.

The default layer strategy retains the first two and last LLM layers as dense. These layers enter P_{fixed} in Eq. 3; any non-default converted-layer scope must be declared. For bases with different widths across layers, apply the hard width interval to every converted layer and prefer candidates whose converted layers all satisfy the recommended interval. The reported homogeneous-width bases use the single ratio in the main algorithm.

Balanced Hybrid has $N_E = qN_S$ with the same integer number q of replicas per shard. Non-integer replication requires an explicit allocation rule and is outside the reported lists and the closed-form scaling derivation. Thus there is no universal divisibility filter on N_E, but a realizable construction must accompany each candidate. The empirical routing guideline $K/N_E \in [1/8, 1/2]$ does not change the sorting key. Pure-path comparator rows in Table 1 may lie outside LEGO's recommended or hard width ranges; they are evaluated representatives of restricted operator families.

Table 17: Operator mapping from the selected tuple. Balanced replication is the reported setting; the general operator name alone does not define shard allocation for a candidate configuration.

Condition	Operator
$(N_S, N_E) = (1, 1)$	Dense identity boundary
$N_S = 1 < N_E$	Upcycle
$1 < N_S = N_E$	Split
$1 < N_S < N_E$	Hybrid
$N_E < N_S$	Split+Drop, explicit diagnostic lists only

Table 18: LEGO configurations retained from the submitted main table. Realized budgets are reported in the author-response experiments and are not reconstructed historical selection logs.

Target total/active	Source	(N_S, N_E, K)	D_I/D_H	Reported total/active
8B/5B	InternVL3-8B	$(16, 16, 8)$	2.64	7.65B/4.79B
8B/3B	InternVL3-2B	$(2, 16, 4)$	11.67	7.84B/2.84B
17B/5B	InternVL3-8B	$(8, 24, 4)$	2.64	Not verified
8B/3B	Qwen2-VL-2B	$(2, 16, 4)$	11.53	Not verified

Historical selection records. The complete ordered $\mathcal{B}$ and Ω_t, per-setting slack and layer scope, and surviving-candidate ranking values still need to be consolidated from the original runs. The nominal $\epsilon = 0.05$ must not be taken as proof that every main-table row satisfies a uniform retrospective replay. The table records evaluated configurations, while Algorithm 1 makes the decision rule explicit. Exact replay of the historical selections remains pending those inputs; no candidate list or ranking value is inferred from the observed benchmark scores.

J Per-task scores for the main construction comparison

The following tables preserve every benchmark cell from the submitted construction table, including the 17B/5B block. Means are recalculated from the displayed component scores; no experimental score is changed. Target labels do not establish verified realized budgets for the 17B/5B block. IVL3 denotes InternVL3 and Q2VL denotes Qwen2-VL; the historical Q2VL-8B name refers to the 7B language-backbone model. The evaluation configuration still needs review to confirm the exact SEED benchmark evaluator variant.

Table 19: Main construction scores: perception and document tasks, unchanged from the submission.

Method	Source/configuration	AI2D$_{w/o}$	TextVQA	OCRBench	ChartQA	DocVQA
Target total/active: 8B/5B						
Upcycle	IVL3-2B-1s8E4A	91.5	79.2	87.5	81.5	88.0
Split	IVL3-8B-16s16E8A	92.1	80.6	87.9	83.0	89.3
LEGO	IVL3-8B-16s16E8A	92.1	80.6	87.9	83.0	89.3
Dense	IVL3-8B-AfterTrain	92.7	80.5	88.6	86.1	92.6
Target total/active: 8B/3B						
Upcycle	IVL3-2B-1s8E2A	88.9	79.9	85.3	81.1	90.3
Split	IVL3-8B-16s16E2A	87.5	76.2	84.1	73.4	83.9
LEGO	IVL3-2B-2s16E4A	89.3	80.3	85.7	81.5	90.7
Dense	IVL3-2B-AfterTrain	87.3	77.2	83.8	80.1	89.4
Target total/active: 17B/5B						
Upcycle	IVL3-2B-1s16E4A	92.1	79.8	88.1	82.1	88.6
Split	IVL3-14B-32s32E2A	82.1	67.5	75.9	56.1	58.1
LEGO	IVL3-8B-8s24E4A	93.4	80.5	88.5	84.6	92.2
Dense	IVL3-14B-AfterTrain	93.9	80.9	88.9	86.6	93.6
Target total/active: 8B/3B (Qwen2-VL)						
Upcycle	Q2VL-2B-1s8E2A	88.9	83.1	84.9	75.9	92.3
Split	Q2VL-8B-16s16E2A	84.5	79.0	80.7	71.8	88.1
LEGO	Q2VL-2B-2s16E4A	89.7	83.9	85.7	76.5	92.9
Dense	Q2VL-2B-AfterTrain	85.3	80.1	81.6	73.5	90.1

Table 20: Main construction scores for information and comprehensive tasks (submission values).

Method	Source/configuration	InfoVQA	MMMU	SEED	MMB1.1-en
Target total/active: 8B/5B					
Upcycle	IVL3-2B-1s8E4A	73.5	55.0	59.5	70.8
Split	IVL3-8B-16s16E8A	74.8	57.3	61.0	74.3
LEGO	IVL3-8B-16s16E8A	74.8	57.3	61.0	74.3
Dense	IVL3-8B-AfterTrain	77.2	61.6	67.5	80.9
Target total/active: 8B/3B					
Upcycle	IVL3-2B-1s8E2A	67.3	46.6	64.6	78.3
Split	IVL3-8B-16s16E2A	61.1	39.1	52.7	66.5
LEGO	IVL3-2B-2s16E4A	67.7	47.0	65.0	78.7
Dense	IVL3-2B-AfterTrain	64.8	47.3	62.9	78.0
Target total/active: 17B/5B					
Upcycle	IVL3-2B-1s16E4A	74.1	55.6	60.1	71.4
Split	IVL3-14B-32s32E2A	56.0	37.6	42.0	48.5
LEGO	IVL3-8B-8s24E4A	83.4	62.1	64.7	76.9
Dense	IVL3-14B-AfterTrain	84.0	65.2	67.9	81.6
Target total/active: 8B/3B (Qwen2-VL)					
Upcycle	Q2VL-2B-1s8E2A	68.6	48.1	62.3	73.6
Split	Q2VL-8B-16s16E2A	64.5	43.9	58.5	69.8
LEGO	Q2VL-2B-2s16E4A	69.2	48.5	62.7	74.0
Dense	Q2VL-2B-AfterTrain	66.7	46.7	61.2	72.8

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and Section 1 state the contributions and their scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Limitations are discussed in Section 5 and Appendix E.2.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

   Justification: Assumptions and derivations appear in Section 3.3 and Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: Detailed data that helps reproduce our results are provided in our appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: A verified public code release and complete reproduction instructions are soon available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: Training and evaluation settings are given in Appendices B–D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: Cconfidence intervals are involved and additional evaluations use point estimates.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: Appendix D.3 reports hardware and approximate costs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The study uses public assets and involves no new human-subject research.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: Positive and negative impacts are discussed in Section 5 and the limitations and broader impacts section.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: No new high-risk pretrained model or scraped dataset is released in this work.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: Assets are cited properly in our paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: Supplementary code and detailed appendix are provided in our paper, which including almost all the key data or experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The study involves no crowdsourcing or new research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The study involves no new human-subject research; IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: LLM tools assisted manuscript writing and polishing; they were not a component of the research method.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.