FROM PATCHING TO PRUNING VISUAL COMPUTATION IN VISION–LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision–language models (VLMs) incur substantial inference cost because every visual token is processed by the attention and MLP projections of every decoder layer, even when token-specific visual computation is unnecessary at many depths. We introduce **Patch-to-Prune (P2P)**, inspired by Mechanistic Interpretability, a training-free framework that converts activation patching from a diagnostic tool into an inference-time computation bypass. **P2P** performs validation-guided forward and backward layer sweeps to identify decoder regions whose visual-token projection outputs can be replaced by fixed neutral proxy activation vectors within a user-specified accuracy tolerance. Unlike conventional token-pruning methods, **P2P** preserves the sequence length, token order, positional information, attention mask, and residual pathways, thereby pruning computation without removing tokens or modifying the pretrained model weights. We evaluate **P2P** on four VLMs from the Qwen2.5-VL and LLaVA families across seven multi-modal benchmarks using mutually disjoint calibration, validation, and test partitions. **P2P** at a 3% tolerance retains around **94%** of dense accuracy while reducing FLOPs by **55%**. Beyond these efficiency gains, our layer-wise analysis suggests that visual processing in VLMs is non-uniformly distributed across decoder depth: early and late layers often require little token-specific visual computation, whereas intermediate layers appear to perform most task-relevant visual integration, enabling later reasoning to rely largely on visual information already embedded in shared residual and textual representations. This makes **P2P** both an efficient inference framework and a causal lens into visual information processing in VLMs.

1 INTRODUCTION

Vision–language models (VLMs) have achieved superior performance across visual question answering, multimodal reasoning, object recognition, and spatial understanding. This capability, however, comes with substantial inference cost. An input image is typically represented by hundreds or thousands of visual tokens, which are processed repeatedly by every decoder layer. Since visual tokens typically dominate the input sequence, repeatedly applying the attention transformations makes the language decoder a major computational bottleneck in VLM inference.

Prior training-free acceleration methods like FastV Chen et al. (2024) and PyramidDrop Xing et al. (2024) tackle this problem mainly by pruning visual tokens: they estimate each token's importance from early attention or similarity scores, then permanently discard tokens judged less relevant. However, this approach ignores how visual information actually flows through the network, and it overlooks the fact that a token's importance in later layers may not decrease monotonically—a token that seems unimportant early on could become critical downstream. This limitation raises a complementary question: which layers are actually responsible for processing visual tokens?

To investigate this problem, we first perform a layer sensitivity analysis through activation patching, which replaces the output of a selected module with a reference activation to identify its causal significance. Our analysis reveals a pronounced non-uniformity: substitution of visual-token projection outputs is well tolerated in many early and late layers, whereas an intermediate band of layers exhibits markedly greater sensitivity to such substitution. This pattern suggests that redundancy within VLM decoders is not uniformly distributed, but is instead systematically structured across depth.

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

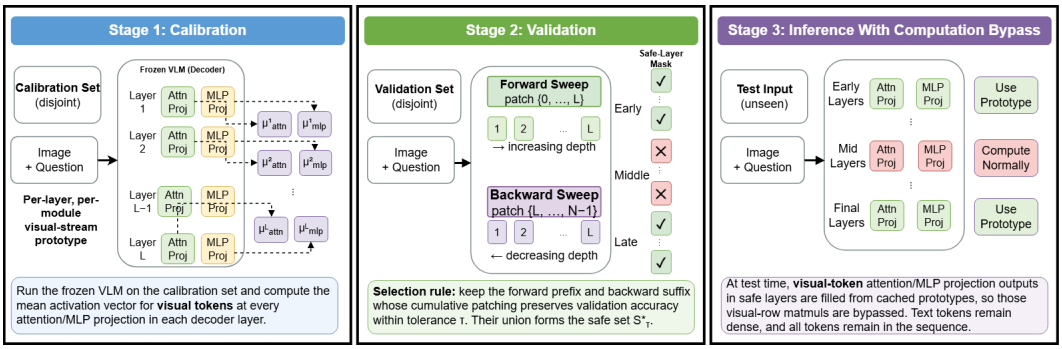


Figure 1: Framework of Patch-to-Prune

Based on our layer sensitivity analysis, we propose **Patch-to-Prune (P2P)** shown in Figure 1, an acceleration framework that converts this diagnostic intervention into an actual inference-time computation bypass. P2P first runs the frozen VLM on a small calibration set and constructs a modality-conditioned activation prototype for every decoder layer and linear module. It then performs cumulative forward and backward patching sweeps on a disjoint validation set. These sweeps identify the layers whose visual-token computations can be replaced by corresponding prototypes while remaining within a specified accuracy tolerance. The union of these selected regions defines the set of *pressable or safe layers* during inference.

At test time, P2P does not perform the full computations. Instead, for visual tokens in safe layers, their outputs of attention projection modules are directly filled with cached activation prototype, without actual computations. Only textual tokens in safe layers and all tokens in other layers go through the typical attention computations. Importantly, P2P prunes *computation* rather than tokens. Input embeddings of visual tokens remain in the sequence throughout the network, and the original sequence length, token order, attention mask, and residual pathways are preserved. This design avoids the irreversible token-removal decisions by conventional pruning methods and enables P2P to be applied directly to pretrained VLMs without modifying their parameters, changing their architecture, or performing additional training. Our main contributions are as follows:

- We introduce P2P, an acceleration framework that uses modality-conditioned activation patching to localize redundant visual-token computations across decoder depth.
- We present a validation-guided, tolerance-controlled layer-selection procedure that provides an interpretable accuracy–efficiency trade-off.
- We convert post-hoc activation replacement into an actual computation bypass that skips selected attention and MLP projection matmuls while preserving all tokens and the original sequence structure.
- We demonstrate consistent FLOP and latency reductions across four VLMs and seven multimodal benchmarks, and show that decoder redundancy is concentrated near the early and late layers of the network. We demonstrate that our P2P-aware finetuning can adapt the model directly under its patched computation graph and substantially recover the accuracy lost under aggressive pressing.

2 Related Work

Efficient vision–language model inference. VLMs commonly represent an image using hundreds or thousands of visual tokens, which substantially increases the computation cost. To address this, a family of methods reduces the visual tokens. PruMerge Shang et al. (2025) selects informative visual tokens and merges redundant tokens into the retained representations. PyramidDrop Xing et al. (2024) extends the idea of FastV **?** with a progressive schedule that partitions the decoder into stages and removes a fraction of the surviving visual tokens at each stage. SparseVLM Zhang et al. (2024) uses text-guided attention to rank visual tokens, selects layer-dependent sparsification ratios, and recycles information from discarded tokens. Collectively, these methods demonstrate that substantial computation can be eliminated.

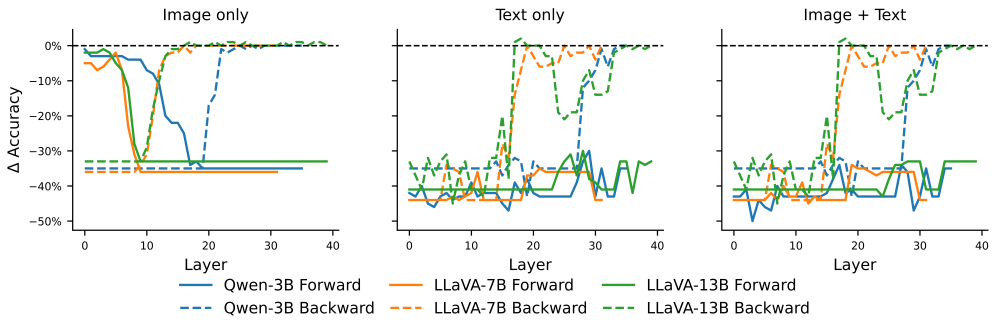


Figure 2: Layer-wise sensitivity on POPE under cumulative forward and backward mean patching of image, text, and joint image–text token streams.

Activation patching and causal localization. Activation patching, also referred to as interchange intervention or causal tracing, determines the causal importance of an internal model component by replacing its activation and measuring the resulting behavioral change Geiger et al. (2021); Zhang & Nanda (2024). It has been used to localize information processing to particular layers, token positions, attention heads, and intermediate representations in language models. Prior work has also studied methodological choices for patching and scalable approximations that reduce the cost of evaluating large numbers of interventions Kramár et al. (2024); Heimersheim & Nanda (2024). These methods are primarily interpretability tools: the original computation is normally executed before its activation is overwritten, so the intervention itself does not accelerate inference.

3 Layer Sensitivity Analysis

To identify which layers are actually responsible for processing vision and text tokens when the model answers questions, we first perform a layer sensitivity analysis.

3.1 Notations

Let $\mathcal{M}$ be a frozen, instruction-tuned VLM whose language backbone consists of N decoder layers. Given an image–question pair, the processor produces a token sequence of length T that interleaves projected visual tokens with text tokens. Each decoder layer $\ell \in \{0, \dots, N-1\}$ contains a set of linear projections $\mathcal{S} = \{W^Q, W^K, W^V, W^O, W^{\text{gate}}, W^{\text{up}}, W^{\text{down}}\}$, i.e., the four attention projections and the three gated-MLP projections. For a module $m \in \mathcal{S}$ at layer ℓ, we write $a_{\ell,m}(t) \in \mathbb{R}^{d_m}$ for its output activation at token position t, where d_m is the module's output length.

3.2 Stream-Conditional Activation Statistics

To identify the layer sensitivity on tokens, we replace the layer outputs with example-level mean activations. Specifically, on a statistics split $\mathcal{D}_{\text{stat}}$ of N_{stat} training examples (disjoint from all evaluation data), for every layer ℓ, module m, and stream $s \in \{\text{img}, \text{txt}\}$, we run the frozen model and record the example-level mean activation

$$\mu_{\ell,m,s} = \frac{1}{N_{\text{stat}}} \sum_{n=1}^{N_{\text{stat}}} \frac{1}{|\mathcal{P}_s^{(n)}|} \sum_{t \in \mathcal{P}_s^{(n)}} a_{\ell,m}^{(n)}(t). \tag{1}$$

Note that here each example contributes equally regardless of its token count, which prevents long sequences (or large images) from dominating the estimate.

3.3 Cumulative Layer Intervention

To localize the causally load-bearing depth band, we run *cumulative* sweeps on a held-out validation split. These sweeps are done on three setups of streams: `image_only` (image positions ← image

means), `text_only` (text positions ← text means), and `dual` (both, same-stream). We perform sweep from two directions with patch layers: (i) **Forward:** patch layers $\mathcal{L}_{\text{fwd}}(L) = \{0, \ldots, L\}$ with L starting from 0 and growing from the input side; and (ii) **Backward:** patch layers $\mathcal{L}_{\text{bwd}}(L) = \{L, \ldots, N-1\}$ with L starting from $N-1$ and decreasing from output. For each value of L during each direction in each stream, we replace the module outputs of patch layers with the example-level mean activation, and record the corresponding accuracy drop on this split. Each sweep point re-registers the intervention on the specific layer set and these points localize how much *token-specific* information each modality contributes at a given depth.

3.4 Observations and Insights

Benchmarks. We evaluate on POPE (tests object-recognition), WhatsUp (tests spatial-relation), and ScienceQA-IMG (image-dependent scienctific knowledge). For each dataset, from the training set, we randomly draw a statistics split $\mathcal{D}_{\text{stat}}$ (with $N_{\text{stat}} = 100$ examples) used exclusively for Eq. 1, and another validation split (100 examples) to perform sweeps. These splits and the final test set are three mutually disjoint sets.

Models and sweep grid. We use Qwen2.5-VL-3B-Instruct and LLaVA-Next (Vicuna-7B and Vicuna-13B). The localization grid crosses $\{\text{model}\} \times \{\text{dataset}\} \times \{\text{forward, backward}\} \times \{\texttt{image_only}, \texttt{text_only}, \texttt{dual}\}$ with mean patching for each value of L.

Observations. Figure 2, Figure 6 (Appendix A.2) and Figure 7 (Appendix A.2) reveal a consistent depth-dependent pattern: replacing visual-token projection outputs is often tolerated across substantial early and late decoder regions (demonstrated by the small accuracy drop of a few first layers in the forward sweep or a few last layers in the backward sweep), whereas extending the intervention into intermediate layers causes markedly larger accuracy degradation. In contrast, patching text-token computations remains broadly damaging, indicating that this redundancy is primarily associated with the visual stream. We hypothesize that visual information follows a staged computational pathway. Early layers begin encoding visual features, intermediate layers perform the primary task-relevant integration into the shared residual stream and textual representations, and later layers generate the answer largely from information already embedded in these representations. This motivates replacing unnecessary visual-token projection computations at pressable depths with a modality-conditioned fixed proxy activation. This preserves the average contribution of each layer while bypassing token-specific computation.

4 Patch-to-Prune Framework

4.1 Overview

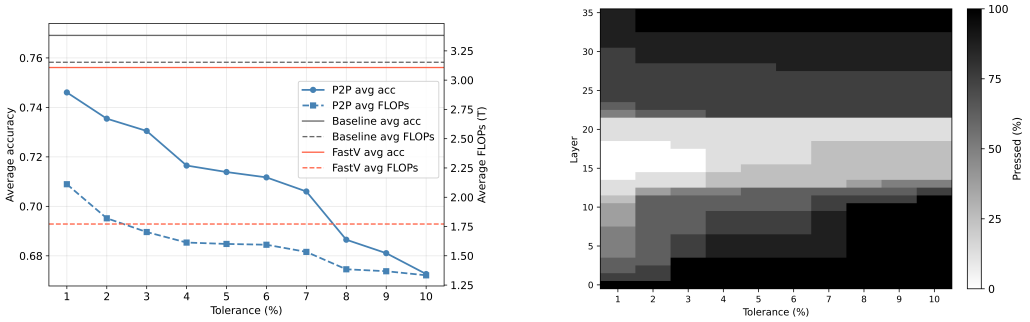
We propose **Patch-to-Prune (P2P)**, a training-free method for reducing the inference cost of VLMs. P2P framework consists of three stages: (i) layer sensitivity analysis with mechanistic profiling of model's dependence on visual computation at each projection module, (ii) validation-guided safe-layer selection, and (iii) computation-bypassed inference.

4.2 Layer Sensitivity Analysis

As discussed in Section 3, P2P first performs layer sensitivity analysis with activation patching to identify decoder layers in which visual token computations can be replaced without causing a large decrease in task accuracy. It profiles the accuracy drop under various patching layer configurations during forward and backward sweeps.

4.3 Safe-Layer Selection

Based on the profiling of the accuracy drop under various patching layer configurations, P2P then selects safe layers. Specifically, given a tolerance τ, we find an admissible patch layer index L such that its cumulative patch preserves validation accuracy, i.e., $\text{Acc}_d(L) \geq (1-\tau)\,\text{Acc}_0$. We search L^*_{fwd} and L^*_{bwd} for the forward and backward sweeps respectively. In the forward search, we increase L gradually until we find two consecutive *inadmissible* boundaries. Then the forward frontier is set



(a) Accuracy and FLOPs as P2P prunes image-stream computation in more decoder layers under increasing tolerance, compared with Vanilla and FastV.

(b) Layer-wise image-stream pruning configurations selected by cumulative forward and backward validation sweeps across datasets and tolerance levels.

Figure 3: Tolerance sensitivity analysis.

to $L^*_{\text{fwd}} = L-1$. The backward search mirrors this from the output side with decreasing L. The final safe set is the union of the resulting prefix and suffix, $\mathcal{S}^*_\tau = \{0, \ldots, L^*_{\text{fwd}}\} \cup \{L^*_{\text{bwd}}, \ldots, N-1\}$, which are fixed after selection and applied unchanged at test time. The layers in $\mathcal{S}^*_\tau$ are safe to remove under a tolerance τ.

4.3.1 Tolerance Sensitivity.

The tolerance τ is the single control parameter of P2P: relaxing it admits more boundaries under the selection rule above, expanding the safe set inward from both ends of the decoder. To characterize this trade-off on Qwen2.5-VL-3B-Instruct, we sweep τ from 1% to 10%, re-running safe-layer selection on the validation split at each setting and evaluating the resulting configuration on the full test set. Figure 3(a) shows that increasing τ produces a consistent accuracy–efficiency trade-off: average accuracy falls from 74.6% at 1% to 70.6% at 7% and 67.1% at 10%, while average computation drops from 2.1 to 1.3 TFLOPs against a 3.1 TFLOP dense reference. Figure 3(b) providing the layer-wise view explains this: pressed safe layers first appear at the top of the decoder and near the input, and as τ grows they expand inward toward the sensitive intermediate band (approximately layers 14–21), which is precisely where accuracy begins to break down. We therefore adopt $\tau = 3\%$ as the default operating point with a better trade-off between accuracy and computations.

4.4 Computation-Bypassed Inference

Once the pressed set $\mathcal{S}^*_\tau$ is fixed by safe-layer selection, the diagnostic intervention is converted into an actual computation bypass. At the selected safe layers, P2P skips the multiplication with attention and the three MLP modules for visual tokens at test time. Their module outputs are filled directly with the cached prototype $\mu^v_{l,m}$ discussed in Section 3. Only the remaining textual tokens go through the original attention computations with the original weights.

4.5 P2P-Aware Fine-Tuning

We propose an optional P2P-aware finetuning under P2P-patched computation graph for accuracy.

Fine-tuning under mean patching. For the safe layers $\mathcal{S}^*_\tau$ identified as safe under an accuracy tolerance τ, we insert LoRA adapters into the attention and MLP projections of these layers. To mitigate the training and test mismatch problem, we use the same P2P-patched computation graph during training. Specifically, for visual tokens in safe layers, we skip their computations in projection modules and directly use the cached mean activation in Eq. 1 to replace their original outputs. Thus the corresponding projection computation is bypassed and no gradient is propagated through the LoRA adapter for those patched positions. The adapters are instead optimized through the remaining active computation, including unpatched textual tokens and modules outside the patched layers. This

encourages the adapters to compensate for the information removed by P2P rather than relying on activations that are unavailable during patched inference.

Training objective. We optimize adapters using teacher-forced causal language modeling. The multimodal input prompt is masked from the loss, and cross-entropy is computed only over the ground-truth assistant-answer tokens:

$$\mathcal{L} = -\sum_{j \in \mathcal{Y}} \log p^{\mathrm{P2P}}_{\boldsymbol{\theta}_0, \boldsymbol{\theta}_{\mathrm{LoRA}}} \left(y_j \mid \mathbf{x}, y_{<j} \right), \tag{2}$$

where $\mathcal{Y}$ denotes the answer-token positions, and $\boldsymbol{\theta}_{\mathrm{LoRA}}$ denotes LoRA parameters. Consequently, P2P-aware fine-tuning adapts the model directly to its patched execution with less training efforts. We further discuss the distinguishing advantages of P2P in Appendix A.1.

5 Evaluation

5.1 Experiment Setup

Datasets. We evaluate P2P on seven benchmarks spanning diverse multimodal capabilities:**WhatsUp-VLM** Kamath et al. (2023),**POPE** (pope) Li et al. (2023) , **ScienceQA-IMG** (sqa) Lu et al. (2022), **GQA** (gqa) Hudson & Manning (2019), **TextVQA** (tvqa) Singh et al. (2019), **CV-Bench** (cvb) Tong et al. (2024), and **AI2D** (ai2d) Kembhavi et al. (2016). For each benchmark, we use 100 samples for proxy estimation and a disjoint set of 100 samples for layer selection. We set the proxy-estimation size to 100 based on the ablation in Figure 5. Finally, we evaluate on a large held-out test set. More details are in Appendix A.6.

Models. We evaluate P2P on four VLMs from two model families, **Qwen** Bai et al. (2023); Wang et al. (2024); Team (2025) and **LLaVA** Liu et al. (2023; 2024a;b), spanning parameter scales of 3B, 7B, and 13B. Specifically, we use Qwen2.5-VL-3/7B-Instruct, and LLaVA-v1.6-Vicuna-7/13B-HF.

Baselines. We compare P2P with the dense model and two training-free visual-token pruning methods, FastV Chen et al. (2024) and PyramidDrop Xing et al. (2024), implemented within the same evaluation framework using identical models, prompts, precision, test examples, and metrics. The Vanilla baseline applies no token removal, activation replacement, or layer removal. We include token pruning baselins (FastV and PyramidDrop) not as competitors but to put P2P in perspective: obviating per-layer visual computation matches the efficiency of token pruning, and opens a complementary direction—pruning tokens *within* the vital layers P2P preserves.

P2P-aware finetuning protocol. For main results, we do not finetune the model to make a fair comparison. In ablation study, we further demonstrate that our finetuning can improve the performance effectively. Our finetuning trains on the 100-sample statistics split that built the mean prototypes and validates on the 100-sample split that produced the saliency profiles, adding no data beyond the training-free pipeline. We train with batch size 1, LoRA rank $r = 1$, scaling factor $\alpha = 1$, and AdamW with a learning rate of 2×10^{-4}. Training is performed for 10 epochs.

5.2 Main Result for P2P

From mechanistic discovery to full-test acceleration. P2P tests whether visual-computation redundancy discovered from small, disjoint subsets generalizes to the full held-out test set. We evaluate three operating points: a conservative 1% tolerance, the default 3% tolerance, and a more aggressive 5% tolerance, which progressively expand the pressed regions to trade accuracy for greater computation reduction.

Accuracy–computation trade-off. Table 1 shows that at the conservative 1% tolerance, P2P preserves most of the Vanilla model accuracy while already removing a substantial amount of computation. For LLaVA-7B, average accuracy (computed from Table 1) decreases only from 0.641 to 0.626, retaining 97.7% of Vanilla accuracy, while average FLOPs decrease from 30.8 to 17.5 T, corresponding to a 43.2% reduction. For LLaVA-13B, P2P retains 98.1% of Vanilla accuracy, with

Table 1: Absolute accuracy, FLOPs (teraFLOPs), and average inference latency (seconds per sample) for LLaVA-7B and 13B across seven benchmarks. We compare Vanilla inference, FastV, PyramidDrop, and P2P at 1%, 3%, and 5% tolerance levels.

Method	Metric	Tol.	wco	wvg	pope	sqa	gqa	tvqa	cvb	ai2d
LLaVA-v1.6-Vicuna-7B										
Vanilla	Acc.	–	0.576	0.58	0.848	0.647	0.639	0.601	0.598	0.642
	FLOPs (T)		32.6	30	32.5	23.6	32.2	34	31.2	30.1
	Latency (s)		0.218	0.199	0.22	0.163	0.531	1.07	0.213	0.201
FastV	Acc.	–	0.559	0.561	0.842	0.646	**0.636**	0.596	0.605	0.634
	FLOPs (T)		17.1	15.8	16.9	12.6	16.7	18	16.4	15.8
	Latency (s)		0.241	0.214	0.235	0.164	0.561	1.16	0.231	0.213
PDrop	Acc.	–	**0.567**	0.572	**0.849**	**0.65**	0.632	**0.599**	**0.606**	0.636
	FLOPs (T)		15.4	14.2	15.1	11.3	**15**	**16.2**	14.8	14.2
	Latency (s)		0.233	0.209	0.23	0.163	0.55	1.13	0.224	0.21
P2P	Acc.	1%	0.522	**0.635**	0.848	0.646	0.63	0.534	0.556	0.64
	FLOPs (T)		10	14.1	32.5	12.3	24	20.9	9.55	16.6
	Latency (s)		0.176	0.17	0.22	0.15	0.467	**0.924**	0.166	**0.175**
P2P	Acc.	3%	0.511	0.574	0.831	0.624	0.598	0.524	0.506	**0.641**
	FLOPs (T)		**3.69**	10.8	16	7.67	21.3	19	6.96	15.8
	Latency (s)		**0.161**	**0.168**	0.184	0.146	**0.447**	0.927	0.167	0.176
P2P	Acc.	5%	0.511	0.565	0.776	0.57	0.581	0.454	0.505	0.627
	FLOPs (T)		**3.69**	**8.27**	**14.2**	**2.38**	16.7	16.2	**6.09**	**13.3**
	Latency (s)		0.164	0.179	**0.182**	**0.144**	0.815	1.38	**0.163**	0.281
LLaVA-v1.6-Vicuna-13B										
Vanilla	Acc.	–	0.754	0.744	0.821	0.675	0.646	0.622	0.605	0.676
	FLOPs (T)		62.7	57.7	62.5	45.7	61.9	65.5	60.1	58
	Latency (s)		0.383	0.35	0.386	0.289	0.94	1.88	0.372	0.357
FastV	Acc.	–	0.736	0.739	0.807	0.669	0.641	0.615	0.603	0.675
	FLOPs (T)		32.8	30.3	32.3	24.2	32	34.4	31.5	30.4
	Latency (s)		0.367	0.326	0.358	0.255	0.864	1.78	0.353	0.329
PDrop	Acc.	–	0.735	0.736	**0.819**	0.67	**0.646**	**0.621**	**0.606**	**0.677**
	FLOPs (T)		29.8	27.5	29.3	22	**28.9**	**31.3**	28.6	27.6
	Latency (s)		0.361	0.327	0.352	0.253	0.855	1.76	0.348	0.328
P2P	Acc.	1%	**0.762**	**0.769**	0.815	**0.674**	0.594	0.613	0.537	0.676
	FLOPs (T)		49.9	51	28.2	22.7	32	62.5	14	58
	Latency (s)		0.344	0.331	0.291	0.239	0.698	1.79	0.248	0.357
P2P	Acc.	3%	0.726	0.748	0.811	0.661	0.594	0.564	0.537	0.613
	FLOPs (T)		18.8	19.8	18.2	18.6	32	39	14	18.5
	Latency (s)		0.27	0.257	**0.266**	**0.227**	0.698	1.48	0.247	0.255
P2P	Acc.	5%	0.675	0.664	0.758	0.651	0.55	0.56	0.392	0.588
	FLOPs (T)		**16**	**17.2**	**15.3**	**13.4**	29.2	36.1	**7.21**	**15.9**
	Latency (s)		**0.261**	**0.255**	0.323	2.11	**0.681**	**1.44**	**0.235**	**0.248**

average accuracy changing from 0.693 to 0.680, while reducing FLOPs from 59.2 to 39.8 T, a 32.8% reduction. The default 3% tolerance provides a stronger accuracy–efficiency trade-off. Table 1 shows that on LLaVA-7B, P2P reduces average FLOPs from 30.8 to 12.6 T, a 59.1% reduction, while retaining 93.8% of Vanilla average accuracy. On LLaVA-13B, average FLOPs decrease from 59.2 to 22.4 T, a 62.2% reduction, while retaining 94.8% of Vanilla accuracy. Averaged across the two model sizes, P2P at 3% therefore retains approximately 94% of Vanilla accuracy while reducing the reported FLOPs by approximately 60%. Increasing the tolerance to 5% further reduces average FLOPs to 10.1 T for LLaVA-7B and 18.8 T for LLaVA-13B, corresponding to reductions of 67.2% and 68.2%, respectively. However, the average accuracies decrease to 0.574 and 0.605. The tolerance therefore controls a genuine trade-off: relaxing it expands the pressed prefix and suffix toward task-sensitive intermediate layers, yielding greater computation reduction but also increasing the risk of removing task-relevant visual processing. More results on Qwen models are in Table 3 of Appendix A.3.

Comparison with the baseline pruning. At the default 3% setting, P2P reaches a substantially lower computation regime than FastV and PyramidDrop. For LLaVA-7B, P2P requires 12.6 T

FLOPs on average, compared with 16.2 T for FastV and 14.5 T for PyramidDrop. For LLaVA-13B, P2P requires 22.4 T, compared with 31.0 T for FastV and 28.1 T for PyramidDrop. P2P therefore reduces average FLOPs by an additional 22.2% relative to FastV and 13.1% relative to PyramidDrop on LLaVA-7B, and by 27.7% and 20.3%, respectively, on LLaVA-13B.

This more aggressive computation reduction is accompanied by lower average accuracy than the token-pruning baselines. At 3%, P2P obtains average accuracies of 0.601 and 0.657 on LLaVA-7B and LLaVA-13B, whereas FastV obtains 0.635 and 0.686, and PyramidDrop obtains 0.639 and 0.689. Thus, P2P provides a distinct operating point rather than uniformly dominating baselines.

Wall-clock acceleration. The reduction in visual-token projection operations translates into measurable wall-clock improvements at the default operating point. In Table 1, on LLaVA-7B, P2P at 3% reduces average latency by 15.4%: from 0.351 to 0.297 seconds per sample, whereas FastV and PyramidDrop show higher latency than Vanilla despite their FLOP reductions. On LLaVA-13B, average latency decreases from $0.619s$ to $0.462s$, a 25.4% reduction. P2P is also faster than both token-pruning baselines at this setting: FastV and PyramidDrop require $0.377s$ and $0.369s$ on LLaVA-7B and $0.578s$ and $0.573s$ seconds on LLaVA-13B, respectively.

These results are notable since lower theoretical FLOPs need not translate into lower latency. Token-pruning baselines add ranking, indexing, and sequence-reorganization overhead, whereas P2P directly skips selected visual-row projection matmuls. Yet P2P is not strictly monotonic either: because each patched layer carries a fixed patching cost, a higher tolerance can reduce FLOPs while increasing latency, so the lowest-FLOP configuration is not always the fastest. Overall, the 3% configuration therefore provides the strongest overall balance among accuracy, FLOPs, and observed wall-clock latency in Table 1.

Generalization across tasks. The full-test results also show that the amount of pressable visual computation is task dependent. At 1%, P2P improves LLaVA-7B accuracy on WhatsUp-VGQA from 0.580 to 0.635. For LLaVA-13B, it improves WhatsUp-COCOQA from 0.754 to 0.762 and WhatsUp-VGQA from 0.744 to 0.769. At 3%, LLaVA-7B nearly preserves its dense AI2D accuracy, obtaining 0.641 compared with 0.642. Other tasks, particularly TextVQA and CV-Bench, do not collapse but are more sensitive to aggressive pressing. We note that the full-test accuracy drop can, on some datasets, exceed the specified tolerance. This motivates a fine-tuning setup that tests whether lightweight LoRA adaptation can compensate for that sensitivity.

Overall, despite this dataset-specific variation, the pressable regions identified yield on an average over all four models and 8 datasets a 55% FLOPs reduction and 22% latency reduction while retaining 94% of dense accuracy—consistent computation savings without collapse, confirming that lightweight mechanistic discovery and activation replacements generalize to large-scale evaluation.

5.3 Ablation Study

Proxy-estimation choice. We use example-level mean activation as the proxy estimation. To justify this choice, we perform ablation by using different replacements: the zero vector (`zero`); Gaussian noise with per-dimension scale derived from $\sigma_{\ell,m,s}$ or a fixed override (`noise`); or a single real token activation drawn uniformly from the cached bank (`random_token`). The results in Figure 4 show that our mean activation leads to the best performance.

Proxy-estimation set size ablation. We further study how many examples are required to estimate the proxy activations. We vary the statistics-set size over $N_{\text{stat}} \in \{1, 25, 50, 100, 250, 500\}$ while keeping the 100-example validation sweep and all other settings unchanged. Our results in Figure 5 show that 100 statistics examples provide a stable proxy efficiently.

Accuracy recovery with P2P-aware fine-tuning. Table 2 compares the frozen dense model, denoted *Vanilla*, with frozen P2P and P2P-aware LoRA. P2P-aware fine-tuning consistently improves over the corresponding frozen P2P configuration across ScienceQA, TextVQA, and CV-Bench while preserving the same patched inference graph. At 1%, P2P-aware LoRA recovers approximately 75% of the average accuracy gap introduced by pressing, 3% recovers approximately 100%, and 5% recovers approximately 59%. The strong recovery makes 3% the default tolerance. P2P-aware LoRA improves accuracy of ScienceQA from 0.624 to 0.629, TextVQA from 0.524 to 0.589, and

Table 2: Full-test accuracy for LLaVA-7B comparing frozen Vanilla, frozen P2P, and P2P+LoRA fine-tuned under mean patching at 1%, 3%, and 5% tolerances

Method	Tol.	sqa	tvqa	cvb
Vanilla	–	0.647	0.601	0.598
P2P	1%	0.646	0.543	0.556
P2P+LoRA	1%	0.665	0.59	0.566
P2P	3%	0.624	0.524	0.506
P2P+LoRA	3%	0.629	0.589	0.628
P2P	5%	0.570	0.454	0.505
P2P+LoRA	5%	0.584	0.544	0.587

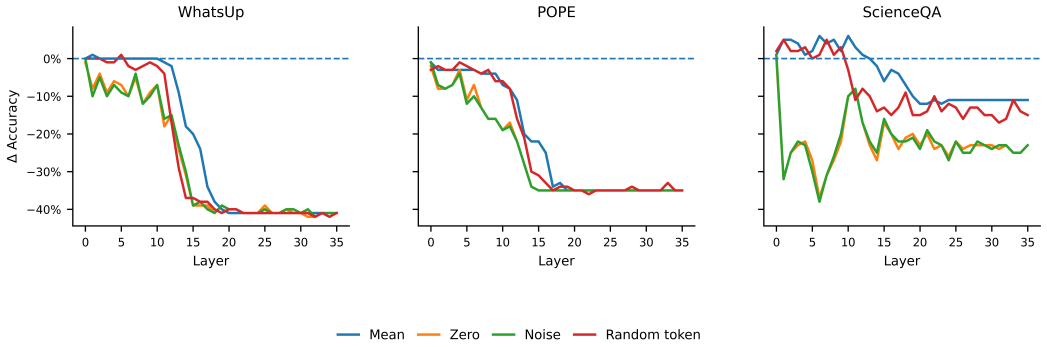


Figure 4: Patch-type ablation using cumulative forward pruning of image-stream attention and MLP computations with mean, zero, noise, or random-token activation replacement.

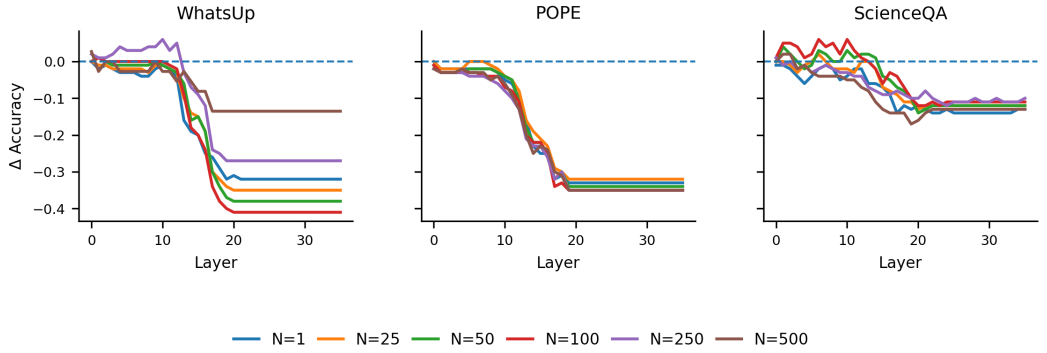


Figure 5: Proxy-estimation set-size ablation using cumulative forward mean patching of image-stream attention and MLP computations with 1, 25, 50, 100, 250, or 500 statistics examples.

CV-Bench from 0.506 to 0.628. P2P shows that diagnostic proxy activations serve as practical substitutes for visual-token computation—effective training-free, and stronger with LoRA adaptation.

6 Conclusion

P2P turns activation patching from a diagnostic tool into a practical inference mechanism: by identifying decoder depths where visual-token projection outputs can be safely replaced with fixed proxies, it bypasses unnecessary computation without removing tokens or disrupting the original sequence structure. Beyond its efficiency gains, P2P reveals that visual processing is concentrated in a task-sensitive intermediate region, while substantial early- and late-layer computation is redundant. This depth-aware perspective opens a promising direction for token pruning, where future methods can decide not only *which* visual tokens matter, but also *where* their computation is truly necessary.

AI USE STATEMENT

In this work, we used generative AI tools for editing paper (clarity, grammar, organization, and presentation), preparing the software code and the latex of this paper. We have not used generative AI tools for algorithm development, idea generation or formation. Algorithm and idea are fully original. We have reviewed all parts of the work that had AI responses.

REFERENCES

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 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*, pp. 19–35. Springer, 2024.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. *Advances in neural information processing systems*, 34:9574–9586, 2021.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*, 2024.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6700–6709, 2019.

Amita Kamath, Jack Hessel, and Kai-Wei Chang. What's "up" with vision-language models? investigating their struggle with spatial reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9161–9175, 2023.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pp. 235–251. Springer, 2016.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. Atp*: An efficient and scalable method for localizing llm behaviour to components. *arXiv preprint arXiv:2403.00745*, 2024.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 292–305, 2023.

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

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

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llavanext: Improved reasoning, ocr, and world knowledge, 2024b.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22857–22867, 2025.

Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8317–8326, 2019.

Qwen Team. Qwen2.5-vl, January 2025. URL https://qwenlm.github.io/blog/qwen2.5-vl/.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai C Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356, 2024.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

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

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, volume 2024, pp. 1651–1678, 2024.

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