# Budget-Aware Submodel Extraction from Vision-Language Foundation Models

Anonymous ECCV 2026 Submission

Paper ID #10839

**Abstract.** Large vision-language foundation models (VLMs) provide strong generalization but are often over-capacitated for real-world deployment, where applications typically involve a small set of tasks and strict computational budgets. Existing approaches either incorporate budget adaptivity during pretraining or perform compression while attempting to preserve universal zero-shot capability, which inherently limits its achievable compression. In this work, we instead formulate deployment as a *task-conditioned, budget-aware submodel extraction* problem: given a pretrained model, a target task set, and a budget, the goal is to derive a structured submodel that maximizes task performance under the constraint. We propose a post-hoc extraction method that searches over heterogeneous structural reductions, including attention dropping, MLP dropping, block dropping, and width reduction. We rank removable units using lightweight task-aware evaluation and progressively selects the operator that yields the best accuracy–budget trade-off at each reduction step. Without modifying the pretraining process, our approach produces task-aligned submodels that maintain strong task accuracy under tight parameter and compute budgets across multiple VLMs.

## 1 Introduction

The emergence of large-scale vision-language foundation models (VLMs) has fundamentally reshaped visual understanding. Trained on web-scale multimodal data, these models demonstrate strong generalization across diverse tasks such as visual question answering [1], image captioning [12, 27], and multimodal reasoning [6]. A growing number of powerful open-source VLMs are now publicly available [2, 14]. Despite architectural differences, they share common characteristics: large scale, broad versatility, and strong adaptation capability. As a result, a single pretrained model can often serve as a universal backbone for a wide range of downstream multimodal tasks.

However, the increasing scale and universality of these models expose a mismatch with real-world deployment. Practical applications are often highly specialized and require only a narrow subset of capabilities [21, 24, 25]. In many cases, large portions of a foundation model's capacity become functionally redundant. For instance, an embedded vision system may only need pose estimation or anomaly detection rather than the full spectrum of dense prediction or reasoning tasks [30]. Moreover, deployment environments frequently have strict

computational constraints. Applications such as robotics and autonomous systems demand real-time inference under limited hardware budgets, where the parameter count and computational overhead of modern foundation models can be prohibitive. As models continue to scale for broader generalization, the gap between large-scale pretraining and budget-constrained deployment widens. Existing efforts focus primarily on improving pretraining or enhancing universal capability, yet provide limited mechanisms for deriving models that align simultaneously with specific task needs, computational budgets, and real-world deployment requirements [11, 16, 28].

Existing approaches to mitigate this mismatch generally fall into two categories. The first follows the once-for-all (OFA) model training [3,4], where budget adaptivity is incorporated during pretraining so that sub-networks of different computational costs can be selected after training. However, such methods assume control over the pretraining process, which is often inaccessible to practitioners relying on publicly available foundation models. The second category consists of post-hoc compression techniques, including structured pruning and sparsification [8, 9], which aim to reduce the size of the model while preserving overall zero-shot generalization. Because these methods seek to maintain broad universal performance, the degree of compression achievable is inherently limited. In contrast, practical deployment typically requires flexible compression levels tailored to specific budgets and focuses on a small subset of domain-specific tasks rather than universal capability [22]. As a result, current paradigms are not explicitly designed to produce aggressively compressed submodels aligned with tasks that match the requirements of the concrete task and the budget.

In this work, we reformulate deployment as a task-conditioned, budget-constrained submodel extraction problem. Given a pretrained VLM, a small set of target tasks, and a specified computational budget, our goal is to derive a structured submodel that maximizes task performance while satisfying the budget constraint. Instead of preserving universal zero-shot capability, we explicitly focus on aligning model capacity with the requirements of specific downstream tasks. To address this problem, we propose a post-hoc extraction framework that searches over heterogeneous structural reductions in transformer-based vision-language models. The framework considers multiple structural operators, including attention removal, MLP removal, block dropping, and MLP width reduction. A lightweight task-aware ranking stage first estimates the relative importance of removable units, after which a granularity-controlled greedy search progressively selects the operator that provides the best accuracy–budget trade-off at each reduction step. This design enables flexible adjustment of model depth and width without modifying the original pretraining process.

By decoupling deployment adaptation from pretraining and explicitly optimizing task-specific performance, our approach produces task-aligned submodels that establish a clear trade-off between task accuracy and computational budget. This perspective provides a practical mechanism for transforming publicly available foundation models into budget-efficient models suitable for diverse deployment scenarios. Our contributions are summarized as follows:

- **Task-Conditioned Submodel Extraction.** We formulate deployment of vision-language foundation models as a budget-aware, task-conditioned submodel extraction problem, which focuses on maximizing task performance under explicit resource constraints instead of preserving universal capability.
- **Heterogeneous Structural Extraction Framework.** We propose a post-hoc extraction framework that searches over heterogeneous structural reductions (attention, MLP, block, and width) and progressively constructs submodels that achieve favorable accuracy–budget trade-offs.
- **Comprehensive Empirical Study.** Through systematic experiments on multiple VLM backbones, including Qwen2.5-VL and LLaVA models, we show that task-aligned submodel extraction consistently achieves strong accuracy under significantly reduced parameter and compute budgets.

## 2 Related Work and Preliminaries

### 2.1 Once-for-all Model

One important way to extract submodels from a pretrained model is to train a Once-for-All (OFA) model [3, 19]. OFA training adopts a progressive shrinking algorithm, which starts from the largest full-capacity model and gradually supports smaller submodels within a single over-parameterized supernet. The final model can flexibly instantiate sub-networks of different depths, widths, kernel sizes, or other architectural dimensions, while sharing the same set of pretrained weights across all configurations. In the context of foundation models, a closely related paradigm is often referred to as many-in-one models [4, 5]. These models follow a similar shrinking philosophy during training, but introduce architectural mechanisms, such as routing modules that dynamically select different submodels or computational paths at inference time. This design is conceptually related to Mixture-of-Experts (MoE) models [17, 29], although the experts in these systems may have heterogeneous sizes and partially shared parameters rather than being completely independent. However, both OFA and many-in-one approaches require modifying the training pipeline or redesigning the model architecture during pretraining, for example by introducing routers, restructuring layers, or explicitly training a supernet that supports progressive shrinking. In practice, many users rely on publicly released foundation models and do not have access to the original training process, making it unrealistic to retrain models with additional architectural constraints. Consequently, while OFA-style approaches provide elegant flexibility in principle, they are difficult to apply in plug-and-play deployment scenarios where the objective is to derive efficient submodels directly from an already pretrained and fixed foundation model without modifying the original training procedure.

### 2.2 Structure Pruning in Transformer-based Foundation Models

Following the Transformer architecture [23], structure pruning in Transformer-based foundation models typically focuses on three structural dimensions: attention modules, MLP (feed-forward) modules, and entire Transformer blocks.

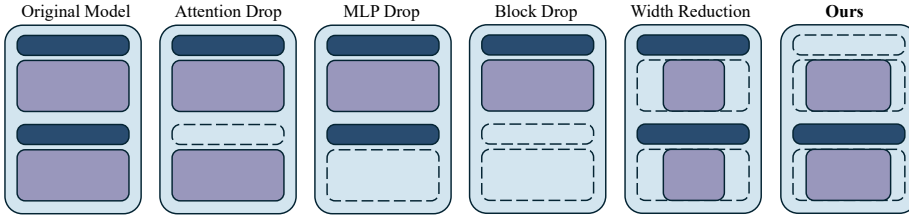


**Fig. 1: Structural pruning strategies in Transformer models.** Dark blue blocks denote attention modules, purple blocks denote MLP modules, dashed blocks indicate removed components, and narrower blocks indicate width reduction. **Ours:** We summarize the performance of different methods on VLMs and propose an automated pipeline to generate candidate sub-models by combining multiple strategies, selecting the best sub-model under target constraints.

Recent compression methods [9,16] explore different strategies along these axes, including attention drop [11], MLP drop [18], block drop, and MLP width reduction [8, 13]. Attention drop primarily aims to reduce the memory and latency overhead associated with the key-value (KV) cache during autoregressive inference, rather than significantly decreasing the total number of model parameters, since attention layers account for only a relatively small fraction of the parameters in Transformer models [11]. In contrast, MLP drop and MLP width reduction are motivated by empirical observations that MLP layers are often over-parameterized and contain substantial redundancy. As MLP modules typically dominate the parameter count in large language models, pruning or shrinking them can substantially reduce model size while maintaining competitive performance. Block drop removes entire Transformer blocks and therefore reduces both attention and MLP parameters simultaneously, but this strategy is relatively coarse-grained and often lacks fine-grained control over parameter allocation and capacity preservation. Although these pruning strategies have been widely studied individually, existing approaches typically rely on a single operator or apply pruning along a fixed structural dimension. As a result, they provide limited flexibility when adapting models to different deployment constraints. Moreover, most prior work focuses on preserving universal or zero-shot generalization across tasks. In contrast, we focus on extracting task-specific sub-models under explicit resource budgets, which relaxes the requirement of maintaining universal capability and enables more aggressive compression strategies tailored to the target deployment scenario.

### 2.3 Compression Methods in Vision-Language Models

Since our goal is submodel extraction, we rely on structured pruning (i.e., compression) algorithms as the core tools. We therefore first conduct an empirical comparison of several representative compression strategies on common VLMs in order to understand their behavior under different deployment constraints.

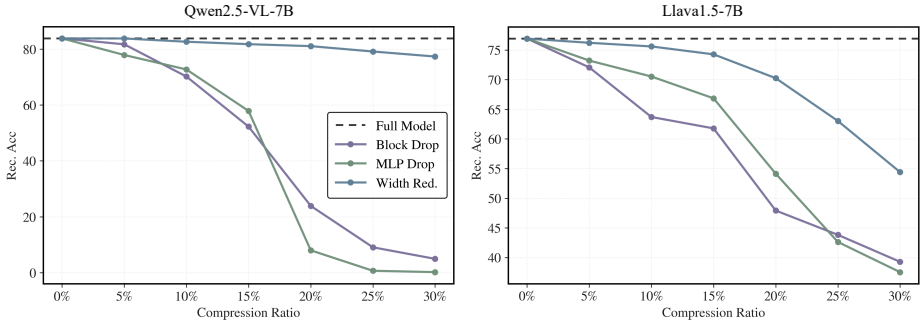


**Fig. 2:** The performance of different compression methods in 7B Qwen2.5-VL and Llava1.5 models. The performance is evaluated on VQA v2. Different compression methods share the same ratio of dropped parameters to the total number of parameters.

For block drop, MLP drop, and MLP width reduction, we align the comparison by controlling the proportion of removed parameters relative to the total parameter count of the original model, ensuring evaluation under the same compression ratio. In contrast, attention drop removes only a small fraction of parameters and mainly improves efficiency by reducing the key-value (KV) cache during autoregressive inference; we therefore evaluate it separately. Figure 2 presents results on instruction fine-tuned Qwen2.5-VL-7B and LLaVA1.5-7B using VQA v2 [7] and MMBench [15] as representative benchmarks. Empirically, under the same parameter drop ratio, MLP width reduction generally preserves performance better than block drop and MLP drop, particularly at higher compression levels. This observation suggests that, in our evaluated settings, redundancy in these VLMs is more concentrated in the width of MLP layers rather than in model depth. We also observe that, under comparable compression ratios and benchmarks, Qwen2.5-VL-7B tends to exhibit steeper performance degradation than LLaVA1.5-7B, indicating relatively lower redundancy in the former model within our experimental setup.

For attention drop, which is more effective for reducing GPU memory usage and inference latency than for reducing parameter count [10, 11], we conduct additional experiments to analyze its efficiency characteristics. As shown in Figure 3, dropping less than 30% of attention layers in Qwen and LLaVA models largely preserves VLM performance. Although this operation removes less than 3% of the total model parameters, it provides noticeable inference speed improvements due to the reduction of KV-cache computation during autoregressive decoding. This efficiency gain is comparable to that achieved by substantially larger parameter reductions through width reduction. These observations indicate that different compression operators affect model efficiency and parameter count in different ways. Therefore, extracting an optimal submodel requires considering the primary deployment bottleneck, such as parameter-constrained scenarios (favoring MLP reduction) or latency-constrained scenarios (favoring attention drop).

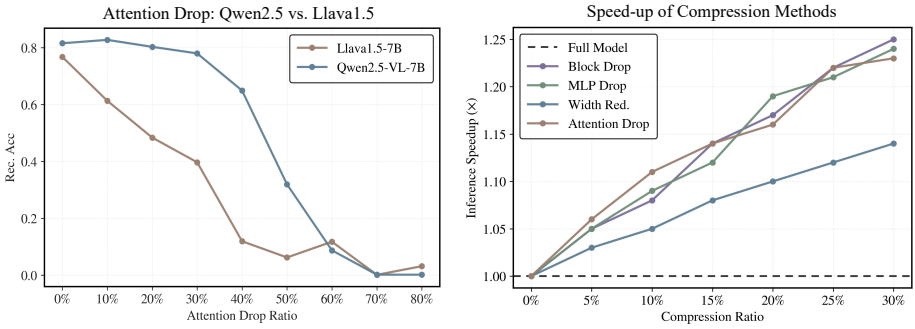


**Fig. 3:** The performance of attention drop in 7B Qwen2.5-VL and Llava1.5 models and the inference speedup compared to mlp-related compression methods.

# 3 Method

## 3.1 Problem Setup

Let $F(\theta)$ denote a pretrained VLM, where $\theta$ represents the full set of model parameters. We denote the original parameter size by $P_{\text{orig}} = |\theta|$. We consider a small set of downstream tasks $\mathcal{T} = \{T_1, \ldots, T_k\}$, where typically $k \in [1, 3]$. These tasks represent the target deployment scenario and correspond to a small number of task-specific benchmarks. We evaluate performance using the averaged validation accuracy across the selected tasks, denoted as $\text{Acc}(F(\theta))$.

Given a target parameter budget $B \leq P_{\text{orig}}$, our goal is to extract a structured submodel $F(\theta')$ with parameter size $P' = |\theta'| \leq B$, such that its performance on the target tasks is maximized.

Formally, we aim to solve:

$$\max_{\theta' \subseteq \theta} \text{Acc}\big(F(\theta')\big) \quad \text{s.t.} \quad |\theta'| \leq B. \tag{1}$$

We emphasize that the objective is explicitly task-conditioned: instead of preserving universal zero-shot capability, we seek the best-performing submodel under a specified parameter constraint for the selected target tasks.

## 3.2 Model-Level Ranking under the Original Model

Before performing budgeted reduction, we first compute an importance ranking for removable structural units based on the original pretrained model. This ranking stage provides a coarse estimate of the relative importance of different components and serves as a candidate ordering for the subsequent extraction process.

For block-level ranking, suppose the model contains $L$ Transformer blocks. We construct $L$ temporary variants, each removing exactly one block while keeping all other components unchanged. Each variant is evaluated on a lightweight

**Algorithm 1:** Granularity-Controlled Submodel Extraction

**Input:** Pretrained model $F(\theta)$; target budget $B$; reduction unit $g$ (size of one MLP); lightweight set $\mathcal{D}_{\text{lite}}$ ($K$ batches, $KB$ samples)
**Output:** Structured submodel parameters $\theta'$ with $|\theta'| \le B$
**Stage 1: Ranking on the original model.**;
For each removable structural unit $u$, compute its drop score on $\mathcal{D}_{\text{lite}}$:
$s(u) = \text{Acc}(F(\theta_{-u}); \mathcal{D}_{\text{lite}})$.
For each MLP layer $\ell$ and channel $i$, compute a channel importance score:
$s_{\ell,i} = \mathbb{E}_{x \sim \mathcal{D}_{\text{lite}}} \left[ |W_{d,i}^{(\ell)} h_i^{(\ell)}| \right]$.
Initialize $\theta^{(0)} \leftarrow \theta$ and $t \leftarrow 0$.
**Stage 2: Budget-aware extraction with fixed granularity.**;
**while** $|\theta^{(t)}| - g > B$ **do**
  `// Candidate set = next available units whose size is approximately` $g$
  $\mathcal{C}^{(t)} \leftarrow \emptyset$;
  Add the next not-yet-dropped unit from each ranking (block, attention, or MLP) into $\mathcal{C}^{(t)}$;
  **foreach** $a \in \mathcal{C}^{(t)}$ **do**
    Apply $a$ to obtain a temporary submodel $F(\theta_a^{(t)})$;
    Evaluate $A(a) = \text{Acc}\big(F(\theta_a^{(t)}); \mathcal{D}_{\text{lite}}\big)$;
  $a^{(t)} \leftarrow \arg\max_{a \in \mathcal{C}^{(t)}} A(a)$;
  $\theta^{(t+1)} \leftarrow \theta^{(t)}_{a^{(t)}}$;
  Advance the iterator only for the ranking type selected by $a^{(t)}$;
  $t \leftarrow t + 1$;
**Stage 3: Final budget matching by width reduction.**;
Let remaining reduction be $\Delta = |\theta^{(t)}| - B$.;
Shrink MLP widths to remove approximately $\Delta$ parameters.;
Return the final submodel parameters $\theta'$.

validation subset consisting of $K$ batches ($KB$ samples in total). Blocks whose removal yields higher accuracy are considered less important, and blocks are ranked accordingly. The same procedure is applied to attention modules and MLP sub-layers: each unit is individually removed once, and its post-removal accuracy determines its ranking.

For MLP channels, instead of repeatedly measuring validation accuracy, we estimate importance using output sensitivity. Given intermediate activation $h$ and output projection weight $W_d$, the contribution of channel $i$ is approximated by the magnitude of $W_{d,i}h_i$. Each MLP layer is ranked independently based on this criterion. After this stage, we obtain a ranking over blocks, attention modules, MLP modules, and channels. Because the evaluation subset is extremely lightweight, this ranking step can be computed efficiently and performed only once before the extraction process. For attention layers, we follow the selection method in [11].

### 3.3 Granularity-Controlled Submodel Extraction

Based on the precomputed ranking, we perform budget-aware submodel extraction in a step-wise manner. Empirically, removing an entire MLP sub-layer provides a stable and effective reduction unit in transformer-based VLMs. We therefore adopt the parameter size of one MLP as the default reduction granularity, which allows the extraction process to proceed with consistent parameter reductions at each step. This design effectively performs operator-level search over heterogeneous structural reductions under a fixed budget constraint.

For example, if the target requires removing approximately 50% of parameters and one MLP corresponds to about 5% of the total parameters, we perform roughly 10 reduction steps. At each step, we consider candidate structural reductions across different operator types (block removal, attention removal, or MLP removal) that correspond to approximately one reduction unit. Each candidate submodel is evaluated on the lightweight validation subset, and the operator that achieves the highest validation accuracy is selected. This step-wise selection process progressively constructs the final submodel while respecting the parameter budget. The procedure continues until the cumulative parameter reduction slightly exceeds the target budget. Finally, we apply MLP width reduction to finely adjust the remaining parameter count and match the target constraint precisely. The same extraction framework can also be applied under an inference speed budget by selecting operators that reduce computational latency.

This design tightly couples the ranking stage with the extraction process: the ranking restricts the candidate pool of removable components, while the granularity-controlled selection determines the combination of structural operators used in the final hybrid submodel.

## 4 Experiments

### 4.1 Experimental Setup

**Models and Submodel Settings.** We evaluate our method on four widely used instruction-following VLM backbones: Qwen2.5-VL-7B, Qwen2.5-VL-3B [26], LLaVA-1.5-7B, and LLaVA-1.5-13B [14]. Unless otherwise specified, we keep the vision encoder frozen during extraction and recovery training, and apply all structural modifications to the language-side Transformer.

**Benchmarks and Datasets.** We use three VQA-style benchmarks as target tasks: VQA v2 [7], MMBench [15], and TextVQA [20], and report accuracy as the evaluation metric. For each benchmark, we sample 2,000 validation/test examples for evaluation to keep experiments consistent and computationally tractable. For the lightweight ranking stage, we use a much smaller fixed subset of 64 samples, which is shared across all candidate evaluations.

**Recovery Training Details.** After submodel extraction, we optionally perform lightweight recovery training for 1 epoch on 5,000 training samples with learning rate $1 \times 10^{-6}$ using bf16 mixed precision. We sweep target compression ratios over $\{5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$ parameter reduction and

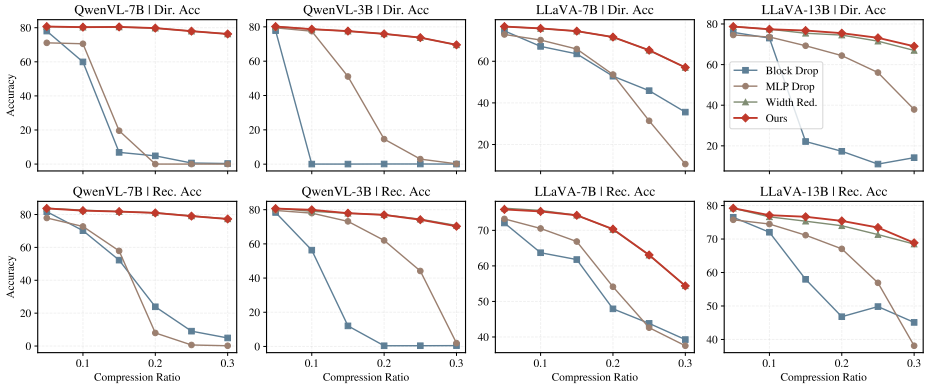


**Fig. 4: Submodel performance under parameter-budgeted compression. Attention drop is excluded since it removes very few parameters.** We compare different drop operators across four models. For each model, the top row reports the direct accuracy of the compressed submodel (Dir. Acc) and the bottom row reports the recovered accuracy after recovery training (Rec. Acc). We exclude attention drop since it is consistently inferior under parameter-budgeted settings. Overall, width reduction yields the strongest accuracy retention across compression ratios, and our method adapts to select the most suitable operator to match the budget while maintaining accuracy competitive with the best baseline.

$\{5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$ inference speedup. Throughout the experiments, we use the parameter size of one MLP layer as the default reduction unit to define the step granularity in our extraction procedure, and one attention layer as the default inference speedup unit. All experiments are conducted on NVIDIA H200 GPUs. Peak memory occupation is calculated on one GPU.

### 4.2 Submodel Extraction Performance

We first study submodel extraction when the deployment constraint is defined by the **parameter budget**. Figure 4 summarizes the accuracy trends across four backbones and multiple compression ratios. A consistent observation is that **MLP width reduction is the most accuracy-preserving operator under parameter-budgeted compression**. Compared to dropping entire blocks or MLP modules, width reduction removes parameters in a finer-grained manner while preserving the overall network structure, which helps maintain the representational capacity required for VQA tasks. This pattern holds consistently across all evaluated models and for both direct accuracy and recovered accuracy.

Although width reduction performs strongly under parameter constraints, it is not always the optimal choice at every operating point. Our framework is designed to **adaptively select the operator that best matches the current budget–accuracy trade-off**. As shown in Figure 4, the extracted submodels consistently achieve performance comparable to or better than the strongest single-operator baseline across most compression ratios. This behavior

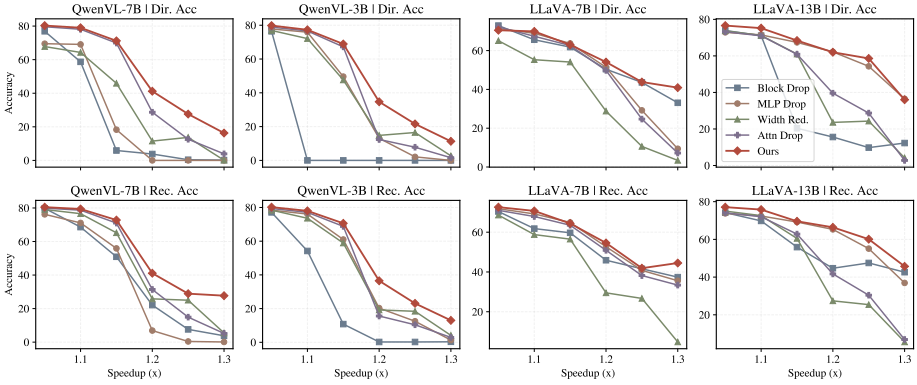


**Fig. 5:** Submodel extraction under an inference speed budget. Our method adaptively selects among heterogeneous operators and achieves consistently strong performance across different speed-up targets.

indicates that combining heterogeneous operators can exploit complementary effects: width reduction is chosen when it provides the best parameter–accuracy trade-off, while other operators are selected when they produce better local improvements.

We next analyze submodel extraction when the deployment constraint is defined by the **inference speed budget**. Figure 5 presents the accuracy trends under different speed-up targets. In contrast to the parameter-budget scenario, width reduction is no longer consistently the most favorable operator. When the required speed-up is relatively small, **attention drop** typically achieves the highest accuracy, since removing attention layers directly reduces the key-value cache computation during autoregressive decoding while affecting only a small portion of the model parameters. As the required speed-up increases, **block drop** becomes more effective. Removing entire Transformer blocks shortens the computational path of the network and therefore provides stronger latency reduction than attention removal alone. In comparison, width reduction offers relatively limited latency benefits despite reducing parameters, because shrinking intermediate tensor dimensions does not always translate to proportional improvements in GPU kernel efficiency.

Overall, these results highlight the importance of **adaptive operator selection under different deployment constraints**. While width reduction dominates under parameter budgets, latency-oriented deployment favors operators that directly reduce computation, such as attention removal and block removal. Our extraction framework can automatically adapt to these different optimization objectives and consistently produce high-performing submodels across both parameter- and latency-budget settings.

| Method | | Peak Alloc.(GB) ↓ | Peak Reserv.(GB)↓ | Time/sample(ms)↓ |
|---|---|---|---|---|
| Ratio | Type | | | |
| Ori. | No Compress. | 15.879 | 16.660 | 301.37 |
| 10% | Block Drop | 14.825 | 16.191 | 289.48 |
| | MLP Drop | 15.031 | 16.253 | 264.01 |
| | Width Red. | 14.957 | 16.168 | 302.10 |
| 20% | Block Drop | 13.520 | 15.115 | 236.90 |
| | MLP Drop | 13.439 | 14.472 | 249.92 |
| | Width Red. | 13.421 | 14.918 | 274.39 |
| 30% | Block Drop | 12.007 | 13.149 | 198.18 |
| | MLP Drop | 11.648 | 12.899 | 204.68 |
| | Width Red. | 11.922 | 13.934 | 246.11 |

**Table 1: GPU memory occupation and per-sample latency under total-parameter budget.** We report CUDA peak allocated/reserved memory and average time per sample during generation on VQAv2 (64 samples) for Qwen2.5-VL-7B.

### 4.3 Inference Efficiency and Memory Analysis

We further analyze the system-level efficiency of different compression operators under a total-parameter budget. Table 1 reports the peak GPU memory usage and the average inference latency per sample on VQAv2.

Increasing the compression ratio consistently reduces GPU memory usage across all operators. For example, the peak allocated memory decreases from 15.879 GB in the original model to around 12 GB at a 30% compression ratio. Among the evaluated operators, MLP Drop tends to provide the largest memory reduction, since MLP layers constitute a substantial portion of both the model parameters and activation memory.

The impact on inference latency, however, differs significantly across operators. Removing entire blocks or MLP modules shortens the computational path and leads to faster inference, reducing the latency from 301.37 ms to around 200 ms at a 30% compression ratio. In contrast, width reduction does not achieve comparable speedups despite reducing parameters, because modifying intermediate tensor shapes can result in less efficient GPU kernel execution.

These results suggest that parameter reduction alone does not necessarily translate into improved inference efficiency. Instead, different structural operators optimize different system bottlenecks, such as parameter count, memory usage, or runtime latency. This observation highlights the importance of selecting compression operators according to the target deployment constraint.

### 4.4 Effect of Recovery Training

An important property of our framework is that the extracted submodel is optimized for a specific downstream task rather than preserving general-purpose capability. This design enables lightweight recovery training on the target dataset to restore task accuracy after structural compression.

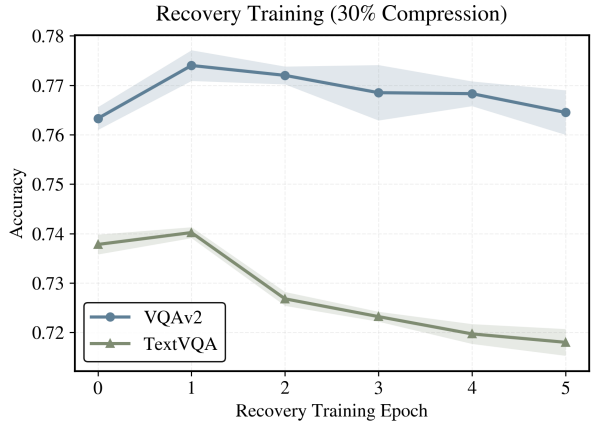


**Fig. 6: Effect of recovery training at 30% compression.** We perform recovery training on the **target task** (VQAv2) after extracting the compressed submodel. The VQAv2 accuracy increases from **0.7633** (compressed, epoch 0) to **0.7740** after one epoch, while TextVQA consistently decreases (e.g., **0.7378** → **0.7180** within 5 epochs), indicating that recovery training prioritizes task-specific performance rather than generalization across tasks. Shaded areas denote the standard deviation.

We analyze this effect under a compression ratio of **30%**, where recovery training is performed on the VQA v2 dataset. Before recovery training, the compressed model achieves an accuracy of **0.7633** on VQAv2. After one epoch of recovery training, the accuracy improves to **0.7740**, and remains relatively stable in subsequent epochs, as shown in Figure 6. These results indicate that a small amount of task-specific fine-tuning can effectively recover part of the accuracy lost during compression.

At the same time, recovery training does not improve performance on other tasks. For example, the accuracy on TextVQA decreases from **0.7378** at epoch 0 to **0.7180** after five epochs of training. This behavior reflects the task-prioritized nature of our approach: since recovery training focuses exclusively on the target dataset, the limited model capacity is gradually adapted to the target task rather than maintaining cross-task generalization.

Although the recovered VQA v2 accuracy remains lower than that of the original uncompressed model, these results demonstrate that recovery training provides an effective mechanism for restoring target-task performance after submodel extraction. This property is particularly useful in practical deployment scenarios where the model is optimized for a specific application and task-specific accuracy is more critical than preserving broad multi-task capability.

### 4.5 Case Study

To better understand the behavior of the proposed submodel extraction framework, we visualize the operator selection process during the search under dif-


**Fig. 7:** Greedy search trajectory under a parameter-priority constraint. Each step shows the operator selected by the search algorithm and the corresponding model accuracy. The search first removes MLP layers and then repeatedly applies width reduction, which provides the most favorable parameter–accuracy trade-off.

ferent deployment constraints. Figure 7 and Figure 8 show two representative cases corresponding to different optimization objectives: parameter-priority and speed-up-priority.

Figure 7 illustrates the search trajectory when parameter reduction is the primary objective. The extraction process first applies MLP Drop and then repeatedly selects width reduction in subsequent steps. This behavior reflects the strong parameter–accuracy trade-off provided by width reduction, which can effectively reduce parameter counts while preserving most of the model structure. As a result, width reduction becomes the dominant operator during the search process, gradually shrinking the model until the desired parameter budget is reached. Importantly, the accuracy degradation remains smooth throughout the process, indicating that the algorithm consistently selects operators that minimize performance loss under the parameter constraint. In contrast, the search trajectory under the speed-up-priority setting shows a different pattern, as illustrated in Figure 8. In this case, the algorithm first removes several attention layers, which directly reduces the computational cost of the model and leads to immediate inference speed improvements. After the attention layers are removed, the algorithm further applies block removal to shorten the computational path of the network. Finally, width reduction is used in the later stage to fine-tune the remaining budget and reach the desired speed-up target.

These two case studies highlight the adaptive nature of the proposed extraction framework. Instead of relying on a fixed compression strategy, the algorithm dynamically selects the most suitable operator according to the current optimization objective. Under parameter constraints it prioritizes width reduction, while under latency constraints it first removes computationally expensive modules such as attention layers. This behavior demonstrates that the proposed method can automatically discover compression strategies that align with different deployment requirements while maintaining strong task accuracy.

More broadly, the consistent search patterns observed in these cases suggest that the framework captures meaningful structural redundancy in the model. The operator selection mechanism therefore provides a practical way to adapt pretrained vision-language models to diverse deployment scenarios with different resource constraints.

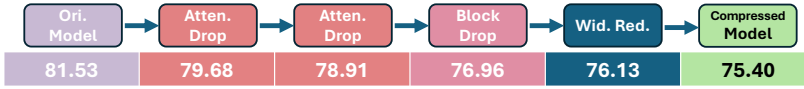


**Fig. 8:** Greedy search trajectory under a speed-up-priority constraint. The algorithm first removes attention layers to quickly reduce inference latency, then removes a block module, and finally applies width reduction to satisfy the remaining budget. This demonstrates that the proposed search strategy can adaptively select different operators depending on the optimization objective.

## 5 Conclusion

In this work, we revisit model compression from a deployment-oriented perspective and formulate budget-aware, task-conditioned submodel extraction for VLMs. Instead of preserving universal zero-shot capability, we emphasize that practical deployment scenarios typically involve a small number of target tasks operating under explicit computational constraints. From this perspective, extracting task-aligned submodels becomes essential for bridging the gap between large-scale pretraining and real-world deployment. Through systematic analysis of representative structural compression strategies in transformer-based vision-language models, including attention drop, MLP drop, block drop, and MLP width reduction, we observe clear differences in their behavior. Attention drop mainly improves inference efficiency with minimal parameter reduction, while block drop provides large parameter savings but often leads to significant performance degradation due to its coarse granularity. MLP drop offers moderate compression but becomes unstable at higher reduction ratios, whereas width reduction consistently preserves task performance better under comparable budgets, suggesting that redundancy in many VLMs is more concentrated in MLP width than in model depth.

Building on these observations, we propose a unified framework that automatically extracts task-specific submodels under a given parameter budget. By combining model-level importance ranking with a granularity-controlled extraction process, the proposed method adaptively selects among heterogeneous structural reductions and dynamically determines the most effective compression strategy at each reduction step. This design enables the framework to flexibly construct high-performing submodels across a wide range of compression ratios without modifying the original pretraining process. Extensive experiments on multiple Qwen-based and LLaVA-based vision-language models demonstrate that our method consistently achieves strong task accuracy under significant parameter reductions. More broadly, our work provides a practical and scalable pathway for transforming large pretrained foundation models into task-aligned and budget-efficient models suitable for real-world deployment.

# References


1. Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C.L., Parikh, D.: Vqa: Visual question answering. In: Proceedings of the IEEE international conference on computer vision. pp. 2425–2433 (2015)
2. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
3. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791 (2019)
4. Cai, R., Muralidharan, S., Heinrich, G., Yin, H., Wang, Z., Kautz, J., Molchanov, P.: Flextron: Many-in-one flexible large language model. arXiv preprint arXiv:2406.10260 (2024)
5. Cai, R., Muralidharan, S., Yin, H., Wang, Z., Kautz, J., Molchanov, P.: Llamaflex: Many-in-one llms via generalized pruning and weight sharing. In: The Thirteenth International Conference on Learning Representations (2025)
6. Chen, B., Xu, Z., Kirmani, S., Ichter, B., Sadigh, D., Guibas, L., Xia, F.: Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14455–14465 (2024)
7. Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., Parikh, D.: 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. pp. 6904–6913 (2017)
8. He, S., Deng, C., Li, A., Yan, S.: Understanding and harnessing sparsity in unified multimodal models. arXiv preprint arXiv:2512.02351 (2025)
9. He, S., Li, A., Chen, T.: Rethinking pruning for vision-language models: Strategies for effective sparsity and performance restoration. arXiv preprint arXiv:2404.02424 (2024)
10. He, S., Sun, G., Shen, Z., Li, A.: Uncovering the redundancy in transformers via a unified study of layer dropping
11. He, S., Sun, G., Shen, Z., Li, A.: What matters in transformers? not all attention is needed. arXiv preprint arXiv:2406.15786 (2024)
12. Hossain, M.Z., Sohel, F., Shiratuddin, M.F., Laga, H.: A comprehensive survey of deep learning for image captioning. ACM Computing Surveys (CsUR) **51**(6), 1–36 (2019)
13. Lei, Y., He, S., Li, A., Yates, A.: Making large language models efficient dense retrievers. arXiv preprint arXiv:2512.20612 (2025)
14. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. Advances in neural information processing systems **36**, 34892–34916 (2023)
15. Liu, Y., Duan, H., Zhang, Y., Li, B., Zhang, S., Zhao, W., Yuan, Y., Wang, J., He, C., Liu, Z., et al.: Mmbench: Is your multi-modal model an all-around player? In: European conference on computer vision. pp. 216–233. Springer (2024)
16. Ma, X., Fang, G., Wang, X.: Llm-pruner: On the structural pruning of large language models. Advances in neural information processing systems **36**, 21702–21720 (2023)
17. Mu, S., Lin, S.: A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications. arXiv preprint arXiv:2503.07137 (2025)
18. Mugnaini, L.G., Yamamoto, B.L., de Alcantara, L.L., Zacarias, V., Bollis, E., Pellicer, L., Costa, A.H.R., Jordao, A.: Efficient llms with amp: Attention heads and

mlp pruning. In: 2025 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2025)
19. Sakuma, Y., Ishii, M., Narihira, T.: Detofa: efficient training of once-for-all networks for object detection using path filter. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1333–1342 (2023)
20. Singh, A., Natarajan, V., Shah, M., Jiang, Y., Chen, X., Batra, D., Parikh, D., Rohrbach, M.: Towards vqa models that can read. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8317–8326 (2019)
21. Tan, A.Z., Yu, H., Cui, L., Yang, Q.: Towards personalized federated learning. IEEE transactions on neural networks and learning systems **34**(12), 9587–9603 (2022)
22. Tiwari, R., Bamba, U., Chavan, A., Gupta, D.K.: Chipnet: Budget-aware pruning with heaviside continuous approximations. arXiv preprint arXiv:2102.07156 (2021)
23. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
24. Wang, T., Zhou, W., Zeng, Y., Zhang, X.: Efficientvlm: Fast and accurate vision-language models via knowledge distillation and modal-adaptive pruning. In: Findings of the Association for Computational Linguistics: ACL 2023. pp. 13899–13913 (2023)
25. Wang, Z., He, Y., Shen, Z., Li, Y., Sun, G., Lee, M., Li, A.: Prada: Black-box llm adaptation with private data on resource-constrained devices. arXiv preprint arXiv:2503.14932 (2025)
26. Xu, Y., Wang, P., Zhang, H., Wang, P., Bai, S., Wang, S., Lin, J., Xie, T., Zhu, Y., Yang, Z., Ding, W., Zhang, X., Wan, J., Tang, J., Xu, H., Ye, J., Chen, K., Liu, X., Wang, J., Ge, W., Dang, K., Cheng, Z., Yang, M., Song, S., Li, Z., Zhong, H., Fu, Z.: Qwen2.5-vl technical report (2025), `https://arxiv.org/abs/2502.13923`
27. You, Q., Jin, H., Wang, Z., Fang, C., Luo, J.: Image captioning with semantic attention. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4651–4659 (2016)
28. Zhao, W., Zou, L., Wang, Z., Yao, X., Yu, B.: Hape: Hardware-aware llm pruning for efficient on-device inference optimization. ACM Transactions on Design Automation of Electronic Systems **30**(4), 1–18 (2025)
29. Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A.M., Le, Q.V., Laudon, J., et al.: Mixture-of-experts with expert choice routing. Advances in Neural Information Processing Systems **35**, 7103–7114 (2022)
30. Zhuang, W., Chen, C., Li, Z., Sajadmanesh, S., Li, J., Huang, J., Sehwag, V., Sharma, V., Shinozaki, H., Garcia, F.C., et al.: Argus: A compact and versatile foundation model for vision. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 4418–4429 (2025)