VisPCO: Visual Token Pruning Configuration Optimization via Budget-Aware Pareto-Frontier Learning for Vision-Language Models

Huawei Ji[1], Yuanhao Sun[1], Yuan Jin[1], Cheng Deng[2], Jiaxin Ding[1]* Luoyi Fu[1], Xinbing Wang[1]

[1]Shanghai Jiao Tong University, Shanghai, China,
[2]University of Edinburgh, Edinburgh, UK
{sjtu3365981, h_iden, lemon0703, jiaxinding, yiluofu, xwang8}@sjtu.edu.cn

Abstract

Visual token pruning methods effectively mitigate the quadratic computational growth caused by processing high-resolution images or long video frames in vision-language models (VLMs). However, existing approaches rely on predefined pruning configurations without determining whether they achieve computation-performance optimality. In this work, we introduce **VisPCO**, a novel framework that formulates visual token pruning as a Pareto configuration optimization problem to automatically identify optimal configurations. Our approach employs continuous relaxation and straight-through estimators to enable gradient-based search, solved via the Augmented Lagrangian method. Extensive experiments across 8 visual benchmarks demonstrate that **VisPCO** effectively approximates the empirical Pareto frontier obtained through grid search and generalizes well across various pruning methods and VLM architectures. Furthermore, through learnable kernel functions, we investigate layer-wise pruning patterns and reveal that multi-step progressive pruning captures VLMs' hierarchical compression structure, achieving superior computation-performance trade-offs compared to single-layer approaches.

1 Introduction

Large-scale vision-language models (LVLMs) process both visual and textual features as input, enabling them to learn unified multimodal representations and perform cross-modal reasoning. Recent studies have shown that higher-resolution image inputs can effectively improve the model's understanding and generation performance (Guo et al., 2024; An et al., 2025; Chen et al., 2024b). Meanwhile, tasks such as video understanding require models to process numerous frames to capture temporal continuity and dynamic semantics (Lin et al., 2024; Xu et al., 2025). Both scenarios significantly increase the number of visual tokens, leading to quadratic growth in computational costs.

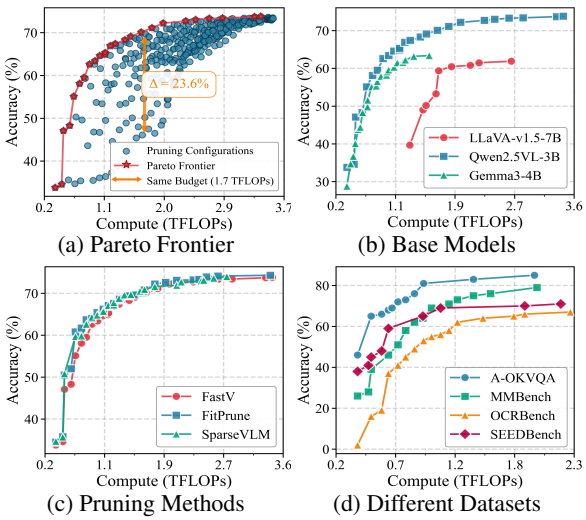

Figure 1: Pareto frontiers across different configurations: (a) The Pareto frontier connects optimal pruning configurations. (b) Pareto frontiers across different VLMs. (c) Pareto frontiers across different pruning methods. (d) Pareto frontiers across different datasets.

To solve this problem, various visual pruning algorithms for VLMs have emerged. These methods mainly focus on designing different importance scoring mechanisms to prune redundant visual tokens at single or multiple layers. For instance, FastV (Chen et al., 2024a), Dynamic-LLaVA (Huang et al., 2024), VTW (Lin et al., 2025), and TOPV (Yang et al., 2025) prune visual tokens at a specific LLM layer using predefined pruning ratios. In contrast, ATP-LLaVA (Ye et al., 2025b), HiMAP (Zhou et al., 2024), and SparseVLM (Zhang et al., 2024) apply dynamic pruning ratios across multiple selected layers. All these works aim to reduce computational costs (e.g., FLOPs) while maintaining model performance.

However, two critical questions remain unexplored in existing works. First, it is unclear whether current pruning configurations (i.e., pruning positions and ratios) achieve the optimal computation-

*Corresponding author

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9281–9301
July 2-7, 2026

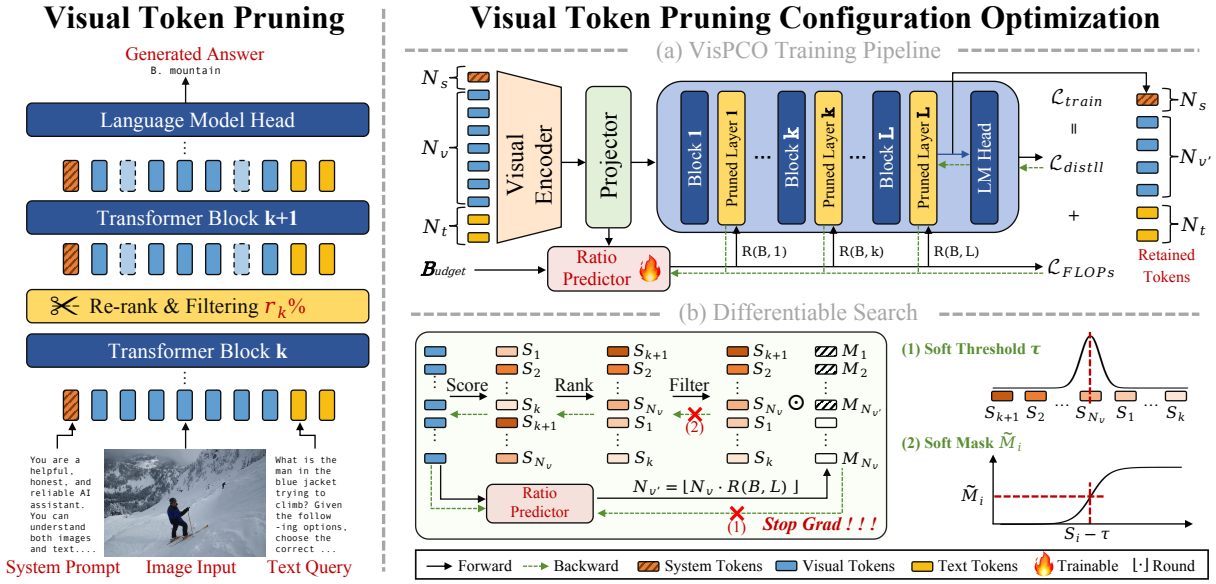

Figure 2: Illustration of our **VisPCO** framework. **(Left)** Overview of the visual token pruning process. After each transformer block, visual tokens are ranked by their importance scores and low-scoring tokens are filtered out. **(Right)** Upper panel: The overall architecture of **VisPCO**, where the trainable Ratio Predictor, a lightweight surrogate network, determines the pruning ratio to guide token compression at each layer. Lower panel: The gradient disconnection problem encountered during end-to-end training of the Ratio Predictor and our proposed solution.

performance trade-off. Second, how to adjust these configurations efficiently to reach the optimal remains an open problem. Addressing these two questions is crucial for the efficient deployment of VLMs in real-world applications, e.g., edge device deployment and mobile vision systems.

To this end, we introduce the concept of Pareto optimality to characterize the optimal computation-performance trade-off (Arrow and Debreu, 2024). Pareto optimality is a classic concept in multi-objective optimization that describes a state where no further improvement can be made among multiple conflicting objectives. In this work, we define a point as Pareto-optimal if we cannot simultaneously reduce computational cost and improve performance. The curve connecting these points is called the Pareto frontier.

In Figure 1, we visualize different pruning configurations in the computation-performance space and their corresponding Pareto frontier. Each point represents the experimental result of one configuration. Figure 1(a) reveals that under the same computational budget, different pruning configurations exhibit significant performance variations. For instance, at a computational budget of 1.7 TFLOPs, the performance gap between different configurations can reach up to 23.6%. Moreover, Pareto-optimal frontiers are not fixed. Instead, they vary with VLM architectures, pruning methods and image complexity as illustrated in Figure 1(b-d). In practice, to determine the best pruning configuration under a fixed computational budget, we typically need to perform grid sampling over the configuration search space. This involves conducting numerous experiments to measure performance and cost across different configurations, and then selecting the optimal one for deployment. However, this process is prohibitively time-consuming and resource-intensive.

In this paper, we propose a computation budget-aware method for **Vis**ual token **P**runing **C**onfiguration **O**ptimization, termed **VisPCO**. This approach employs a learnable surrogate model to automatically predict pruning configurations on the Pareto frontier given a computational budget, thereby achieving optimal model performance. As shown in Figure 2, unlike traditional grid search methods, **VisPCO** employs efficient gradient descent for search, significantly reducing search costs. To address the discrete and non-differentiable nature of visual token pruning, we introduce continuous relaxation techniques and straight-through estimators for end-to-end optimization. For the optimization objective, we formulate it as a Pareto optimization problem with non-convex inequality constraints and solve it using the Augmented Lagrangian method (Nocedal and Wright, 2006). Furthermore, we investigate the layer-wise pruning patterns in VLMs. Specifically, we explore how the optimal pruning ratio varies across different layers

when progressively compressing visual tokens. To model this variation, we use learnable kernel functions to parameterize the pruning ratio distribution across layers. By evaluating how well different kernel functions approximate the Pareto frontier, we identify the intrinsic pruning patterns that lead to optimal computation-performance trade-offs. Our contributions can be summarized as follows:

- We propose **VisPCO**, a differentiable framework that automatically finds Pareto-optimal configurations via gradient-based methods, eliminating the prohibitive cost of exhaustive grid search.
- Experiments on 8 benchmarks demonstrate that **VisPCO** effectively approximates the empirical Pareto frontier and generalizes across various pruning methods and VLM architectures.
- We reveal non-uniform visual token redundancy across layers via learnable pruning kernels, showing multi-step pruning is most effective under tight budgets. Our code is available at `https://github.com/JHW5981/VisPCO`.

2 Related Work

2.1 Visual Token Pruning

Visual token pruning accelerates VLMs by reducing computational costs from processing hundreds of visual tokens. Single-layer approaches perform one-shot reduction at specific layers: FastV (Chen et al., 2024a) prunes tokens after layer 2 using attention scores, Dynamic-LLaVA (Huang et al., 2024) dynamically adjusts token retention ratios based on input characteristics, VTW (Lin et al., 2025) withdraws all tokens after sufficient absorption, and TopV (Yang et al., 2025) optimizes configurations at inference time. Multi-layer progressive methods distribute reduction across multiple layers: ATP-LLaVA (Ye et al., 2025b) adaptively prunes tokens at different depths with layer-specific ratios, SparseVLM (Zhang et al., 2024) progressively reduces tokens across layers with recycling mechanisms, and PyramidDrop (Xing et al., 2024) implements stage-wise pyramid reduction that preserves more tokens in shallow layers. These strategies achieve 40-90% computational savings while maintaining competitive performance. However, the pruning configurations in these methods are either predefined or heuristically determined, and it remains unclear whether they achieve the optimal computation-performance trade-off.

2.2 Pruning Configuration Optimization

While most pruning methods rely on predefined configurations, recent works explore adaptive strategies to optimize pruning ratios across layers. FitPrune (Ye et al., 2025a) employs binary search over attention statistics to minimize distribution divergence and generate layer-wise pruning recipes. G-Search (Zhao et al., 2025) combine greedy search with Bayesian-optimized sigmoid functions to approximate optimal retention ratios. ATP-LLaVA (Ye et al., 2025b) introduces learnable modules for training-based optimization of layer-specific sparsity. SparseVLM (Zhang et al., 2024) employs rank-based adaptive determination of per-layer sparsification ratios. More recent methods explore input-adaptive configurations: AIM (Zhong et al., 2025) develops scheduler-controlled pruning with adjustable parameters, and MADTP (Cao et al., 2024) utilizes learnable thresholds for instance-wise adaptive pruning. Despite these advances, existing approaches focus primarily on performance preservation rather than budget-aware optimization. They lack mechanisms to systematically identify near-optimal configurations under varying computational constraints.

3 VisPCO

3.1 Pareto Optimization

We formulate the visual pruning configuration optimization problem as finding the optimal layer-wise pruning ratios $\mathbf{r} = [r_1, r_2, \ldots, r_L] \in [0, 1]^L$, where L denotes the number of layers and r_i represents the token retention ratio at layer i (relative to the original number of visual tokens). Our objective is to identify a configuration $\bar{\mathbf{r}}$ on the Pareto frontier that achieves optimal model performance under a given computational budget.

To quantify performance degradation, we define the pruned VLM output logits as $\hat{l}$ and the original output logits as l, and measure their discrepancy using KL divergence:

$$\mathcal{L}_{\text{distill}}(\mathbf{r}) = D_{KL}\left(\text{softmax}(\hat{l}) \,\middle\|\, \text{softmax}(l)\right). \tag{1}$$

A smaller value of $\mathcal{L}_{\text{distill}}$ indicates less impact of pruning on model performance. Meanwhile, we define the computational cost function as:

$$F(\mathbf{r}) = \sum_{i=1}^{L} \left[24(N_t + r_i N_v)D^2 + 4(N_t + r_i N_v)^2 D\right] \tag{2}$$

where N_t is the number of text tokens, N_v is the number of visual tokens, and D is the hidden dimension. Detailed derivation of the FLOPs computation is provided in Appendix A. Therefore, the Pareto optimization problem for the computation-performance trade-off can be formulated as the following constrained optimization problem:

$$\begin{aligned} \min \quad & \mathcal{L}_{\text{distill}}(\mathbf{r}) \\ \text{s.t.} \quad & F(\mathbf{r}) \leq B, \end{aligned} \tag{3}$$

where B is the given computational budget. Considering that the objective function $\mathcal{L}_{\text{distill}}$ is typically non-convex, we employ the Augmented Lagrangian Method for numerical iterative solving.

Definition 1 (Augmented Lagrangian Method)**.** *Consider the equality-constrained optimization problem:*

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \textit{s.t.} \quad & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, l \end{aligned}$$

where $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is the objective function to be minimized, and $h_j(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ are the equality constraint functions. The augmented Lagrangian function is defined as:

$$\phi(\mathbf{x}, \mathbf{v}, \lambda) = f(\mathbf{x}) - \sum_{j=1}^{l} v_j h_j(\mathbf{x}) + \frac{\lambda}{2} \sum_{j=1}^{l} h_j^2(\mathbf{x}), \tag{4}$$

where $\mathbf{v} = [v_1, \ldots, v_l]$ is the Lagrange multiplier vector and $\lambda > 0$ is the penalty parameter.

Theorem 1 (Adapted from Bertsekas, 2014)**.** *Let $\bar{\mathbf{x}}$ and $\bar{\mathbf{v}}$ satisfy the second-order conditions for a local optimal solution of the problem. Then there exists $\lambda' \geq 0$ such that for all $\lambda > \lambda'$, $\bar{\mathbf{x}}$ is a strict local minimizer of $\phi(\mathbf{x}, \bar{\mathbf{v}}, \lambda)$.*

The proof of Theorem 1 is provided in Appendix B.1. Based on this theorem, we can develop an iterative algorithm with bounded λ, avoiding the ill-conditioning of quadratic penalty methods (Nocedal and Wright, 2006) and the convergence difficulties of standard Lagrangian methods (Bertsekas, 2014). Specifically, at iteration k, let $\mathbf{x}^{(k)}$ be the minimizer of (4) with respect to $\mathbf{x}$. The multiplier update rule is:

$$v_j^{(k+1)} = v_j^{(k)} - \lambda h_j(\mathbf{x}^{(k)}), \quad j = 1, \ldots, l \tag{5}$$

Through this iterative update, we have $\mathbf{v}^{(k)} \to \bar{\mathbf{v}}$ and $\mathbf{x}^{(k)} \to \bar{\mathbf{x}}$, with convergence rate typically measured by $\|h(\mathbf{x}^{(k)})\| / \|h(\mathbf{x}^{(k-1)})\|$.

Returning to our optimization problem (3), we introduce an auxiliary variable y to convert the inequality constraint into an equality constraint:

$$\begin{aligned} \min \quad & \mathcal{L}_{\text{distill}}(\mathbf{r}) \\ \text{s.t.} \quad & B - F(\mathbf{r}) - y^2 = 0. \end{aligned} \tag{6}$$

The corresponding augmented Lagrangian function is defined as:

$$\begin{aligned} \tilde{\phi}(\mathbf{r}, y, w, \lambda) = \mathcal{L}_{\text{distill}}(\mathbf{r}) - w \left(B - F(\mathbf{r}) - y^2\right) \\ + \frac{\lambda}{2} \left(B - F(\mathbf{r}) - y^2\right)^2. \end{aligned} \tag{7}$$

By completing the square with respect to y, we can eliminate the dependence on y and obtain the simplified augmented Lagrangian (see Appendix B.2):

$$\phi(\mathbf{r}, w, \lambda) = \mathcal{L}_{\text{distill}}(\mathbf{r}) + \frac{1}{2\lambda} \left(z^2 - w^2\right), \tag{8}$$

where $z = \max\big(0,\ w - \lambda\big(B - F(\mathbf{r})\big)\big)$. The problem is thus transformed into minimizing the unconstrained objective $\phi(\mathbf{r}, w, \lambda)$. Using the iterative algorithm in Algorithm 1, we can obtain the optimal pruning configuration $\bar{\mathbf{r}}$.

3.2 Differentiable Configuration Search

Despite having the Pareto optimization objective Eq. (8) and iterative Algorithm 1, computing $\nabla_{\mathbf{r}} \mathcal{L}_{\text{distill}}(\mathbf{r})$ faces two critical non-differentiability challenges, as illustrated in the bottom right of Figure 2. First, the discretization of retained token counts introduces non-differentiability. For layer i with pruning ratio r_i and N_v visual tokens, the retained token count $k_i = \lfloor r_i \cdot N_v \rfloor$ involves a floor operation that causes vanishing gradients, preventing backpropagation-based updates of r_i. Second, selecting the top-k_i tokens based on importance scores involves discrete operations that block gradient flow. We propose the following two methods to address these challenges respectively:

Continuous Relaxation. To address the first challenge, we adopt a continuous relaxation strategy. We retain the floating-point form $\tilde{k}_i = r_i \cdot N_v$ and design a soft interpolation method using a Gaussian kernel to estimate the selection threshold. Specifically, we first sort the importance scores of all visual tokens in descending order to obtain $\{s_{i1}, s_{i2}, \ldots, s_{iN_v}\}$, where s_{ij} denotes the score of the j-th token in layer i after sorting. Traditional hard thresholding directly uses the score at position

$\lfloor \tilde{k}_i \rfloor$ as the threshold, which leads to vanishing gradients. Instead, we employ Gaussian kernel-based soft interpolation to maintain differentiability:

$$w_{ij} = \exp\left(-\frac{(j-\tilde{k}_i)^2}{2\sigma^2}\right), \quad \tau_i = \frac{\sum_{j=1}^{N_v} w_{ij} s_{ij}}{\sum_{j=1}^{N_v} w_{ij}}, \tag{9}$$

where w_{ij} is the Gaussian weight for the j-th token in layer i. The parameter σ controls the kernel width, balancing between approximation accuracy and gradient stability. As $\sigma \to 0$, the soft threshold τ_i converges to the hard threshold $s_{i\lfloor \tilde{k}_i \rfloor}$. The gradient of τ_i with respect to r_i can be expressed as:

$$\frac{\partial \tau_i}{\partial r_i} = N_v \sum_{j=1}^{N_v} \frac{w_{ij}(j-\tilde{k}_i)}{\sigma^2 \sum_{l=1}^{N_v} w_{il}} (s_{ij} - \tau_i), \tag{10}$$

which remains well-defined for all $r_i \in (0, 1)$, enabling smooth gradient flow via backpropagation.

Straight-Through Estimator. To address the second challenge, we employ the Straight-Through Estimator (STE) strategy. Inspired by Gumbel-Softmax (Jang et al., 2016), we use discrete hard decisions in the forward pass and continuous soft approximations in the backward pass. Given the threshold τ_i, for each visual token j in layer i, we compute both a hard selection mask and a soft mask:

$$m_{ij} = \mathbb{I}[s_{ij} \geq \tau_i], \quad \tilde{m}_{ij} = \text{sigm}\left(\frac{s_{ij} - \tau_i}{T}\right), \tag{11}$$

where $\mathbb{I}[\cdot]$ is the indicator function, $\text{sigm}(\cdot)$ is the sigmoid function, and T is the temperature parameter. The soft mask $\tilde{m}_{ij}$ provides a smooth approximation: as $T \to 0^+$, $\tilde{m}_{ij} \to m_{ij}$. The STE combines both masks through:

$$\hat{m}_{ij} = \tilde{m}_{ij} + \text{sg}(m_{ij} - \tilde{m}_{ij}), \tag{12}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operation. This formulation ensures $\hat{m}_{ij} = m_{ij}$ in the forward pass, while the backward gradient satisfies:

$$\frac{\partial \mathcal{L}}{\partial \tau_i} = \sum_j \frac{\partial \mathcal{L}}{\partial \hat{m}_{ij}} \cdot \frac{\partial \tilde{m}_{ij}}{\partial \tau_i} = -\sum_j \frac{\partial \mathcal{L}}{\partial \hat{m}_{ij}} \cdot \frac{\tilde{m}_{ij}(1-\tilde{m}_{ij})}{T}, \tag{13}$$

where $\frac{\partial \mathcal{L}}{\partial \hat{m}_{ij}}$ denotes the upstream gradient propagated from subsequent layers and is well-defined since $\hat{m}_{ij}$ is treated as a differentiable proxy of m_{ij} during backpropagation. These provide biased but low-variance gradient estimators that enable end-to-end optimization of both token scores and adaptive thresholds.

Algorithm 1 Training pipeline of **VisPCO**

Input: Initial configuration $\mathbf{r}^{(0)}$, initial Lagrange multiplier $w^{(1)}$, penalty parameter λ, convergence threshold $\epsilon > 0$, update coefficients $\alpha > 1$, $\beta \in (0, 1)$
Output: Locally optimal configuration $\bar{\mathbf{r}}$ and optimal multiplier $\bar{w}$
Set $k = 1$
while True **do**
  Starting from $\mathbf{r}^{(k-1)}$, solve optimization problem $\min \phi(\mathbf{r}, w, \lambda)$
  / Train using gradient descent */*
  Obtain solution $\mathbf{r}^{(k)}$
  if $\|B - F(\mathbf{r}^{(k)})\| < \epsilon$ **then**
    / Constraint satisfied, training converged */*
    break
  end if
  if $\frac{\|B-F(\mathbf{r}^{(k)})\|}{\|B-F(\mathbf{r}^{(k-1)})\|} \geq \beta$ **then**
    / Update penalty parameter */*
    $\lambda \leftarrow \alpha\lambda$
  end if
  / Update Lagrange multiplier */*
  $w^{(k+1)} \leftarrow w^{(k)} - \lambda(B - F(\mathbf{r}^{(k)}))$
  $k \leftarrow k + 1$
end while
return $\bar{\mathbf{r}} = \mathbf{r}^{(k)}$, $\bar{w} = w^{(k)}$

3.3 Learnable Kernel Functions

With **VisPCO**, we can automatically search for configurations on the Pareto frontier in a differentiable manner. To further investigate how different pruning patterns affect the Pareto frontier, we impose structural constraints on the pruning configuration search space. In practice, visual token pruning exhibits a monotonically non-increasing pattern across layers: deeper layers tend to retain fewer tokens. We leverage this prior by introducing **learnable kernel functions** to parameterize the layer-wise pruning ratios. This design offers two key benefits: (1) it provides interpretability for visual token pruning behavior, and (2) it reduces the parameter search space, ensuring both optimization stability and computational efficiency.

We consider two pruning scenarios: single-layer pruning and multi-layer pruning. For the first case, we employ a parameterized p-sigmoid kernel to model a sharp transition at layer k:

$$\mathcal{K}_s(i; k, r, \gamma) = 1 + (r-1) \cdot \text{sigm}(\gamma(i-k)), \tag{14}$$

where i is the layer index, k is the pruning position, $r \in (0, 1]$ controls the final retention ratio, $\gamma > 0$ is the sharpness parameter, and $\text{sigm}(\cdot)$ is the sigmoid function. With sufficiently large γ, the transition approaches a step function retaining all tokens before layer k and applying retention ratio r thereafter, effectively approximating single-layer pruning while remaining differentiable.

For the second case, we explore several kernel functions to capture diverse pruning patterns. First, inspired by the Ebbinghaus forgetting curve from cognitive psychology (Fuchs, 2000), we design an exponential decay kernel to investigate whether VLMs exhibit similar attention decay patterns for visual tokens across layers:

$$\mathcal{K}_{\text{e}}(i; k, r) = r \cdot e^{-k \cdot i}. \tag{15}$$

Second, we consider a linear decay kernel that models uniform, gradual token reduction:

$$\mathcal{K}_{\text{l}}(i; k, r) = -k \cdot i + r. \tag{16}$$

Third, motivated by findings in (Zhao et al., 2025) showing that attention score rankings remain similar across layers and follow a sigmoid-like curve, we adopt a gentle p-sigmoid kernel. Following Eq. (14), we use a smaller γ to capture smooth, progressive pruning transitions. Finally, inspired by hierarchical representation learning where deep networks perform feature extraction at different levels, we introduce a multi-step sigmoid kernel to model the hypothesis that VLMs compress information at multiple critical layers:

$$\mathcal{K}_{\text{ms}}(i; k, r, M) = 1 - \sum_{j=1}^{M} \frac{1-r}{M} \cdot \sigma\left(k\left(i - \frac{(2j-1)L}{2M}\right)\right), \tag{17}$$

where M is the number of pruning steps, and the j-th step is centered at layer $\frac{(2j-1)L}{2M}$, which evenly distributes the steps across layers. This design creates M evenly-spaced decision points for progressive information compression across layers. Collectively, these learnable kernels capture a wide spectrum of pruning patterns, enabling the model to discover task-specific compression strategies.

The parameters k and r are dynamically predicted by a lightweight surrogate neural network f_θ. As illustrated in the top-right of Figure 2, the network takes as input the concatenated visual and textual embeddings along with the computational budget B, and computes layer-wise retention ratios $r_i = \mathcal{K}(i; k, r)$ according to the selected pruning pattern. All outputs r_i are clipped to $[0, 1]$. Through gradient-based optimization, the surrogate network learns which layers and tokens are critical. This provides mechanistic insights into how VLMs prioritize visual information across layers.

4 Experiments

4.1 Implementation Details

We use Qwen2.5VL-3B (Bai et al., 2025) as the base model and construct our training set by downsampling 30K samples from LLaVA-Instruct-150K (Liu et al., 2023). To mitigate the long-tail distribution of image resolutions, we apply resolution-based resampling to prevent performance degradation on rare image sizes. Our evaluation spans three categories of benchmarks: visual question answering (A-OKVQA (Schwenk et al., 2022), VizWiz (Bigham et al., 2010), SEED-Bench (Li et al., 2023)), multimodal reasoning (MMBench (Liu et al., 2024a), MME (Fu et al., 2025)), and chart understanding (ChartQA (Masry et al., 2022), OCRBench (Liu et al., 2024b), TextVQA (Singh et al., 2019)).

For single-layer pruning, we set the penalty parameter $\lambda = 100$, convergence threshold $\epsilon = 0.01$, and update coefficients $\alpha = 2$, $\beta = 0.5$. The Gaussian kernel width $\sigma = 10$ and temperature $T = 0.1$ control the continuous relaxation and straight-through estimator, respectively. We use the AdamW optimizer with a learning rate of 4×10^{-4} and batch size of 16. All experiments are conducted on 8 NVIDIA H20 GPUs (96GB each). Other training configurations are provided in Appendix C.1.

4.2 Pareto Frontier Approximation

We apply **VisPCO** to three representative pruning methods: FastV (Chen et al., 2024a), FitPrune (Ye et al., 2025a), and SparseVLM (Zhang et al., 2024), which employ different importance scoring mechanisms for visual tokens. Table 1 compares performance before and after applying **VisPCO** under different FLOPs budgets. For each method, results without **VisPCO** are obtained by sampling multiple configurations and averaging their performance, with standard deviations reported ($\pm$ std). Figure 3(left) illustrates the empirical Pareto frontier obtained through grid search and the predicted Pareto frontier by **VisPCO**, with computational budget on the x-axis and average accuracy across 8

Table 1: Comparison of pruning methods with and without **VisPCO** on eight benchmarks under different budgets. Results without **VisPCO** are averaged over multiple sampled configurations meeting the budget constraint (± std).

Method	AOKVQA	VizWiz	SEED	MMB	MME†	ChartQA	OCRB	TextVQA	Avg (%)
Upper Bound, 100% Budget, ∼3.56 TFLOPs									
Qwen2.5VL-3B	90.2	75.1	75.6	79.8	84.2	64.1	74.6	81.3	78.1
Reduce FLOPs Budget to 90%, ∼3.20 TFLOPs									
⌞ FastV	88.2 ± 0.4	72.9 ± 0.9	72.4 ± 0.9	76.4 ± 0.5	81.3 ± 0.5	62.2 ± 0.8	71.6 ± 0.7	79.1 ± 0.6	75.5 ± 0.7
+ VisPCO	88.4	73.8	73.2	76.9	81.7	62.9	72.3	79.5	76.1
⌞ SparseVLM	88.5 ± 0.3	73.1 ± 0.5	73.4 ± 0.4	76.9 ± 0.6	82.1 ± 0.3	62.2 ± 0.7	71.9 ± 0.6	79.5 ± 0.6	76.0 ± 0.5
+ VisPCO	88.6	73.5	73.8	77.5	82.4	62.9	72.5	80.0	76.4
⌞ FitPrune	89.1 ± 0.5	73.9 ± 0.4	74.2 ± 0.5	77.6 ± 0.4	82.5 ± 0.6	63.1 ± 0.6	72.5 ± 0.5	79.9 ± 0.3	76.2 ± 0.5
+ VisPCO	89.6	74.1	74.6	77.9	82.8	63.5	72.9	81.2	77.1
Reduce FLOPs Budget to 50%, ∼3.56 TFLOPs									
⌞ FastV	74.7 ± 10.1	60.3 ± 9.6	61.5 ± 8.1	62.4 ± 9.3	68.8 ± 9.1	51.6 ± 9.9	59.2 ± 9.1	65.9 ± 10.8	63.1 ± 9.5
+ VisPCO	84.8	69.4	67.6	71.2	77.1	58.1	67.8	75.9	71.5
⌞ SparseVLM	75.9 ± 9.8	62.6 ± 8.2	63.1 ± 7.2	63.9 ± 8.6	69.9 ± 8.2	51.9 ± 8.3	62.4 ± 6.9	66.6 ± 9.8	64.5 ± 8.4
+ VisPCO	85.2	69.0	68.1	71.9	77.6	58.4	67.9	76.3	71.8
⌞ FitPrune	77.1 ± 8.7	63.4 ± 7.7	63.9 ± 6.8	64.5 ± 8.2	70.8 ± 7.9	52.8 ± 8.1	63.3 ± 6.4	67.6 ± 9.4	65.4 ± 7.9
+ VisPCO	85.9	69.4	68.4	72.4	77.9	58.8	68.2	76.6	72.2
Reduce FLOPs Budget to 10%, ∼0.36 TFLOPs									
⌞ FastV	33.3 ± 2.3	30.4 ± 1.6	44.5 ± 2.7	33.0 ± 2.5	39.7 ± 1.4	29.8 ± 4.1	8.3 ± 2.1	33.7 ± 2.8	31.6 ± 2.4
+ VisPCO	35.5	31.7	46.9	35.5	40.1	33.2	10.1	36.1	33.6
⌞ SparseVLM	33.6 ± 2.1	31.2 ± 1.3	44.9 ± 2.5	33.9 ± 2.3	40.3 ± 1.1	30.5 ± 3.7	9.1 ± 2.0	34.4 ± 2.2	32.2 ± 2.2
+ VisPCO	35.5	31.5	47.1	35.8	40.4	33.3	10.2	36.3	33.8
⌞ FitPrune	33.8 ± 2.1	31.5 ± 1.1	45.3 ± 2.4	34.2 ± 2.2	40.6 ± 1.0	30.9 ± 3.5	9.6 ± 1.9	34.6 ± 2.1	32.6 ± 2.0
+ VisPCO	35.6	31.6	47.3	35.8	40.9	33.5	10.4	36.4	33.9

visual benchmarks on the y-axis.

Performance gains at moderate budgets. As shown in Table 1, **VisPCO**'s benefits vary significantly across different computational budget regimes. At extreme budgets, configuration selection has limited impact. For example, at 90% budget, performance varies by less than 1 percentage point across different configurations, as resources are abundant enough that most configurations perform well. Similarly, at very low budgets, severe resource constraints limit all configurations. In contrast, moderate budgets (e.g., 50%) present a critical regime where configuration choice significantly impacts performance—different configurations can vary by up to 19 percentage points. This substantial performance gap demonstrates the importance of principled configuration optimization and validates the need for methods like **VisPCO**.

Quality of frontier approximation. As shown in Figure 3(left), the predicted Pareto frontiers exhibit near-perfect alignment with empirical frontiers. This validates the effectiveness of our kernel-based approximation approach in capturing the true computation-performance trade-off landscape. Table 2 presents a comprehensive comparison with existing approaches, including predefined pruning configuration strategies, training-based methods, and random search baselines (Random-N denotes selecting the best from N random samples). We evaluate both search efficiency and performance. **VisPCO** outperforms all baseline methods including VTW (Lin et al., 2025), G-Search (Zhao et al., 2025), ATP-LLaVA (Ye et al., 2025b), MADTP (Cao et al., 2024), AIM (Zhong et al., 2025), and moderate random search variants, while requiring only 1 hour of training time. Notably, MADTP requires additional training of MAG and DTP modules (6+ hours) and incurs substantially higher FLOPs (3.91T). AIM is training-free with a fixed layer-wise pruning strategy; at comparable FLOPs (2.20T vs. 2.33T), **VisPCO** achieves clearly superior performance (MMB: 43.6 vs. 39.5; TQA: 81.3 vs. 74.6; SEED: 47.5 vs. 43.8).

Table 2: Comparison of configuration search methods. Time represents search time to identify optimal configurations (training time for **VisPCO**). Random-N denotes random sampling with N evaluations. All methods target similar computational budgets.

Methods	FLOPs (T) ↓	Time (h) ↓	MMB ↑	SEED ↑	TQA ↑
VTW	2.34	1+	36.5	44.1	73.2
G-Search	2.58	-	42.1	47.5	80.2
ATP-LLaVA	2.23	48+	37.2	45.5	77.2
MADTP	3.91	6+	42.4	46.8	79.1
AIM	2.33	-	39.5	43.8	74.6
Random-40	2.18	12+	31.2	36.6	70.9
Random-80	2.20	24+	42.8	47.7	80.1
Random-160	2.20	48+	44.0	48.1	82.4
VisPCO	2.20	1+	43.6	47.5	81.3

4.3 Cross-Model Generalization

To validate the generalization capability of **VisPCO** across different VLM architectures, we apply it to Gemma3-4B (Team et al., 2025) and LLaVA-

†MME scores are normalized to percentages.

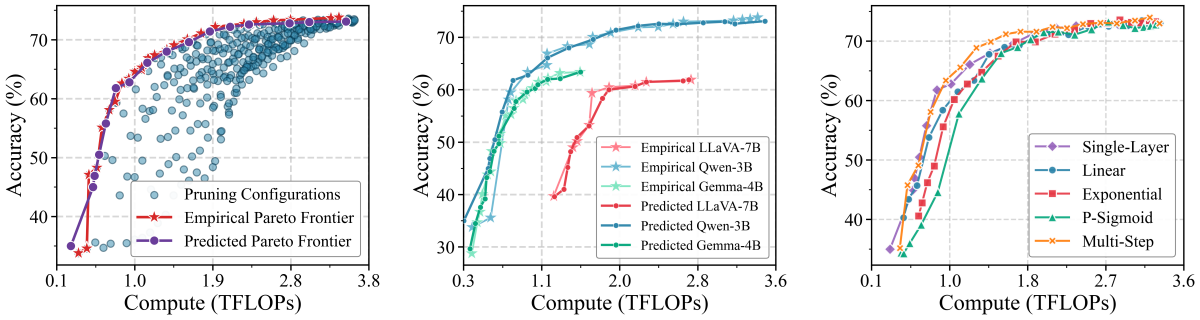

Figure 3: Experimental results of **VisPCO**. (Left) Comparison between empirical and predicted Pareto frontiers. (Middle) Comparison between empirical and predicted Pareto frontiers across different VLM architectures. (Right) Comparison of Pareto frontiers among different pruning patterns.

Table 3: Performance of different VLMs with and without **VisPCO** at 50% FLOPs budget ($\pm$ std).

Models	AOKVQA	MMBench	TextVQA	Avg (%)
LLaVA-7B	50.3 ± 10.7	26.5 ± 11.5	64.2 ± 11.4	47.0 ± 11.2
+ VisPCO	60.8	38.0	75.2	58.0
Gemma3-4B	41.2 ± 12.4	44.7 ± 12.9	33.6 ± 12.3	39.8 ± 12.5
+ VisPCO	53.4	57.3	45.5	52.1
Qwen2.5VL-3B	74.7 ± 10.1	62.4 ± 9.3	65.9 ± 10.8	67.7 ± 10.1
+ VisPCO	84.8	71.2	75.9	77.3

Table 4: Hardware performance measurements at 50% compute budget across models. TTFT: time to first token. Throughput: tokens generated per second. Avg. Perf.: average accuracy across benchmarks.

Method	Budget	TTFT (ms) ↓	Throughput (tokens/s) ↑	Avg. Perf. ↑
Qwen2.5VL-3B	100%	83 ± 3	18 ± 2	78.1
+ FastV	50%	74 ± 2	20 ± 3	63.1
+ FastV + VisPCO	50%	76 ± 3	20 ± 2	**71.5**
LLaVA-v1.5-7B	100%	114 ± 12	14 ± 6	63.9
+ FastV	50%	96 ± 10	16 ± 6	41.4
+ FastV + VisPCO	50%	97 ± 8	16 ± 4	**52.3**
Gemma3-4B	100%	94 ± 6	16 ± 4	68.9
+ FastV	50%	81 ± 7	18 ± 5	38.8
+ FastV + VisPCO	50%	81 ± 6	18 ± 5	**50.8**

v1.5-7B (Liu et al., 2023). Table 3 presents representative results using FastV (Chen et al., 2024a) under a 50% FLOPs budget.

Consistency across architectures. While Qwen2.5VL exhibits a wider performance range compared to Gemma3 and LLaVA, **VisPCO** consistently selects optimal configurations. Figure 3(middle) shows that predicted Pareto frontiers closely match the empirical frontiers across all architectures, demonstrating robust generalization. Notably, Qwen2.5VL's frontier is positioned in the upper-left region, indicating superior accuracy-efficiency characteristics. This advantage stems from its native image resolution, whereas Gemma3 and LLaVA resize inputs to fixed dimensions. This suggests preserving original dimensions with learned pruning is potentially more effective than aggressive preprocessing for efficient VLMs.

Hardware efficiency. To verify that FLOPs reductions translate into practical speedups, we measure time-to-first-token (TTFT) and throughput on an NVIDIA H20 GPU. As shown in Table 4, at a 50% compute budget, FastV with **VisPCO** maintains the same hardware efficiency as FastV alone—comparable TTFT and throughput—while recovering substantial performance lost from pruning. For example, on Qwen2.5VL-3B, **VisPCO** improves average performance from 63.1% to 71.5% with negligible latency overhead (76 vs. 74 ms TTFT). Similar patterns hold across LLaVA-v1.5-7B and Gemma3-4B, confirming that **VisPCO** preserves the underlying pruning method's hardware efficiency while substantially improving task performance.

Table 5: Comparison of performance under different pruning patterns at 50% FLOPs budget.

Patterns	Kernels	AOK	MMB	TQA	Avg (%)
Single-Layer	-	84.8	71.2	75.9	77.3
Multi-Layer	Linear	82.6	70.9	74.9	76.1
	Exponential	82.2	70.4	74.4	75.7
	P-Sigmoid	81.9	69.6	74.1	75.2
	Multi-Step	84.9	71.8	76.7	77.8

4.4 Analysis of Pruning Patterns

We investigate the performance differences between single-layer and multi-layer pruning strategies, as well as the impact of different kernel choices in multi-layer configurations. Table 5 presents the Pareto-optimal results for different pruning patterns under a 50% computational budget. Figure 3(right) illustrates and compares the Pareto frontiers across different pruning patterns.

Strategic pruning pattern selection. As shown in Table 5, multi-layer pruning with the multi-step kernel achieves the best performance under a 50% budget, outperforming both single-layer pruning and other kernel variants (linear, exponential, sig-

moid). Figure 3(right) reveals that this advantage is budget-dependent. When computational budget exceeds 50%, all pruning patterns converge to comparable performance and closely approximate the empirical Pareto frontier, making strategy selection less critical. However, below 50% budget, notable differences emerge among pruning patterns, with the multi-step kernel showing clear superiority.

Implications for VLM design. The multi-step kernel's superior performance at low budgets reveals important architectural insights. Visual token redundancy emerges at specific layers rather than uniformly across the network. Certain layers introduce redundancy through attention or feature transformations, while others preserve essential representations. The multi-step kernel identifies these critical compression points, enabling targeted pruning while retaining key information. These findings provide practical guidance. When resources are sufficient (budget >50%), simple single-layer pruning achieves near-optimal performance. Under tight constraints (budget <50%), multi-step layer-wise pruning is recommended to better exploit VLMs' hierarchical compression structure.

5 Conclusion

In this paper, we introduced **VisPCO**, a novel computation budget-aware framework for automatically optimizing visual token pruning configurations in vision-language models. By formulating the problem as Pareto optimization with continuous relaxation, **VisPCO** enables efficient end-to-end gradient-based training to automatically identify optimal pruning configurations for any given computational budget. This approach significantly reduces search costs compared to traditional exhaustive grid search methods. Extensive experiments across 8 visual benchmarks demonstrate that our method generalizes well across various pruning strategies and VLM architectures. Furthermore, our investigation through learnable kernel functions reveals that progressive multi-step pruning consistently outperforms both single-layer and other multi-layer kernel approaches, providing valuable insights for efficient VLM design in resource-constrained deployment scenarios.

Limitations

Although our framework demonstrates strong performance across diverse benchmarks and model architectures, several limitations remain to be addressed in future work. First, our experiments primarily focus on single-image tasks; further validation is needed to assess how effectively our optimized pruning configurations generalize to multi-image and video inputs, which involve more complex temporal and spatial redundancies. Second, our proposed kernel functions provide a structured and interpretable way to model pruning distributions. Future work could explore extending this approach to learn more flexible, non-parametric or input-adaptive patterns, potentially capturing even more nuanced task-specific pruning strategies.

Ethics Statement

This work focuses on optimizing visual token pruning configurations for vision-language models to improve computational efficiency. Our method does not involve the collection or use of private or sensitive data; all experiments are conducted on publicly available benchmarks. We do not foresee direct negative societal impacts from this research. By reducing the computational cost of VLMs, our work may contribute to lowering energy consumption and carbon emissions associated with large-scale model inference, thereby promoting more sustainable and accessible AI deployment.

Acknowledgments

We sincerely thank the students and engineers at the Data Intelligence Research Center, Shanghai Jiao Tong University, for their assistance during the development of this work. This work was supported by NSF China under Grant No.T2421002, 92579104, 62525209, T2542021.

References

Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Chunsheng Wu, and 1 others. 2025. Llava-onevision-1.5: Fully open framework for democratized multimodal training. *arXiv preprint arXiv:2509.23661*.

Kenneth J Arrow and Gerard Debreu. 2024. Existence of an equilibrium for a competitive economy. In *The Foundations of Price Theory Vol 5*, pages 289–316. Routledge.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others.

2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Dimitri P Bertsekas. 2014. *Constrained optimization and Lagrange multiplier methods*. Academic press.

Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and 1 others. 2010. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, pages 333–342.

Jianjian Cao, Peng Ye, Shengze Li, Chong Yu, Yansong Tang, Jiwen Lu, and Tao Chen. 2024. Madtp: Multimodal alignment-guided dynamic token pruning for accelerating vision-language transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15710–15719.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024a. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35. Springer.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, and 1 others. 2024b. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, and 1 others. 2025. Mme: A comprehensive evaluation benchmark for multimodal large language models. In *The 39th Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Thomas Fuchs. 2000. Das gedächtnis des leibes. *Phänomenologische Forschungen*, 5(1):71–89.

Zonghao Guo, Ruyi Xu, Yuan Yao, Junbo Cui, Zanlin Ni, Chunjiang Ge, Tat-Seng Chua, Zhiyuan Liu, and Gao Huang. 2024. Llava-uhd: an lmm perceiving any aspect ratio and high-resolution images. In *European Conference on Computer Vision*, pages 390–406. Springer.

Wenxuan Huang, Zijie Zhai, Yunhang Shen, Shaosheng Cao, Fei Zhao, Xiangfeng Xu, Zheyu Ye, Yao Hu, and Shaohui Lin. 2024. Dynamic-llava: Efficient multimodal large language models via dynamic vision-language context sparsification. *arXiv preprint arXiv:2412.00876*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. 2023. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. 2024. Video-llava: Learning united visual representation by alignment before projection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5971–5984.

Zhihang Lin, Mingbao Lin, Luxi Lin, and Rongrong Ji. 2025. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 5334–5342.

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

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, and 1 others. 2024a. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. 2024b. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pages 2263–2279.

Jorge Nocedal and Stephen J Wright. 2006. *Numerical optimization*. Springer.

Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. 2022. A-okvqa: A benchmark for visual question answering using world knowledge. In *European conference on computer vision*, pages 146–162. Springer.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and 1 others. 2024. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*.

Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Kelly Peng, Yao Lu, and Song Han. 2025. Streamingvlm: Real-time understanding for infinite video streams. *arXiv preprint arXiv:2510.09608*.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Chendi Li, Jinghua Yan, Yu Bai, Ponnuswamy Sadayappan, Xia Hu, and Bo Yuan. 2025. Topv: Compatible token pruning with inference time optimization for fast and low-memory multimodal vision language model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19803–19813.

Weihao Ye, Qiong Wu, Wenhao Lin, and Yiyi Zhou. 2025a. Fit and prune: Fast and training-free visual token pruning for multi-modal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 22128–22136.

Xubing Ye, Yukang Gan, Yixiao Ge, Xiao-Ping Zhang, and Yansong Tang. 2025b. Atp-llava: Adaptive token pruning for large vision language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24972–24982.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. 2024. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*.

Shiyu Zhao, Zhenting Wang, Felix Juefei-Xu, Xide Xia, Miao Liu, Xiaofang Wang, Mingfu Liang, Ning Zhang, Dimitris N Metaxas, and Licheng Yu. 2025. Accelerating multimodal large language models by searching optimal vision token reduction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 29869–29879.

Yiwu Zhong, Zhuoming Liu, Yin Li, and Liwei Wang. 2025. Aim: Adaptive inference of multi-modal llms via token merging and pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20180–20192.

Yi Zhou, Hui Zhang, Jiaqian Yu, Yifan Yang, Sangil Jung, Seung-In Park, and ByungIn Yoo. 2024. Himap: Hybrid representation learning for end-to-end vectorized hd map construction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15396–15406.

A FLOPs Computation

In this section, we provide a detailed derivation of the floating-point operations (FLOPs) computation for Transformer layers in vision-language models.

For a standard Transformer layer, the primary computational costs come from the self-attention mechanism and the feed-forward network (FFN). Given a sequence length N and hidden dimension D, we compute the FLOPs for each component separately.

A.1 Self-Attention Mechanism

The self-attention mechanism consists of the following operations:

(1) Linear projections: Three projection matrices $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{D\times D}$ map the input to Query, Key, and Value representations. Each matrix multiplication requires $2ND^2$ floating-point operations (multiplying input of size $N \times D$ with weight of size $D \times D$), thus:

$$\text{FLOPs}_{\text{QKV}} = 3 \times 2ND^2 = 6ND^2. \tag{18}$$

(2) Attention score: Computing $\mathbf{QK}^T \in \mathbb{R}^{N\times N}$, where $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{N\times D}$:

$$\text{FLOPs}_{\text{Score}} = 2N^2D. \tag{19}$$

(3) Attention weighting: Computing $\text{Softmax}(\mathbf{QK}^T/\sqrt{D})\mathbf{V}$, i.e., multiplying $\mathbb{R}^{N\times N}$ with $\mathbb{R}^{N\times D}$:

$$\text{FLOPs}_{\text{Weight}} = 2N^2D. \tag{20}$$

Note: The FLOPs for the Softmax operation are relatively small and typically neglected.

(4) Output projection: Projecting back to the original dimension through $\mathbf{W}_O \in \mathbb{R}^{D\times D}$:

$$\text{FLOPs}_{\text{Output}} = 2ND^2. \tag{21}$$

Therefore, the total FLOPs for the self-attention mechanism is:

$$\text{FLOPs}_{\text{Attention}} = 8ND^2 + 4N^2D. \tag{22}$$

A.1.1 Feed-Forward Network

The standard FFN consists of two linear layers with an intermediate dimension D_{ffn}:

$$\text{FFN}(\mathbf{x}) = \mathbf{W}_2 \cdot \text{GELU}(\mathbf{W}_1 \cdot \mathbf{x}). \tag{23}$$

where $\mathbf{W}_1 \in \mathbb{R}^{D\times D_{\text{ffn}}}$ and $\mathbf{W}_2 \in \mathbb{R}^{D_{\text{ffn}}\times D}$. The total FLOPs for the FFN is:

$$\text{FLOPs}_{\text{FFN}} = 4ND_{\text{ffn}}D. \tag{24}$$

A.1.2 Total FLOPs per Layer

Combining self-attention and FFN, the total FLOPs for a single Transformer layer is:

$$\text{FLOPs}_{\text{layer}} = 8ND^2 + 4N^2D + 4ND_{\text{ffn}}D. \quad (25)$$

In standard Transformer architectures, $D_{\text{ffn}} = 4D$, which gives:

$$\text{FLOPs}_{\text{layer}} = 24ND^2 + 4N^2D. \quad (26)$$

We neglect relatively small computational costs such as LayerNorm and residual connections.

A.2 Total FLOPs for Vision-Language Models

For vision-language models, the input sequence consists of text tokens and visual tokens. Let N_t denote the number of text tokens and N_v denote the initial number of visual tokens. The total number of tokens at layer i is:

$$N_i = N_t + r_i N_v \quad (27)$$

where $r_i \in [0, 1]$ represents the retention ratio of visual tokens at layer i.

For a Transformer model with L layers, the total computational cost is:

$$F(\mathbf{r}) = \sum_{i=1}^{L} \left[24(N_t + r_i N_v)D^2 + 4(N_t + r_i N_v)^2 D\right] \quad (28)$$

where:

- The first term $24(N_t + r_i N_v)D^2$ corresponds to linear projections in self-attention and the FFN
- The second term $4(N_t+r_i N_v)^2 D$ corresponds to the quadratic complexity of attention matrix computation
- $\mathbf{r} = [r_1, r_2, \ldots, r_L]$ is the vector of visual token retention ratios across layers

This formula indicates that as visual tokens are pruned (r_i decreases), the model's computational cost is significantly reduced, especially the quadratic complexity term.

B Theoretical Analysis

B.1 Proof of Theorem 1

Consider the equality-constrained optimization problem:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, l, \end{aligned} \quad (29)$$

where $f, h_j : \mathbb{R}^n \to \mathbb{R}$ are twice continuously differentiable functions. The augmented Lagrangian function for this problem is given by Equation (4).

Let $\bar{\mathbf{x}}$ be a local optimal solution of problem (29) that satisfies the second-order sufficient conditions. That is, there exists a Lagrange multiplier vector $\bar{\mathbf{v}} = [\bar{v}_1, \ldots, \bar{v}_l]^T$ such that:

$$\nabla f(\bar{\mathbf{x}}) - \mathbf{A}\bar{\mathbf{v}} = 0, \quad (30)$$

$$h_j(\bar{\mathbf{x}}) = 0, \quad j = 1, \ldots, l, \quad (31)$$

and for every nonzero vector $\mathbf{d}$ satisfying $\mathbf{d}^T \nabla h_j(\bar{\mathbf{x}}) = 0$ for $j = 1, \ldots, l$, we have:

$$\mathbf{d}^T \nabla^2_{\mathbf{x}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{v}})\mathbf{d} > 0, \quad (32)$$

where

$$\mathbf{A} = [\nabla h_1(\bar{\mathbf{x}}), \ldots, \nabla h_l(\bar{\mathbf{x}})], \quad (33)$$

and $\mathcal{L}(\mathbf{x}, \mathbf{v}) = f(\mathbf{x}) - \mathbf{v}^T \mathbf{h}(\mathbf{x})$ is the standard Lagrangian function.

By assumption, $\bar{\mathbf{x}}$ is a Karush-Kuhn-Tucker (KKT) point of problem (29), thus:

$$\nabla_{\mathbf{x}} \phi(\bar{\mathbf{x}}, \bar{\mathbf{v}}, \lambda) = 0. \quad (34)$$

We now prove that the Hessian matrix $\nabla^2_{\mathbf{x}} \phi(\bar{\mathbf{x}}, \bar{\mathbf{v}}, \lambda)$ is positive definite at $\bar{\mathbf{x}}$ for sufficiently large λ.

From Equation (4), we can derive:

$$\begin{aligned} \nabla^2_{\mathbf{x}} \phi(\mathbf{x}, \bar{\mathbf{v}}, \lambda) &= \nabla^2 f(\mathbf{x}) - \sum_{j=1}^{l} \bar{v}_j \nabla^2 h_j(\mathbf{x}) \\ &+ \sigma \sum_{j=1}^{l} h_j(\mathbf{x}) \nabla^2 h_j(\mathbf{x}) + \lambda \sum_{j=1}^{l} \nabla h_j(\mathbf{x}) \nabla h_j(\mathbf{x})^T \quad (35) \\ &= \nabla^2 f(\mathbf{x}) - \sum_{j=1}^{l} (\bar{v}_j - \lambda h_j(\mathbf{x})) \nabla^2 h_j(\mathbf{x}) \\ &+ \lambda \sum_{j=1}^{l} \nabla h_j(\mathbf{x}) \nabla h_j(\mathbf{x})^T = \mathbf{Q} + \lambda \mathbf{A}\mathbf{A}^T, \end{aligned}$$

where

$$\mathbf{Q} = \nabla^2 f(\mathbf{x}) - \sum_{j=1}^{l} (\bar{v}_j - \lambda h_j(\mathbf{x})) \nabla^2 h_j(\mathbf{x}), \quad (36)$$

$$\mathbf{A} = [\nabla h_1(\mathbf{x}), \ldots, \nabla h_l(\mathbf{x})]. \quad (37)$$

At the point $\bar{\mathbf{x}}$, we have:

$$\nabla^2_{\mathbf{x}} \phi(\bar{\mathbf{x}}, \bar{\mathbf{v}}, \lambda) = \bar{\mathbf{Q}} + \lambda \bar{\mathbf{A}} \bar{\mathbf{A}}^T, \quad (38)$$

where $\bar{\mathbf{Q}}$ and $\bar{\mathbf{A}}$ denote the evaluations at $\bar{\mathbf{x}}$.

Let $\text{rank}(\bar{\mathbf{A}}) = r \leq l$, and let $\mathbf{B} \in \mathbb{R}^{n\times r}$ be an orthonormal basis matrix for $\bar{\mathbf{A}}$ (i.e., $\mathbf{B}^T\mathbf{B} = \mathbf{I}_r$), meaning the r columns of $\mathbf{B}$ form an orthonormal basis for the subspace spanned by the l columns of $\bar{\mathbf{A}}$. Thus, we have:

$$\bar{\mathbf{A}} = \mathbf{BC}, \tag{39}$$

where $\mathbf{C} = \mathbf{B}^T\bar{\mathbf{A}}$ has rank r.

For any nonzero vector $\mathbf{u} \in \mathbb{R}^n$, we decompose it as:

$$\mathbf{u} = \mathbf{p} + \mathbf{Bq}, \tag{40}$$

where $\mathbf{p}$ satisfies $\mathbf{B}^T\mathbf{p} = \mathbf{0}$. Clearly, $\bar{\mathbf{A}}^T\mathbf{p} = \mathbf{0}$, which implies:

$$\nabla h_j(\bar{\mathbf{x}})^T\mathbf{p} = 0, \quad j = 1, \ldots, l. \tag{41}$$

Now, we can write $\mathbf{u}^T\nabla^2_{\mathbf{x}}\phi(\bar{\mathbf{x}}, \bar{\mathbf{v}}, \lambda)\mathbf{u}$ as:

$$\begin{aligned}
&\mathbf{u}^T\nabla^2_{\mathbf{x}}\phi(\bar{\mathbf{x}}, \bar{\mathbf{v}}, \lambda)\mathbf{u} \\
&= (\mathbf{p} + \mathbf{Bq})^T(\bar{\mathbf{Q}} + \lambda\bar{\mathbf{A}}\bar{\mathbf{A}}^T)(\mathbf{p} + \mathbf{Bq}) \\
&= \mathbf{p}^T\bar{\mathbf{Q}}\mathbf{p} + 2\mathbf{p}^T\bar{\mathbf{Q}}\mathbf{Bq} + \mathbf{q}^T\mathbf{B}^T\bar{\mathbf{Q}}\mathbf{Bq} \\
&+ \lambda\mathbf{q}^T\mathbf{C}\mathbf{C}^T\mathbf{q}.
\end{aligned} \tag{42}$$

Since $\bar{\mathbf{x}}$ is a local optimal solution of problem (29) satisfying the second-order sufficient conditions, there exists a constant $\alpha > 0$ such that:

$$\mathbf{p}^T\bar{\mathbf{Q}}\mathbf{p} \geq \alpha\|\mathbf{p}\|^2. \tag{43}$$

Let b be the largest singular value of $\bar{\mathbf{Q}}\mathbf{B}$, let $e = \|\mathbf{B}^T\bar{\mathbf{Q}}\mathbf{B}\|_2$, and let $\mu > 0$ be the smallest eigenvalue of $\mathbf{CC}^T$. Then:

$$\mathbf{u}^T\nabla^2_{\mathbf{x}}\phi(\bar{\mathbf{x}}, \bar{\mathbf{v}}, \lambda)\mathbf{u} \geq \alpha\|\mathbf{p}\|^2 - 2b\|\mathbf{p}\|\|\mathbf{q}\| + (\lambda\mu - e)\|\mathbf{q}\|^2. \tag{44}$$

Since $\mathbf{u} \neq \mathbf{0}$, the vectors $\mathbf{p}$ and $\mathbf{q}$ cannot both be zero. Therefore, if we choose λ sufficiently large such that:

$$\lambda\mu - e - \frac{b^2}{\alpha} > 0, \tag{45}$$

that is,

$$\lambda > \frac{b^2 + \alpha e}{\alpha\mu}, \tag{46}$$

then we always have:

$$\mathbf{u}^T\nabla^2_{\mathbf{x}}\phi(\bar{\mathbf{x}}, \bar{\mathbf{v}}, \lambda)\mathbf{u} > 0. \tag{47}$$

Therefore, there exists:

$$\lambda' = \frac{b^2 + \alpha e}{\alpha\mu}. \tag{48}$$

When the penalty parameter $\lambda > \lambda'$, the matrix $\nabla^2_{\mathbf{x}}\phi(\bar{\mathbf{x}}, \bar{\mathbf{v}}, \lambda)$ is positive definite. Combined with Equations (34) and (47), we conclude that $\bar{\mathbf{x}}$ is a strict local minimizer of $\phi(\mathbf{x}, \bar{\mathbf{v}}, \lambda)$. This completes the proof.

B.2 Elimination of y via Quadratic Completion

Starting from the augmented Lagrangian function in Equation (7), we apply the technique of completing the square to eliminate the auxiliary variable y.

Let $g(\mathbf{r}) = B - F(\mathbf{r})$ denote the constraint function. We can rewrite Equation (7) as:

$$\begin{aligned}
&\tilde{\phi}(\mathbf{r}, y, w, \lambda) \\
&= \mathcal{L}_{\text{distill}}(\mathbf{r}) - w(g(\mathbf{r}) - y^2) + \frac{\lambda}{2}(g(\mathbf{r}) - y^2)^2 \\
&= \mathcal{L}_{\text{distill}}(\mathbf{r}) + \left[-w(g(\mathbf{r}) - y^2) + \frac{\lambda}{2}(g(\mathbf{r}) - y^2)^2\right].
\end{aligned} \tag{49}$$

Completing the square with respect to y^2, we have:

$$\begin{aligned}
&- w(g(\mathbf{r}) - y^2) + \frac{\lambda}{2}(g(\mathbf{r}) - y^2)^2 \\
&= \frac{\lambda}{2}\left[(g(\mathbf{r}) - y^2) - \frac{w}{\lambda}\right]^2 - \frac{w^2}{2\lambda} \\
&= \frac{\lambda}{2}\left[y^2 - \left(g(\mathbf{r}) - \frac{w}{\lambda}\right)\right]^2 - \frac{w^2}{2\lambda}.
\end{aligned} \tag{50}$$

To minimize $\tilde{\phi}$ with respect to y, we analyze the optimal value of y^2. The term $\frac{\lambda}{2}\left[y^2 - \left(g(\mathbf{r}) - \frac{w}{\lambda}\right)\right]^2$ is minimized when:

$$y^2 = g(\mathbf{r}) - \frac{w}{\lambda} = \frac{1}{\lambda}(\lambda g(\mathbf{r}) - w). \tag{51}$$

However, since $y \in \mathbb{R}$, we must have $y^2 \geq 0$. Therefore, the optimal value is:

$$y^2 = \max\left\{0, \frac{1}{\lambda}(\lambda g(\mathbf{r}) - w)\right\}. \tag{52}$$

This can be expressed equivalently as:

$$y^2 = \begin{cases} \frac{1}{\lambda}(\lambda g(\mathbf{r}) - w), & \text{if } \lambda g(\mathbf{r}) - w \geq 0, \\ 0, & \text{if } \lambda g(\mathbf{r}) - w < 0. \end{cases} \tag{53}$$

Substituting the optimal y^2 back into Equation (50), we obtain:

$$-w(g(\mathbf{r}) - y^2) + \frac{\lambda}{2}(g(\mathbf{r}) - y^2)^2 = \frac{1}{2\lambda}\left(z^2 - w^2\right), \tag{54}$$

where

$$\begin{aligned}
z &= \max\{0, w - \lambda(B - F(\mathbf{r}))\} \\
&= \max\{0, w - \lambda g(\mathbf{r})\}.
\end{aligned} \tag{55}$$

Therefore, the simplified augmented Lagrangian function, after eliminating y, is:

$$\phi(\mathbf{r}, w, \lambda) = \mathcal{L}_{\text{distill}}(\mathbf{r}) + \frac{1}{2\lambda}\left(z^2 - w^2\right), \tag{56}$$

where $z = \max\{0, w - \lambda(B - F(\mathbf{r}))\}$ and $g(\mathbf{r}) = B - F(\mathbf{r})$ represents the constraint satisfaction.

C Experiment Details

C.1 Experiment Settings

C.1.1 Training Dataset

We find that directly training on the original dataset leads to suboptimal performance: **VisPCO**'s predicted Pareto frontier for high-resolution images concentrates on low computational budgets, diverging from the empirical frontier under high budget regimes. We analyze the training dataset and observe that the distribution of image areas exhibits significant skewness, heavily concentrated on smaller areas, as shown in the left panel of Figure 4. This imbalance is detrimental to learning appropriate pruning ratios, as it leads to poor generalization on high-resolution images.

To address this issue, we preprocess the training images using histogram equalization to balance the area distribution. Specifically, we divide the image area range into uniform bins and apply stratified sampling to ensure balanced representation across all area intervals. For each bin, we either oversample images (for underrepresented bins) or subsample images (for overrepresented bins) to achieve approximately equal counts per bin. This rebalancing procedure ensures that the training distribution covers the full spectrum of image resolutions uniformly, enabling **VisPCO** to learn robust pruning configurations for both low and high-resolution images. The left panel of Figure 4 shows the original skewed distribution, while the right panel illustrates the balanced distribution after equalization.

C.1.2 Evaluation Datasets

We utilize the evaluation datasets provided by VLMEvalKit, which includes curated question-answer pairs and images from various vision-language benchmarks. Our evaluation spans three categories of tasks: visual question answering, multimodal reasoning, and chart understanding. For MME, to maintain comparability with other benchmarks, we report the ratio of correct answers to total questions, normalizing the final evaluation results to the range [0, 1]. An example evaluation case is shown in Figure 5.

A-OKVQA (Schwenk et al., 2022) is a knowledge-based visual question answering dataset that requires models to leverage external commonsense and world knowledge beyond visual content. It contains 1,145 questions across diverse image types, challenging models to perform reasoning that combines visual understanding with factual knowledge.

VizWiz (Bigham et al., 2010) is a visual question answering dataset collected from blind users who took images and asked questions about them. The dataset contains over 4,319 image-question pairs with natural, real-world scenarios, often featuring challenging conditions such as poor image quality, blur, or unusual viewpoints, making it particularly valuable for evaluating model robustness.

SEEDBench (Li et al., 2023) is a comprehensive benchmark for evaluating multimodal large language models across multiple dimensions. It includes 14,232 multiple-choice questions spanning nine evaluation dimensions including scene understanding, instance identity, spatial relation, and visual reasoning, providing a holistic assessment of model capabilities.

MMBench (Liu et al., 2024a) (Multimodal Benchmark) is a systematically designed objective benchmark for evaluating various abilities of vision-language models. It covers 20 ability dimensions organized into three categories: perception (e.g., object localization, OCR), reasoning (e.g., social reasoning, physical commonsense), and knowledge (e.g., celebrity recognition, landmark identification).

MME (Fu et al., 2025) (Multi-Modal Evaluation) is a comprehensive evaluation benchmark measuring both perception and cognition abilities. It consists of 14 subtasks including existence, count, position, color, posters, celebrity, scene, landmark, artwork, OCR, commonsense reasoning, numerical calculation, text translation, and code reasoning. We normalize scores to [0, 1] for consistency with other benchmarks.

ChartQA (Masry et al., 2022) focuses on question answering about statistical charts and plots. The dataset contains over 2,000 human-written questions covering bar charts, line plots, and pie charts, requiring models to perform visual reasoning, data extraction, and numerical computation from chart images.

OCRBench (Liu et al., 2024b) is a comprehensive benchmark for evaluating optical character recognition and text understanding capabilities in vision-language models. It includes diverse text recognition scenarios such as scene text, handwritten text, document text, and multilingual text, assessing both basic OCR accuracy and text-based reasoning abilities.

TextVQA (Singh et al., 2019) requires models to read and reason about text in images to answer

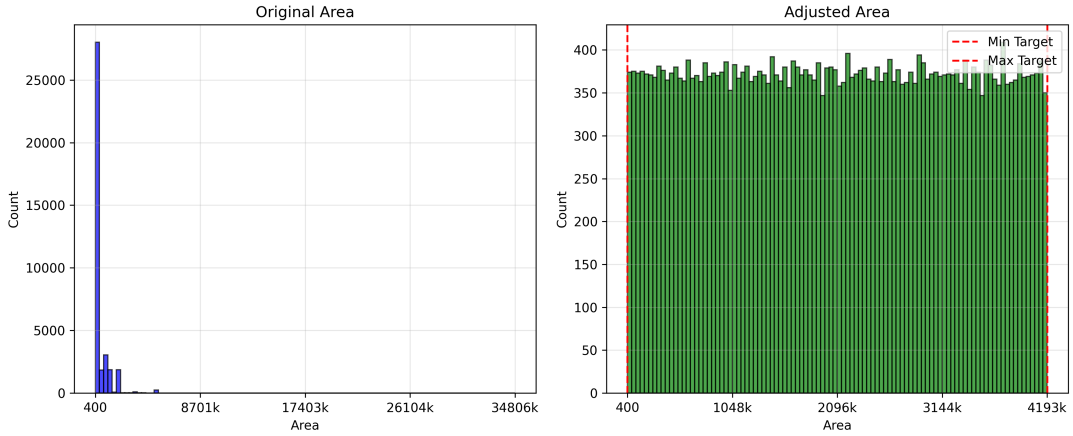

Figure 4: Distribution histogram of image areas in the training dataset before and after applying histogram equalization to balance area diversity. The left panel shows the original distribution heavily concentrated on smaller image areas, while the right panel demonstrates the more balanced distribution after the equalization process.

User: "<image>
What is the laptop on? Given the following options, choose the correct answer:
A. counter B. bed
C. island D. table
Only give the correct choice:"
Assistant: "The answer is D. table"

Figure 5: An example evaluation case from the VLMEvalKit benchmark. The figure demonstrates a typical question-answer pair with the corresponding image, showing how the model processes visual and textual inputs to generate responses for evaluation.

questions. The dataset contains 1,000 images from OpenImages, where answering questions necessitates reading and understanding scene text, making it essential for evaluating text-aware visual reasoning capabilities.

C.1.3 Pruning Configuration Sampling

To identify the empirical Pareto frontier that serves as the ground truth for evaluating **VisPCO**, we employ a comprehensive sampling-based approach. Our methodology consists of three steps: (1) systematically sampling a large number of pruning configurations across the search space, (2) evaluating each configuration's performance across multiple benchmarks and measuring its computational cost in FLOPs, and (3) extracting the Pareto-optimal configurations from the evaluated results.

Sampling Strategy. Our sampling strategy operates at the layer level to capture fine-grained pruning patterns. For a vision-language model with L Transformer layers, we independently sample the visual token retention ratio for each layer from layer 1 to layer L. Specifically, the retention ratio r_i for layer i is sampled from the discrete set $\{0.01, 0.06, 0.11, \ldots, 0.96, 0.99\}$, with a uniform step size of 0.05. This granularity balances comprehensive coverage of the configuration space with computational feasibility. For the Qwen2.5-VL-3B model with L=36 layers, this sampling scheme generates a total of 700 distinct pruning configurations spanning diverse computational budgets.

Evaluation Protocol. For each sampled configuration, we perform a complete evaluation to obtain both its performance and computational cost. Performance is measured by averaging accuracy across our eight evaluation benchmarks, providing a comprehensive assessment of model capabilities. Computational cost is calculated using the FLOPs formula derived in Appendix A, accounting for both the attention mechanism and feed-forward network operations across all layers.

Pareto Frontier Extraction. Given the set of evaluated configurations $\mathcal{C} = \{(p_i, f_i)\}_{i=1}^{N}$, where $p_i \in [0, 1]$ represents the normalized average performance (higher is better) and f_i represents the computational cost in TFLOPs (lower is better) for configuration i, we identify the Pareto frontier us-

Table 6: Hyperparameters for main experiments comparing different methods across multiple VLMs.

Model + Method	λ	α	ϵ	β	σ	T	lr	B
Qwen2.5-VL-3B + FastV	100	5	0.005	0.5	10	0.1	1e-4	16
Qwen2.5-VL-3B + SparseVLM	100	5	0.005	0.5	10	0.1	1e-4	16
Qwen2.5-VL-3B + FitPrune	100	5	0.005	0.5	10	0.1	1e-4	16
Gemma3-4B + FastV	1	5	0.005	0.5	10	0.1	5e-4	16
Gemma3-4B + SparseVLM	1	5	0.005	0.5	10	0.1	5e-4	16
Gemma3-4B + FitPrune	1	5	0.005	0.5	10	0.1	5e-4	16
LLaVA-v1.5-7B + FastV	100	10	0.01	0.5	10	0.1	5e-5	16
LLaVA-v1.5-7B + SparseVLM	100	10	0.01	0.5	10	0.1	5e-5	16
LLaVA-v1.5-7B + FitPrune	100	10	0.01	0.5	10	0.1	5e-5	16

Table 7: Hyperparameters for ablation studies on different pruning scheduling strategies.

Model + Strategy	λ	α	ϵ	β	σ	T	lr	B
Qwen2.5-VL-3B + Linear	100	5	0.01	0.5	1	0.1	1e-4	16
Qwen2.5-VL-3B + Exponential	100	10	0.005	0.5	1	0.1	5e-5	16
Qwen2.5-VL-3B + P-sigmoid	100	5	0.01	0.5	1	0.1	1e-4	16
Qwen2.5-VL-3B + Multi-step	1	10	0.05	0.5	5	0.1	1e-4	16

ing the Pareto dominance criterion. Formally, a configuration (p_i, f_i) is said to dominate another configuration (p_j, f_j) if and only if:

$$p_i \geq p_j \quad \text{and} \quad f_i \leq f_j. \tag{57}$$

The Pareto frontier $\mathcal{P}$ consists of all non-dominated configurations:

$$\begin{aligned} \mathcal{P} = \{(p_i, f_i) \in \mathcal{C} \mid \nexists (p_j, f_j) \in \mathcal{C} \\ \text{such that } (p_j, f_j) \text{ dominates } (p_i, f_i)\}. \end{aligned} \tag{58}$$

These Pareto-optimal configurations represent the best achievable trade-offs between performance and computational efficiency, forming the empirical frontier against which we evaluate **VisPCO**'s predictions. This extensive sampling and evaluation process requires significant computational resources (approximately 48+ GPU hours for 700 configurations), highlighting the practical necessity of efficient optimization methods like **VisPCO**.

C.1.4 Hyperparameter Settings

We provide detailed hyperparameter configurations for our experiments in Tables 6 and 7. The key hyperparameters and their roles are as follows:

λ denotes the penalty parameter in the augmented Lagrangian method, controlling the strength of constraint enforcement. ϵ is the convergence threshold that determines when the optimization terminates. α and β are the update coefficients for the Lagrangian multiplier and penalty parameter, respectively, governing the convergence dynamics. σ controls the Gaussian kernel width for continuous relaxation of discrete pruning decisions,

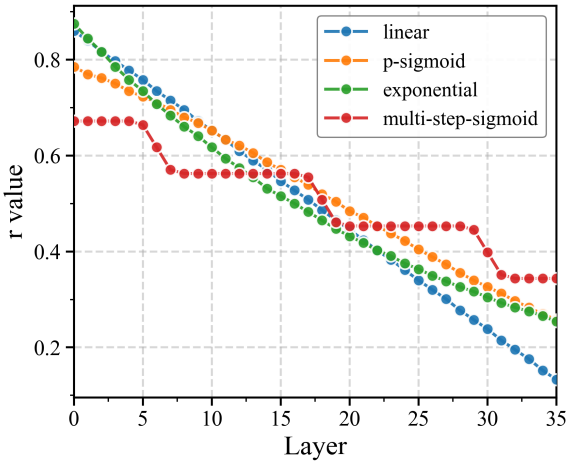

Figure 6: Comparison of layer-wise pruning ratios for different kernel functions under 50% computational budget. Linear kernel produces gradual transitions across layers, Exponential concentrates pruning in later layers, P-Sigmoid creates smooth S-shaped curves, and Multi-Step generates progressive discrete transitions. The diversity of patterns enables comprehensive exploration of different pruning strategies.

with larger values leading to smoother approximations. T is the temperature parameter for the straight-through estimator, balancing between gradient flow and discretization sharpness during training. lr denotes the learning rate for the AdamW optimizer, and B indicates the batch size (number of samples per training iteration).

C.2 More Experiment Results

We provide more detailed experimental results in this section. First, we present the results of different pruning methods under various computational budgets with **VisPCO** in Table 8. Second, we show the results of applying **VisPCO** to different base VLMs in Table 9. Third, we report the results of **VisPCO** with different pruning patterns in Table 10.

C.3 Case Studies of Predicted Pruning Configurations

We present case studies of pruning configurations predicted by **VisPCO** on Qwen2.5-VL-3B to provide insights into its behavior under different computational budgets. Figure 7 illustrates the layer-wise pruning curves predicted by **VisPCO** under various budget constraints, along with the corresponding visual token retention patterns at different layers. These visualizations reveal how **VisPCO** adaptively adjusts its pruning strategy in response to varying resource constraints.

The visualization reveals several key observations. First, as the computational budget becomes more constrained, **VisPCO** adopts increasingly aggressive pruning strategies, with pruning occurring earlier in the network and achieving lower retention ratios. This demonstrates the model's ability to adaptively allocate computational resources based on budget constraints. Second, the predicted configurations exhibit smooth transitions across layers, validating the effectiveness of our continuous relaxation approach.

Additionally, Figure 6 presents a comparison of different kernel functions (Linear, Exponential, P-Sigmoid, Multi-Step) for multi-layer pruning under a 50% computational budget. The layer-wise pruning ratios reveal distinct patterns: linear kernels produce gradual transitions, exponential kernels concentrate pruning in later layers, p-sigmoid kernels create smooth S-shaped curves, and multi-step kernels generate progressive discrete transitions. These diverse patterns enable **VisPCO** to explore different trade-offs.

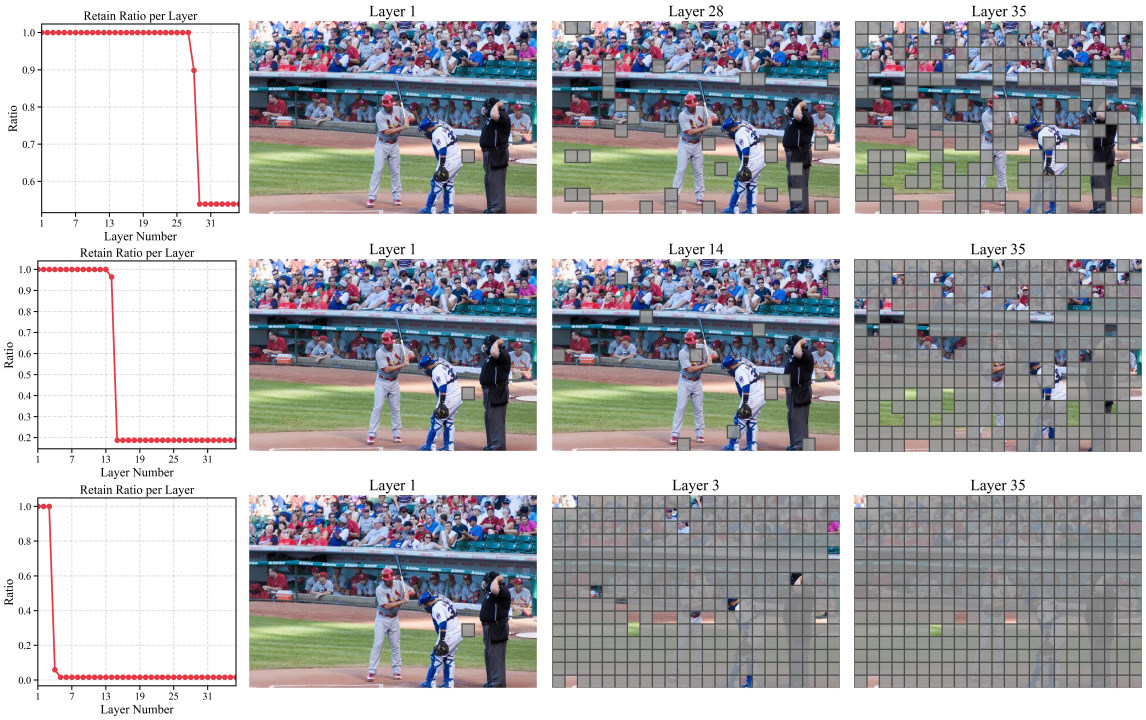

Figure 7: Layer-wise pruning configurations predicted by **VisPCO** under different computational budgets. The left panel shows the retention ratio curves across layers for budgets ranging from 10% to 90%. The right panel visualizes the actual visual token retention at selected layers (from top to bottom: 90%, 50%, 10% budgets), demonstrating how aggressive pruning (lower budgets) leads to earlier and more extensive token removal.

Table 8: Detailed comparison of pruning methods with and without **VisPCO** across eight vision-language benchmarks under different computational budgets. Results without **VisPCO** are averaged over multiple randomly sampled configurations that satisfy the budget constraint, with standard deviations reported in parentheses.

Method	AOKVQA	VizWiz	SEED	MMB	MME†	ChartQA	OCRB	TextVQA	Avg (%)
Upper Bound, 100% Budget, ∼3.56 TFLOPs									
Qwen2.5VL-3B	90.2	75.1	75.6	79.8	84.2	64.1	74.6	81.3	78.1
Reduce FLOPs Budget to 90%, ∼3.20 TFLOPs									
⌞ FastV	88.2 ± 0.4	72.9 ± 0.9	72.4 ± 0.9	76.4 ± 0.5	81.3 ± 0.5	62.2 ± 0.8	71.6 ± 0.7	79.1 ± 0.6	75.5 ± 0.7
+ VisPCO	88.4	73.8	73.2	76.9	81.7	62.9	72.3	79.5	76.1
⌞ SparseVLM	88.5 ± 0.3	73.1 ± 0.5	73.4 ± 0.4	76.9 ± 0.6	82.1 ± 0.3	62.2 ± 0.7	71.9 ± 0.6	79.5 ± 0.6	76.0 ± 0.5
+ VisPCO	88.6	73.5	73.8	77.5	82.4	62.9	72.5	80.0	76.4
⌞ FitPrune	89.1 ± 0.5	73.9 ± 0.4	74.2 ± 0.5	77.6 ± 0.4	82.5 ± 0.6	63.1 ± 0.6	72.5 ± 0.5	79.9 ± 0.3	76.2 ± 0.5
+ VisPCO	89.6	74.1	74.6	77.9	82.8	63.5	72.9	81.2	77.1
Reduce FLOPs Budget to 80%, ∼2.84 TFLOPs									
⌞ FastV	87.0 ± 1.6	71.6 ± 2.1	71.2 ± 2.1	75.1 ± 1.8	80.1 ± 1.6	61.0 ± 2.2	70.2 ± 1.9	77.7 ± 2.2	74.2 ± 1.9
+ VisPCO	88.3	73.7	73.1	76.9	81.6	62.8	72.0	79.4	75.7
⌞ SparseVLM	87.1 ± 1.8	71.7 ± 2.4	71.4 ± 2.1	75.3 ± 2.1	80.2 ± 2.2	61.6 ± 2.2	70.5 ± 2.1	78.3 ± 2.0	74.5 ± 2.1
+ VisPCO	88.4	73.9	73.5	77.4	82.1	62.6	72.4	79.6	76.2
⌞ FitPrune	87.3 ± 2.2	72.3 ± 2.1	72.5 ± 2.3	75.8 ± 2.2	80.6 ± 2.2	61.6 ± 2.5	70.6 ± 2.5	78.4 ± 1.6	74.9 ± 2.2
+ VisPCO	89.3	73.8	74.3	77.4	82.6	63.3	72.5	79.9	76.6
Reduce FLOPs Budget to 70%, ∼2.50 TFLOPs									
⌞ FastV	83.9 ± 4.5	69.2 ± 4.4	68.4 ± 4.8	73.1 ± 3.8	76.2 ± 4.5	59.3 ± 4.3	66.2 ± 4.9	75.5 ± 4.1	71.5 ± 4.4
+ VisPCO	88.0	73.4	72.8	76.7	80.9	62.4	71.1	78.9	75.5
⌞ SparseVLM	84.1 ± 4.7	69.4 ± 4.6	68.5 ± 5.1	73.3 ± 4.0	76.3 ± 4.5	59.5 ± 4.5	66.3 ± 5.0	75.7 ± 4.2	71.6 ± 4.6
+ VisPCO	88.1	73.6	72.9	76.7	81.0	62.5	71.1	79.1	75.6
⌞ FitPrune	84.2 ± 4.6	69.5 ± 4.7	68.6 ± 5.2	73.5 ± 4.1	76.6 ± 4.5	59.4 ± 4.7	66.5 ± 5.2	75.8 ± 4.3	71.8 ± 4.7
+ VisPCO	88.2	73.7	73.1	76.7	81.1	62.8	71.0	79.3	75.7
Reduce FLOPs Budget to 60%, ∼2.14 TFLOPs									
⌞ FastV	80.2 ± 5.9	68.4 ± 5.4	66.6 ± 6.8	69.4 ± 5.3	74.5 ± 5.5	56.4 ± 5.3	65.2 ± 5.1	71.7 ± 5.1	69.2 ± 5.4
+ VisPCO	86.0	71.4	70.8	74.7	78.9	60.4	69.1	76.9	73.5
⌞ SparseVLM	80.3 ± 6.2	68.5 ± 5.5	66.7 ± 7.1	69.6 ± 5.6	74.8 ± 5.8	56.4 ± 5.6	65.7 ± 5.5	71.8 ± 5.3	69.4 ± 5.7
+ VisPCO	86.2	71.6	70.9	74.9	79.2	60.6	69.3	77.1	73.7
⌞ FitPrune	80.4 ± 6.1	68.6 ± 5.4	66.8 ± 7.0	69.8 ± 5.8	74.9 ± 5.9	56.7 ± 5.9	65.9 ± 5.7	71.9 ± 5.3	69.5 ± 5.8
+ VisPCO	86.3	71.7	71.0	75.1	79.4	60.9	69.6	77.1	73.9
Reduce FLOPs Budget to 50%, ∼1.78 TFLOPs									
⌞ FastV	74.7 ± 10.1	60.3 ± 9.6	61.5 ± 8.1	62.4 ± 9.3	68.8 ± 9.1	51.6 ± 9.9	59.2 ± 9.1	65.9 ± 10.8	63.1 ± 9.5
+ VisPCO	84.8	69.4	67.6	71.2	77.1	58.1	67.8	75.9	71.5
⌞ SparseVLM	75.9 ± 9.8	62.6 ± 8.2	63.1 ± 7.2	63.9 ± 8.6	69.9 ± 8.2	51.9 ± 8.3	62.4 ± 6.9	66.6 ± 9.8	64.5 ± 8.4
+ VisPCO	85.2	69.0	68.1	71.9	77.6	58.4	67.9	76.3	71.8
⌞ FitPrune	77.1 ± 8.7	63.4 ± 7.7	63.9 ± 6.8	64.5 ± 8.2	70.8 ± 7.9	52.8 ± 8.1	63.3 ± 6.4	67.6 ± 9.4	65.4 ± 7.9
+ VisPCO	85.9	69.4	68.4	72.4	77.9	58.8	68.2	76.6	72.2
Reduce FLOPs Budget to 40%, ∼1.42 TFLOPs									
⌞ FastV	65.6 ± 12.2	50.3 ± 11.4	52.6 ± 10.2	53.3 ± 11.3	61.4 ± 11.2	42.5 ± 11.7	51.4 ± 11.2	57.5 ± 12.7	54.3 ± 11.5
+ VisPCO	77.6	61.7	62.6	64.4	72.6	54.1	62.5	69.9	65.7
⌞ SparseVLM	66.8 ± 12.6	50.9 ± 11.8	53.2 ± 10.9	53.5 ± 11.8	62.3 ± 11.3	42.9 ± 11.9	51.8 ± 11.3	58.1 ± 12.9	54.9 ± 11.8
+ VisPCO	77.9	61.9	62.7	64.9	72.9	54.4	62.6	70.2	65.9
⌞ FitPrune	66.9 ± 12.5	50.7 ± 12.2	53.0 ± 11.1	53.7 ± 11.9	62.7 ± 11.5	42.8 ± 12.2	51.7 ± 11.5	58.3 ± 12.6	55.0 ± 11.9
+ VisPCO	78.4	62.2	62.8	65.0	73.1	54.6	62.6	70.4	66.1
Reduce FLOPs Budget to 30%, ∼1.06 TFLOPs									
⌞ FastV	62.5 ± 8.1	46.4 ± 8.2	48.7 ± 7.6	49.4 ± 7.5	57.2 ± 8.9	38.8 ± 8.1	46.7 ± 8.3	53.6 ± 9.1	50.4 ± 8.2
+ VisPCO	70.2	54.6	55.5	54.5	65.7	46.4	54.4	62.8	58.0
⌞ SparseVLM	62.6 ± 8.3	46.5 ± 8.3	48.9 ± 7.7	49.5 ± 7.7	57.4 ± 9.0	38.9 ± 8.3	46.9 ± 8.5	53.8 ± 9.3	50.6 ± 8.4
+ VisPCO	70.5	54.8	55.9	54.8	65.8	46.9	55.1	63.4	58.4
⌞ FitPrune	62.7 ± 8.2	46.7 ± 8.2	49.0 ± 7.6	49.7 ± 7.9	57.3 ± 9.1	39.1 ± 8.1	46.7 ± 8.6	53.9 ± 9.4	50.6 ± 8.4
+ VisPCO	70.6	54.9	56.0	54.7	65.8	47.1	55.3	63.1	58.4
Reduce FLOPs Budget to 20%, ∼0.72 TFLOPs									
⌞ FastV	42.5 ± 4.1	39.4 ± 4.2	46.7 ± 3.6	39.4 ± 4.5	47.2 ± 4.9	34.8 ± 4.1	12.7 ± 5.3	43.6 ± 5.1	38.3 ± 4.5
+ VisPCO	46.6	43.1	50.1	43.9	51.2	38.9	17.8	48.5	42.5
⌞ SparseVLM	42.6 ± 4.2	39.5 ± 4.3	46.9 ± 3.7	39.6 ± 4.6	47.4 ± 5.0	34.9 ± 4.2	12.8 ± 5.4	43.7 ± 5.3	38.4 ± 4.6
+ VisPCO	46.7	43.2	50.2	43.9	51.3	39.1	17.9	48.6	42.6
⌞ FitPrune	42.5 ± 4.3	39.4 ± 4.4	46.7 ± 3.9	39.7 ± 4.7	47.6 ± 5.1	34.7 ± 4.1	12.6 ± 5.2	43.8 ± 5.4	38.3 ± 4.6
+ VisPCO	46.8	43.3	50.4	44.1	51.4	39.2	18.0	48.7	42.7
Reduce FLOPs Budget to 10%, ∼0.36 TFLOPs									
⌞ FastV	33.3 ± 2.3	30.4 ± 1.6	44.5 ± 2.7	33.0 ± 2.5	39.7 ± 1.4	29.8 ± 4.1	8.3 ± 2.1	33.7 ± 2.8	31.6 ± 2.4
+ VisPCO	35.5	31.7	46.9	35.5	40.1	33.2	10.1	36.1	33.6
⌞ SparseVLM	33.6 ± 2.1	31.2 ± 1.3	44.9 ± 2.5	33.9 ± 2.3	40.3 ± 1.1	30.5 ± 3.7	9.1 ± 2.0	34.4 ± 2.2	32.2 ± 2.2
+ VisPCO	35.5	31.5	47.1	35.8	40.4	33.3	10.2	36.3	33.8
⌞ FitPrune	33.8 ± 2.1	31.5 ± 1.1	45.3 ± 2.4	34.2 ± 2.2	40.6 ± 1.0	30.9 ± 3.5	9.6 ± 1.9	34.6 ± 2.1	32.6 ± 2.0
+ VisPCO	35.6	31.6	47.3	35.8	40.9	33.5	10.4	36.4	33.9

Table 9: Performance of **VisPCO** applied to different base vision-language models across eight benchmarks under various computational budgets. The results demonstrate the generalizability and effectiveness of **VisPCO** across different model architectures and sizes.

Method	AOKVQA	VizWiz	SEED	MMB	MME†	ChartQA	OCRB	TextVQA	Avg (%)
Upper Bound, 100% Budget									
LLaVA-7B	72.3	93.1	52.1	48.2	50.4	42.3	64.3	88.2	63.9
Gemma3-4B	80.1	61.2	69.9	72.3	79.3	53.8	64.2	70.3	68.9
Reduce FLOPs Budget to 90%									
LLaVA-7B	71.3 ± 0.5	91.5 ± 0.8	50.3 ± 0.6	47.3 ± 0.6	48.4 ± 0.8	40.9 ± 0.7	62.8 ± 0.7	86.2 ± 0.5	62.3 ± 0.7
+ VisPCO	71.8	92.3	50.8	47.7	49.2	41.5	63.5	86.5	62.9
Gemma3-4B	78.8 ± 0.5	59.8 ± 0.9	67.7 ± 0.7	70.5 ± 0.6	77.6 ± 0.9	51.8 ± 0.8	62.3 ± 0.7	58.8 ± 0.5	65.9 ± 0.7
+ VisPCO	79.3	60.5	68.4	71.1	78.5	52.5	62.9	59.2	66.6
Reduce FLOPs Budget to 80%									
LLaVA-7B	70.1 ± 1.6	89.7 ± 1.7	48.7 ± 1.3	45.4 ± 1.3	46.2 ± 1.7	38.5 ± 1.6	60.6 ± 1.7	84.1 ± 1.6	60.2 ± 1.9
+ VisPCO	71.7	91.4	49.9	46.7	47.6	40.0	62.3	85.6	61.9
Gemma3-4B	76.6 ± 1.4	57.9 ± 1.7	65.6 ± 1.6	68.4 ± 1.6	75.3 ± 1.8	49.8 ± 1.8	60.2 ± 1.6	56.6 ± 1.5	63.8 ± 1.6
+ VisPCO	77.8	59.5	67.2	69.9	77.0	51.5	61.8	57.9	65.3
Reduce FLOPs Budget to 70%									
LLaVA-7B	66.2 ± 4.7	85.7 ± 4.5	44.6 ± 4.3	41.3 ± 4.4	42.3 ± 4.6	34.6 ± 4.7	56.3 ± 4.5	80.3 ± 4.1	56.4 ± 4.5
+ VisPCO	70.7	89.3	48.9	45.7	46.6	39.1	60.7	84.4	60.7
Gemma3-4B	70.2 ± 5.1	51.7 ± 5.4	59.5 ± 5.5	62.3 ± 5.2	69.1 ± 4.7	43.7 ± 5.4	54.3 ± 5.4	50.6 ± 6.2	57.7 ± 5.4
+ VisPCO	75.3	56.9	65.1	67.5	73.5	58.9	59.7	56.7	64.2
Reduce FLOPs Budget to 60%									
LLaVA-7B	62.1 ± 6.6	82.9 ± 6.2	41.7 ± 6.4	38.4 ± 5.6	38.1 ± 7.0	31.6 ± 6.5	53.5 ± 6.1	76.1 ± 6.3	53.1 ± 6.3
+ VisPCO	68.1	88.3	47.5	43.6	44.1	37.0	59.6	81.3	58.7
Gemma3-4B	54.1 ± 6.1	46.6 ± 6.3	54.4 ± 6.6	57.2 ± 7.3	65.1 ± 6.8	38.6 ± 6.2	49.4 ± 6.5	45.6 ± 7.2	53.1 ± 6.6
+ VisPCO	60.2	52.9	60.9	64.1	71.1	44.3	54.9	51.3	57.5
Reduce FLOPs Budget to 50%									
LLaVA-7B	50.3 ± 10.7	70.8 ± 10.3	30.8 ± 11.3	26.5 ± 11.5	26.1 ± 11.4	21.3 ± 10.6	41.2 ± 11.6	64.2 ± 11.4	41.4 ± 11.1
+ VisPCO	60.8	80.5	42.1	38.0	37.4	31.6	52.8	75.2	52.3
Gemma3-4B	41.2 ± 12.4	33.5 ± 12.6	41.8 ± 11.5	44.7 ± 12.9	52.3 ± 11.8	25.7 ± 12.3	37.6 ± 12.7	33.6 ± 12.3	38.8 ± 12.3
+ VisPCO	53.4	45.9	53.1	57.3	62.8	37.9	50.3	45.5	50.8
Reduce FLOPs Budget to 40%									
LLaVA-7B	48.4 ± 8.6	68.5 ± 8.4	29.7 ± 8.8	24.1 ± 9.4	24.2 ± 9.8	19.5 ± 8.6	39.3 ± 9.4	62.7 ± 9.9	39.6 ± 9.1
+ VisPCO	56.8	76.9	38.4	33.5	34.0	27.9	48.7	71.7	48.5
Gemma3-4B	39.4 ± 8.1	31.1 ± 9.4	39.8 ± 9.7	42.8 ± 8.9	50.4 ± 9.5	23.6 ± 8.4	35.8 ± 8.9	31.7 ± 8.3	36.8 ± 8.9
+ VisPCO	47.5	40.1	48.7	51.5	59.9	31.4	44.2	39.9	45.4
Reduce FLOPs Budget to 30%									
LLaVA-7B	41.3 ± 6.7	61.6 ± 6.1	22.8 ± 6.5	17.2 ± 6.5	17.4 ± 6.9	14.6 ± 6.7	36.4 ± 6.3	60.8 ± 8.2	34.0 ± 6.7
+ VisPCO	47.8	67.7	29.2	23.6	24.2	20.9	42.5	68.9	40.6
Gemma3-4B	31.4 ± 6.1	24.1 ± 6.4	31.8 ± 6.7	36.8 ± 6.9	44.4 ± 6.5	20.6 ± 6.4	31.8 ± 6.9	25.7 ± 6.3	30.8 ± 6.5
+ VisPCO	37.5	30.1	38.5	43.5	50.2	27.0	38.3	31.9	37.1
Reduce FLOPs Budget to 20%									
LLaVA-7B	40.8 ± 2.8	60.9 ± 2.2	21.7 ± 2.6	17.1 ± 2.9	16.9 ± 3.1	13.8 ± 2.5	34.6 ± 2.5	60.1 ± 2.3	37.2 ± 2.6
+ VisPCO	43.6	63.1	24.3	19.9	19.8	16.3	37.1	62.4	39.8
Gemma3-4B	31.2 ± 2.2	23.8 ± 2.4	31.9 ± 2.2	36.4 ± 2.5	44.6 ± 2.6	20.3 ± 2.6	31.7 ± 2.3	25.6 ± 2.3	30.7 ± 2.4
+ VisPCO	33.4	26.2	34.0	38.9	47.2	22.9	34.0	27.9	33.1
Reduce FLOPs Budget to 10%									
LLaVA-7B	40.1 ± 0.7	60.8 ± 0.8	21.4 ± 0.6	17.2 ± 0.4	16.7 ± 0.2	13.5 ± 0.3	34.3 ± 0.9	60.2 ± 0.5	36.8 ± 0.6
+ VisPCO	40.7	61.5	22.0	17.6	16.8	13.8	35.0	60.6	33.5
Gemma3-4B	30.1 ± 0.3	22.8 ± 0.5	27.4 ± 0.6	33.3 ± 0.8	41.5 ± 0.7	19.2 ± 0.6	30.6 ± 0.6	23.5 ± 0.4	28.6 ± 0.6
+ VisPCO	30.4	23.2	27.9	34.1	42.2	19.7	31.2	23.9	29.1

Table 10: Performance comparison of **VisPCO** with different pruning patterns across vision-language benchmarks. The table shows detailed results for single-layer pruning and various multi-layer pruning strategies including Linear, Exponential, P-Sigmoid, and Multi-Step patterns.

Method	Kernels	AOKVQA	VizWiz	SEED	MMB	MME†	ChartQA	OCRB	TextVQA	Avg (%)
Upper Bound, 100% Budget, ∼3.56 TFLOPs										
Qwen2.5VL-3B	-	90.2	75.1	75.6	79.8	84.2	64.1	74.6	81.3	78.1
Reduce FLOPs Budget to 90%, ∼3.20 TFLOPs										
Single-Layer	-	88.4	73.8	73.2	76.9	81.7	62.9	72.3	79.5	76.1
Multi-Layer	Linear	88.2	73.1	72.5	76.4	81.3	61.8	72.0	79.1	75.6
	Exponential	87.9	72.8	72.3	76.4	81.3	61.7	71.9	79.0	75.4
	P-Sigmoid	87.5	72.3	72.1	76.1	81.0	61.3	71.8	78.6	75.1
	Multi-Step	88.5	73.9	73.3	76.9	81.9	62.9	72.2	79.8	76.2
Reduce FLOPs Budget to 80%, ∼2.84 TFLOPs										
Single-Layer	-	88.3	73.7	73.1	76.9	81.6	62.8	72.0	79.4	75.7
Multi-Layer	Linear	87.7	73.2	72.6	76.3	81.4	62.4	71.3	78.8	75.5
	Exponential	87.4	72.7	72.2	75.9	81.0	62.2	70.9	78.8	75.1
	P-Sigmoid	87.1	72.3	71.8	74.9	80.2	61.3	69.9	77.7	74.4
	Multi-Step	88.5	73.9	73.4	77.2	81.9	63.2	72.4	79.7	76.3
Reduce FLOPs Budget to 70%, ∼2.50 TFLOPs										
Single-Layer	-	88.0	73.4	72.8	76.7	80.9	62.4	71.1	78.9	75.5
Multi-Layer	Linear	87.4	72.8	72.2	76.3	80.5	62.1	70.8	78.3	75.1
	Exponential	87.2	72.5	72.0	76.1	80.4	61.8	70.6	78.2	74.9
	P-Sigmoid	86.8	72.3	71.8	75.7	80.1	61.6	70.4	78.0	74.6
	Multi-Step	87.7	72.9	72.5	76.6	80.8	62.0	70.6	78.7	75.2
Reduce FLOPs Budget to 60%, ∼2.14 TFLOPs										
Single-Layer	-	86.0	71.4	70.8	74.7	78.9	60.4	69.1	76.9	73.5
Multi-Layer	Linear	84.5	69.7	68.9	72.7	77.1	58.1	68.1	74.4	71.7
	Exponential	83.2	68.5	67.7	71.4	76.3	56.8	67.5	73.1	70.6
	P-Sigmoid	82.9	68.1	66.5	70.1	75.3	56.2	66.5	72.4	69.8
	Multi-Step	86.3	71.7	71.1	74.9	79.2	60.6	69.3	77.3	73.8
Reduce FLOPs Budget to 50%, ∼1.78 TFLOPs										
Single-Layer	-	84.8	69.4	67.6	71.2	77.1	58.1	67.8	75.9	71.5
Multi-Layer	Linear	82.6	67.7	65.4	70.9	76.2	57.5	66.5	74.9	70.2
	Exponential	82.2	67.3	65.1	70.4	75.3	57.1	65.5	74.4	69.7
	P-Sigmoid	81.9	67.0	64.8	69.6	74.7	56.8	65.2	74.1	69.3
	Multi-Step	84.9	69.5	68.6	71.8	77.9	59.2	68.5	76.7	72.1
Reduce FLOPs Budget to 40%, ∼1.42 TFLOPs										
Single-Layer	-	77.6	61.7	62.6	64.4	72.6	54.1	62.5	69.9	65.7
Multi-Layer	Linear	76.2	59.4	60.3	62.6	70.3	52.7	61.3	67.7	63.8
	Exponential	76.1	59.2	59.6	62.3	69.6	52.3	50.7	67.4	62.2
	P-Sigmoid	73.3	56.3	56.7	60.5	66.9	49.1	47.5	65.7	59.5
	Multi-Step	77.8	62.1	62.9	64.8	73.3	55.3	62.6	66.1	65.7
Reduce FLOPs Budget to 30%, ∼1.06 TFLOPs										
Single-Layer	-	70.2	54.6	55.5	54.5	65.7	46.4	54.4	62.8	58.0
Multi-Layer	Linear	65.5	49.6	50.8	49.3	61.3	41.8	49.4	57.8	53.2
	Exponential	65.3	49.5	50.6	49.2	60.9	41.6	49.3	57.7	53.0
	P-Sigmoid	60.9	45.3	47.2	44.7	55.9	38.3	45.9	52.4	48.8
	Multi-Step	71.3	55.7	56.2	55.6	66.2	47.6	55.6	63.2	58.9
Reduce FLOPs Budget to 20%, ∼0.72 TFLOPs										
Single-Layer	-	46.6	43.1	50.1	43.9	51.2	38.9	17.8	48.5	42.5
Multi-Layer	Linear	44.8	41.7	48.2	41.4	48.7	36.4	15.3	46.4	40.4
	Exponential	43.9	40.2	47.2	40.4	47.4	35.8	14.5	45.4	39.4
	P-Sigmoid	42.1	39.2	46.1	39.7	46.6	34.3	13.8	44.5	38.3
	Multi-Step	46.7	43.2	50.5	43.2	51.4	38.8	18.6	47.9	42.5
Reduce FLOPs Budget to 10%, ∼0.36 TFLOPs										
Single-Layer	-	35.5	31.7	46.9	35.5	40.1	33.2	10.1	36.1	33.6
Multi-Layer	Linear	35.2	31.4	46.7	35.2	39.8	32.9	9.9	35.7	33.6
	Exponential	34.4	30.9	46.4	34.9	39.1	31.9	9.3	35.2	32.8
	P-Sigmoid	34.6	31.3	46.8	35.2	39.4	32.3	9.8	35.5	33.1
	Multi-Step	35.4	31.1	46.3	35.3	39.7	32.8	10.0	35.8	33.3