PRUNE LESS, PAY MORE: IMPERCEPTIBLE BUDGET INFLATION ATTACKS ON TOKEN PRUNING IN VISION-LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The growing number of visual tokens processed by Vision-Language Models (VLMs) increases inference costs, motivating token pruning and compression techniques that reduce computational demand. Increasingly, these techniques adapt the number of retained tokens to the complexity of each input. The count is determined by statistical thresholds, learned importance predictors, or image-complexity estimators. In this work, we show that this flexibility introduces a hidden opportunity for adversarial cost amplification: an imperceptibly perturbed image can cause a VLM to retain substantially more visual tokens than its clean counterpart while preserving task performance. Consequently, users may incur greater resource consumption without noticing changes in either the image's appearance or the quality of the model's response. Unlike existing attacks that manipulate token selection within a fixed budget to degrade accuracy, our attack targets the adaptive token budget itself. We propose the Budget Inflation Attack (BIA), which optimizes imperceptible image perturbations to induce content-adaptive compressors to retain more tokens, undermining their intended efficiency benefits. Across multiple adaptive pruning mechanisms and datasets, BIA increases retained visual token counts by significantly while leaving task accuracy essentially unchanged. This token inflation can increase inference computation, latency, and energy consumption. More importantly, it may translate into higher user charges when billing reflects the additional processing. Our findings reveal that adaptive visual token pruning creates an overlooked attack surface: adversaries can erode efficiency and consume resource budgets while preserving the apparent utility of the system. Our code is available at https://anonymous.4open.science/r/PLPM/README.md.

1 INTRODUCTION

Vision-Language Models (VLMs) have gained significant attention for their ability to jointly understand visual inputs and natural-language queries, supporting applications such as visual question answering, document understanding, and image description (Bai et al., 2023; Li et al., 2023; Liu et al., 2023). A typical VLM uses a vision encoder to transform an image into visual features represented as a sequence of visual tokens, which are subsequently processed by a language model (Li et al., 2026). Processing a larger number of visual tokens increases computation, memory consumption, and inference latency. Visual-token pruning techniques address this cost by removing redundant or less informative tokens while preserving the visual information needed for downstream tasks (Yang et al., 2024; Shang et al., 2025; Chen et al., 2024; Zhang et al., 2025; Alvar et al., 2025). However, different inputs may require different amounts of visual information: a simple image may be adequately represented using only a small number of tokens, whereas a complex scene or text-heavy document may benefit from a larger token budget (Li et al., 2024; Wang et al., 2024). Consequently, recent *adaptive* pruning methods determine the number of retained tokens from the input itself using statistical thresholds, learned importance predictors, or estimates of visual complexity (Shang et al., 2025; Lan et al., 2025; Ye et al., 2024).

This adaptivity introduces an important security implication: the input can influence not only *which* visual information is processed, but also *how much* computation the VLM allocates to processing

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.

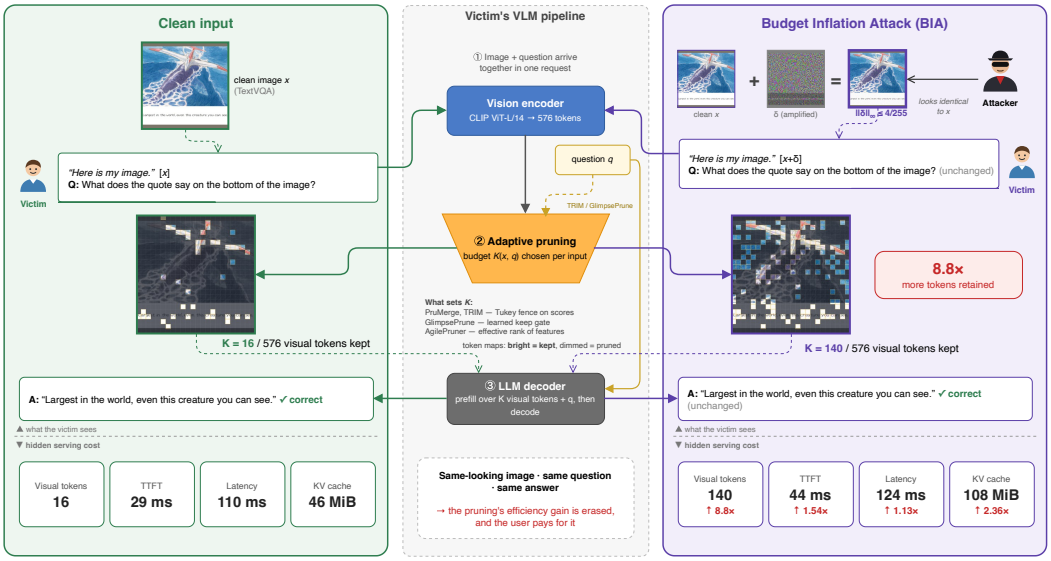


Figure 1: In the illustrated samples, the adversarial images cause adaptive pruning to retain more visual tokens than the clean images while preserving the correct answers. The larger token budgets increase time to first token, inference latency, and KV-cache storage, reducing the efficiency benefits of pruning.

that input. This motivates our central question: *Can a small, visually inconspicuous image perturbation cause an adaptive pruning mechanism to retain substantially more visual tokens?*

Existing attacks on compressed VLMs primarily manipulate the information that survives pruning, for example by causing useful visual evidence to be discarded or misleading evidence to be retained, ultimately degrading task performance (Zhang et al., 2026a;b). These attacks primarily target *which* information survives compression. In contrast, we target *how many* visual tokens an adaptive pruning mechanism retains. This attack surface is also distinct from prior attacks on VLM inference efficiency. Verbose-image attacks, for example, increase resource consumption by inducing VLMs to generate longer responses (Gao et al., 2024); our attack instead targets the visual-token budget during multimodal input processing, without explicitly optimizing the response length. DeSparsify (Yehezkel et al., 2024) provides the closest precedent by showing that adversarial inputs can reduce token sparsification in vision transformers. We study a complementary setting in adaptive VLM inference, where the retained visual-token count can be determined by heterogeneous and sometimes question-dependent mechanisms, and investigate whether manipulation of this budget translates into measurable VLM serving overhead.

To study this problem, we introduce the **Budget Inflation Attack (BIA)**, which optimizes bounded image perturbations against the internal signal that determines the retained-token budget. Rather than assuming a single budget-selection rule, we instantiate BIA across several classes of adaptive pruning mechanisms. For threshold-based methods such as PruMerge (Shang et al., 2025), BIA manipulates the token-score distribution so that more tokens exceed the retention threshold. For learned gating mechanisms (Zeng et al., 2026), it increases the tendency of visual tokens to be retained. For complexity-based methods such as AgilePruner (Baek et al., 2026), it increases the feature statistic used to determine the token budget. After optimization, we export the adversarial image and process it through the victim's normal inference pipeline, measuring the number of tokens actually retained together with downstream task performance. Thus, BIA attacks the *resource-allocation decision* induced by adaptive pruning rather than merely changing an intermediate token-selection pattern.

Our contributions are summarized as follows:

- We reveal that the flexibility of adaptive pruning technique to reduce the visual token can become an attack surface. Our research shows that adversarial image perturbations attacks are feasible and causes the victim to retain more tokens for the same task.
- We introduce BIA to increase retained visual tokens. We develop objectives for different budget mechanisms and analyze the limits imposed by their selection rules. We also assess its resource impact through time to first token (TTFT), inference latency, and KV-cache size.
- We evaluate cross-method transfer and introduce a randomly sampled ensemble strategy. We also study universal patches and transfer to unseen questions to examine the scope of the attacker's required knowledge. Our attack works across all the adaptive pruning mechanism.
- We evaluate BIA across four dataset covering diverse visual-language tasks. We measure token inflation alongside task accuracy and visual quality. We also discuss about two countermeasure of our attack. We hope the community can recognize this vulnerability to develop more robust pruning technique without hampering the task performance.

2 Background and Related Works

Visual token compression. Recent advances in VLMs have motivated methods that reduce the computational cost of processing visual tokens. VisionZip and FastV use predefined token budgets or pruning ratios (Yang et al., 2024; Chen et al., 2024). For a fixed initial token grid, these settings determine the retained count regardless of image complexity. Other methods adapt the retained count to the input. PruMerge, TRIM, and GlimpsePrune use content-dependent scores or learned importance predictors (Shang et al., 2025; Song et al., 2025; Zeng et al., 2026). Our work targets these adaptive budget mechanisms, including an adaptive-count variant motivated by AgilePruner (Baek et al., 2026).

Attacks on VLMs and inference efficiency. Prior attacks target either answer accuracy or inference resources. Compression-Aware Attack and CAGE exploit visual token compression to remove useful information or retain distorted evidence which causes incorrect answers (Zhang et al., 2026a;b). For inference efficiency, DeSparsify attacks adaptive token sparsification in vision transformers (Yehezkel et al., 2024). Verbose-image attacks increase VLM resource consumption by encouraging longer generated responses (Gao et al., 2024).

3 Threat Model

Attacker goal and capabilities. We consider a VLM inference service that uses an adaptive visual-token pruning mechanism to reduce computation by selecting the number of retained tokens from the input. The attacker can modify only the image supplied to the victim; they cannot alter the model weights, pruning parameters, decoding procedure, or serving configuration. In the primary setting, the attacker generates a perturbed image $x' = x + \delta$ subject to an ℓ_∞ constraint, $\|\delta\|_\infty \leq \epsilon$, while keeping the image resolution fixed. The attacker's primary goal is to increase the number of retained visual tokens, thereby increasing the resource consumption of the victim's inference pipeline. A secondary goal is to preserve the usefulness of the VLM response; accordingly, we evaluate token inflation jointly with task accuracy rather than treating prediction degradation as attack success.

Attacker knowledge. We consider both white-box and black-box attacker knowledge. In the **white-box setting**, the attacker knows the victim VLM, pruning mechanism, and inference configuration and can compute gradients through the components that determine the retained-token budget. In the **black-box setting**, the attacker does not know the pruning mechanism deployed by the victim and has no access to its gradients; instead, adversarial images are generated using one or more locally available surrogate pruning mechanisms and transferred to the unseen target without further optimization. For ensemble-based transfer, the target mechanism is excluded from both gradient computation and candidate selection.

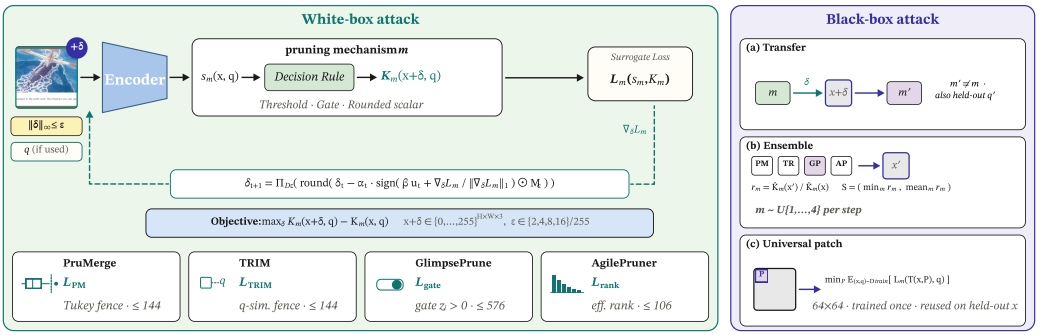


Figure 2: Overview of BIA attack generation. In the white-box setting, BIA optimizes image perturbations against an individual pruning mechanism using an objective matched to its budget rule. For an unknown deployed mechanism, we consider cross-mechanism transfer, universal patches reused across images, and ensemble optimization over multiple source mechanisms.

4 The Budget Inflation Attack

We attack four visual-token pruning techniques: PruMerge (Shang et al., 2025), AgilePruner (Baek et al., 2026), TRIM (Song et al., 2025), and GlimpsePrune (Zeng et al., 2026). Each is evaluated under two levels of attacker knowledge: White-Box Attacks and Black-Box Attacks. Figure 2 shows the overview of our Budget Inflation Attack.

4.1 Budget inflation objective

For a fixed mechanism m, the desired count change is:

$$\max_{\delta \in \mathcal{D}_\epsilon(x)} \big[K_m(x+\delta, q) - K_m(x, q)\big]. \tag{1}$$

$$\mathcal{D}_\epsilon(x) = \{\delta : \|\delta\|_\infty \le \epsilon_{\text{int}}, \quad x+\delta \in \{0,\dots,255\}^{H\times W\times 3}\}. \tag{2}$$

We optmize a continuous loss on the signal that determines these boundaries. The victim then evaluates the exported image to obtain the actual count and answer.

4.2 White-Box Settings

For a mechanism m the attacker wants to raise the retained count $K_m(x+\delta, q)$ at fixed resolution, which is Equation 1. The count is a sum of indicators over a threshold test or a rounded scalar, so we optimize a continuous loss L_m on the quantity the rule thresholds or rounds, and read the actual count from the victim's own forward pass on the exported file.

Shared optimizer. All four mechanisms are attacked with one optimizer, momentum sign-gradient ascent (Dong et al., 2018) projected onto the integer lattice. For P pixel channels,

$$g_t = \frac{\nabla_\delta L_m}{\|\nabla_\delta L_m\|_1/P + \xi}, \qquad u_{t+1} = \beta u_t + g_t, \qquad \delta_{t+1} = \Pi_{\mathcal{D}_\epsilon}(\text{round}\,(\delta_t - \alpha_t\ \text{sign}(u_{t+1}) \odot M_c)), \tag{3}$$

where M_c is the content mask and $\Pi_{\mathcal{D}_\epsilon}$ projects onto the feasible set of Equation 2, so every iterate stays within the budget and remains a valid image. The gradient is normalized by its mean absolute value; the constraint itself stays L_∞. We use $\beta = 1$, a step size annealed on a cosine schedule from 2.0 down to 0.75 intensity levels, and $\delta_0 = 0$. Straight-through rounding supplies gradients through the integer cast. Each attack returns the iterate with the highest count under the victim's own arithmetic rather than the final one; the counts used for that check are attack-head estimates,

and every number we report comes from a reloaded PNG. Only the loss differs between mechanisms. Each subsection below states what the rule counts and the loss we optimize against it

4.2.1 PRUMERGE

PruMerge scores each visual token by the attention the CLIP `[cls]` token pays to it and keeps the upper outliers of that distribution under a Tukey fence, $a_i > Q_3 + 1.5\,\mathrm{IQR}$, plus one merged summary token. The rule reads no text, so one perturbation serves every question about an image. In the deployed model the attention is computed in half precision, where about 93% of the 576 values round to exact zero; the served count is therefore decided by how many values clear the fp16 subnormal scale, not by the float32 fence. Our loss targets that quantity. With τ the reference scale and $n_{\mathrm{esc}} = \sum_i \sigma\big((\log a_i - \log\tau)/T\big)$ a soft count of the values above it,

$$L_{\mathrm{PM}} = \frac{1}{144}\Big[(144 - n_{\mathrm{esc}}) + 3\,(n_{\mathrm{esc}} - 144)\Big] \;+\; \mathrm{Var}(a_L) - \lambda\big(\bar{a}_U - \bar{a}_L\big). \tag{4}$$

The first term is two-sided and targets the structural limit of 144 from below, penalizing overshoot more heavily, because raising scores past the fence also raises the fence. The remaining terms compress the lower three quarters of the distribution and separate the upper quarter from it.

4.2.2 TRIM

TRIM ranks visual tokens by their CLIP-space similarity to the question's text embedding, applies a softmax, and keeps the upper outliers under the same Tukey fence, again with one aggregated token appended. Its budget therefore depends on the question as well as the image. Writing p for the score distribution, Q_3 and d for its third quartile and interquartile range, and $\widetilde{F} = Q_3 + 1.5d$ for the fence, we push the b largest scores across it while softly counting everything above it,

$$L_{\mathrm{TRIM}} = \frac{1}{b\widetilde{F}} \sum_{i \in B_b} \big((1+\kappa)\widetilde{F} - p_i\big) \;-\; \frac{\lambda_{\mathrm{s}}}{n} \sum_i \sigma\!\left(\frac{(p_i - Q_3)/d - 1.5}{\tau}\right). \tag{5}$$

The band B_b grows during the attack and is capped at the structural limit, so the objective never asks for a count the rule cannot serve.

4.2.3 GLIMPSEPRUNE

GlimpsePrune appends a learned glimpse token, reads its attention inside the decoder, and fuses it with hierarchical visual features to produce one keep logit z_i per visual token. A token survives when $\sigma(z_i) > 0.5$, that is when $z_i > 0$. The decision is made once, at decoder layer 21 of 32, and is question dependent. Unlike a fence, the threshold is fixed and the logits are produced independently, so no order statistic constrains how many tokens can pass; the full pool of 576 is reachable. We simply drive every logit above the threshold with a margin,

$$L_{\mathrm{gate}} = \frac{1}{n} \sum_i (m - z_i). \tag{6}$$

Because the gate sits behind 22 half-precision decoder layers, the loss is dynamically rescaled: a fixed scale overflows on some samples, and a non-finite gradient would leave the perturbation silently unchanged.

4.2.4 AGILEPRUNER

AgilePruner uses effective rank as a measure of feature diversity. Its main configuration keeps a fixed token budget; the adaptive-count variant we attack is described in there as well. It maps the effective rank of the projected visual tokens to a budget around a reference of 88 tokens.

We instantiate that description as

$$K_{\mathrm{S}}(x) = \mathrm{clip}_{[1,576]}\Big(\mathrm{round}\big[\,88 \cdot \mathrm{clip}_{[0.8,1.2]}\big(r(X)/90\big)\big]\Big), \tag{7}$$

where the reference rank 90 is the released code's constant and the upper clamp is our reading of "increasing it proportionally"; the paper states no upper limit. That clamp is what bounds the count at 106. The statistic is text-blind, so one perturbation per image suffices. We maximize effective rank directly, capped just above the point where the clamp saturates,

$$L_{\text{rank}} = -\frac{\min\big(r(X), 112\big)}{90}, \tag{8}$$

with $r(X)$ computed from the eigenvalues of the centred Gram matrix of the projected tokens X. The Gram is accumulated in double precision: in float32 a large part of the spectrum collapses to the deployed clamp and passes no gradient, and effective rank rises precisely by lifting that tail.

4.3 Black-Box Settings

Universal patch. Let P be one 64×64 RGB patch. An operator $T(x, P)$ places it at the upper-left corner of the image content on the canvas. We optimize

$$\min_{P \in \{0,\dots,255\}^{64\times 64\times 3}} \mathbb{E}_{(x,q)\sim\mathcal{D}_{\text{train}}}[L_m(T(x, P), q)]. \tag{9}$$

The patch is shared across all evaluation images. It is trained once for each source mechanism. A separate ensemble patch samples the source mechanism at each training step. Patch pixels need only stay in the valid RGB range. There is no small perturbation bound inside the patch.

The main protocol anchors to image content, not automatically to the square canvas corner. Training and evaluation images are disjoint.

Ensemble image attack. For each image and question, we sample one of the four mechanisms uniformly at each step. Only that loss supplies the gradient. We use 160 steps and check candidates every 20 steps. Sampling gives roughly equal updates in expectation, not exactly equal counts.

Let $\widehat{K}_m$ denote the attack head's count estimate. Candidate selection uses

$$r_m(x') = \frac{\widehat{K}_m(x', q)}{\max(1, \widehat{K}_m(x, q))}, \qquad S(x') = \left(\min_m r_m(x'),\ \frac{1}{|\mathcal{M}|}\sum_m r_m(x')\right). \tag{10}$$

We maximize S lexicographically. This first protects the weakest count ratio. An additive score alone can trade a decrease in one method for a large gain in another. The denominator floor handles zero counts during search. It is not the definition of reported dataset inflation.

The result is one adversarial image for a given image-question pair. Every victim receives that same file. For black-box ensemble transfer, a target must be excluded from both gradient updates and candidate checks. The algorithm for ensemble image attack is shown in Appendix A.2 .

5 Evaluation

5.1 Experimental Setup

Models. Our evaluation primarily focuses on the widely adopted LLAVA (Liu et al., 2023) model. It is used as a backbone model in existing token compression research. We used two variants of LLAVA-1.5 model: 7B and 13B. The results of 13B are shown in Appendix. The core configurations use LLaVA-1.5-based models with 336x336 canvases and 576 initial visual tokens. We have analysed recent state of the art pruning mechanism which are PruMerge (Shang et al., 2025), AgilePrune (Baek et al., 2026), TRIM (Song et al., 2025), and GlimsePrune (Zeng et al., 2026).

Datasets and Baselines. We evaluated on four established dataset which evaluate diverse capabilites: TextVQA (Singh et al., 2019), POPE (Yifan Li & Wen, 2023), MME (Fu et al., 2025), and MMB (Liu et al., 2024). These datasets differ in the types of questions asked about the images. TextVQA focuses on detailed text in images, so it requires keeping specific image patches. POPE asks simple yes or no questions about objects to test whether important visual information is lost during pruning. MME and MMB cover broader image understanding and reasoning using short-answer and multiple-choice questions. We compare the effectiveness our attack against the clean image result and also with the random perturbation image result.

Table 1: White-box attacks on POPE, MME, MMBench, and TextVQA with $\epsilon = 4/255$. We report mean retained visual tokens and accuracy (%).

		POPE		MME		MMBench		TextVQA	
Mechanism	Image	Token Count	Accuracy	Token Count	Accuracy	Token Count	Accuracy	Token Count	Accuracy
PruMerge	Clean	36.3	74.26%	40.1	55.77%	37.9	66.58%	40.9	53.96%
	Random	36.84	74.18%	39.34	55.69%	38.62	66.49%	40.82	53.88%
	BIA	**88.4**	**73.31%**	**96**	**53.24%**	**84.8**	**63.83%**	**90.1**	**49.10%**
TRIM	Clean	118.33	89.4%	127.29	61.04%	126.57	72.28%	128.7	53.5%
	Random	119.63	89.31%	127.54	60.98%	126.70	72.34%	128.83	53.62%
	BIA	**140.12**	**87.4%**	**140.26**	**61.11%**	**140.92**	**72.69%**	**144.08**	**53.38%**
AgilePrune-S	Clean	98.8	85.5%	91.41	61.54%	84.96	71.24%	94.43	56.36%
	Random	98.90	85.42%	91.59	61.47%	84.79	71.31%	94.52	56.21%
	BIA	**106**	**65.79%**	**106**	**47.62%**	**105.94**	**56.98%**	**106**	**48.13%**
AgilePrune-E	Clean	105.8	85.37%	94.85	61.51%	84.33	71.53%	100.5	56.57%
	Random	105.91	85.29%	95.23	61.62%	84.58	71.41%	100.60	56.43%
	BIA	**356.29**	**77.79%**	**353.88**	**49.45%**	**280.17**	**59.72%**	**287.93**	**49.18%**
GlimpsePrune	Clean	108.41	88.5%	201.18	66.48%	65.85	73.82%	74.99	57.75%
	Random	108.19	88.37%	199.77	66.31%	64.40	73.68%	74.02	57.61%
	BIA	**575.99**	**81%**	**575.99**	**59.82%**	**575.98**	**67.63%**	**575.95**	**50.59%**

Implementation settings. Perturbations are ℓ_∞-bounded and optimized on the integer pixel lattice. The perturbation budget is swept over $\{2, 4, 8, 16\}/255$. All four mechanisms use momentum sign-gradient steps (Dong et al., 2018) with decay $\mu = 1.0$, an ℓ_1-normalized gradient and a cosine step size annealed from 2.0 to 0.75 intensity levels. The ensemble samples one mechanism uniformly per step rather than summing four losses whose scales differ by an order of magnitude. Universal patches are 64×64, unconstrained within the patch, and trained for 1500 iterations over 200 image-question pairs disjoint from every evaluation split. Attacks and evaluations use a single NVIDIA RTX 5090.

Evaluation metrics. We examine the following metrics to evaluate the effectiveness of the attack:

- **Retained visual tokens:** the number of visual tokens the compressor forwards into the language model. We report the clean and adversarial means and their ratio $R_K = \sum_i K_{a,i} / \sum_i K_{c,i}$, a ratio of sums rather than a mean of per-example ratios.
- **Accuracy:** performance on the victim's original task.
- **KV-cache size:** the per-request key-value cache implied by the retained count. At batch size one the model weights dominate peak memory, so the cache is the quantity that actually limits how many concurrent requests a device can hold.
- **Latency:** total wall-clock time per request including decoding. Generated answer length is logged alongside it, so a latency change cannot be attributed to longer outputs.
- **Time to first token (TTFT):** prompt processing up to the first generated token. We report it separately from total latency because this attack inflates the visual *prefix*, so its effect concentrates in prefill, whereas prior slowdown attacks inflate decoding.

5.2 White-Box Attacks

Table 1 shows that BIA increases retained visual tokens across all evaluated mechanisms and datasets. BIA also outperforms random perturbations showing the effectiveness of our attack. GlimpsePrune retains nearly all 576 tokens after the attack, with inflation reaching $8.75\times$ on MMBench. PruMerge and AgilePrune-E exhibit increases of up to $2.44\times$ and $3.73\times$ respectively. TRIM and AgilePrune-S show smaller increases because their token counts are bounded. TRIM's threshold rule selects at most 144 tokens and AgilePrune-S caps the retained count at 106. Thus, we can't increase the visual token count beyond that limit. The accuracy impact varies across mechanisms. TRIM changes relatively little, whereas the other methods like the AgilePruner variants have larger larger losses. These results demonstrate that BIA can increase the visual token count in all the pruning mechanism across all the datasets.

Table 2: Cross-mechanism transfer on TextVQA with $\epsilon = 4/255$. Only mean retained visual-token counts are reported. Rows identify the source attack used to generate the images, and columns identify the pruning mechanism used for evaluation.

Input Image	PruMerge	TRIM	GlimpsePrune	AgilePrune-S	AgilePrune-E
Clean Image	40.9	128.7	74.99	94.43	100.50
PruMerge Attacked Image	90.1	124.20	68.46	100.09	111.46
TRIM Attacked Image	40.5	144.08	74.0	93.98	100.56
GlimpsePrune Attacked Image	52.6	117.30	575.95	97.6	93.05
AgilePrune-E Attacked Image	102.6	124.3	105.94	106.2	287.93

Table 3: Universal-patch and ensemble attacks on TextVQA. Rows identify the input image variant, and columns identify the evaluated pruning mechanism. We report mean retained tokens and accuracy (%). Patches are 64×64; the per-image ensemble attack uses $\epsilon = 8/255$.

	PruMerge		TRIM		AgilePrune-S		AgilePrune-E		GlimpsePrune	
Input Image	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.
Clean	40.8	54.5	128.7	53.5	94.5	56.8	100.6	57.0	75.8	58.1
PruMerge patch	**98.6**	51.6	129.5	52.2	101.5	53.9	113.9	53.5	76.0	54.5
TRIM patch	47.8	51.2	**129.4**	52.9	95.7	55.7	102.0	56.1	69.7	57.1
AgilePrune patch	47.7	52.1	129.2	53.3	**103.9**	55.7	129.3	55.9	68.4	56.4
GlimpsePrune patch	52.9	51.3	129.3	52.5	100.5	54.3	111.9	54.4	**368.0**	56.1
Ensemble patch	82.8	52.0	129.4	52.7	102.2	54.5	120.8	54.8	78.7	55.7
Ensemble Attack	119.2	47.3	136.4	47.9	103.0	48.6	115.7	48.6	457.2	48.4

5.3 Transfer across mechanism

Table 2 shows that BIA generally produces larger token increases when evaluated on the pruning mechanism used to generate the attack. Transfer to other mechanisms is inconsistent and can even reduce the retained count. An exception is AgilePrune-E where adversarial images increase token counts on PruMerge, GlimpsePrune, and AgilePrune-S, although they do not increase TRIM's count. Its transfer to PruMerge is particularly strong. These results suggest that objectives optimized for one budget rule do not reliably generalize to other rules. This motivates evaluating ensemble optimization as a way to seek more consistent inflation when the deployed pruning mechanism is unknown. The attack also transfers across model scale. Replaying the same perturbations through LLaVA-1.5-13B without re-optimizing them. The 13B results are shown to Appendix A.1.

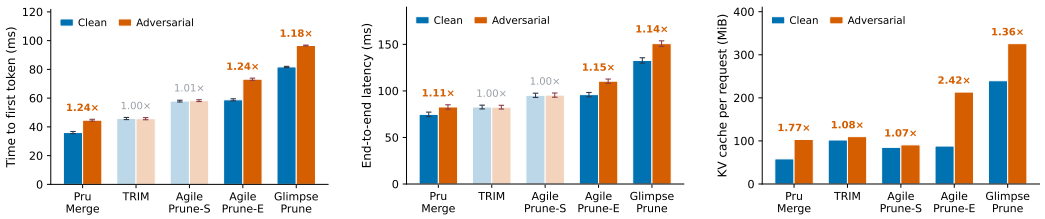
5.4 Black-Box Attacks

Universal Patches. We train the patches on 500 images and evaluate them on 2,666 disjoint held-out images containing 4,224 questions. For each source pruning mechanism, we learn one universal patch and evaluate it across the target mechanisms in Table 3. Each patch is reused unchanged across evaluation images. We also train an ensemble patch using multiple source mechanisms and apply the same learned pattern to every evaluation image. The results show that patch effectiveness depends on the source and target.

Ensemble Attack. Unlike a universal patch, the ensemble attack optimizes a separate adversarial image for each evaluated example using multiple source mechanisms. The same resulting image is then evaluated across the pruning mechanisms. Table 3 shows that this approach increases retained tokens in every evaluated configuration. Compared with the ensemble patch, it produces larger token counts on PruMerge, TRIM, AgilePruner-S, and GlimpsePrune. These results show broader token inflation across the evaluated mechanisms.

5.5 Resource Cost

Figure 3 compares the resource cost of clean and adversarial inputs. BIA increases TTFT by up to $1.24\times$ on PruMerge and AgilePruner-E, while end-to-end latency increases by up to $1.15\times$ on



(a) Time to first token (b) End-to-end latency (c) KV cache per request

Figure 3: Resource cost of clean and adversarial inputs at $\epsilon = 8/255$ on an NVIDIA RTX 5090.

AgilePruner-E. The largest relative increase occurs in KV-cache storage, which rises by $2.42\times$ on AgilePruner-E and $1.77\times$ on PruMerge. In contrast, TRIM and AgilePruner-S show little change in runtime and smaller cache increases, consistent with their limited token-budget headroom. These results show that token inflation can increase inference time and memory demand.

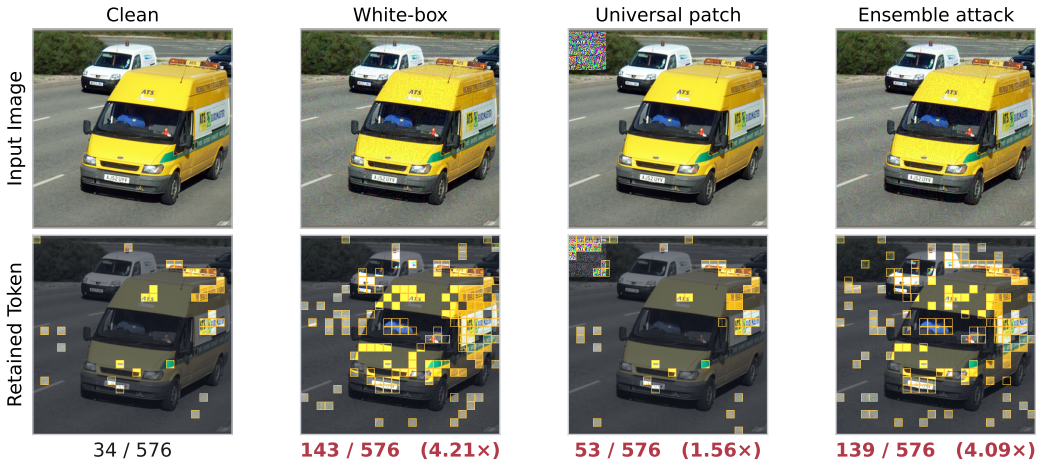


Figure 4: **Budget inflation on PruMerge.** *Top:* the input image under three attacks. *Bottom:* which of the 576 visual patches PruMerge retains, survivors in full color. Perturbations are ℓ_∞-bounded at $\epsilon = 8/255$.

5.6 Visual Quality

Figure 4 compares clean images with their adversarial counterparts. The bounded perturbations produce subtle visual changes while preserving the main image content. To the question *"What is the license plate number of this vehicle?"* the model gives same answer in all four cases, while the retained budget rises from 34 to 143 tokens.

6 Conclusions, Limitations, and Discussion

In this work, we presented BIA to increase retained token counts and resource use in adaptive visual pruning. We have examined both white box and black box scenario. Our both attack performs well by inflating the visual token. A limitation is that BIA cannot inflate a token count in a fixed toke pruning mechanism. However, a restrictive fixed budget will discard information needed for difficult questions. This suggests two simple countermeasures. First, an explicit token cap can bound inflation attack but it will reduce task accuracy. Second, image smoothing will weaken our perturbations, but it can also remove fine details from the images. As a result, the visual information needed for a particular question might be lost. Our findings motivate pruning methods that balance resource efficiency, task performance, and robustness to adversarial inputs.

AI USE STATEMENT

We used generative AI solely as a writing and editing aid to polish and refine text that had already been written by the authors. We have not used generative AI tools to generate the research ideas, conduct the research, perform the analysis, develop the methodology, generate experimental results. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies a security weakness in adaptive visual token pruning. Our goal is to help developers understand and reduce the risk of adversarial resource consumption. We evaluate them only on models and computing resources that we are authorized to use, without targeting third-party services. Our experiments use existing benchmark datasets and do not involve human participants. We report both token inflation and changes in task accuracy to make the effects and limitations clear. We also discuss potential countermeasures and their trade-offs to support the development of more robust pruning methods.

REPRODUCIBILITY STATEMENT

The methodology section describes the attack objectives and optimization procedures. The experimental setup specifies the datasets, pruning configurations, and evaluation metrics. We provide attack and evaluation code, environment requirements, and instructions for reproducing the experiments in our https://anonymous.4open.science/r/PLPM/README.md.

REFERENCES

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 9392–9401, 2025.

Changwoo Baek, Jouwon Song, Sohyeon Kim, and Kyeongbo Kong. Agilepruner: An empirical study of attention and diversity for adaptive visual token pruning in large vision-language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2NLkhPex1M.

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, 2023. URL https://arxiv.org/abs/2308.12966.

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 fornbsp;large vision-language models. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXXXI*, pp. 19–35, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73003-0. doi: 10.1007/978-3-031-73004-7_2. URL https://doi.org/10.1007/978-3-031-73004-7_2.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum, 2018. URL https://arxiv.org/abs/1710.06081.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. Mme: A comprehensive evaluation benchmark for multimodal large language models, 2025. URL https://arxiv.org/abs/2306.13394.

Kuofeng Gao, Yang Bai, Jindong Gu, Shu-Tao Xia, Philip Torr, Zhifeng Li, and Wei Liu. Inducing high energy-latency of large vision-language models with verbose images. In *ICLR*, 2024.

Zhibin Lan, Liqiang Niu, Fandong Meng, Wenbo Li, Jie Zhou, and Jinsong Su. Avg-llava: An efficient large multimodal model with adaptive visual granularity, 2025. URL https://arxiv.org/abs/2410.02745.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Zhang Li, Biao Yang, Qiang Liu, Zhiyin Ma, Shuo Zhang, Jingxu Yang, Yabo Sun, Yuliang Liu, and Xiang Bai. Monkey: Image resolution and text label are important things for large multi-modal models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26753–26763, 2024. doi: 10.1109/CVPR52733.2024.02527.

Zongxia Li, Xiyang Wu, Hongyang Du, Fuxiao Liu, Huy Nghiem, and Guangyao Shi. A survey of state of the art large vision language models: Alignment, benchmark, evaluations and challenges, 2026. URL https://arxiv.org/abs/2501.02189.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916. Curran Associates, Inc., 2023. doi: 10.52202/075280-1516. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf.

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

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. *ICCV*, 2025.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8309–8318, 2019. doi: 10.1109/CVPR.2019.00851.

Dingjie Song, Wenjun Wang, Shunian Chen, Xidong Wang, Michael X. Guan, and Benyou Wang. Less is more: A simple yet effective token reduction method for efficient multi-modal LLMs. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 7614–7623, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.508/.

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, 2024. URL https://arxiv.org/abs/2409.12191.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. *arXiv preprint arXiv:2412.04467*, 2024.

Xubing Ye, Yukang Gan, Yixiao Ge, Xiao-Ping Zhang, and Yansong Tang. Atp-llava: Adaptive token pruning for large vision language models, 2024. URL https://arxiv.org/abs/2412.00447.

Oryan Yehezkel, Alon Zolfi, Amit Baras, Yuval Elovici, and Asaf Shabtai. Desparsify: Adversarial attack against token sparsification mechanisms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=D4yRz3s7UL.

Kun Zhou Jinpeng Wang Wayne Xin Zhao Yifan Li, Yifan Du and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=xozJw0kZXF.

Quan-Sheng Zeng, Yunheng Li, Qilong Wang, Peng-Tao Jiang, Zuxuan Wu, Ming-Ming Cheng, and Qibin Hou. A glimpse to compress: Dynamic visual token pruning for large vision-language models. *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2026. doi: 10.1109/TCSVT.2026.3702147.

Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. *arXiv preprint arXiv:2412.01818*, 2025.

Xiaomei Zhang, Zhaoxi Zhang, Leo Yu Zhang, Yanjun Zhang, Guanhong Tao, and Shirui Pan. Less is more – until it breaks: Security pitfalls of vision token compression in large vision-language models, 2026a. URL https://arxiv.org/abs/2601.12042.

Xinwei Zhang, Hangcheng Liu, Li Bai, Hao Wang, Qingqing Ye, Tianwei Zhang, and Haibo Hu. On the adversarial robustness of large vision-language models under visual token compression, 2026b. URL https://arxiv.org/abs/2601.21531.

A Appendix

A.1 MODEL SCALE: LLAVA-1.5-13B

We replay the perturbations crafted against LLaVA-1.5-7B through LLaVA-1.5-13B without re-optimizing them, so Table 4 measures how much of the inflation is a property of the compressor rather than of the language model behind it. Both models are evaluated on the same 5,000-question TextVQA split, with the same saved PNGs, on the same GPUs.

Four of the five configurations reproduce their 7B behaviour at 13B. PruMerge and TRIM decide entirely inside the frozen CLIP ViT-L/14-336 tower, which is identical across the two checkpoints, and their retained counts carry over unchanged. AgilePrune-S and AgilePrune-E read post-projector features whose width grows from 4096 to 5120, so their absolute counts shift while the direction and scale of the effect do not. GlimpsePrune is the exception: its gate is a trained component of the model, retrained at each scale, and the perturbation that saturates the 7B gate reaches only 172.85 of 576 tokens at 13B. Inflation therefore transfers across model scale wherever the pruning decision is made before the language model, and has to be re-earned where the decision is learned with it.

Table 4: White-box attack replayed across model scale on TextVQA (5,000 questions, $\epsilon = 8/255$). The same adversarial images are used for both models; no perturbation is re-optimized for 13B. Token counts are per-image means of the visual tokens forwarded into the language model.

	LLaVA-1.5-7B				LLaVA-1.5-13B			
	Token Count		Accuracy (%)		Token Count		Accuracy (%)	
Configuration	Clean	Adv.	Clean	Adv.	Clean	Adv.	Clean	Adv.
PruMerge	40.93	131.10	53.89	48.00	40.93	131.10	55.20	50.60
TRIM	119.56	144.19	47.38	51.21	119.10	144.19	48.84	53.52
AgilePrune-S	94.45	106.00	56.39	54.77	91.23	102.36	58.74	57.17
AgilePrune-E	100.52	351.21	56.57	46.00	94.76	341.25	58.62	48.62
GlimpsePrune	74.98	575.98	57.87	50.03	94.39	172.85	60.87	54.90

A.2 ENSEMBLE ATTACK

Algorithm 1 gives the per-image ensemble procedure. One mechanism is sampled uniformly at each step and only its loss supplies that step's gradient. The losses are sampled rather than summed because they differ by an order of magnitude in scale, and with a signed update a sum would let one mechanism dominate.

A.3 TRANSFER ACROSS QUESTIONS

Table 5: Transfer across questions on POPE. Each adversarial image is generated using one question and then evaluated on different held-out questions about the same image. We report clean and adversarial token counts and accuracy for the optimization question and the held-out questions.

Configuration	Uses q	Group	Clean Count	Adv. Count	Clean Accuracy	Adv. Accuracy
PruMerge	No	Same question	36.29	124.64	79.60	76.20
		Unseen question	36.29	124.64	75.12	77.77
AgilePrune-S	No	Same question	98.82	104.83	93.40	78.20
		Unseen question	98.82	104.83	82.52	76.84
TRIM	Yes	Same question	113.85	131.70	94.20	81.00
		Unseen question	103.91	111.41	78.64	72.69
GlimpsePrune	Yes	Same question	143.58	509.14	92.20	71.60
		Unseen question	115.75	211.18	83.72	78.28

A.4 PERTURBATION BUDGET SWEEP

Algorithm 1 Per-image ensemble attack

Require: clean canvas x, question q, mechanism set $\mathcal{M}$, budget ϵ, steps T, scoring period k, content mask M_c
Ensure: adversarial canvas $x^\star$

$\delta \leftarrow 0, \quad u \leftarrow 0$
$K_m^{\text{clean}} \leftarrow \widehat{K}_m(x, q)$ for all $m \in \mathcal{M}$
$x^\star \leftarrow x, \quad S^\star \leftarrow (1, 1)$ ▷ clean image is the incumbent
for $t = 1$ **to** T **do**
  $\alpha_t \leftarrow 0.75 + \frac{1}{2}(2.0 - 0.75)\big(1 + \cos(\pi t/T)\big)$
  $m \sim \text{Uniform}(\mathcal{M})$ ▷ resampled every step
  $x' \leftarrow \text{round}\big(\text{clip}(x + \delta,\, 0,\, 255)\big)$ ▷ straight-through
  $g \leftarrow \nabla_\delta L_m(x', q)$
  $g \leftarrow g \,/\, \big(\|g\|_1 / P\big)$
  $u \leftarrow \beta u + g$
  $\delta \leftarrow \Pi_{\mathcal{D}_\epsilon}\big(\text{round}(\delta - \alpha_t \,\text{sign}(u) \odot M_c)\big)$
  if $t \bmod k = 0$ **or** $t = T$ **then**
    $c \leftarrow \text{round}\big(\text{clip}(x + \delta,\, 0,\, 255)\big)$
    $r_m \leftarrow \widehat{K}_m(c, q) \,/\, \max(1,\, K_m^{\text{clean}})$ for all $m \in \mathcal{M}$
    $S \leftarrow \big(\min_m r_m,\ \frac{1}{|\mathcal{M}|}\sum_m r_m\big)$
    if $S > S^\star$ lexicographically **then**
      $x^\star \leftarrow c, \quad S^\star \leftarrow S$
    end if
  end if
end for
return $x^\star$

Table 6: Perturbation budget sweep on the TextVQA Dataset. Tokens are per-image means of the visual tokens forwarded into the language model

Configuration	ϵ	Tokens	Acc. (%)	SSIM	PSNR (dB)
PruMerge	clean	41.30	55.10	1.0000	–
	2	54.92	52.88	0.9820	44.97
	4	90.10	51.26	0.9427	39.60
	8	131.07	48.53	0.8347	34.09
	16	141.48	47.45	0.6351	28.57
TRIM	clean	128.42	55.79	1.0000	–
	2	143.66	53.70	0.9824	43.85
	4	144.08	53.38	0.9395	37.95
	8	144.16	53.05	0.8359	32.09
	16	144.23	52.19	0.6833	26.33
AgilePrune-S	clean	94.70	58.27	1.0000	–
	2	106.00	58.52	0.9818	44.27
	4	106.00	57.51	0.9520	40.30
	8	106.00	57.42	0.9140	38.19
	16	106.00	57.19	0.9049	37.83
AgilePrune-E	clean	100.83	58.24	1.0000	–
	2	250.30	54.85	0.9824	43.78
	4	306.10	50.10	0.9390	37.84
	8	350.06	45.50	0.8336	31.96
	16	375.00	42.95	0.6790	26.15
GlimpsePrune	clean	75.97	58.27	1.0000	–
	2	573.12	53.37	0.9826	43.89
	4	575.96	51.88	0.9418	38.14
	8	575.98	51.74	0.8462	32.56
	16	575.99	51.30	0.7108	27.36