Chiseling for Efficiency: Omitting Non-salient Visual Information for VLMs

Anonymous ACL submission

Abstract

Vision-Language Models (VLMs) demand ever-higher input resolutions for fine-grained reasoning, causing patch tokens to explode and inflating compute, memory, and energy. Existing remedies either rigidly prune tokens after the vision encoder runs, or rely on incomplete localizers that miss the query's actual visual evidence. We present CHISEL, which carefully *chisels* an input image down to query-relevant regions before the vision encoder by pairing query-noun grounding with a small language model that recovers abstract semantics. It then fuses the extracted regions with a context-faithful, layout-preserving scheme, compressing the visual input without sacrificing accuracy. A pipelined runtime overlaps preprocessing with concurrent VLM execution, effectively amortizing the additional computation. Backbone-agnostic and single-pass, CHISEL cuts visual tokens by up to 83.9% while improving accuracy by up to 17.37 pp and end-to-end throughput by up to 178% on HR-Bench and V^*Bench across three VLM backbones.

1 Introduction

Recent advances in AI have led to a surge of interest in multimodal models. Vision-Language Models (VLMs), a prominent class of multimodal models, jointly process visual and linguistic information within a single model. Specifically, visual features extracted by a vision encoder are projected into the embedding space of a language model, enabling the language model to interpret and reason over visual inputs in the same manner as text. This architecture extends the powerful reasoning capabilities of language models to the visual domain, and its effectiveness has been demonstrated across a wide range of applications.

In such VLMs, handling high-resolution images is both essential and costly. On the demand side, VLMs must span a wide spectrum of detail—from fine-grained character reading to scene-level reasoning (Mathew et al., 2021; Singh et al., 2019; Gurari et al., 2018). Since the relevant detail cannot be anticipated in advance, the encoder must preserve fine-grained information across the entire image. On the cost side, the patches produced by the vision encoder are consumed directly as Large Language Model (LLM) tokens, so any increase in resolution causes a rapid explosion in sequence length. This tension between coverage and compute makes high-resolution processing a central bottleneck for practical VLMs. For example, Qwen2-VL (Wang et al., 2024) scales vision tokens with input resolution to preserve detail, but suffers from a token explosion that sharply inflates compute and memory demands.

Earlier VLMs sidestepped this trade-off by resizing inputs to a fixed resolution, both to match the pretraining setup of their vision encoders and to keep the number of visual tokens fed to the language model bounded (Liu et al., 2024a; Li et al., 2023; Dai et al., 2023; Alayrac et al., 2022; Bai et al., 2023). Such resizing, however, inevitably discards critical information from high-resolution inputs.

This has motivated tiling-based pipelines (Ye et al., 2023; Li et al., 2024b; Lin et al., 2023; Liu et al., 2024b; Li et al., 2024a), which partition the input image into fixed-size tiles, encode each tile independently through the vision encoder, and concatenate the resulting tokens before feeding them to the language model; a downsampled view of the entire image is included alongside the tiles to preserve global context.

While this design preserves fine-grained content better than aggressive resizing, the number of visual tokens still scales with image area, leading to excessive computation and substantial memory overhead, particularly in attention-based VLMs. Furthermore, because self-attention in the vision encoder is confined within each tile, cross-tile con-

text is not captured during feature extraction; long-range visual structure must be reconstructed by the language model from concatenated tile-wise tokens, often degrading accuracy.

Recent sophisticated approaches to efficient high-resolution VLM inference still suffer from a limitation: they reduce tokens either too late or in the wrong place. Token-pruning methods prune a preset fraction of tokens by attention score after the vision encoder, regardless of input content. The upstream token explosion therefore remains unaddressed, while the rigid policy often discards visual evidence essential to the query and disrupts semantic coherence (Chen et al., 2024). Other methods perform query-guided region extraction directly from raw images, yet rely on incomplete localization heuristics that often fail to localize the relevant visual evidence and introduce additional overhead (Guo et al., 2025). Such limitations constrain the preservation of fine-grained visual evidence under constrained compute budgets, as existing approaches either defer compression until after large-scale token expansion or allocate computation through imprecise region localization.

Motivated by these limitations, we propose `Chisel`, a framework that surgically isolates the query-relevant regions of an image, and selectively fuses only their essential content into a compact input for the LLM. By adaptively compressing visual information prior to large-scale token expansion, `Chisel` fundamentally decouples image resolution from downstream token growth. By compressing the image before VLM inference, `Chisel` is orthogonal to existing inner-LLM token-pruning methods. `Chisel` exhibits the following key features:

- **Incisive Region of Interest (ROI) extraction:** `Chisel` extracts query-relevant ROIs directly from the raw image by pairing noun-phrase grounding with a lightweight auxiliary model that recovers the abstract semantics and actions that surface nouns alone miss (§3.1).
- **Context- and layout-faithful fusing:** `Chisel` arranges all extracted regions into a single composite that preserves the source spatial layout, while merging clustered objects into shared coarse crops to retain the surrounding context that small targets rely on for recognition (§3.2).
- **Latency-hiding pipelining:** `Chisel` decomposes its preprocessing into fine-grained stages, dispatches each across the heterogeneous compute resources, and schedules them into the idle slots exposed by in-flight VLM executions—hiding the added cost (§3.3).

We evaluate `Chisel` on two high-resolution VQA benchmarks (HR-Bench and V^*Bench) across three VLM backbones spanning native dynamic resolution, tile-based encoding, and Perceiver-style resampling. `Chisel` improves accuracy by up to +22.62 pp on a single category and +17.37 pp on overall accuracy, while cutting visual tokens by up to 83.9% (48.8% on average). Under pipelined execution, `Chisel` boosts end-to-end throughput by 34–178% across the three backbones. Notably, `Chisel` improves accuracy in 16 of 18 cells, whereas competing methods vary substantially across models and benchmarks.

2 Related Work

Existing efforts to handle high-resolution images in VLMs cluster along two complementary axes, as summarized in Table 1: reducing the number of visual tokens consumed by the VLM to ease the compute and memory bottleneck (§2.1), and preserving or expanding the input to enhance accuracy on fine-grained content (§2.2). `Chisel` addresses both goals jointly—cutting token counts while concentrating capacity on query-relevant regions.

2.1 Token Reduction

Resampling-based compression. Some VLMs compress the many patch features produced by the vision encoder into a predetermined, small set of learnable queries via cross-attention, thereby bounding the number of visual tokens fed to the LLM. Flamingo (Alayrac et al., 2022) introduces a perceiver resampler that distills patch features into 64 latent queries; Qwen-VL (Bai et al., 2023) adopts a single-layer cross-attention adapter with 256 queries; and MiniCPM-V (Yao et al., 2024) employs a similar resampling stage. These designs bound memory, but the compression bottleneck irreversibly discards fine-grained detail and is fixed at the architecture level, requiring training to adjust.

Vision token pruning. A separate line of work prunes visual tokens within the language model itself by exploiting attention patterns. FastV (Chen et al., 2024) drops low-attention image tokens after the early LLM layers, observing that visual attention rapidly decays. SparseVLM (Zhang et al.,

Table 1: Comparison of related work for high-resolution VLM processing. Methods are grouped by their primary goal—token reduction versus high-resolution handling without reduction.

Method	Primary Goal	Stage	Approach
`Chisel` (Chisel)	Token reduction	Pre-ViT	ROI extraction + fusing
CROP-PLC (Guo et al., 2025)	Token reduction	Post-ViT, Pre-LLM	Localization-guided adaptive compression
CROP-ILP (Guo et al., 2025)	Token reduction	Inside LLM	Localization-guided pruning
Flamingo (Alayrac et al., 2022)	Token reduction	Post-ViT, Pre-LLM	Perceiver Resampler
Qwen-VL (Bai et al., 2023)	Token reduction	Post-ViT, Pre-LLM	Cross-attention adapter
MiniCPM-V (Yao et al., 2024)	Token reduction	Post-ViT, Pre-LLM	Resampler
FastV (Chen et al., 2024)	Token reduction	Inside LLM	Attention-based pruning
SparseVLM (Zhang et al., 2024)	Token reduction	Inside LLM	Query-guided pruning
TEVA (Jiang et al., 2025)	High-res accuracy	Pre-ViT	Dual-resolution input
SEAL (Wu and Xie, 2024)	High-res accuracy	Multi-pass	Iterative zoom-in search
DC^2 (Wang et al., 2025)	High-res via aggregation	Multi-pass	Recursive division + text aggregation

2024) selects visually grounded text tokens, scores image tokens via a query-guided criterion, and recycles pruned tokens to limit information loss.

Query-aware region selection. Closer to our work in spirit, several methods detect query-relevant image regions and feed them selectively to the VLM. CROP (Guo et al., 2025) uses a lightweight 2B VLM to localize a coarse-grid contextual region, then reduces tokens via either a learnable Pre-LLM Compression module (CROP-PLC) or a training-free Inner-LLM Pruning scheme (CROP-ILP).

Despite their distinct mechanisms, existing token-reduction methods retain structural limitations for truly high-resolution inputs: resampling is fixed at the architecture level and indiscriminately removes detail (Flamingo, Qwen-VL, MiniCPM-V); LLM-stage pruning operates only after the vision encoder, leaving ViT-stage token explosion unresolved (FastV, SparseVLM); and query-aware region selection still processes the full image through the main VLM and an additional localizer before reduction, while collapsing multiple scattered targets into a single bounding box (CROP). `Chisel`, in contrast, reduces tokens upstream of the ViT in a backbone-agnostic manner, leaving the encoder to operate within a controlled token budget while high-resolution detail in query-relevant regions is preserved. As a pre-VLM stage, `Chisel` is complementary to existing inner-LLM token-pruning methods and can be stacked with them for compounding gains.

2.2 Compute Reallocation for High-Resolution Reasoning

A separate line of work improves high-resolution accuracy without reducing—and often by adding—visual compute, either by enriching the VLM input with high-resolution crops or by invoking the VLM repeatedly over selected regions.

TEVA (Jiang et al., 2025) pairs an LLM with Grounding-DINO (Liu et al., 2024c) to localize query-relevant targets, then feeds the VLM a token sequence that combines a low-resolution view of the full image for global context with high-resolution crops of those targets for local detail—spending more tokens where they matter and fewer elsewhere under a fixed budget.

DC^2 (Wang et al., 2025) is a training-free pipeline that recursively *Divides* a high-resolution image into patches, prompts an MLLM to describe each patch (*Conquer*), and *Combines* the resulting descriptions to enrich the VLM's understanding of the full image.

SEAL (Wu and Xie, 2024) takes a search-based route: it uses an LLM's commonsense priors to predict where a query target is likely to appear, then iteratively zooms into candidate regions, re-querying the VLM at each step until the target is resolved at sufficient resolution.

Across these three lines, accuracy gains on high-resolution inputs come either at the cost of architectural changes that hinder portability, or repeated VLM forward passes whose count grows with patch count or search depth; none of them addresses the underlying token explosion at the ViT stage. `Chisel`, in contrast, reduces visual tokens before the main VLM forward pass in a backbone-agnostic manner, requiring neither architectural modification nor repeated VLM passes. By concentrating tokens on query-relevant regions at the input stage, `Chisel` mitigates ViT-stage token explosion while preserving high-resolution detail where needed.

3 Methodology

The `Chisel` framework is designed to chisel an input image down to only those regions bearing substantive relevance to a given query, discarding all else. To this end, we develop an incisive procedure that extracts regions of interest (ROIs) directly from the query, carefully engineered to recover targets that naive grounding tends to miss (§3.1). Since cropping risks discarding not only essential surrounding context but also inter-object spatial relationships, `Chisel` further fuses the ROI fragments under a layout-preserving scheme in which the stitched representation mirrors rather than distorts the source layout (§3.2). Finally, to ensure that this added preprocessing does not inflate end-to-end latency, `Chisel` introduces a pipelining strategy that decomposes the preprocessing into fine-grained stages and slots them into the idle bubbles of the existing inference pipeline, hiding their cost behind concurrent execution (§3.3). Figure 1 illustrates the overall algorithmic flow of the Chisel framework, highlighting how the image is compressed prior to VLM processing.

3.1 Incisive ROI Extraction

The most straightforward approach to ROI extraction is to apply a simple parser (e.g., spaCy) to the query—extracting noun phrases and grounding each to a bounding box in the image. This baseline, however, suffers from a fundamental limitation: it cannot recover ROIs for phrases carrying abstract or conceptual meaning. Consider the phrase *"waiting time"*: a query that hinges on it must localize concrete referents such as displays, signs, boards, or clocks—none of which appear as surface nouns in the query itself. A direct surface-noun-to-object mapping cannot bridge this *semantic indirection*.

`Chisel` bridges this gap with an SLM (e.g., Qwen2.5-1.5B-Instruct) that emits the target noun phrases directly from the query. Beyond supplying concrete referents for abstract terms, this approach naturally filters out surface nouns that prove unnecessary for grounding—a pruning effect the parser-based baseline cannot provide.

The resulting target list is then fed as a text prompt to an open-vocabulary detector (GroundingDINO-T (Liu et al., 2024c)), which returns the bounding boxes of the corresponding image regions. In high-resolution inputs, query-relevant content may be confined to only a small, localized portion of the image; in such cases, missing a target (false negative) hurts accuracy far more than including an irrelevant region (false positive). Given this asymmetry, we adopt a recall-first policy by setting the detection confidence threshold to a low, empirically tuned value of 0.25.

3.2 Context- and Layout-Faithful Fusing

The extracted ROIs identify the anchor regions to retain; how to splice them coherently is a separate design question. A naive crop-and-concatenate, for instance, is doubly lossy: tight crops strip the surrounding context many targets need to be interpreted, and the inter-object geometry collapses once tiles are stacked. `Chisel` therefore fuses the ROIs through a context- and layout-faithful scheme that merges clustered, overlapping objects into a single coarse crop to preserve the semantics of their joint appearance, while arranging all ROIs into a composite that preserves the source layout.

When forwarding the extracted crops to the VLM, a straightforward option is to feed each crop independently. This approach, however, has two drawbacks. First, it scales VLM inference cost linearly with the number of crops. Each crop requires its own vision-encoder pass and adds a separate token sequence to the LLM input, causing compute, latency, and KV-cache footprint to grow proportionally. Second, it discards the spatial relationships among the crops, which are often essential for the model to produce accurate answers.

`Chisel` merges the crops into a single composite image that is as compact as possible yet spatially faithful to the original layout. Simpler alternatives would be to concatenate the crops or to take a single large crop enclosing all bounding boxes; however, the former discards inter-object positions and the latter reintroduces substantial non-ROI regions, negating the token savings. `Chisel` instead consolidates the crops via orthogonal beam splicing: it stripes out every row and column that contains no ROIs, and splices the surviving rows and columns back together into a single composite image. Figure 2 illustrates the procedure of orthogonal beam splicing on an example with four bounding boxes. `Chisel` first removes every row containing no bounding box and splices the survivors, then performs the same operation column-wise. The result is a compact composite preserving the relative positions of all targets.

This scheme has the following benefits. First, feeding a single compact composite to the LLM eliminates both the multi-forward overhead and the

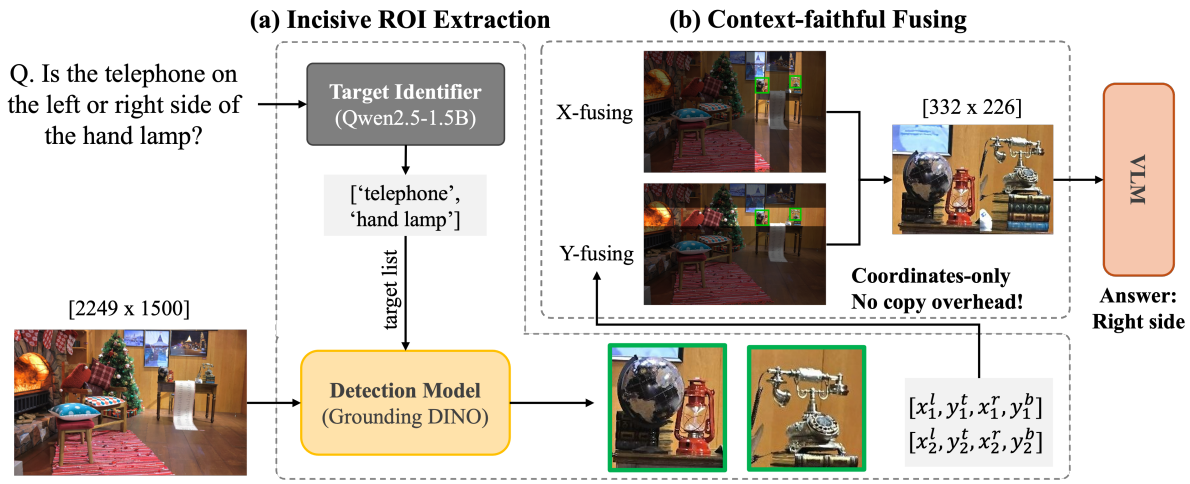


Figure 1: Algorithmic flow of `Chisel`. (1) **Incisive ROI Extraction** (§3.1): query nouns are grounded in the image to yield ROI bounding boxes; the abstract semantics that surface nouns alone cannot capture are recovered by a lightweight LLM (Qwen2.5-1.5B-Instruct). (2) **Context-Faithful Fusing** (§3.2): the detected regions are cropped and fused while preserving the relative spatial relationships among objects, yielding a single composite image as input to the VLM. Note that, for this 2249×1500 example, `Chisel` feeds the VLM a composite as small as 333×226, obtained by cropping and stitching only the query-relevant regions at their original resolution—rather than downscaling the entire image and losing fine-grained detail.

token explosion. Second, inheriting the source's row–column ordering preserves the relative spatial positions among objects. Third, beam splicing yields a rectangular composite without any padding or resizing; instead, it enriches the semantic context of adjacent or overlapping objects by retaining the in-between background regions.

3.3 Pipelined Heterogeneous Execution

The preprocessing of `Chisel` before the VLM forward pass inevitably adds cost and end-to-end latency. Although this overhead can be mitigated through the reduction in LLM inference time induced by token compression, such compensation is not uniformly realized across inference regimes. In particular, when the LLM aggressively shrinks the token count via input resizing, or when a SLM is employed, `Chisel` delivers substantially higher accuracy but may suffer a net latency penalty due to the preprocessing cost.

To address this limitation, we propose a pipelined execution scheme that hides the preprocessing overhead behind concurrent VLM execution. The key idea is to decompose `Chisel` into two sub-tasks: a compute-bound Extraction stage that performs target identification and visual grounding via an SLM, and a memory-bound Fusing stage that composes the image through orthogonal beam splicing. We dispatch Extraction to the GPU and Fusing to the CPU, enabling concurrent execution of the two. Moreover, the GPU-resident Extraction tasks are scheduled into the idle bubbles arising from the VLM execution of preceding requests, thereby masking the preprocessing cost by harvesting otherwise-wasted resources.

VLM execution can be decomposed into as many as three pipeline stages (Vision Encoder, Prefill, and Decode), or into coarser groupings thereof. Extraction sub-tasks are then scheduled into the idle SM slots exposed by an in-flight VLM execution, hiding their cost behind concurrent stages. GPU compute resources are managed at the granularity of Streaming Multiprocessors (SMs); a task can be dispatched to an SM only when the SM has sufficient capability to meet its resource demand.

Figure 3 illustrates an example in which each light-blue lane represents the device's full SM capacity, and the colored boxes inside denote pipeline stages, with width indicating duration and height indicating SM occupancy. The unfilled area of each lane is idle SM capacity available to concurrent stages. When a pending task's resource requirement exceeds the SM's remaining capacity, the task cannot be scheduled and the resources sit idle. `Chisel` exploits these idle slots by injecting extraction sub-tasks into them, thereby improving GPU utilization while reducing user-perceived latency and increasing throughput. As a result, while the sequential execution exposes the full latency overhead introduced by `Chisel` to the user, the pipelined execu-

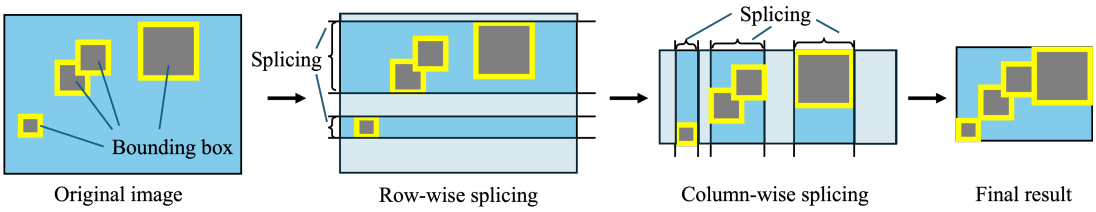


Figure 2: Orthogonal beam splicing. `Chisel` strips ROI-free rows and then ROI-free columns, splicing the survivors into a compact composite that preserves the relative positions of all targets.

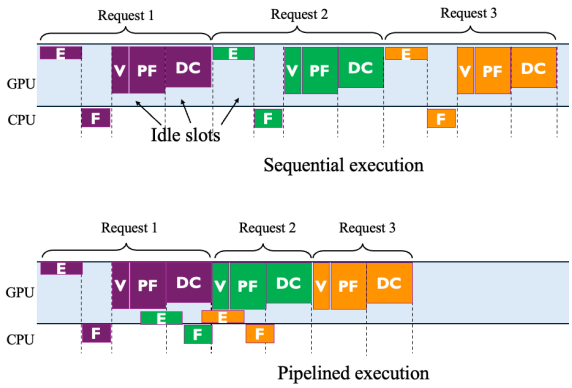


Figure 3: Execution timelines for three consecutive requests. E: Extraction, F: Fusing, V: Vision Encoder, PF: Prefill, DC: Decode.

tion successfully hides this additional time behind the VLM execution.

4 Experiments

We evaluate `Chisel` on two high-resolution VQA benchmarks across three backbone VLMs, reporting accuracy (§4.2), throughput alongside the visual-token footprint (§4.3), and ablations of its ROI extraction and fusing (§4.4).

4.1 Setup

Models and Benchmarks. As `Chisel` is backbone-agnostic, we evaluate it across three VLMs spanning three vision-encoding methods: Qwen2.5-VL-3B (Bai et al., 2025) with native dynamic resolution; the tile-based LLaVA-OneVision-7B (Li et al., 2024a); and the resampler-based MiniCPM-V-2.6 (Yao et al., 2024). For evaluation, we use two visual question answering (VQA) benchmarks designed specifically for high-resolution inputs. HR-Bench (Wang et al., 2025) contains 1600 multiple-choice questions over images at 4K and 8K resolutions, spanning fine-grained single-instance perception and cross-instance reasoning. V^*Bench (Wu and Xie, 2024) comprises 191 multiple-choice questions on natural high-resolution photographs, targeting attribute recognition and spatial reasoning over small or distant objects. Both benchmarks are scored via option-NLL (negative log-likelihood) to eliminate option-order bias.

Baselines. We compare `Chisel` against four baselines. Vanilla feeds the original image as input. FastV (Chen et al., 2024) applies attention-based visual token pruning at the early LLM layers. CROP-ILP (Guo et al., 2025) uses a localization module (Qwen2-VL-2B) to select contextual regions and applies a training-free inner-LLM pruning scheme. DC^2 (Wang et al., 2025) is a training-free pipeline that recursively divides a high-resolution image into four sub-patches until they reach the vision encoder's resolution, then describes each patch via an MLLM and aggregates the descriptions. Compatibility and reproducibility caveats are summarized in Appendix A.

Hardware and Metrics. All experiments are run on a single NVIDIA RTX PRO 6000 (Blackwell) GPU; the full hardware/software environment is reported in Appendix B (Table 3). Accuracy is measured with option-level NLL—the fraction of multiple-choice questions for which the model assigns the lowest negative log-likelihood to the gold option; throughput is queries per second under steady-state batched serving.

4.2 Accuracy

Overall Analysis. Table 2 reports per-method accuracy on two high-resolution VQA benchmarks across three backbone VLMs. Within each benchmark, the first category targets perception of single objects: FSP (HR-Bench) tests fine-grained attributes and details, and Attr (V^*Bench) tests attribute recognition. The second category targets reasoning across multiple objects: FCP (HR-Bench) covers spatial relations, comparisons, and counting,

Model	Method	HR-Bench			V^*Bench		
		FSP	FCP	Overall	Attr	Spatial	Overall
Qwen2.5-VL-3B	Vanilla	78.75	45.88	62.31	75.65	57.89	68.59
	FastV	**82.50 (+3.75)**	44.00 (-1.88)	63.25 (+0.94)	**84.35 (+8.70)**	60.53 (+2.64)	**74.87 (+6.28)**
	CROP-ILP	26.62 (-52.12)	28.75 (-17.13)	27.69 (-34.63)	41.74 (-33.91)	56.58 (-1.32)	47.64 (-20.94)
	DC^2	57.00 (-21.75)	47.25 (+1.38)	52.12 (-10.19)	65.22 (-10.43)	**61.84 (+3.95)**	63.87 (-4.71)
	CHISEL	79.12 (+0.37)	**51.75 (+5.87)**	**65.44 (+3.13)**	66.96 (-8.69)	60.53 (+2.64)	64.40 (-4.19)
LLaVA-OV-7B	Vanilla	56.50	47.75	52.12	74.78	57.89	68.06
	FastV	59.88 (+3.38)	48.75 (+1.00)	54.31 (+2.19)	73.91 (-0.87)	67.11 (+9.22)	71.20 (+3.14)
	CROP-ILP	50.00 (-6.50)	40.38 (-7.38)	45.19 (-6.94)	49.57 (-25.22)	60.53 (+2.63)	53.93 (-14.14)
	DC^2	64.62 (+8.13)	49.38 (+1.62)	57.00 (+4.87)	**78.26 (+3.48)**	67.11 (+9.21)	**73.82 (+5.76)**
	CHISEL	**74.25 (+17.75)**	**52.38 (+4.63)**	**63.31 (+11.19)**	75.65 (+0.87)	**71.05 (+13.16)**	73.82 (+5.76)
MiniCPM-V-2.6	Vanilla	59.50	45.25	52.38	60.87	64.47	62.30
	FastV	48.00 (-11.50)	40.75 (-4.50)	44.38 (-8.00)	46.09 (-14.78)	57.89 (-6.58)	50.79 (-11.51)
	CROP-ILP	47.75 (-11.75)	40.50 (-4.75)	44.12 (-8.26)	49.57 (-11.30)	52.63 (-11.84)	50.79 (-11.51)
	DC^2	44.00 (-15.50)	38.62 (-6.63)	41.31 (-11.07)	49.57 (-11.30)	55.26 (-9.21)	51.83 (-10.47)
	CHISEL	**82.12 (+22.62)**	**57.38 (+12.13)**	**69.75 (+17.37)**	**73.04 (+12.17)**	**65.79 (+1.32)**	**70.16 (+7.86)**

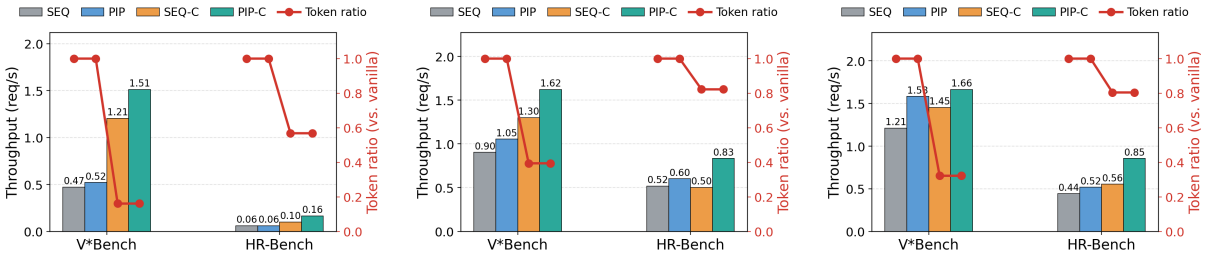
Table 2: Accuracy results (%) per benchmark and category. **HR-Bench**: FSP (Fine-grained Single-instance Perception), FCP (Fine-grained Cross-instance Perception). V^***Bench**: Attr (Attribute Recognition), Spatial (Spatial Relationship). **Bold** indicates the best result in each column within a model group; underlined entries indicate the second-best. Parentheses indicate absolute change against the same-model vanilla baseline.

and Spatial (V^*Bench) covers spatial relationships. Overall is the per-question accuracy aggregated over all categories within a benchmark.

As shown in the table, `Chisel` achieves the highest accuracy in 12 of 18 (model, benchmark, category) cells and ranks first or second in 16 of them. Competing methods either prune a statically chosen fraction of tokens regardless of the query or perform incomplete cropping that misses important objects, producing accuracy that varies widely across settings. `Chisel`, by contrast, applies query- and image-aware extraction and fusing, yielding consistently high accuracy in nearly every cell.

`Chisel` is most effective on backbones that most drastically compress their visual inputs query-agnostically, such as MiniCPM-V-2.6. It uses a Perceiver-style resampler to collapse visual features into a small, fixed token pool, and thus loses much of the fine-grained detail in its vanilla representation; `Chisel` crops the image to only its query-relevant regions before the resampler is applied, allowing much of that detail to survive resampling. As a result, `Chisel` raises overall accuracy by 17.37 pp on HR-Bench (52.38 to 69.75) and 7.86 pp on V^*Bench (62.30 to 70.16). In contrast, other methods rigidly prune tokens inside the LLM by a preset budget, blind to how much information has already been lost from the input; on heavily compressed inputs such as the above, this inevitably sacrifices substantial accuracy. LLaVA-OV-7B uses tile-based encoding without comparable compression and therefore retains moderate detail even in its vanilla representation; even here, `Chisel` delivers consistent gains of 11.19 pp on HR-Bench (52.12 to 63.31) and 5.76 pp on V^*Bench (68.06 to 73.82). Qwen2.5-VL-3B scales its token count proportionally to input resolution, leaving its vanilla version with already-high accuracy. Even in this setting, `Chisel` trims the unnecessary regions and reduces visual tokens by up to 83.9% while still improving accuracy in 4 of the 6 cells (see Appendix C for token-reduction details). The lone exception is V^*Bench, where Qwen2.5-VL-3B's native dynamic-resolution encoding already captures these moderate-resolution images at full density, leaving `Chisel` little super-resolution headroom to add (Attr -8.69, Overall -4.19 pp).

Per-category analysis. For single-object queries (FSP on HR-Bench, Attr on V^*Bench), preserving precise detail of small targets is critical to accuracy. `Chisel` captures small but salient objects at high resolution via ROI extraction, delivering the largest gains on backbones whose indiscriminate compression destroys this detail (+22.62 pp on HR-Bench FSP and +12.17 pp on V^*Bench Attr with MiniCPM-V-2.6). In contrast, FastV applies attention-based pruning that discards a fixed token fraction, hurting accuracy when small-target tokens are missed (14.78 pp below vanilla on V^*Bench Attr with MiniCPM-V-2.6). Although FastV performs favorably on token-rich, high-resolution in-



(a) Qwen2.5-VL-3B (b) LLaVA-OV-7B (c) MiniCPM-V-2.6

Figure 4: Throughput (req/s) across two benchmarks (V^*Bench, HR-Bench) for three backbone VLMs. **SEQ**: vanilla baseline; **PIP**: vanilla with pipelining only; **SEQ-C**: `Chisel` without pipelining; **PIP-C**: full `Chisel` with pipelining (proposed). The red line is the visual-token ratio vs. vanilla (right axis).

puts such as Qwen2.5-VL, its effectiveness varies considerably across backbones with different visual compression schemes. DC^2 loses small targets that fall on quadrant boundaries or are judged unimportant, producing inconsistent single-object accuracy.

For multi-object reasoning (FCP, Spatial), preserving multiple regions and their relative layout is what matters. `Chisel` composes all detected ROIs into a single layout-preserving composite that retains relational cues, outperforming vanilla in every cell. FastV benefits Spatial since multi-object tokens retain rough left/right ordering even after pruning; DC^2's multi-region aggregation similarly favors relational queries, but recursive splitting fragments the global layout inconsistently.

4.3 Throughput with Pipelined Execution

Figure 4 reports throughput across the four execution modes on three backbones, together with the visual-token ratio for each setting. Overall, when chiseling is paired with pipelined execution, it introduces essentially no overhead and consistently improves throughput, by an average of 178% on Qwen2.5-VL-3B, 46% on LLaVA-OV-7B, and 34% on MiniCPM-V-2.6.

Under sequential execution, the throughput impact of chiseling depends heavily on how aggressively the token count is reduced. For example, on V^*Bench with Qwen2.5-VL-3B and LLaVA-OV-7B, where chiseling cuts token counts by 83.9% and 60.6% respectively, sequential execution still delivers throughput gains of 155% and 44%. The reduced visual-token count directly shortens VLM forward time, more than offsetting the added image preprocessing cost. When token reduction is more modest, throughput can dip slightly (e.g., LLaVA-OV-7B on HR-Bench, where SEQ-C falls 3% below SEQ), but pipelining masks the chiseling overhead in such cases, leading to throughput improvements across all settings. Notably, despite these drastic throughput gains, `Chisel` improves accuracy in most settings, and even on token-rich, high-resolution Qwen2.5-VL-3B, retaining just $\sim$16% of the visual tokens costs only a 4.2 pp drop in overall accuracy.

4.4 Ablation Study

We ablate three components of `Chisel`'s ROI extraction and fusing (Table 6): the *target identifier* (SLM vs. spaCy), the *image-fusing* policy (composite vs. individual crops), and the *detection model*. Fusing and the SLM identifier drive most of `Chisel`'s relational-category gains ($+11.6$ and $+2.4$ pp), while accuracy further depends on the detector: GDINO-T (our default) leads in five of six cells, with OmDet-T trailing by 3.6–7.6 pp on HR-Bench and YOLO-World-S the weakest. We defer full ablation tables and analysis to Appendix D.

5 Conclusion

We presented `Chisel`, a backbone-agnostic pipeline that compresses high-resolution images into a layout-preserving composite of query-relevant regions before VLM inference. Across three backbones, `Chisel` improves accuracy by +7 pp on average while cutting visual tokens by 48.8% and boosting throughput by 86%. The gains are largest on backbones that compress visual inputs most aggressively—such as the resampler-based MiniCPM-V-2.6—precisely where prior token-reduction methods sacrifice the most detail. Operating upstream of the vision encoder, `Chisel` complements inner-LLM pruning and needs no retraining, making it a drop-in front-end for high-resolution VLM serving.

6 Limitations

While Chisel preserves the relative layout among ROIs through orthogonal beam splicing, it does not retain absolute pixel distances after removing ROI-free rows and columns. As a result, queries requiring precise metric reasoning may remain challenging in some cases, particularly in domains such as visual navigation or robotic perception. In such settings, the original ROI coordinates could be provided as auxiliary input, or the splicing stage could be replaced with geometry-preserving downsampling. We leave these extensions for future work.

Chisel also depends on two external components: an SLM for extracting target noun phrases and an open-vocabulary detector for localizing them. Consequently, overall performance is influenced by the quality of both components. As shown in Table 6, different detectors yield noticeable accuracy differences even under identical settings, suggesting that the choice of detector and SLM remains an important practical consideration.

Appendix

A Reproducibility Notes for Baselines

CROP-ILP: Reproduction Caveats. Faithful reproduction of CROP-ILP is subject to practical constraints. The public codebase only partially documents the end-to-end execution path, checkpoint provenance, and the exact implementation details used for training and evaluation. The CROP-ILP results reported in this paper are therefore best-effort reproductions within the information that is publicly available, and may differ from the official numbers reported in the original paper.

CROP-ILP: Training Data Composition Mismatch. We were unable to exactly match the 610k-sample training mixture described in CROP's Table 1 (410k region-localization + 200k instruction). The public training pipeline appears to use a different region source and a capped subset of instruction data, which may affect the effective sample distribution and downstream accuracy.

TEVA: Excluded from Comparison. TEVA (Jiang et al., 2025) (Token-Efficient Vision-Language Model) is a learning-based, dynamic region-proposal method for high-resolution VLMs that we initially considered for inclusion as a baseline. However, its official code, training data, and model checkpoints are unavailable, making independent reproduction infeasible; we therefore exclude TEVA from our quantitative comparison.

B Implementation Details of the Pipelined Execution

Figure B illustrates the request flow through the three pipeline stages and the data and GPU resource sharing among them. Table 3 summarizes the hardware and software environment used for all experiments. Since flash-attn (Dao et al., 2022) does not yet support the Blackwell architecture, we fall back to PyTorch's built-in SDPA; all VLMs are loaded in bfloat16 and decoded greedily for reproducibility.

Component	Specification
GPU	NVIDIA RTX PRO 6000 Blackwell, 96 GB GDDR7, 300 W TDP
CPU	AMD EPYC 9124 (16C / 32T)
Host memory	192 GB DDR5
Driver / CUDA	580.119.02 / 13.0
PyTorch	2.8.0+cu128
transformers	4.53.2
Attention	PyTorch SDPA
Precision	bfloat16

Table 3: Experimental environment.

Process Layout and Backbone Sharing. We realize the four pipeline stages—Detect, comprising Extraction (GPU) and Fusing (CPU), Prefill (GPU, including vision encoding), and Decode (GPU)—as long-lived processes that load their respective models once at startup and remain resident across requests. Naively running Prefill and Decode in separate processes would require two copies of the LLM backbone on the GPU; we eliminate this duplication by sharing the same physical weight buffer through CUDA IPC handles, registering it in both address spaces and paying the multi-gigabyte VRAM cost only once. The GPU work across all three stages further shares a single GPU context through NVIDIA MPS, whose time-slice scheduling reassigns idle SMs to whichever stage has pending work.

Inter-stage image transfer. Because Chisel runs the Fusing stage on the CPU, the fused image must cross the CPU–GPU boundary to reach Prefill. We minimize this cost using shared memory: rather than the multiple kernel-mediated copies of a standard IPC path, we allocate a POSIX shared-memory ring that both producer and consumer

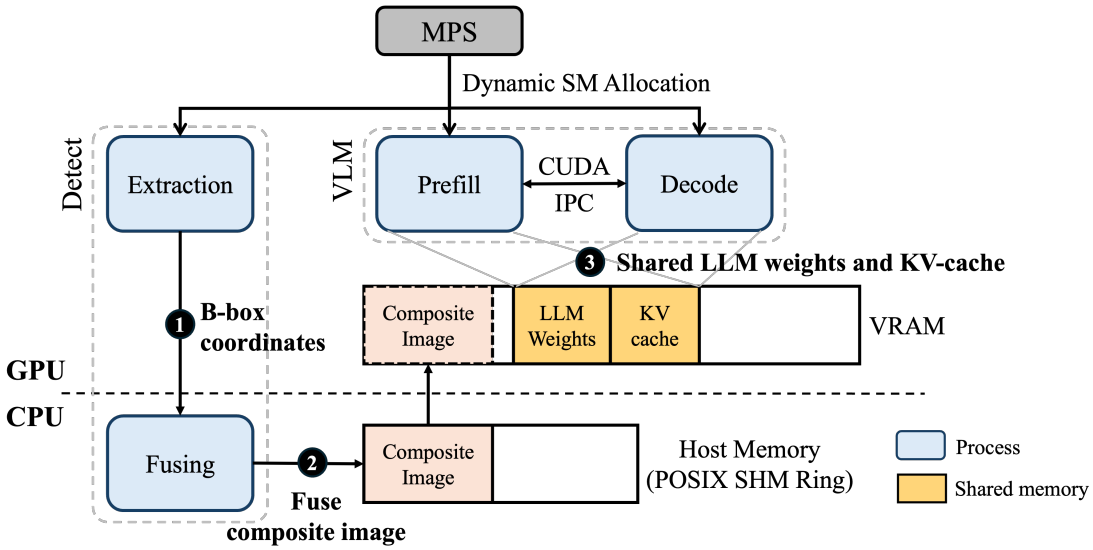


Figure 5: Pipelined heterogeneous execution of `Chisel` across the Detect, Prefill, and Decode stages (Appendix B).

`mmap` into a common region, leaving only a single host-to-device DMA transfer.

KV cache handoff. After Prefill, the KV cache resides entirely in GPU memory; only a handle (pointer plus metadata) crosses the IPC channel to Decode. Decode then performs autoregressive generation directly on this in-place cache without any further data movement.

CPU-side composite synthesis. The composite synthesis pipeline is implemented with NumPy and `cv2.resize`, both of which are bounded by structured block-level memory copies. Running this stage on CPU keeps it off the GPU's critical path and lets it overlap naturally with the GPU-bound stages under concurrency.

C Additional Performance Results.

Token Count. Table 4 reports the per-query visual token count. For `Chisel`, we measure the count entering the VLM, since it compresses before VLM inference. For FastV and CROP-ILP, we report the count after their pruning at the second LLM layer. DC^2 does not prune and matches the vanilla count. FastV prunes a static fraction by attention score (50% by default), giving a roughly 50% reduction across settings. CROP-ILP localizes a single crop and prunes the remaining tokens inside the LLM down to a *preset* pruning ratio—we set it to 77.8%, following CROP—so its reduction is near-uniform (≈78%) by construction rather than input-dependent. `Chisel`, in contrast, adapts to the input. On V^*Bench it cuts visual tokens by more than 60% for every backbone (60.6–83.9%; e.g. 67.8% on MiniCPM-V-2.6 and 83.9% on Qwen2.5-VL-3B); the savings are smaller on HR-Bench (17.8–43.2%), where the much higher 4K–8K resolution makes the open-vocabulary detector localize loosely—often returning large boxes, or missing the target and falling back to the full image—so the composite retains much of the input. Either way, `Chisel` sizes its reduction to each input rather than pruning a fixed fraction, thereby avoiding the over-pruning of informative regions; as a result, unlike the other methods, whose accuracy fluctuates across models and benchmarks, `Chisel` stays consistently accurate in most settings.

Pipelining Analysis. Table 5 reports throughput, GPU SM utilization, idle time, and energy per sample (J/smp) across the four execution modes. PIP-C achieves the highest throughput and the lowest J/smp in nearly every setting. Pipelining is the key driver: comparing SEQ-C with PIP-C, idle time drops substantially (e.g., 18.7% to 7.6% on Qwen2.5-VL-3B with V^*Bench), SM utilization rises by 43–77% across the six configurations, and throughput improves by 15–66%. The shorter wall-clock time and higher GPU utilization together lower energy per sample by 19–30% over SEQ-C and by 48% on average over the vanilla SEQ baseline. Importantly, higher SM utilization and lower idle% do not always indicate better efficiency. On HR-Bench with Qwen2.5-VL-3B—whose token count scales with input resolution—the vanilla baseline runs an enormous full-resolution prefill (16,224 tokens) that saturates the GPU, so it shows

Model	Method	HR-Bench	V^*Bench
Qwen2.5-VL-3B	Vanilla	16,224	4,472
	FastV	8,183 (-49.6%)	2,291 (-48.8%)
	CROP-ILP	3,463 (-78.7%)	993 (-77.8%)
	DC^2	–	–
	CHISEL	9,208 (-43.2%)	720 (-83.9%)
LLaVA-OV-7B	Vanilla	7,303	7,320
	FastV	3,709 (-49.2%)	3,701 (-49.4%)
	CROP-ILP	1,622 (-77.8%)	1,625 (-77.8%)
	DC^2	–	–
	CHISEL	6,001 (-17.8%)	2,881 (-60.6%)
MiniCPM-V-2.6	Vanilla	585	583
	FastV	311 (-46.8%)	314 (-46.1%)
	CROP-ILP	130 (-77.8%)	129 (-77.8%)
	DC^2	–	–
	CHISEL	470 (-19.7%)	188 (-67.8%)

Table 4: Average visual token count per query on HR-Bench and V^*Bench. Parentheses indicate the relative change against the same-model vanilla baseline.

high SM utilization (0.82–0.87) and near-zero idle yet remains slow (3,345 s wall). `Chisel` cuts this work by 43%, so PIP-C exhibits lower utilization (0.46) and higher idle (20.5%) while still delivering 2.7× the throughput (0.060 → 0.164 req/s) at far lower latency. SM utilization and idle% should thus be read together with throughput rather than in isolation.

D Ablation Study: Full Results

We ablate `Chisel`'s ROI extraction and fusing along three axes, toggling one at a time around the full configuration (Table 6): the *target identifier* that decides *what* to localize, the *image-fusing* policy that decides whether the localized ROIs are fed to the VLM *individually* or composited into a single layout-preserving image, and the *detection model* that grounds the targets into bounding boxes.

`Chisel-noun` vs. `Chisel`. We denote by `Chisel-noun` the lightweight variant that extracts detection targets with a spaCy (noun-phrase parser) instead of the auxiliary SLM (§3.1): it grounds the query's surface nouns directly and runs CPU-only. Table 6 compares it against the SLM-based `Chisel`. The SLM attains the higher Overall on most settings, with its advantage concentrated in the cross-instance and spatial categories (FCP, Spatial), where implicit or multiple targets must be inferred, exactly what surface nouns alone miss. `Chisel-noun` nonetheless remains competitive and even leads on a few single-instance settings, making it a practical CPU-only fallback when the auxiliary LLM is unavailable.

Model	Sc.	SM_util	idle%	thrpt	wall(s)	J/smp
V^*Bench						
Qwen2.5-VL-3B	SEQ	0.450	27.6	0.472	404.7	507
	PIP	0.495	23.6	0.521	366.7	486
	SEQ-C	0.272	18.7	1.205	158.5	136
	PIP-C	0.409	**7.6**	**1.512**	126.3	**108**
LLaVA-OV-7B	SEQ	0.556	39.7	0.903	211.6	231
	PIP	0.692	26.8	1.055	181.0	212
	SEQ-C	0.437	13.6	1.301	146.8	145
	PIP-C	0.742	**4.3**	**1.622**	117.7	**117**
MiniCPM-V-2.6	SEQ	0.379	48.2	1.211	157.7	139
	PIP	0.521	35.2	1.582	120.8	117
	SEQ-C	0.332	15.2	1.453	131.5	115
	PIP-C	0.552	**7.5**	**1.664**	114.8	**91**
HR-Bench						
Qwen2.5-VL-3B	SEQ	0.822	2.1	0.060	3345.2	4591
	PIP	0.871	**0.0**	0.060	3334.9	4580
	SEQ-C	0.322	39.7	0.101	1978.1	2237
	PIP-C	0.460	20.5	**0.164**	1217.7	**1645**
LLaVA-OV-7B	SEQ	0.354	60.7	0.518	386.0	313
	PIP	0.429	55.1	0.601	333.0	286
	SEQ-C	0.301	45.3	0.503	397.5	323
	PIP-C	0.532	**31.3**	**0.834**	239.7	**228**
MiniCPM-V-2.6	SEQ	0.152	77.2	0.443	451.9	271
	PIP	0.170	78.6	0.519	385.3	239
	SEQ-C	0.188	54.5	0.555	360.7	247
	PIP-C	0.325	**47.5**	**0.855**	234.0	**173**

Table 5: Pipelining analysis across backbones and benchmarks. **SEQ**: vanilla baseline; **PIP**: vanilla with pipelining only; **SEQ-C**: `Chisel` without pipelining; **PIP-C**: full `Chisel` (proposed). Throughput and J/smp are measured over the serving window (model-load excluded) at 200 requests (V^*Bench uses all 191); SM_util and idle% are from the per-stage GPU trace. **Bold** = best per model×benchmark (highest thrpt, lowest idle%, lowest J/smp). PIP-C attains the lowest idle and highest throughput in nearly every setting.

Individual vs. layout-preserving fusing. After localization, `Chisel` composes the detected ROIs into a single layout-preserving image instead of forwarding each crop separately (§3.2). This helps the relational categories most: across the three backbones, fusing raises FCP (HR-Bench) by 8.00–11.38 pp and Spatial (V^*Bench) by 6.58–22.37 pp over feeding crops individually.

Detection model. We finally vary the open-vocabulary detector that grounds the SLM's target phrases, holding the SLM and the fusing scheme fixed. GroundingDINO-Tiny (GDINO-T)—`Chisel`'s default—attains the highest Overall in five of the six (model, benchmark) cells, with OmDet-Turbo (OmDet-T) ahead only on Qwen2.5-VL-3B / V^*Bench; on HR-Bench, OmDet-T trails GDINO-T by 3.6–7.6 pp across the three backbones. YOLO-World-S is the weakest of the three: it fails to localize the small targets in these benchmarks tightly, trailing GDINO-T by roughly 4 pp on V^*Bench and 9 pp on HR-Bench on aver-

Ablation	Model	Method	HR-Bench			V^*Bench		
			FSP	FCP	Overall	Attr	Spatial	Overall
Target Identifier	Qwen2.5-VL-3B	`Chisel-noun`	81.25	48.25	64.75	70.43	60.53	**66.49**
		`Chisel`	79.88	50.88	**65.38**	66.96	60.53	64.40
	LLaVA-OV-7B	`Chisel-noun`	74.25	48.38	**61.31**	74.78	68.42	72.25
		`Chisel`	71.88	49.75	60.81	74.78	71.05	**73.30**
	MiniCPM-V-2.6	`Chisel-noun`	78.38	48.12	63.25	74.78	63.16	70.16
		`Chisel`	80.75	53.00	**66.88**	73.91	65.79	**70.68**
Image Fusing	Qwen2.5-VL-3B	Individual	75.25	39.50	57.38	73.04	53.95	**65.45**
		Fusing	79.88	50.88	**65.38**	66.96	60.53	64.40
	LLaVA-OV-7B	Individual	74.75	41.75	58.25	75.65	48.68	64.92
		Fusing	71.88	49.75	**60.81**	74.78	71.05	**73.30**
	MiniCPM-V-2.6	Individual	82.25	45.00	63.62	73.04	52.63	64.92
		Fusing	80.75	53.00	**66.88**	73.91	65.79	**70.68**
Detection Model	Qwen2.5-VL-3B	YOLO-World-S	79.38	47.38	63.38	71.30	59.21	**66.49**
		OmDet-T	72.25	44.62	58.44	67.83	64.47	**66.49**
		GDINO-T	79.12	51.75	**65.44**	66.96	60.53	64.40
	LLaVA-OV-7B	YOLO-World-S	60.75	45.62	53.19	73.04	59.21	67.54
		OmDet-T	71.12	48.25	59.69	75.65	67.11	72.25
		GDINO-T	74.25	52.38	**63.31**	75.65	71.05	**73.82**
	MiniCPM-V-2.6	YOLO-World-S	66.75	40.88	53.81	62.61	60.53	61.78
		OmDet-T	72.88	51.38	62.12	66.96	69.74	68.06
		GDINO-T	82.12	57.38	**69.75**	73.04	65.79	**70.16**

Table 6: Ablation of `Chisel`'s ROI extraction and fusing along three axes. **Target Identifier**: `Chisel-noun` (spaCy noun-phrase parser, CPU-only) vs. `Chisel` (Qwen2.5-1.5B SLM). **Image Fusing**: *Individual* (each ROI crop fed to the VLM separately) vs. *Fusing* (`Chisel`'s layout-preserving composite of all ROIs into one image). **Detection Model**: GroundingDINO-Tiny (GDINO-T, `Chisel`'s default) vs. OmDet-Turbo (OmDet-T) vs. YOLO-World-S as the open-vocabulary detector that grounds the targets. HR-Bench: FSP/FCP; V^*Bench: Attr (attribute) / Spatial. **Bold** marks the higher Overall within each model$\times$ablation pair. Accuracy (%) computed per category over the full test sets.

age. These gaps confirm that `Chisel`'s accuracy is bounded by the detector's quality and justify GDINO-T as the default.

E Qualitative Analysis

We qualitatively examine `Chisel` applied to LLaVA-OneVision on HR-Bench and V^*Bench, collecting both successful and failure cases that span single-object, object-relationship, and fine-detail queries.

Successful Region Isolation. Figure 6 presents cases in which the vanilla baseline answers incorrectly (✗) while `Chisel` answers correctly (✓); in each pair the left image is the original input fed to the vanilla VLM and the right is the `Chisel` composite. For single-object questions, `Chisel` reliably localizes even very small targets, shrinking the input to as little as 0.25% of a 4032×4032 image while still resolving the queried attribute. For questions about the relationship between objects, `Chisel` merges the detected boxes into a single composite that preserves their relative spatial arrangement, so the VLM retains the layout cues needed to answer positional queries. Because every region of interest is kept at its native resolution and only the query-relevant areas are stitched together, `Chisel` sharply reduces the visual-token count while preserving the fine-grained detail that aggressive downsampling would otherwise destroy.

Detection-Induced Failures. The benefit of `Chisel` hinges on the open-vocabulary detector grounding the correct region; when grounding fails, the composite carries the wrong evidence and the VLM errs. Figure 7 shows three representative cases. In the top example, the target telephone covers only 0.18% of the 2250×1500 image and is recessed inside an open doorway; no candidate clears our 0.25 confidence threshold, and `Chisel` falls back to the full image. Even with the full image as input, the target occupies too small a region for the VLM to recognize, and inference fails. In the middle example, the pink banner is localized correctly,

but the green trash bin occupies a mere 0.0013% of the 4032×2304 image and is missed entirely; the detector instead grounds the large green scaffolding net as the "green trash bin," and the composite carries the wrong object, flipping the relative-position answer (ground truth: 'left and slightly behind'; prediction: 'right and slightly in front'). In the bottom example, the red suitcase spans only 0.018% of the 2394×1500 image, and GroundingDINO assigns its highest "red suitcase" confidence to the similarly colored red stool (0.434) rather than to the true carrier (0.26); only the stool is retained. Across these cases the failure mode is consistent: prohibitively small targets, occlusion, or same-color and more salient distractors cause the detector to miss the object or ground the wrong one—a failure that is especially damaging for relational queries, where a single misgrounded reference inverts the spatial answer. Systematically mining such failure cases to augment open-vocabulary detectors is a promising direction for future work.

Question: What is the relative position of the person in light brown hat compared to the person wearing pink shirt?
Answer(LLaVA-OV): Right ✗ | **Answer(CHISEL):** Left ✓

[4032x4032]

Reduced by 99.75%
[187x216]

Question: What's the relative position of the orange building compared to the cylindrical green building?
Answer(LLaVA-OV): Behind ✗ | **Answer(CHISEL):** Left ✓

[4032x2304]

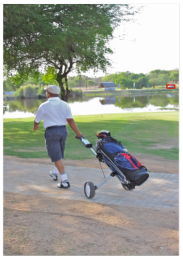
Reduced by 86.82%
[806x1519]

Question: What is the relative position of the person in the red jacket compared to the large tree?
Answer(LLaVA-OV): In front ✗ | **Answer(CHISEL):** Right ✓

[4032x2721]

Reduced by 45.42%
[2460x2434]

Question: What is the color of the SUV car?
Answer(LLaVA-OV): Black ✗ | **Answer(CHISEL):** Silver ✓



[1500x2246]

Reduced by 99.94%
[69x31]

Question: What is the color of the pink-haired woman's handbag?
Answer(LLaVA-OV): Brown ✗ | **Answer(CHISEL):** White ✓

[4032x4032]

Reduced by 99.85%
[187x216]

Question: Is the bucket on the left or right side of the cyclist?
Answer(LLaVA-OV): Left ✗ | **Answer(CHISEL):** Right ✓

[2666x1500]

Reduced by 87.14%
[928x554]

Question: Is the trash can on the left or right side of the black chair?
Answer(LLaVA-OV): Right ✗ | **Answer(CHISEL):** Left ✓

[1500x2250]

Reduced by 99.62%
[179x71]

Question: Is the bucket on the left or right side of the cyclist?
Answer(LLaVA-OV): Left ✗ | **Answer(CHISEL):** Right ✓

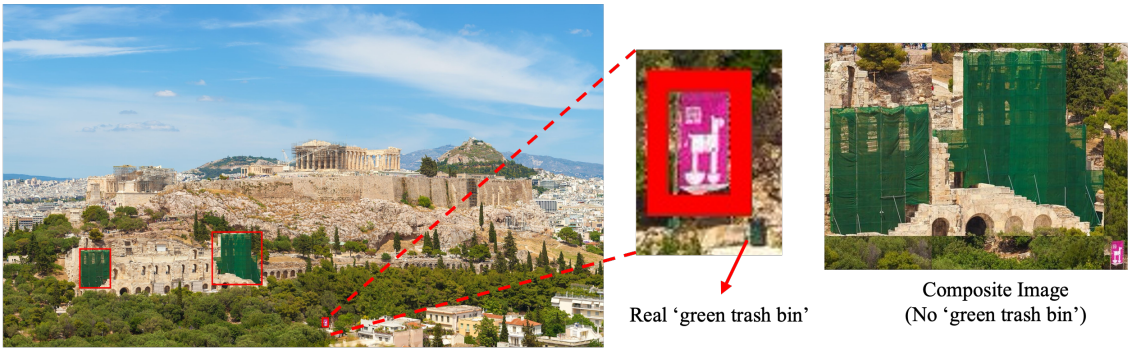
[2230x1500]

Reduced by 99.25%
[201x125]

Figure 6: Successful cases of `Chisel` on LLaVA-OneVision: the vanilla baseline fails (✗) but `Chisel` succeeds (✓). In each pair, the left image is the original input fed to the vanilla VLM and the right is the `Chisel` composite.

Question: What is the color of the telephone? **Target list: [‘telephone’]**
Answer(LLaVA-OV): White ✗ | **Answer(CHISEL):** White ✗ **GT**: Orange ✓

Question: What is the relative position of the pink banner compared to the green trash bin? **Target list: [‘pink banner’, ‘green trash bin’]**
Answer(LLaVA-OV): Left and slightly behind ✓ | **Answer(CHISEL):** Right and slightly in front ✗



Question: Is the red suitcase on the left or right side of the red stool? **Target list: [‘red suitcase’, ‘red stool’]**
Answer(LLaVA-OV): Right ✓ | **Answer(CHISEL):** Left ✗

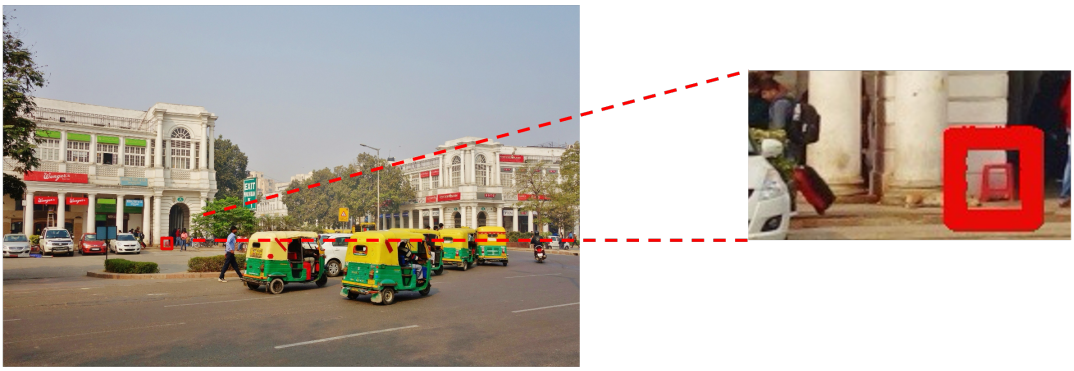


Figure 7: Failure cases of `Chisel` on LLaVA-OneVision (✗); red boxes and zoomed insets mark the detector’s output. **Top:** a recessed telephone with no candidate above threshold; **middle:** a missed green trash bin with the green scaffolding net mis-grounded in its place; **bottom:** a red suitcase outranked by a same-color red stool. See text for details.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, and 1 others. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

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

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Jiaqi Song, Kai Liu, Junyang Wang, and 1 others. 2025. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024. 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.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in neural information processing systems*, 36:49250–49267.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jiawei Guo, Feifei Zhai, Pu Jian, Qianrun Wei, and Yu Zhou. 2025. Crop: Contextual region-oriented visual token pruning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9767–9783.

Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. 2018. VizWiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yitong Jiang, Jinwei Gu, Tianfan Xue, Ka Chun Cheung, Pavlo Molchanov, Hongxu Yin, and Sifei Liu. 2025. Token-efficient vlm: High-resolution image understanding via dynamic region proposal. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24147–24158.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and 1 others. 2024a. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

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

Ziyi Lin, Chris Liu, Renrui Zhang, Peng Gao, Longtian Qiu, Han Xiao, Han Qiu, Chen Lin, Wenqi Shao, Keqin Chen, Jiaming Han, Aojun Huang, Yichi Yue, Dongzhi Zhang, Hongsheng Li, and Yu Qiao. 2023. SPHINX: The joint mixing of weights, tasks, and visual embeddings for multi-modal large language models. *arXiv preprint arXiv:2311.07575*.

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

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024b. LLaVA-NeXT: Improved reasoning, ocr, and world knowledge. https://llava-vl.github.io/blog/2024-01-30-llava-next/.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, and 1 others. 2024c. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*.

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 (CVPR)*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, and 1 others. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. 2025. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 7907–7915.

Penghao Wu and Saining Xie. 2024. V?: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13084–13094.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, and 1 others. 2024. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*.

Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Yuhao Dan, Chenlin Zhao, Guohai Xu, Chenliang Li, Junfeng Tian, Qi Qian, Ji Zhang, Qin Zhang, Liang He, Xianjie Yang, and Fei Huang. 2023. UReader: Universal OCR-free visually-situated language understanding with multimodal large language model. In *Findings of the Association for Computational Linguistics: EMNLP*.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and 1 others. 2024. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*.