PIXELPRUNE: PIXEL-LEVEL ADAPTIVE VISUAL TOKEN REDUCTION VIA PREDICTIVE CODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Document and GUI understanding are among the highest-value applications of Vision-Language Models (VLMs), yet they impose exceptionally heavy computational burden: fine-grained text and small UI elements demand high-resolution inputs that produce tens of thousands of visual tokens. We observe that this cost is largely wasteful: across document and GUI benchmarks, 29–78% of patches within an image are pixel-level duplicates of others. We propose **PixelPrune**, which exploits this pixel-level redundancy through predictive-coding-based compression, pruning redundant patches *before* the Vision Transformer (ViT) encoder. Because it operates in pixel space prior to any neural computation, PixelPrune accelerates the entire inference pipeline—ViT encoder, patch merger, and LLM decoder—and applies equally to training. PixelPrune is parameter-free and supports pixel-lossless (τ=0) as well as controlled lossy (τ>0) compression. Without any training, it stays on average within 0.9 points of full-token accuracy on documents and similarly close on GUI question answering. On GUI grounding, where predicting precise coordinates makes the model more sensitive to the mismatch between full-token training and pruned inference, distillation from the model's own full-token outputs restores full-token accuracy. Trained from scratch on GUI grounding with PixelPrune, a model matches full-token training and can be served with either pruned or full-token input. PixelPrune speeds up inference by 3.0–4.2× and training by 1.9×.

1 INTRODUCTION

Vision-Language Models (VLMs) (Liu et al., 2023; Li et al., 2025a; Chen et al., 2024d;c; Bai et al., 2025a) achieve impressive performance across multimodal tasks by integrating visual encoders with large language models. However, their computational cost scales directly with the number of visual tokens processed during both training and inference. This cost is especially pronounced in document and GUI scenarios, which demand high-resolution inputs for fine-grained text rendering and small UI elements. This burden is further amplified by the growing adoption of NaViT-style vision encoders (Dehghani et al., 2023)—exemplified by the Qwen-VL series (Wang et al., 2024; Bai et al., 2025b;a; Qwen Team, 2026)—which process images at native resolution and can produce tens of thousands of patches per image. For example, a typical document page in DocVQA averages 1775×2082 pixels (Appendix A), producing over 14,000 ViT patches (p=16) and roughly 3,600 LLM tokens after 2×2 merging.

Existing VLM token reduction methods primarily operate at the semantic or feature level—relying on attention-based pruning (Chen et al., 2024a; Yang et al., 2025), adaptive pruning-and-merging (Shang et al., 2024), or diversity-based selection (Alvar et al., 2025). Most are designed for general-domain images (Shao et al., 2025). In structured document and GUI domains, however, a more fundamental source of redundancy exists: the raw pixel content itself. Documents are dominated by white margins and uniform page backgrounds; GUI screenshots contain large solid-color panels, toolbars, and status bars—all of which produce many pixel-identical patches. As shown in Figure 1, large portions of patches correspond to such uniform regions. Our analysis across document and GUI benchmarks reveals that 29–78% of patches within an image are pixel-level duplicates of others, depending on the domain (Appendix A). This indicates that pixel-level content alone is a powerful signal for token reduction, eliminating redundancy at its source without any neural forward pass.

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.

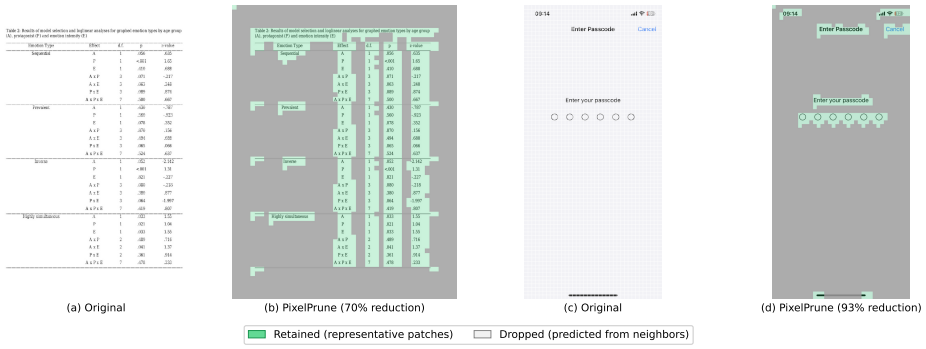

Figure 1: PixelPrune patch selection on a document image (*left*) and GUI screenshot (*right*). Kept patches keep their original color; dropped ones are grayed out. Token reduction: 70% and 93%.

Moreover, most existing VLM acceleration methods reduce only the visual tokens processed by the LLM—via projector-based compression (Li et al., 2023; 2024) or the pruning and merging methods above—leaving the ViT encoder's cost untouched, and are validated mainly on fixed-resolution architectures (e.g., LLaVA (Liu et al., 2023)) where token counts are moderate. This matters for NaViT-based VLMs, where the ViT input grows with image resolution and its self-attention cost grows quadratically. As Figure 2 shows, the vision encoder accounts for up to 86% of prefill latency at 4096^2 for Qwen3-VL-2B, a fraction that grows with resolution. A more direct approach is to prune visual tokens *before* they enter the ViT, accelerating the entire pipeline from the start.

We propose **PixelPrune**, which addresses this gap by applying predictive coding (Weinberger et al., 2000; Boutell, 1997)—the same principle underlying lossless formats such as PNG and JPEG-LS—at the patch level *before* vision encoding. Each patch is predicted from its causal spatial neighbors; if the prediction matches (within a threshold τ), the patch is omitted as redundant. The method requires no learnable parameters, supports both pixel-lossless (τ=0) and controlled lossy (τ>0) compression, and is fully compatible with FlashAttention (Dao et al., 2022). Applied without training, PixelPrune costs little accuracy on documents and GUI question answering but substantially more on GUI grounding. Predicting precise coordinates makes grounding sensitive to the mismatch between full-token training and pruned inference: the retained patches are intact, but their arrangement—a grid with holes—never occurs in pre-training. Training removes this mismatch in two ways: knowledge distillation (KD) from the unpruned model adapts a pre-trained model to the pruned layout, and training with PixelPrune from scratch avoids the mismatch altogether while also accelerating training.

Our contributions are threefold: (1) we quantify the pervasive pixel-level patch redundancy in document and GUI images (Appendix A) and propose PixelPrune, a parameter-free method that exploits this redundancy by content-adaptive selection on the ViT's input patch grid (pre-ViT), with theoretical recoverability guarantees (§3.3); (2) because PixelPrune operates before the ViT, it accelerates the *entire* pipeline—ViT encoder, patch merger, and LLM decoder—in both inference and training, with up to 4.2× faster prefill and 1.9× faster training; and (3) we evaluate PixelPrune with and without training (§4.2–4.3.2): training-free, it stays on average within 1 point of the unpruned model on documents and GUI question answering; on GUI grounding, where training-free pruning loses more, either KD, which brings ScreenSpot-V2 and ScreenSpot-Pro to the unpruned model's accuracy, or training from scratch with PixelPrune, which matches full-token training and supports both pruned and full-token inference, closes the gap.

2 Related Work

Token Reduction in Vision Transformers. ViT (Dosovitskiy et al., 2021) attention scales quadratically with patch count, and MAE (He et al., 2022) shows that 75% of patches can be masked while still permitting accurate reconstruction. DynamicViT (Rao et al., 2021), EViT (Liang et al., 2022) and ToMe (Bolya et al., 2023) prune or merge tokens on intermediate features, which requires partial forward computation. RLT (Choudhury et al., 2024) drops temporally repeated patches before a video

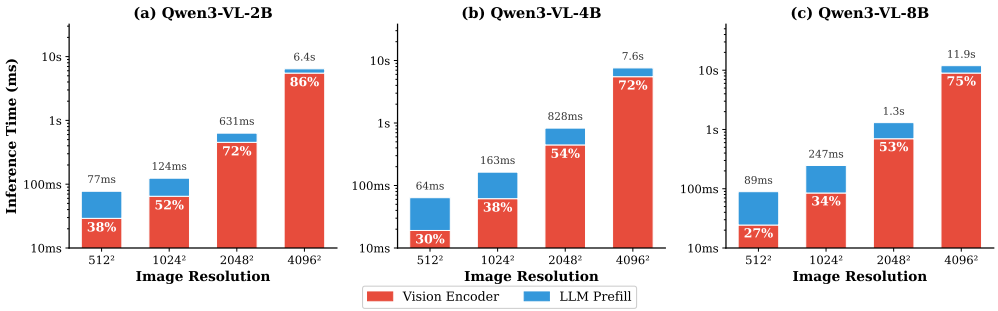

Figure 2: Prefill latency breakdown across Qwen3-VL scales (2B, 4B, 8B) at five resolutions (256^2–4096^2, log scale). Each bar splits into **Vision Encoder** (red, including ViT and Patch Merger) and **LLM Prefill** (blue). At 4096^2, the vision encoder accounts for 86%, 72%, and 75% of total prefill time for the 2B, 4B, and 8B models respectively.

transformer; PixelPrune pursues the spatial counterpart within a single image, with 2D prediction and exact recoverability.

Efficient Vision-Language Models. Token reduction for VLMs (Liu et al., 2023; Bai et al., 2023; Wang et al., 2024) acts at different points of the pipeline (§3.1). Most methods are *post-ViT*, reducing tokens after the vision encoder: projector-based compression (Li et al., 2023; 2024), pruning or merging of encoder outputs (Shang et al., 2024; Yang et al., 2025; Alvar et al., 2025), pruning inside the LLM by decoder attention (Chen et al., 2024a; He et al., 2024; Xing et al., 2024), which typically materializes attention weights at odds with FlashAttention (Dao et al., 2022), and multi-stage combinations (Liu et al., 2024a; Zheng et al., 2025). All of them run the encoder at full length, which dominates prefill for small and medium VLMs (Figure 2). Others reduce tokens inside the ViT—IPCV (Chen et al., 2025) on intermediate features, LLaVA-UHD v4 (Fang et al., 2026) in shallow layers, and CIVIC (Yang et al., 2026) with a compact sequence trained by distillation—but still need a partial encoder pass. Image-domain methods act before the encoder by changing the image: PACE (Liu et al., 2026) adaptively downsamples redundant inputs before extracting tokens after the encoder, and DocKylin (Zhang et al., 2025) crops blank rows and columns of documents. PixelPrune is *pre-ViT*: it removes patches from the ViT's input grid and leaves the image itself unchanged.

Document and GUI Understanding. Models in these domains handle high resolution by image tiling (Hu et al., 2024a;b; Liu et al., 2024d; Wei et al., 2024) or dual encoders (Hong et al., 2024), and GUI agents such as UI-TARS (Qin et al., 2025) and Jedi (Xie et al., 2025) target accuracy rather than efficiency. IndexPrune (Son et al., 2025) is also pre-ViT but prunes by text saliency: a classifier trained on text-detector labels keeps text-bearing patches, whereas PixelPrune drops only patches predictable from their neighbors, whatever their content. For GUI grounding, ShowUI (Lin et al., 2024) skips redundant tokens inside ViT self-attention, still routing every patch through the ViT's projections and FFN; FocusUI (Ouyang et al., 2026) trains an instruction-conditioned patch scorer and replaces each run of dropped tokens with a marker to keep positions continuous; and the training-free AQuaUI (Li et al., 2026) merges tokens over an adaptive quadtree of the screenshot while preserving their positions. PixelPrune is instruction-agnostic, parameter-free and exactly recoverable (§3.3).

3 Method

3.1 Preliminaries

We summarize the VLM visual pipeline using Qwen3-VL (Bai et al., 2025a) as an example.

Architecture overview. An input image first passes a resolution-normalization step (`smart_resize` in Qwen3-VL), which rescales it to meet a pixel budget while preserving aspect ratio and aligns both sides to multiples of 32 pixels, the merged patch size corresponding to one LLM token; the normalized image is then divided into $N = (H/p) \times (W/p)$ non-overlapping patches of size $p \times p$ (p=16). Each patch P_i is projected to a D_v-dimensional embedding $\mathbf{e}_i$ and paired with

its 2D grid coordinate $\mathbf{p}_i = (r_i, c_i)$, which is injected via interpolated absolute positional embeddings and 2D RoPE. The sequence $\{(\mathbf{e}_i, \mathbf{p}_i)\}_{i=1}^{N}$ is processed by a L_v-layer ViT with $O(N^2 D_v)$ attention cost per layer, then spatially compressed by Patch Mergers ($M \times M$ grouping, M=2; including DeepStack (Meng et al., 2024) mergers at selected intermediate ViT layers) to $N_m = N/M^2$ tokens, which are concatenated with T text tokens to form the LLM input. In the LLM, spatial layout is encoded via MRoPE (Bai et al., 2025a), which assigns each visual token a 3D coordinate (temporal, height, width). Crucially, spatial information throughout the pipeline is determined by **positional encodings rather than sequence order**, so feeding a *subset* of patches while preserving their original coordinates is an architecturally supported operation. Such position-preserving pruning is not tied to Qwen3-VL's encodings: it also works with absolute ViT embeddings and 1D RoPE (InternVL3.5) and with linear-attention LLM layers that see only token order (Qwen3.5), as Appendix D shows.

Where token reduction can act. We distinguish two domains in which tokens can be reduced. In the *image domain*, resizing, tiling or cropping changes the image before patchification. In the *token domain*, tokens are removed from the visual sequence inside the ViT or the LLM. PixelPrune acts at the very start of the token domain: on top of the default `smart_resize`, it keeps a subset of the ViT's input patch grid with original pixels and coordinates, before any neural computation. We call this *pre-ViT* pruning, and pruning of the encoder output or inside the LLM *post-ViT*. We also tried applying PixelPrune's patch selection at deeper ViT layers instead, which brings no accuracy gain (Appendix J).

3.2 PixelPrune: Patch Compression via Predictive Coding

Given N patch–position pairs $\{(P_i, \mathbf{p}_i)\}_{i=1}^{N}$, the goal is to identify a compact subset $\mathcal{S} \subseteq \{1, \ldots, N\}$ of patches to retain such that downstream task performance is preserved. To identify redundant patches, we draw on **predictive coding** from lossless image compression: classical formats such as PNG (Boutell, 1997) and JPEG-LS (Weinberger et al., 2000) predict each pixel from its spatial neighbors and encode the prediction residual, which is typically small and thus highly compressible. We adapt this idea from individual pixels to $Mp \times Mp$ pixel blocks (32×32 for Qwen3-VL with ViT patch size p=16 and merge factor M=2), each corresponding to one LLM token after patch merging—the smallest independently removable unit. Blocks are scanned in a fixed order; each is predicted from its causal neighbors (those already visited) and omitted if the prediction matches. This formulation naturally provides an **exact recoverability guarantee** (§3.3): a decoder applying the same prediction rule can reconstruct all omitted patches.

Formally, for each patch $P_{r,c}$, PixelPrune omits it when the prediction is close enough:

$$\text{Omit}(P_{r,c}) = \mathbb{I}\big[\, \text{dist}(P_{r,c},\, \hat{P}_{r,c}) \leq \tau \,\big] \tag{1}$$

where $\tau \geq 0$ controls matching strictness; the top-left patch has no causal neighbor and is always kept.

3.2.1 Prediction Strategy

The predictor $\hat{P}_{r,c}$ determines which spatial redundancy patterns can be captured. We consider three strategies that form a **1D-to-2D progression** within the same predictive coding framework, differing only in scan order and prediction rule:

(i) Raster and **(ii) Serpentine** flatten the 2D grid into a 1D sequence—in row-major order, or alternating scan direction between rows—and predict each patch from its immediate predecessor: $\hat{P}_i = P_{i-1}$. Raster captures horizontal continuity but misses cross-row redundancy, as the last patch of row r and the first of row r+1 are generally far apart; Serpentine's alternating direction makes the transition between rows spatially local, partially capturing vertical continuity.

(iii) Pred-2D scans in raster order but predicts each patch from three causal neighbors using a simple selection rule (analogous to the median-edge predictor in LOCO-I (Weinberger et al., 2000)). Let A=$P_{r,c-1}$, B=$P_{r-1,c}$, and C=$P_{r-1,c-1}$ denote the left, upper, and upper-left neighbors (Figure 3):

$$\hat{P}_{r,c} = \begin{cases} A & \text{if } C = B \neq A \quad \text{(upper and upper-left agree} \Rightarrow \text{predict from left)} \\ B & \text{if } C = A \neq B \quad \text{(left and upper-left agree} \Rightarrow \text{predict from upper)} \\ A & \text{otherwise (default: predict from left)} \end{cases} \tag{2}$$

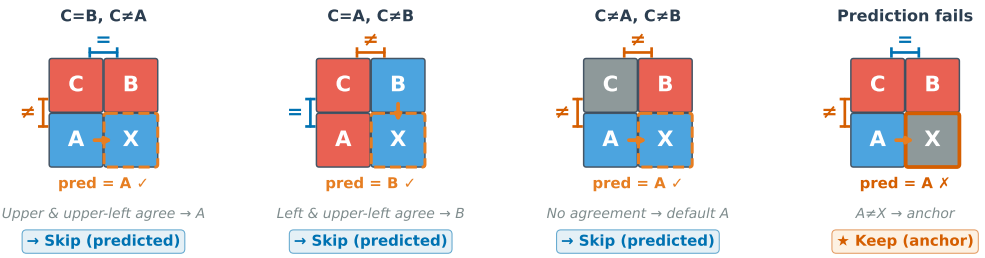

Figure 3: Pred-2D prediction. Three causal neighbors of the target patch X—A (left), B (upper), C (upper-left)—determine the prediction $\hat{X}$; X is omitted if it matches $\hat{X}$.

By checking agreement among three neighbors, the rule exploits horizontal and vertical redundancy at once.

All three strategies run in $O(N)$ time—negligible next to the ViT's $O(N^2)$ attention—and guarantee exact recoverability (§3.3); we adopt **Pred-2D** by default, as it consistently achieves the lowest retain ratio at comparable accuracy (Table 6).

3.2.2 Matching Criterion

Our default threshold is $\tau{=}0$, i.e., strict pixel equality: a patch is omitted only if it is *identical* to its prediction, mirroring lossless compression. This already removes 23–74% of tokens on document and GUI images; near-exact matching ($\tau{>}0$) with MAE or Max distance is ablated in §4.5.2.

At $\tau{=}0$ exact equality is transitive, so patches are compared with their neighbors' *original* values in parallel on GPU; at $\tau{>}0$ the encoder runs sequentially to stay in sync with the decoder.

3.2.3 Position-Preserving Pipeline Integration

Only the retained patch–position pairs $\{(P_k, \mathbf{p}_k)\}_{k\in\mathcal{S}}$ enter the pipeline, each keeping its **original 2D grid coordinate** rather than being re-indexed, so positional encodings (e.g., 2D RoPE and MRoPE) still reflect the spatial layout while fewer patches flow through the ViT, Patch Merger and LLM.

3.3 Theoretical Guarantees

Exactness matters most for pre-ViT pruning: a post-ViT method removes tokens after self-attention has mixed their content into the tokens it keeps, whereas a patch removed before the ViT is never seen by any layer, and its information cannot be recovered from other tokens. PixelPrune therefore comes with formal recoverability guarantees at the pixel level.

Proposition 1 (Exact Reconstruction, $\tau{=}0$)**.** *The original patch sequence $\{P_1, \ldots, P_N\}$ is exactly recoverable from $\{(P_k, \mathbf{p}_k)\}_{k\in\mathcal{S}}$ via a deterministic decoder that applies the same prediction rule in the same scan order. (Proof in Appendix B.)*

Keeping one copy of each distinct block (global deduplication) would drop slightly more, but each dropped block then needs an extra stored index of the block it copies, whereas VLMs encode only positions; used directly, it is not recoverable and loses information.

Near-exact matching ($\tau{>}0$). Relaxing equality to $\mathrm{dist}(\cdot,\cdot) \le \tau$ admits patches differing only by compression artifacts, sensor noise, or smooth gradients. Since near-similarity is not transitive, naive chaining could let errors drift unboundedly; PixelPrune prevents this: the encoder uses the **predicted value** (not the original) for each omitted neighbor, keeping encoder and decoder in sync.

Proposition 2 (Bounded Reconstruction Error, $\tau{>}0$)**.** *When the encoder uses predicted values for omitted neighbors, every omitted patch $i \notin \mathcal{S}$ satisfies $\mathrm{dist}(P_i, \hat{P}_i) \le \tau$, with no error accumulation across patches. (Proof in Appendix B.)*

Table 1: Training-free document results (Qwen3-VL-2B, %). **Stage**: where tokens are removed (§3.1). Parentheses: PixelPrune's retain ratio.

Method	Stage	OCRBench (66.1%)	DocVQA (70.1%)	InfoVQA (76.7%)	ChartQA (73.1%)	AI2D (69.2%)	MML-Doc (50.3%)	olmOCR (56.9%)	Avg
Full	–	86.9	92.7	72.3	78.8	76.6	28.1	48.9	69.2
FastV	post-ViT	82.9	83.4	70.4	70.7	75.7	24.8	36.9	63.5
VisionZip	post-ViT	85.1	89.4	71.4	**77.1**	76.1	25.9	43.5	66.9
DivPrune	post-ViT	86.3	90.7	71.0	77.0	**76.9**	26.2	43.4	67.3
Random	pre-ViT	76.0	66.8	62.9	57.3	72.2	17.0	34.7	55.3
ConnComp	pre-ViT	77.2	80.5	66.1	62.8	73.9	18.4	37.2	59.4
PixelPrune	pre-ViT	**87.3**	**92.0**	**71.5**	77.0	75.8	**26.5**	**48.0**	**68.3**

4 EXPERIMENTS

We evaluate documents training-free (§4.2) and GUI tasks in three settings (§4.3): training-free, with distillation (§4.3.1) and trained from scratch (§4.3.2), followed by efficiency (§4.4) and ablations (§4.5).

4.1 EXPERIMENTAL SETUP

Models and benchmarks. Main results use Qwen3-VL-2B (Bai et al., 2025a); Appendices C–E extend them to larger scales (4B, 8B), other architectures (Qwen3.5-2B, InternVL3.5-2B) and general-domain images. We evaluate seven *document* benchmarks—DocVQA (Mathew et al., 2021), AI2D (Kembhavi et al., 2016), ChartQA (Masry et al., 2022), InfoVQA (Mathew et al., 2022), OCRBench (Liu et al., 2024c), MMLongBench-Doc (Ma et al., 2024) and olmOCRBench (Poznanski et al., 2025)—and *GUI grounding* on ScreenSpot V2 (Wu et al., 2025) (Web/Mobile/Desktop) and ScreenSpot Pro (Li et al., 2025b) (six professional domains), where a prediction is correct if the predicted click point falls inside the target element's bounding box. We also evaluate GUI question answering on ScreenQA-Short (Hsiao et al., 2025), using the 7,558 test questions whose answer appears on the screen, and on MMBench-GUI L1 (Wang et al., 2025c).

Baselines. Besides *Full*, which keeps all visual tokens, we compare five reduction methods under a matched *per-image token budget*, keeping for each image exactly as many tokens as PixelPrune retains: three published post-ViT methods—*FastV* (Chen et al., 2024a) prunes inside the LLM by decoder attention, *VisionZip* (Yang et al., 2025) and *DivPrune* (Alvar et al., 2025) select from the encoder's output—and two pre-ViT controls that isolate the selection criterion: *Random*, and *ConnComp*, which, following ShowUI (Lin et al., 2024), keeps a random subset of each connected group of pixel-identical patches. Appendix F gives metrics and baseline details.

Implementation. All experiments use exact matching (τ=0) on 32×32 blocks (§3.2) and Qwen3-VL's default pixel budget (256^2 to 4096^2). Inference uses 8 NVIDIA H20 GPUs, batch size 1 and greedy decoding (at most 16,384 new tokens for olmOCRBench, 512 otherwise), scored with VLMEvalKit (Duan et al., 2024). *Retain ratios* are dataset-level, $\sum_i |\mathcal{S}_i| / \sum_i N_i$ over retained and total visual tokens per image. GUI training uses the grounding subset of Jedi (Xie et al., 2025) (1.75M samples): distillation (§4.3.1) mixes it 1:1 with the Windows grounding data of ScaleCUA (Liu et al., 2025), and training from scratch (§4.3.2) uses it alone (Appendix G).

4.2 DOCUMENT UNDERSTANDING

Table 1 presents training-free results on 7 document understanding benchmarks with Qwen3-VL-2B. PixelPrune's retention is content-adaptive, with no preset budget: it keeps 50.3% of the tokens on MMLongBench-Doc and 56.9% on olmOCRBench, but 73.1% on ChartQA and 76.7% on InfoVQA. PixelPrune has the highest average of all reduction methods, and the ranking holds at 4B and 8B (Appendix C) and on Qwen3.5-2B (Qwen Team, 2026) and InternVL3.5-2B (Wang et al., 2025a) (Appendix D); its gap to Full is 0.9 points at 2B and shrinks to 0.4 and 0.2 points at 4B and 8B. Among pre-ViT methods, the selection criterion decides accuracy: Random and ConnComp prune at the same stage and budget yet fall 9–13 points below PixelPrune. PixelPrune also outperforms every post-ViT baseline, by 1.0–4.8 points on average, even though it selects from raw pixels while they select from full ViT features.

Table 2: Training-free GUI question answering (Qwen3-VL-2B, %): token F1 on ScreenQA-Short and accuracy on MMBench-GUI L1. Parentheses: PixelPrune's retain ratio.

	Full	FastV	VisionZip	DivPrune	**PixelPrune**
ScreenQA-Short (53.3%)	91.8	73.5	81.5	89.9	**91.3**
MMBench-GUI L1 (50.7%)	86.6	76.8	80.9	84.0	**85.2**

Table 3: GUI grounding (Qwen3-VL-2B, %). Pre-ViT cells give *training-free* / *+KD* accuracy (§4.3.1); post-ViT baselines are training-free only. Parentheses: PixelPrune's retain ratio.

Method	**ScreenSpot V2**				**ScreenSpot Pro**						
	Web (39.7%)	**Mobile** (49.1%)	**Desktop** (61.9%)	**Avg**	**Dev.** (64.3%)	**Creative** (59.0%)	**CAD** (65.6%)	**Sci.** (26.3%)	**Office** (56.4%)	**OS** (61.9%)	**Avg**
Full	88.1	92.0	91.6	90.6	45.8	41.6	40.2	53.1	70.4	50.0	50.2
Post-ViT, training-free											
FastV	**77.3**	86.4	**85.9**	**83.2**	43.8	**37.8**	**39.5**	35.8	63.9	**48.5**	44.9
VisionZip	73.0	81.8	83.8	79.6	**44.8**	37.0	36.0	**41.3**	**65.7**	45.4	**45.0**
DivPrune	75.3	**87.0**	83.8	82.0	30.1	31.4	34.9	21.3	54.3	31.1	33.8
Pre-ViT, training-free / +KD											
Random	48.5/62.7	62.7/79.0	68.0/75.4	59.7/72.4	29.4/39.1	24.6/31.4	26.4/28.4	12.6/19.7	37.8/44.8	29.6/34.7	26.8/33.0
ConnComp	69.8/79.4	82.0/89.2	75.1/80.8	75.7/83.2	35.8/44.5	27.9/34.0	28.7/31.0	22.0/40.6	47.8/60.0	31.1/43.9	32.2/42.3
PixelPrune	72.1/**86.3**	86.2/**91.6**	**85.9/92.5**	81.4/**90.1**	37.8/**50.5**	28.4/**41.1**	35.2/**41.8**	21.3/**53.9**	55.7/**69.6**	40.3/**49.5**	36.5/**51.1**

4.3 GUI UNDERSTANDING

GUI tasks fall into two kinds. In question answering the answer is text, and training-free PixelPrune behaves as on documents, staying within 1.4 points of Full and ahead of the other reduction methods (Table 2). Grounding, the core skill of GUI agents, instead asks for a pixel coordinate, and here training-free PixelPrune loses 9–14 points. We therefore study how training closes this gap, by distillation (§4.3.1) or by training with PixelPrune from scratch (§4.3.2).

4.3.1 KNOWLEDGE DISTILLATION

Without training, every reduction method loses clearly on grounding: the post-ViT baselines drop 5–16 points, and PixelPrune 9.2 on ScreenSpot V2 and 13.7 on Pro (training-free values in Table 3). Predicting precise coordinates makes grounding sensitive to the mismatch between full-token training and pruned inference. For PixelPrune the retained pixels are bit-exact at τ=0, but their *arrangement*—a grid with holes—is unseen in pre-training; post-ViT methods face a smaller mismatch, since they prune features the ViT has already computed on the full grid. Because PixelPrune loses no pixel information (Proposition 1), its gap comes from this mismatch rather than from missing content, and training on pruned input should close it.

We therefore distill the model on pruned input from the original model on full input (+KD) (Hinton et al., 2015), with $\mathcal{L} = \frac{1}{2}\mathcal{L}_{\text{CE}} + \frac{1}{2}\mathcal{L}_{\text{KL}}(p_{\text{prune}} \| p_{\text{full}})$. PixelPrune, Random and ConnComp share one recipe (300 steps with the ViT learning rate scaled by 0.25; Appendix G), each at PixelPrune's per-image budget; following their standard usage, the post-ViT baselines stay training-free. After distillation, PixelPrune reaches the accuracy of the unpruned model (+KD values in Table 3): it is 0.5 points below Full on V2 and 0.9 points above it on Pro, and no Pro sub-task falls more than 0.8 points below Full. Random and ConnComp, distilled the same way, still trail PixelPrune by 7–18 points, so the recovery comes from PixelPrune's selection rather than from training alone.

4.3.2 TRAINING FROM SCRATCH

Training from scratch serves two purposes: it tests PixelPrune as a way to accelerate training itself, and it compares pruned and full-token training on equal footing, with the same initialization, data and loss. To this end, we replace the language model of Qwen3-VL-2B-Instruct with the text-only Qwen3-1.7B-Instruct and train the resulting model on Jedi with the standard cross-entropy loss (no KD). We compare three configurations: **Full→Full** (standard training and inference), **PixelPrune→PixelPrune** (compressed tokens for both), and **PixelPrune→Full** (compressed training, full inference). Training efficiency is reported in Section 4.4.2.

Table 4: Training from scratch on GUI grounding (%). Train/Infer: whether full or PixelPrune-pruned tokens are used in training and at inference.

		ScreenSpot V2				ScreenSpot Pro						
Train	**Infer**	**Web**	**Mobile**	**Desktop**	**Avg**	**Dev.**	**Creative**	**CAD**	**Sci.**	**Office**	**OS**	**Avg**
Full	Full	79.9	81.6	83.8	81.8	**31.8**	26.7	15.3	**35.0**	49.6	**30.6**	31.5
PixelPrune	PixelPrune	79.9	85.6	**88.9**	84.8	30.8	27.3	15.7	34.6	**52.6**	27.6	31.4
PixelPrune	Full	**80.5**	**86.0**	88.6	**85.1**	30.4	**27.9**	**16.1**	**35.0**	50.4	29.6	**31.6**

Table 5: Inference efficiency on MMLongBench-Doc and ScreenSpot Pro Scientific (Qwen3-VL-2B). Selector: patch-selection time per sample; speedups are relative to Full.

Dataset	**Method**	**Selector** (ms)	**Retain** (%)	**FLOPs** (TFLOPs)	**ViT Lat.** (ms)	**TTFT** (ms)	**KV Cache** (GB)	**FLOPs** Speedup	**TTFT** Speedup
MML-Doc	Full	–	100.0	2710	4898	18477	13.2	1.0×	1.0×
	PixelPrune	19.1	50.3	770	2143	6114	7.2	**3.5**×	**3.0**×
SS-Pro Sci.	Full	–	100.0	71	625	832	0.82	1.0×	1.0×
	PixelPrune	1.2	26.2	11	115	198	0.28	**6.5**×	**4.2**×

Training with PixelPrune costs no accuracy: PixelPrune→PixelPrune is on par with Full→Full on ScreenSpot Pro and higher on V2 (Table 4; convergence curves in Appendix H). A model trained with PixelPrune also works with full-token input: PixelPrune→Full is on par with or better than Full→Full, so a PixelPrune-trained checkpoint can serve with PixelPrune for low latency or fall back to full tokens where a serving stack does not yet integrate the selector.

4.4 Efficiency Analysis

4.4.1 Inference Efficiency

Table 5 reports prefill-stage efficiency on two benchmarks at opposite ends of the spectrum: ScreenSpot Pro Scientific (single-page, 26.2% retention) and MMLongBench-Doc (multi-page, 50.3% retention), measured on 8 NVIDIA H20 GPUs at batch size 1 after discarding the first sample per GPU as warmup. Across both, PixelPrune speeds up time-to-first-token by 3.0–4.2×, much of it from the ViT itself (2.3–5.4×), which post-ViT pruning cannot accelerate, while FLOPs fall 3.5–6.5×, KV-cache memory falls 45–66%, and the selector's overhead stays negligible. Because ViT attention is quadratic in sequence length, its saving grows faster than the pruning ratio (Appendix I).

4.4.2 Training Efficiency

Because PixelPrune acts on raw pixels before any neural computation, it applies unchanged during training. In the from-scratch experiment of §4.3.2, at 40.8% retention, forward and backward passes each run 2.0× faster, wall-clock training time falls 1.9× and peak memory falls by a third (Table A6). Post-ViT methods are harder to use in training: their criterion is computed from encoder outputs or decoder attention, which adds overhead such as materializing attention weights and leaves the ViT at full length in both the forward and backward passes. PixelPrune depends only on pixels, so the patches to keep can already be determined during data preprocessing.

4.5 Ablation Studies

We ablate the two design choices of PixelPrune (§3.2): the prediction strategy, which sets what redundancy can be captured, and the matching criterion, which sets when a prediction matches.

4.5.1 Comparison of Prediction Strategies

We compare the three strategies of §3.2.1, Raster, Serpentine and Pred-2D, which form a 1D-to-2D progression, on three document and three GUI benchmarks (Table 6). Moving from 1D to 2D prediction lowers retention on five of the six benchmarks, most on ChartQA (76.7% → 73.1%) and GUI-Web (42.5% → 39.7%), where panels, bars and backgrounds extend over several rows.

Table 6: Prediction strategies (Qwen3-VL-2B). Document: training-free; GUI: each strategy distilled separately with the recipe of §4.3.1. Ret./Acc.: retain ratio / accuracy (%).

	Document						GUI					
	DocVQA		InfoVQA		ChartQA		Web		Mobile		Desktop	
Method	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**
Raster	70.1	92.0	77.8	71.4	76.7	**77.6**	42.5	85.6	50.5	**92.2**	64.4	91.9
Serpentine	**70.0**	**92.1**	77.4	71.2	74.9	77.1	41.8	**87.4**	50.1	**92.2**	64.1	92.2
Pred-2D	70.1	92.0	**76.7**	**71.5**	**73.1**	77.0	**39.7**	86.3	**49.1**	91.6	**61.9**	**92.5**

Table 7: Distance metrics and thresholds (τ normalized to $[0, 1]$). Document: training-free; GUI: the PixelPrune (+KD) model of §4.3.1, evaluated at each τ without retraining.

		Document						GUI					
		DocVQA		InfoVQA		ChartQA		Web		Mobile		Desktop	
Distance	τ	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**
Full (100%)		100	92.7	100	72.3	100	78.8	100	88.1	100	92.0	100	91.6
Exact (τ=0)		70.1	92.0	76.7	71.5	73.1	77.0	39.7	**86.3**	49.1	91.6	61.9	92.5
MAE	0.005	61.8	92.0	66.7	70.4	53.1	74.4	36.0	84.7	35.9	**91.8**	42.7	91.0
	0.020	37.0	90.9	56.9	68.4	35.0	69.4	31.3	82.2	27.1	**91.8**	30.7	87.1
	0.050	28.6	86.8	47.1	65.8	23.3	43.5	25.7	76.7	20.6	88.6	23.3	82.0
	0.100	21.2	69.5	35.1	55.4	13.2	25.8	17.8	59.5	14.4	77.4	14.9	58.7
Max	0.005	69.8	**92.1**	75.5	71.2	72.6	**77.2**	39.2	84.9	44.7	91.6	53.9	**92.8**
	0.020	66.1	91.8	71.7	**71.6**	69.3	76.4	38.0	84.4	38.2	**91.8**	45.7	91.0
	0.050	53.1	91.8	68.0	71.1	62.2	76.6	37.1	84.9	34.4	91.6	40.1	90.4
	0.100	45.1	91.7	62.4	70.5	56.8	74.8	35.8	84.2	30.1	91.6	35.9	89.8

Accuracy stays comparable across all three, so the strategy mainly affects compression, and we adopt **Pred-2D**, the most compressive overall.

4.5.2 Effect of Matching Threshold

We compare MAE and Max (maximum per-pixel difference), both normalized to $[0, 1]$, over several thresholds τ (Table 7). Max is the safer choice for near-exact matching, matching or beating MAE in all but one setting: MAE averages per-pixel differences, so a patch that differs only in a few pixels, such as a thin line or a small glyph, can still count as a match, which collapses ChartQA at τ=0.05 (43.5 vs. 76.6 for Max), whereas Max keeps DocVQA within about 1 point of Full even at τ=0.10, where retention falls to 45.1%. Near-exact matching also extends PixelPrune beyond documents and GUI: on general-domain images, where exact duplicates are rare and τ=0 keeps most tokens, accuracy on six benchmarks stays within 2.1 points of Full across all tested τ (Appendix E).

5 Conclusion

We presented **PixelPrune**, a parameter-free token reduction method that exploits 2D pixel redundancy via predictive coding with formal recoverability guarantees. Selecting patches on raw pixels before any neural computation, it accelerates the entire pipeline—ViT encoder, patch merger, and LLM—with up to 4.2× faster prefill and 1.9× faster training, while its retention adapts to the content of each image. Without training, PixelPrune preserves accuracy on documents and GUI question answering; on GUI grounding, distillation restores full-token accuracy, and a model trained from scratch with PixelPrune matches full-token training and can be served with either pruned or full-token input. PixelPrune stays the most accurate reduction method across scales (4B, 8B) and architectures (Qwen3.5, InternVL3.5), and with near-exact matching it remains within 2.1 points of Full on general-domain images.

AI USE STATEMENT

In this work, we used generative AI tools for implementing the baseline methods (§4) and for cleaning and reformatting datasets. We have not used generative AI tools for creating synthetic datasets, building theoretical models or conceptual frameworks, making mathematical claims or proofs, generating hypotheses, designing methods or experiments, or interpreting results, and translation and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools for drafting sections of the paper and polishing its writing. We have reviewed all AI-assisted work. The baseline implementations were unit-tested and checked against the procedures described in their original papers, the processed data were checked against the source datasets, and all drafted and polished text was reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *CVPR*, 2025.

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

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025a. URL https://arxiv.org/abs/2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *ICLR*, 2023.

Thomas Boutell. PNG (portable network graphics) specification version 1.0. *RFC*, 2083:1–102, 1997. doi: 10.17487/RFC2083.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *ECCV*, 2024a.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models? In *NeurIPS*, 2024b.

Yuan Chen, Zichen Wen, Yuzhou Wu, Xuyang Liu, Shuang Chen, Junpeng Ma, Weijia Li, Conghui He, and Linfeng Zhang. IPCV: Information-preserving compression for MLLM visual encoders. *arXiv preprint arXiv:2512.18747*, 2025.

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

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024d.

Rohan Choudhury, Guanglei Zhu, Sihan Liu, Koichiro Niinuma, Kris M. Kitani, and László Jeni. Don't look twice: Faster video transformers with run-length tokenization. *arXiv preprint arXiv:2411.05222*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *NeurIPS*, 2022.

Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim M Alabdulmohsin, et al. Patch n' pack: Navit, a vision transformer for any aspect ratio and resolution. In *NeurIPS*, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Haodong Duan, Xinyu Fang, Junming Yang, et al. VLMEvalKit: An open-source toolkit for evaluating large multi-modality models. *arXiv preprint arXiv:2407.11691*, 2024.

Kechen Fang, Yihua Qin, Chongyi Wang, Wenshuo Ma, Tianyu Yu, and Yuan Yao. LLaVA-UHD v4: What makes efficient visual encoding in MLLMs? *arXiv preprint arXiv:2605.08985*, 2026.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Yefei He, Feng Chen, Jing Liu, Wenqi Shao, Hong Zhou, Kaipeng Zhang, and Bohan Zhuang. Zipvl: Efficient large vision-language models with dynamic token sparsification. *arXiv preprint arXiv:2410.08584*, 2024.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, et al. Cogagent: A visual language model for gui agents. In *CVPR*, 2024.

Yu-Chung Hsiao, Fedir Zubach, Gilles Baechler, Srinivas Sunkara, Victor Carbune, Jason Lin, Maria Wang, Yun Zhu, and Jindong Chen. ScreenQA: Large-scale question-answer pairs over mobile app screenshots. In *NAACL*, 2025.

Anwen Hu, Haiyang Xu, Jiabo Ye, Ming Yan, Liang Zhang, Bo Zhang, Chen Li, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. mplug-docowl 1.5: Unified structure learning for ocr-free document understanding. In *Findings of EMNLP*, 2024a.

Anwen Hu, Haiyang Xu, Liang Zhang, Jiabo Ye, Ming Yan, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. mplug-docowl2: High-resolution compressing for ocr-free multi-page document understanding. *arXiv preprint arXiv:2409.03420*, 2024b.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *ECCV*, 2016.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *TMLR*, 2025a.

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 *ICML*, 2023.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: Gui grounding for professional high-resolution computer use. In *ACM MM*, 2025b.

Wentong Li, Yuqian Yuan, Jian Liu, Dongqi Tang, Song Wang, Jie Qin, Jianke Zhu, and Lei Zhang. Tokenpacker: Efficient visual projector for multimodal llm. *arXiv preprint arXiv:2407.02392*, 2024.

Yuankai Li, Tinghui Zhu, Ha Min Son, Zhe Zhao, Xin Liu, and Muhao Chen. AQuaUI: Visual token reduction for GUI agents with adaptive quadtrees. *arXiv preprint arXiv:2605.19260*, 2026.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *ICLR*, 2022.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. *arXiv preprint arXiv:2411.17465*, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

Junjie Liu, Shengyuan Ye, and Xu Chen. PACE: A unified condense-and-extract paradigm for fast VLM inference. *arXiv preprint arXiv:2608.27206*, 2026.

Ting Liu, Liangtao Shi, Richang Hong, Yue Hu, Quanjun Yin, and Linfeng Zhang. Multi-stage vision token dropping: Towards efficient multimodal large language model. *arXiv preprint arXiv:2411.10803*, 2024a.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. Mmbench: Is your multi-modal model an all-around player? In *ECCV*, 2024b.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xucheng Yin, Cheng-lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: On the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 2024c.

Yuliang Liu, Biao Yang, Qiang Liu, Zhang Li, Zhiyin Ma, Shuo Zhang, and Xiang Bai. Textmonkey: An ocr-free large multimodal model for understanding document. *arXiv preprint arXiv:2403.04473*, 2024d.

Zhaoyang Liu, Jingjing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, Shenglong Ye, Qingyun Li, Xuan Dong, Yue Yu, Chenyu Lu, YunXiang Mo, Yao Yan, Zeyue Tian, Xiao Zhang, Yuan Huang, Yiqian Liu, Weijie Su, Gen Luo, Xiangyu Yue, Biqing Qi, Kai Chen, Bowen Zhou, Yu Qiao, Qifeng Chen, and Wenhai Wang. ScaleCUA: Scaling open-source computer use agents with cross-platform data. *arXiv preprint arXiv:2509.15221*, 2025.

Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, Pan Zhang, Liangming Pan, Yu-Gang Jiang, Jiaqi Wang, Yixin Cao, and Aixin Sun. Mmlongbench-doc: Benchmarking long-context document understanding with visualizations. In *NeurIPS*, 2024.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *ACL Findings*, 2022.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *WACV*, 2021.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and C.V. Jawahar. Infographicvqa. In *WACV*, 2022.

Lingchen Meng, Jianwei Yang, Rui Tian, Xiyang Dai, Zuxuan Wu, Jianfeng Gao, and Yu-Gang Jiang. Deepstack: Deeply stacking visual tokens is surprisingly simple and effective for lmms. In *NeurIPS*, 2024.

Mingyu Ouyang, Kevin Qinghong Lin, Mike Zheng Shou, and Hwee Tou Ng. FocusUI: Efficient UI grounding via position-preserving visual token selection. In *CVPR*, 2026.

Jake Poznanski, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Aman Rangapur, Luca Soldaini, Kyle Lo, and Chris Wilhelm. olmocr: Unlocking trillions of tokens in pdfs with vision language models. *arXiv preprint arXiv:2502.18443*, 2025.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC*, 2020.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *NeurIPS*, 2021.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. *arXiv preprint arXiv:2403.15388*, 2024.

Kele Shao, Keda Tao, Kejia Zhang, Sicheng Feng, Mu Cai, Yuzhang Shang, Haoxuan You, Can Qin, Yang Sui, and Huan Wang. A survey of token compression for efficient multimodal large language models. *arXiv preprint arXiv:2507.20198*, 2025.

Jaemin Son, Sujin Choi, and Inyong Yun. Index-preserving lightweight token pruning for efficient document understanding in vision-language models. *CoRR*, abs/2509.06415, 2025.

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

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025a.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *AAAI*, 2025b.

Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, Weiyun Wang, Xiangyu Zhao, Jixuan Chen, Haodong Duan, Tianbao Xie, Shiqian Su, Chenyu Yang, Yue Yu, Yuan Huang, Yiqian Liu, Xiao Zhang, Xiangyu Yue, Weijie Su, Xizhou Zhu, Wei Shen, Jifeng Dai, and Wenhai Wang. MMBench-GUI: Hierarchical multi-platform evaluation framework for GUI agents. *arXiv preprint arXiv:2507.19478*, 2025c.

Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, Chunrui Han, and Xiangyu Zhang. General ocr theory: Towards ocr-2.0 via a unified end-to-end model. *arXiv preprint arXiv:2409.01704*, 2024.

Marcelo J. Weinberger, Gadiel Seroussi, and Guillermo Sapiro. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. *IEEE Transactions on Image Processing*, 9(8):1309–1324, 2000.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal LLMs. In *CVPR*, 2024.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. Os-atlas: Foundation action model for generalist gui agents. In *ICLR*, 2025.

xAI. Realworldqa: An AI benchmark for real-world understanding. https://huggingface.co/datasets/xai-org/RealWorldQA, 2024.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, Yiheng Xu, Junli Wang, Doyen Sahoo, Tao Yu, and Caiming Xiong. Scaling computer-use grounding via user interface decomposition and synthesis. *arXiv preprint arXiv:2505.13227*, 2025.

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

Fengze Yang, Bo Yu, Xuewen Luo, Cathy Liu, and Chenxi Liu. CIVIC: End-to-end sequence compactness for efficient vision-language models. *arXiv preprint arXiv:2605.28115*, 2026.

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. In *CVPR*, 2025.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *CVPR*, 2024.

Jiaxin Zhang, Wentao Yang, Songxuan Lai, Zecheng Xie, and Lianwen Jin. Dockylin: A large multimodal model for visual document understanding with efficient visual slimming. In *AAAI*, 2025.

Dehua Zheng, Mouxiao Huang, Borui Jiang, Hailin Hu, and Xinghao Chen. Towards lossless ultimate vision token compression for vlms. *arXiv preprint arXiv:2512.09010*, 2025.

A Pixel-Level Redundancy and Resolution Statistics

Table A1 summarizes the pixel-level patch redundancy and resolution characteristics of all benchmarks. We divide each image into non-overlapping 32×32 blocks and count its distinct block contents. **Duplicate Ratio** is the dataset-level fraction of blocks that repeat a content already present, $1 - \sum_i U_i / \sum_i N_i$, where U_i and N_i are the numbers of distinct and total blocks in the i-th image. This global dedup statistic characterizes the overall pixel-level redundancy of each benchmark; Pixel-Prune's actual retain ratio (§4) is higher because its causal prediction can only exploit redundancy along the scan order.

We also report the **Mean Resolution** (width×height in pixels) of each benchmark's original images; the duplicate statistics are computed after Qwen3-VL's default resolution preprocessing (`min_pixels`=256×256, `max_pixels`=4096×4096). Resolution directly affects the number of visual tokens: higher-resolution benchmarks produce more patches and thus more opportunities for redundancy, but the relationship is not strictly monotonic—content density matters as well.

On average, GUI images exhibit higher duplication (51.4% mean across the 9 ScreenSpot sub-tasks) than documents (39.7% mean across the 7 document benchmarks), driven by large solid-color regions such as backgrounds, panels, and status bars. Among documents, MMLongBench-Doc leads (52.4%, multi-page white margins) while InfoVQA is lowest (28.6%, dense infographics).

B Proofs of Theoretical Guarantees

Proof of Proposition 1 (Exact Reconstruction). We prove by induction on scan order. The anchor patch at $(0, 0)$ has no causal neighbors and is always retained. For each subsequent position i, omission occurs only when $P_i = \hat{P}_i$. The decoder, processing patches in the same scan order, recomputes the identical prediction $\hat{P}_i$ from neighbors that have already been recovered exactly (by inductive hypothesis); thus the decoder outputs $\hat{P}_i = P_i$. The conclusion follows by induction. □

Proof of Proposition 2 (Bounded Reconstruction Error). At $\tau{=}0$ the omitted patch is identical to its prediction, so the proof reduces to Proposition 1. At $\tau{>}0$, near-similarity is not transitive—a chain of pairwise-similar patches can drift arbitrarily—so the encoder uses the predicted value (not the original) for each omitted patch when it serves as a neighbor for subsequent predictions. Since encoder and decoder maintain identical state (both use predicted values for omitted patches), they compute identical predictions $\hat{P}_i$. An omitted patch satisfies $\mathrm{dist}(P_i, \hat{P}_i) \leq \tau$ by construction;

Table A1: Pixel-level redundancy and resolution per benchmark (32×32 patches, exact match).

Benchmark	#Images	Mean Resolution	Duplicate Ratio (%)
Document Understanding			
OCRBench	1,000	615×732	38.5
DocVQA	1,286	1775×2082	33.4
InfoVQA	500	1141×2929	28.6
ChartQA	2,500	768×583	40.8
AI2D	1,204	609×486	37.7
MMLongBench-Doc	5,784	1284×1405	52.4
olmOCRBench	1,403	1556×2007	46.6
GUI Understanding			
SSv2-Web	251	2397×1348	63.6
SSv2-Mobile	248	1387×2339	56.2
SSv2-Desktop	231	1369×907	42.3
SSPro-Dev.	299	3624×2119	43.0
SSPro-Creative	341	3040×1616	46.6
SSPro-CAD	261	3448×1190	40.3
SSPro-Scientific	254	2717×1616	77.6
SSPro-Office	230	2980×1778	51.1
SSPro-OS	196	3926×2118	41.7

Table A2: Training-free document results (%) on Qwen3-VL-4B and 8B. Parentheses: PixelPrune's retain ratio.

Model	Method	Stage	OCRBench (66.1%)	DocVQA (70.1%)	InfoVQA (76.7%)	ChartQA (73.1%)	AI2D (69.2%)	MML-Doc (50.3%)	olmOCR (56.9%)	Avg
Qwen3-VL-4B	Full	–	88.1	94.8	79.5	82.9	82.8	36.7	54.0	74.1
	FastV	post-ViT	85.6	91.8	78.5	79.6	82.0	32.1	44.9	70.6
	VisionZip	post-ViT	87.5	94.1	**79.3**	82.0	**83.6**	33.0	51.8	73.0
	DivPrune	post-ViT	86.7	91.2	77.8	79.0	82.2	29.7	45.8	70.3
	Random	pre-ViT	76.9	70.2	70.4	64.6	78.5	20.0	37.9	59.8
	ConnComp	pre-ViT	78.2	84.4	73.5	70.0	79.2	23.8	41.5	64.4
	PixelPrune	pre-ViT	**88.0**	**94.7**	79.2	**82.4**	81.7	**36.3**	**53.8**	**73.7**
Qwen3-VL-8B	Full	–	90.6	95.7	83.1	83.4	82.7	38.0	54.7	75.5
	FastV	post-ViT	87.2	91.5	81.3	78.0	83.4	34.1	45.3	71.5
	VisionZip	post-ViT	90.0	**95.5**	**82.9**	82.3	**84.2**	37.4	53.8	75.2
	DivPrune	post-ViT	89.9	92.6	81.3	80.4	83.3	30.3	48.2	72.3
	Random	pre-ViT	79.9	72.8	75.5	66.0	79.4	23.5	40.0	62.4
	ConnComp	pre-ViT	82.2	86.3	77.4	67.8	80.7	26.8	43.2	66.4
	PixelPrune	pre-ViT	**90.4**	**95.5**	82.8	**82.9**	82.8	**38.2**	**54.3**	**75.3**

since the decoder outputs exactly $\hat{P}_i$ for that patch, the per-patch error is at most τ. Because every neighbor used in a prediction is the encoder/decoder-aligned value, errors do not accumulate across patches. □

C Scaling to Larger Backbones

Table A2 repeats the training-free document evaluation on Qwen3-VL-4B and 8B under the same per-image token budget. The ranking is stable across scales: PixelPrune remains the strongest non-Full method at both 4B (73.7 Avg) and 8B (75.3 Avg), and its gap to Full narrows as capacity grows (0.9/0.4/0.2 points at 2B/4B/8B).

Table A3: Training-free document results (%) on Qwen3.5-2B and InternVL3.5-2B. Parentheses: PixelPrune's retain ratio.

Method	Stage	OCRBench	DocVQA	InfoVQA	ChartQA	AI2D	MML-Doc	olmOCR	Avg
Qwen3.5-2B		(66.1%)	(70.1%)	(76.7%)	(73.1%)	(69.2%)	(50.3%)	(56.9%)	
Full	–	87.3	93.5	75.4	78.8	79.6	32.4	49.9	71.0
FastV	post-ViT	80.3	79.9	71.6	70.1	78.6	24.5	37.8	63.3
VisionZip	post-ViT	86.3	91.4	**74.5**	**78.0**	79.4	29.5	46.8	69.4
DivPrune	post-ViT	84.8	85.9	70.6	75.6	**79.6**	26.1	40.1	66.1
Random	pre-ViT	75.8	67.5	65.1	60.1	75.1	18.3	36.0	56.8
ConnComp	pre-ViT	77.9	81.3	67.6	65.1	76.5	21.4	38.2	61.1
PixelPrune	pre-ViT	**87.9**	**92.5**	74.3	77.9	78.7	**31.6**	**48.7**	**70.2**
InternVL3.5-2B		(84.4%)	(75.1%)	(83.4%)	(60.2%)	(64.3%)	(68.3%)	(57.7%)	
Full	–	82.9	88.3	69.2	81.6	79.0	16.5	60.2	68.2
FastV	post-ViT	81.1	80.5	66.4	79.1	78.0	14.7	46.8	63.8
VisionZip	post-ViT	81.0	81.3	67.5	77.0	77.3	15.1	49.1	64.0
DivPrune	post-ViT	71.9	69.3	59.6	64.2	72.8	12.3	42.6	56.1
Random	pre-ViT	72.1	67.4	61.0	62.7	74.4	11.9	40.0	55.6
ConnComp	pre-ViT	74.6	76.1	62.9	69.6	77.0	13.5	42.7	59.5
PixelPrune	pre-ViT	**82.7**	**87.9**	**68.5**	**80.4**	**78.1**	**16.8**	**57.0**	**67.3**

D Cross-Architecture Validation

We repeat the training-free document evaluation of Table 1 on two further 2B models (Table A3), with the same five reduction methods as Table 1, all at PixelPrune's per-image token budget. **Qwen3.5-2B** (Qwen Team, 2026) keeps the NaViT-style ViT with separate spatial positional encodings but uses a hybrid LLM that interleaves full-attention layers with linear-attention layers carrying no explicit positional encoding, which tests whether position preservation still suffices when the LLM's positional signal is weaker. **InternVL3.5-2B** (Wang et al., 2025a) instead splits each image into fixed 448×448 tiles, adds learned absolute position embeddings in the ViT before pruning, and uses 1D RoPE in the LLM; there, every reduction method gives the retained tokens their original position indices rather than consecutive ones. FastV prunes after the third LLM layer on InternVL3.5 and after the fourth on Qwen3.5, whose third layer uses linear attention and produces no attention map.

PixelPrune has the highest average among the reduction methods on both models. On Qwen3.5-2B it is 0.8 points below Full (70.2 vs. 71.0) and 0.8 above VisionZip; VisionZip or DivPrune is slightly ahead on InfoVQA, ChartQA and AI2D, by 0.1–0.9 points. On InternVL3.5-2B it is the most accurate reduction method on every benchmark, 0.9 points below Full on average and 3.3 above VisionZip. On both models, Random and ConnComp, which prune at the same pre-ViT stage, trail PixelPrune by 8–13 points on average. On InternVL3.5-2B, MMLongBench-Doc barely separates the methods: with dozens of pages per sample, the fixed tile budget leaves many pages a single thumbnail tile, and even Full reaches only 16.5.

E General-Domain Evaluation

Natural images rarely contain pixel-identical patches, so this setting mainly tests near-exact matching (§4.5.2). We apply PixelPrune training-free to Qwen3-VL-2B on six general benchmarks—MMBench (Liu et al., 2024b), MMMU (Yue et al., 2024), MMStar (Chen et al., 2024b), RealWorldQA (xAI, 2024), HR-Bench (Wang et al., 2025b) and V*Bench (Wu & Xie, 2024) (Table A4)—with the Max distance at several thresholds τ, and compare each setting with Full.

Exact matching (τ=0) retains 80–100% of tokens with an accuracy change of at most 1.1 points. Near-exact matching further reduces retention: at τ=0.05, retention drops to 72–89% yet accuracy stays within 0.6 points of Full on most benchmarks. Even at τ=0.10 (68–83% retention), accuracy drops by at most 2.1 points.

F Evaluation and Baseline Details

Metrics. Each benchmark uses its standard metric as implemented in VLMEvalKit: ANLS for DocVQA and InfoVQA, relaxed accuracy for ChartQA, accuracy for AI2D, the OCRBench score (out of 1,000, reported as a percentage), overall F1 for MMLongBench-Doc, and the unit-test pass rate for

Table A4: General-domain results (Qwen3-VL-2B, training-free, Max distance): retain ratio and accuracy (%) per benchmark.

	MMBench		MMMU		MMStar		RealWorldQA		HR-Bench		V*Bench	
τ	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**
Full (no dedup)	100	77.1	100	50.3	100	55.1	100	66.3	100	70.6	100	71.2
0 (exact)	94.7	**77.1**	80.1	49.2	81.2	55.1	97.7	66.3	96.3	71.0	99.7	71.2
0.005	94.2	76.9	79.2	49.7	80.5	55.1	97.6	66.1	94.9	70.8	99.3	71.2
0.02	91.8	76.9	76.1	49.1	77.2	**55.3**	96.6	65.8	84.6	71.4	94.0	**73.8**
0.05	88.0	76.5	72.4	**50.3**	73.3	55.0	89.2	66.1	77.3	**72.0**	87.7	72.8
0.10	83.5	76.5	68.0	48.2	68.8	54.8	75.4	**67.1**	70.1	71.9	82.3	71.2

olmOCRBench. Multiple-choice benchmarks (AI2D and the general-domain benchmarks) extract the chosen option by rule matching and fall back to an LLM judge when that fails; MMLongBench-Doc uses an LLM judge to extract answers. GUI question answering uses token F1 on ScreenQA-Short and accuracy on MMBench-GUI L1 (unweighted across difficulty levels, unlike the official score), and the general-domain benchmarks use accuracy. For GUI grounding, the model outputs a point in Qwen3-VL's relative $[0, 1000]$ coordinates, which counts as correct if it falls inside the target element's bounding box. Retain ratios are pooled over images rather than averaged per sample, so that large images, which dominate compute, carry proportional weight.

Baselines. Every reduction method keeps, for each image, as many visual tokens (after the 2×2 merge) as PixelPrune retains for that image, and every retained token keeps its position id from the full grid. FastV ranks visual tokens by the average attention they receive in the third LLM layer and removes the rest from all later layers; we use three layers instead of the original two because Qwen3-VL injects DeepStack features into its first three LLM layers. VisionZip, lacking a `[CLS]` token in Qwen3-VL, ranks tokens by the attention they receive in the last ViT layer, keeps the top-ranked ones and merges the remainder into a few contextual tokens, splitting the budget 13:1 between the two. DivPrune greedily selects the subset of encoder outputs with the largest minimum pairwise cosine distance.

G Training Details

All training uses NVIDIA H20 GPUs with DeepSpeed ZeRO-2 (Rajbhandari et al., 2020), FlashAttention-2 (Dao et al., 2022), gradient checkpointing and data packing.

Distillation (§4.3.1). PixelPrune, the two pre-ViT controls and all GUI ablations share one recipe: starting from Qwen3-VL-2B-Instruct, we train for 300 steps on 8 GPUs with AdamW (learning rate 10^{-5}, cosine decay), scaling the ViT learning rate by 0.25, on a 1:1 mix of Jedi and the Windows grounding data of ScaleCUA (Liu et al., 2025) (445K samples after cleaning). Each method prunes with its own criterion at PixelPrune's per-image token budget during both training and evaluation.

Training from scratch (§4.3.2) uses the same Jedi subset on 32 GPUs $\times$ 18 samples/GPU, the largest batch that fits the Full configuration, for 4,000 steps. The PixelPrune selector is implemented as a vectorized GPU module attached to the ViT entry point, adding negligible per-step overhead.

H Training from Scratch: Convergence and Efficiency

Table A5 reports intermediate evaluation results during training from scratch (Section 4.3.2), comparing **Full** and **PixelPrune** configurations. Each configuration uses the same visual token setting for both training and inference: Full trains and evaluates with all tokens, while PixelPrune trains and evaluates with compressed tokens.

Both configurations converge comparably, reaching similar accuracy by step 2,400 and plateauing around step 2,400–3,200. PixelPrune achieves slightly higher final performance (mean 58.1 vs. 56.6 at step 4,000), possibly because reduced token count provides implicit regularization.

Table A5: Training-from-scratch convergence: grounding accuracy (%) every 400 steps. Full: all tokens in training and evaluation; PixelPrune: pruned tokens in both.

		ScreenSpot V2				ScreenSpot Pro						
Mode	**Step**	**Web**	**Mobile**	**Desktop**	**Avg**	**Dev.**	**Cre.**	**CAD**	**Sci.**	**Office**	**OS**	**Avg**
Full	400	66.1	78.2	75.1	73.2	14.7	14.1	5.0	20.5	22.6	10.7	14.6
	800	74.4	84.4	82.3	80.4	22.4	19.6	12.6	24.4	29.1	15.3	20.6
	1200	78.0	87.2	85.9	83.7	24.7	22.6	14.9	29.5	40.9	24.5	26.2
	1600	78.7	88.4	85.0	84.1	29.8	28.2	13.4	31.5	42.6	25.5	28.5
	2000	78.7	84.4	83.5	82.2	25.1	23.5	14.2	27.6	40.9	25.0	26.0
	2400	79.4	84.4	87.4	83.8	29.4	24.9	16.9	34.3	48.3	29.1	30.5
	2800	80.1	78.6	85.6	81.5	32.1	26.7	15.3	35.4	49.1	30.6	31.5
	3200	79.9	84.8	85.0	83.2	31.8	26.1	16.5	33.9	49.1	27.6	30.8
	3600	80.1	83.4	85.0	82.9	31.1	27.3	16.1	34.3	48.7	28.6	31.0
	4000	79.9	81.6	83.8	81.8	31.8	26.7	15.3	35.0	49.6	30.6	31.5
PixelPrune	400	61.1	81.4	76.3	73.0	13.4	10.9	4.6	15.7	17.8	11.2	12.3
	800	72.3	86.0	82.3	80.2	21.7	19.1	7.7	26.4	29.6	13.8	19.7
	1200	75.5	87.6	84.1	82.4	22.4	20.2	13.0	29.5	36.1	21.9	23.9
	1600	76.2	87.0	87.1	83.5	27.4	26.4	14.9	33.9	42.6	26.0	28.5
	2000	78.3	87.2	86.8	84.1	27.4	23.5	14.6	30.7	41.7	28.1	27.7
	2400	79.6	85.2	88.3	84.4	30.1	27.6	15.7	34.6	50.9	27.0	31.0
	2800	79.4	86.4	86.8	84.2	27.8	27.9	14.2	34.3	50.0	29.1	30.5
	3200	80.8	87.4	87.7	85.3	31.4	27.3	14.6	34.3	51.3	28.6	31.2
	3600	79.6	85.8	88.0	84.5	30.1	28.4	16.1	34.3	50.4	30.1	31.6
	4000	79.9	85.6	88.9	84.8	30.8	27.3	15.7	34.6	52.6	27.6	31.4

Table A6 reports the training efficiency of the two configurations (§4.4.2). Both keep sample packing at the original token lengths, so they process the same samples per step.

Table A6: Training efficiency in the from-scratch experiment (§4.3.2); speedup relative to Full.

Method	**Retain**	**Forward (s)**	**Backward (s)**	**Peak Mem (GB)**	**Wall Time (h)**
Full	100.0%	7.9	35.3	57.8	49.1
PixelPrune	40.8%	3.9	18.0	38.4	25.3
Speedup	–	**2.0**×	**2.0**×	**↓33.6%**	**1.9**×

I Detailed FLOPs Analysis

We derive the FLOPs formula for the Qwen3-VL architecture, which comprises three stages: ViT encoding, Patch Merger (including DeepStack), and LLM processing. For an input image with N patches ($N = (H/p) \times (W/p)$ for ViT patch size p) and T text tokens:

$$\begin{aligned}\text{FLOPs}_{\text{total}} = &\underbrace{L_v\big[\, N\big(8D_v^2 + 4D_v D_{f,v}\big) + 4N^2 D_v \,\big]}_{\text{(a) ViT Encoder}} \\ &+ \underbrace{L_m \,\frac{N}{M^2}\big(2D_{\text{in}}^2 + 2D_{\text{in}} D_l\big)}_{\text{(b) Patch Merger \& DeepStack}} + \underbrace{L_l\big[\, N_l C_l + 4N_l^2 D_l \,\big]}_{\text{(c) LLM}}\end{aligned} \tag{3}$$

where $N_l = N/M^2 + T$ is the total LLM sequence length (M=2 is the spatial merge factor, T is the number of text tokens), $D_{f,v}$ is the ViT FFN dimension, $D_{\text{in}} = M^2 \cdot D_v$ is the Patch Merger input dimension, $L_m = 1 + |\mathcal{I}_{\text{ds}}|$ accounts for the main merger plus DeepStack mergers, and C_l is the per-token linear cost of the LLM (defined in Eq. 4 below).

(a) ViT Encoder. Each of the L_v ViT blocks contains multi-head self-attention (MHSA) and a feed-forward network (FFN). Linear projections contribute $8D_v^2 + 4D_v D_{f,v}$ FLOPs per token (QKV,

output, and two FFN layers); attention maps contribute $4N^2D_v$ FLOPs per layer. The ViT computes attention *per image* (controlled by `cu_seqlens`), so N refers to the patch count of a single image.

(b) Patch Merger & DeepStack. After $M \times M$ spatial merge, each token has dimension $D_{\text{in}} = M^2 D_v$. The merger MLP consists of `fc1`: $D_{\text{in}} \to D_{\text{in}}$ ($2\frac{N}{M^2}D_{\text{in}}^2$ FLOPs) and `fc2`: $D_{\text{in}} \to D_l$ ($2\frac{N}{M^2}D_{\text{in}}D_l$ FLOPs). The main merger runs once on the final ViT output; each DeepStack merger runs on an intermediate layer's output, giving a total count of L_m.

(c) LLM. Qwen3-VL uses Grouped-Query Attention (GQA) with n_q query heads and n_{kv} key-value heads, and a gated MLP (SwiGLU). The per-token linear cost per decoder layer is:

$$C_l = \underbrace{2D_l(2n_q + 2n_{kv})d_h}_{\text{QKV + output proj.}} + \underbrace{6D_l D_{f,l}}_{\text{Gated MLP (gate + up + down)}} \tag{4}$$

where $d_h = D_l/n_q$ and $D_{f,l}$ is the LLM FFN dimension. The Q and output projections each have dimension $D_l \times D_l$, while K and V projections have dimension $D_l \times (n_{kv}d_h)$. Attention maps add $4N_l^2 D_l$ FLOPs per layer (since $n_q d_h = D_l$, this is independent of n_{kv}).

Architecture parameters. For Qwen3-VL-2B: (a) ViT: L_v=24, D_v=1024, $D_{f,v}$=4096, n_h=16; (b) Patch Merger: M=2, $D_{\text{in}} = 4096$, D_l=2048, $|\mathcal{I}_{\text{ds}}|$=3; (c) LLM: L_l=28, D_l=2048, $D_{f,l}$=6144, n_q=16, n_{kv}=8, d_h=128.

FLOPs savings with PixelPrune. Let $N_s = |\mathcal{S}| \leq N$ denote the number of retained patches after PixelPrune (§3.2). Replacing N with N_s in the formula above yields: (i) **ViT**: attention FLOPs reduce from $O(N^2)$ to $O(N_s^2)$ (quadratic saving), linear FLOPs reduce proportionally; (ii) **Patch Merger**: scales linearly, from $O(N/M^2)$ to $O(N_s/M^2)$; (iii) **LLM**: the sequence length reduces from N_l to $N_l' = N_s/M^2 + T$, so attention reduces from $O(N_l^2)$ to $O(N_l'^2)$. When $N_s \ll N$, the end-to-end reduction is substantial across all three stages. Plugging the architecture parameters and measured N_s into Eq. 3 yields the theoretical FLOPs reported in Table 5.

J Token Reduction at Different ViT Depths

We ablate the pruning position within the Qwen3-VL-2B ViT encoder (24 blocks; DeepStack at blocks 5, 11, 17). All settings apply the same retain mask at different depths—after the 0th, 4th, 17th, or final ViT block, or before the ViT (PixelPrune default)—so the retained tokens share the same spatial positions but carry features from different depths.

Table A7: Pruning at different ViT depths (Qwen3-VL-2B, %). Document: training-free; GUI: each depth distilled separately with the recipe of §4.3.1.

	Document			GUI		
Depth	**DocVQA**	**InfoVQA**	**ChartQA**	**Web**	**Mobile**	**Desktop**
After ViT	**92.2**	71.0	76.8	**87.4**	92.0	**93.4**
After 17th	91.9	71.1	76.1	86.3	92.4	92.8
After 4th	**92.2**	71.4	**77.0**	86.3	**92.6**	91.3
After 0th	92.1	71.3	76.9	84.9	91.8	92.8
Before ViT	92.0	**71.5**	**77.0**	86.3	91.6	92.5

Table A7 reveals two findings. First, accuracy is largely insensitive to pruning depth: no single layer consistently dominates, and PixelPrune (before ViT) performs on par with all later-pruning variants despite the latter preserving richer learned features for the kept tokens. Second, earlier pruning yields greater computational savings. After-ViT/17th/4th/0th pruning saves 0%/29%/83%/96% of ViT FLOPs for removed tokens, respectively; only PixelPrune (before ViT) saves 100%, delivering full-pipeline acceleration.