FastOCR: Dynamic Visual Fixation via KV Cache Pruning for Efficient Document Parsing

Anonymous authors
Paper under double-blind review

Abstract

Vision-Language Models (VLMs) have shown strong promise on Optical Character Recognition (OCR), yet the sheer number of visual tokens needed to encode dense documents makes inference prohibitively expensive. Existing methods like SnapKV evict visual tokens, which often fails on OCR tasks given the high information density of document images. We observe that, although document images appear globally dense and seemingly unprunable, a VLM's top-attended visual-token sets are line-local and drift gradually across decoding steps, much as a human reader fixates on successive words rather than on a whole page at once. Turning it into a method, we propose **FastOCR**, a training-free framework with two complementary modules: *Focal-Guided Pruning* identifies a small set of focal layers and selects the most task-relevant visual tokens at each step, while *Cross-Step Fixation Reuse* warm-starts each step from the previous one. FastOCR changes only which cached entries are attended and evicts none, so every token stays recoverable at every later step. Under matched nominal reference attention budgets, FastOCR outperforms every baseline we evaluate on OmniDocBench and olmOCR-Bench, and generalizes across various VLMs. On Qwen2.5-VL it retains **98.3%** of the unpruned model's accuracy on OmniDocBench and **97.3%** on olmOCR-Bench, with a **5%** visual-token budget at pruned layers. Latency measurements reach up to **3.4**$\times$ attention speedup on olmOCR.

1 Introduction

Vision-language models (VLMs) (Alayrac et al., 2022; Liu et al., 2023; Bai et al., 2025) have achieved remarkable success in optical character recognition (OCR), with dedicated methods such as Donut (Kim et al., 2021), Nougat (Blecher et al., 2024), and DeepSeek-OCR (Wei et al., 2025) converting diverse documents into structured text. However, OCR requires faithfully transcribing every character on a page, producing far more visual tokens than typical visual understanding tasks and making inference prohibitively expensive due to the quadratic cost of attention (Vaswani et al., 2017). In this work, we focus on *training-free KV cache pruning* to accelerate VLMs on OCR tasks.

Many visual-token and KV-cache compression methods use *physical eviction*, permanently discarding visual tokens. Some methods perform eviction during visual encoding (Yang et al., 2025; Shang et al., 2024) or the prefill stage (Chen et al., 2024; Zhang et al., 2025c), while others (Zhang et al., 2023; Li et al., 2024; Cai et al., 2024) allocate the cache budget in a modality-agnostic way and ignore the distinction between image and text tokens. As illustrated in Figure 1(a), this strategy works well for general image understanding, where a few salient patches suffice to recognize the subject. For dense OCR, such removal can destroy information needed later in decoding. Evicting patches from a document image destroys character-level information irreversibly, causing the model to misread "Classic Boat Museum" as "Glassed Bottle Museum." Once a visual token is evicted, it cannot be revisited when a later decoding step needs it. Query-aware sparse attention such as Quest (Tang et al., 2024) instead retains the cache and selects which entries to access at each step.

We draw inspiration from human reading (Rayner, 1998), where a reader does not perceive an entire page at once but instead *fixates* on a small area before shifting gaze to the next. We observe an analogous phenomenon in VLMs. Although document images appear globally dense and seemingly unprunable, the model's top-attended visual tokens form line-local sets that shift gradually as generation proceeds. For example, as Figure 1(b) shows, when transcribing "Classic Boat Museum"

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.

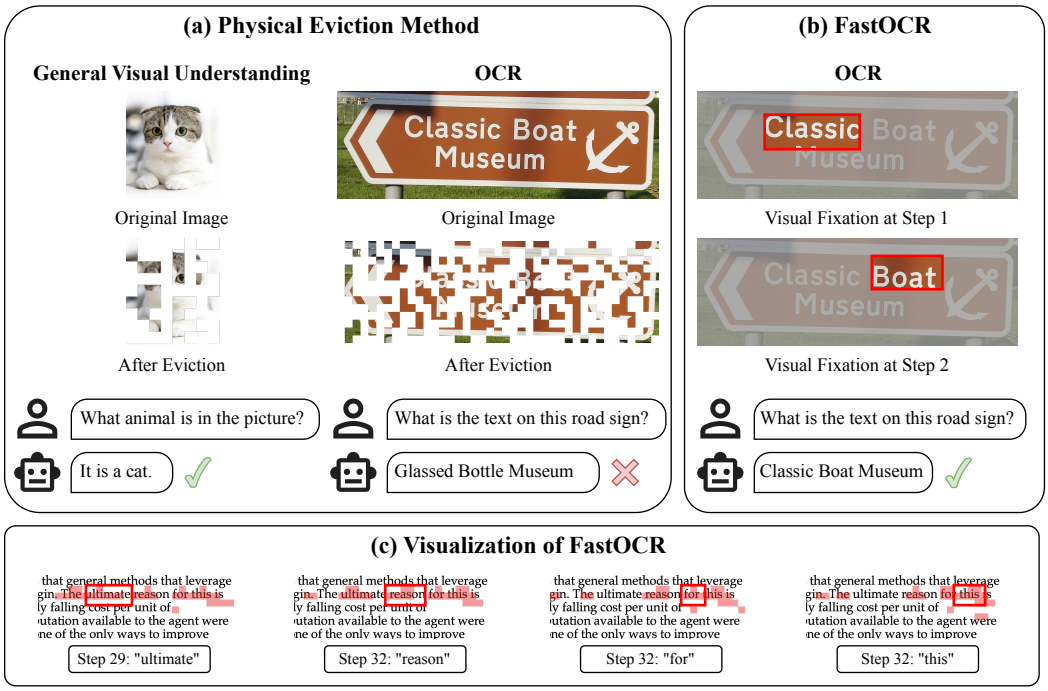

Figure 1: Comparison of FastOCR with existing KV cache pruning methods. FastOCR dynamically retains only the most relevant visual tokens at each decoding step, achieving substantial speedup with minimal accuracy loss. (c) visualizes FastOCR in action, where the red regions highlight the visual tokens actually attended to during inference.

the model first fixates on the region around "Classic" and then shifts to "Boat". We observe this structure in unpruned Qwen2.5-VL at fixed layers: consecutive top-attended token sets overlap with IoU 0.51, versus 0.14 after shuffling step order; FastOCR with model-selected focal layers yields similar values, 0.50 and 0.15 (Section 3.1). This Dynamic Visual Fixation phenomenon recasts the intractable global pruning problem as a tractable local, dynamic one, where we only need to adjust which tokens are attended at each step rather than permanently evict them upfront. Figure 1(c) shows that FastOCR attends to only a small red-highlighted region each decoding step.

Building upon these insights, we propose FastOCR, a dynamic visual fixation framework via KV cache pruning for efficient VLM-based document parsing. FastOCR comprises two complementary modules. *Focal-Guided Pruning* (FGP) determines where the model looks at each decoding step. It identifies a small set of focal layers whose attention is most concentrated on image regions, selects the top-attended image tokens at these layers, and propagates the selection to all remaining layers. *Cross-Step Fixation Reuse* (CSFR) guides how the model's fixation moves across time. Since the attended region shifts only gradually between consecutive steps, the focal tokens from the previous step are reused to warm-start the current one. Together, the two modules attend to only a small fraction of visual tokens at each decoding step, avoiding any permanent information loss while substantially reducing attention computation.

Our main contributions are summarized as follows: (1) We characterize Dynamic Visual Fixation, the temporal structure of VLM attention during OCR decoding, and quantify it against controls that separate this structure from set size and from a static preference for particular tokens: the attended visual set is line-local and drifts gradually from step to step. (2) We propose FastOCR, comprising Focal-Guided Pruning and Cross-Step Fixation Reuse, which turns that structure into a training-free acceleration method: it dynamically attends to a small subset of visual tokens at each decoding step while keeping the full KV cache intact, so that no token is ever irrecoverable. (3) On Qwen2.5-VL and dots.ocr, FastOCR outperforms all four baselines on both benchmarks; two additional models test quality retention. Qwen2.5-VL retains **98.3%** of unpruned OmniDocBench accuracy with a **5%**

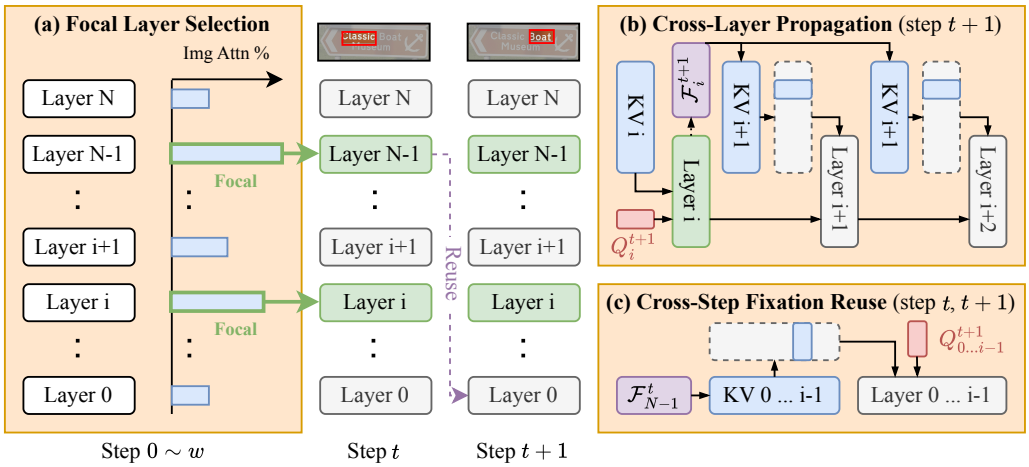

Figure 2: Overview of FastOCR (Section 3). Focal-Guided Pruning comprises (a) Focal Layer Selection, a warmup phase identifying focal layers (green), and (b) Cross-Layer Propagation. (c) Cross-Step Fixation Reuse carries the previous step's focal tokens into the next as a warm start.

visual-token budget at pruned layers. Latency measurements reach up to $\mathbf{3.4}\times$ attention speedup on olmOCR.

2 Related Work

OCR with VLMs, and the cost of visual tokens. General-purpose VLMs (Alayrac et al., 2022; Liu et al., 2023; Bai et al., 2025) and dedicated document models (Kim et al., 2021; Li et al., 2025; Poznanski et al., 2025) both parse documents accurately, and both share one bottleneck: a page yields far more visual tokens than a typical understanding task, in a regime where almost every token carries a character. Most methods reduce this cost by *physical eviction*: a token, once dropped, cannot be recovered. VLM-specific methods drop visual tokens during visual encoding (Yang et al., 2025; Shang et al., 2024) or at and after prefill (Chen et al., 2024; Zhang et al., 2025c), and a recent line replaces attention-based scoring with explicit diversity objectives (Alvar et al., 2025; Zhang et al., 2025a), but the decision is still made once, before decoding begins. Modality-agnostic LLM methods (Zhang et al., 2023; Li et al., 2024; Cai et al., 2024) neither distinguish image from text tokens nor exploit document structure. ZipVL (He et al., 2024) combines sparse prefill attention with discarding less important KV entries; VL-Cache (Tu et al., 2024) uses layer-adaptive budgets and modality-aware scoring to compress the cache. These methods reduce the visual information retained for subsequent decoding, which can be costly on dense documents.

Layer-adaptive and non-evicting selection. Two lines of work are closer to FastOCR. The first observes, as we do, that layers are *not* interchangeable: AdaptInfer (Zhang et al., 2025b) schedules pruning around turning-point layers, SAINT (Jeddi et al., 2025) adapts pruning strength per layer phase, and ST3 (Zhuang et al., 2024) exploits later layers that reproduce earlier attention. We therefore do not claim layer-wise heterogeneity as a novel observation. What differs is the consequence: these methods profile layers to decide *which tokens to remove*, permanently, whereas FastOCR uses layer heterogeneity to decide *which layers must re-examine the full cache*, so that tokens not selected at one step remain available at the next. The second line, represented by Quest (Tang et al., 2024), retains the cache and restricts attention instead. Quest shares FastOCR's commitment that no entry is ever destroyed, making it the most informative comparison for our claims, so we evaluate it directly rather than only citing it; it differs in what drives the per-step selection, using modality-agnostic page bounds where FastOCR exploits a prior specific to OCR decoding.

Reading behaviour and OCR-specific acceleration. Baek et al. (2025) identify *OCR heads* and show that they are *statically* activated — an observation about *which components* read, an identity fixed over generation, whereas Dynamic Visual Fixation concerns *where* attention lands and how that

location moves between decoding steps. Their finding motivates using a short warmup to identify candidate focal layers; the usefulness of that fixed set for acceleration is an empirical question. Studies of cross-modal information flow (Zhang et al., 2025d; Saporita et al., 2026) show that visual evidence is integrated into prompt-token representations during prefill, a pathway FastOCR leaves unmodified by construction (Appendix G). Closest in aim, RTPrune (Wan et al., 2026) uses decoding-time observations to design high-norm token selection and optimal-transport merging for DeepSeek-OCR. Merging changes the visual representation supplied to the decoder. FastOCR instead preserves each cached entry and changes the accessed subset through per-step re-selection, cross-layer propagation, and cross-step reuse.

3 METHOD

FastOCR uses Dynamic Visual Fixation to reduce the visual cache entries accessed during document decoding. Figure 2 and Algorithm 1 summarize the pipeline. Let L denote the number of transformer layers, indexed by $l \in \{0, \ldots, L-1\}$, and let t index decoding forwards. We write Q_l^t for the current query at layer l and KV_l for its full resident cache.

The framework organizes selection across both depth and time. *Focal-Guided Pruning* (Section 3.2) first profiles each layer's image-attention ratio to choose a fixed set of focal layers $\mathcal{C}$ (Figure 2(a)). At each subsequent decoding step, these layers re-examine the full cache, refresh the selected image-token set $\mathcal{F}_l^t$, and pass it to intervening non-focal layers (Figure 2(b)). Thus the locations that perform selection are fixed after profiling, while the selected tokens change with the decoding query. *Cross-Step Fixation Reuse* (Section 3.3, Figure 2(c)) initializes the selection passed beyond layer 0 from the preceding step's deepest focal set $\mathcal{F}_{\text{last}}$, when available and layer 0 is non-focal. Layer 0 itself also computes full attention; reuse determines the image positions accessed by the following non-focal layers.

3.1 DYNAMIC VISUAL FIXATION

We draw inspiration from human reading (Rayner, 1998), where a reader does not perceive an entire page at once but instead *fixates* on a small area before shifting gaze to the next. We observe an analogous phenomenon in VLMs: although document images appear globally dense and seemingly unprunable, a model's top-attended visual tokens form line-local sets that shift gradually across decoding steps. This Dynamic Visual Fixation phenomenon recasts the intractable global pruning problem as a tractable local, dynamic one, in which we need only adjust which tokens are attended at each step rather than permanently evict them upfront.

We measure this on OmniDocBench across all 10 document types, both without pruning and during FastOCR decoding. At fixed layers $\{27, 31, 33\}$ in unpruned Qwen2.5-VL, consecutive top-5% image-token sets overlap with IoU 0.51, versus 0.14 after shuffling step order, and their centroid moves 3.3 patches versus 9.8. FastOCR with model-selected focal layers shows similar temporal structure on dots.ocr and Qwen2.5-VL: IoU 0.53 and 0.50, or 3.1–$3.4\times$ their respective step-shuffled controls. Across the three measurements, 26–27% of selected tokens fall in three patch rows, versus about 9% for size-matched uniform samples. The set is line-local but retains a thin long-range tail, so its bounding box is nearly page-wide. Appendix H details the two measurement protocols, their controls, and the FastOCR breakdown by document type.

3.2 FOCAL-GUIDED PRUNING

Focal-Guided Pruning (FGP) determines which image tokens each layer attends to at every decoding step. It consists of two components: *Focal Layer Selection* identifies a small set of layers with high image-attention ratios, and *Cross-Layer Propagation* uses these focal layers to select the top-attended image tokens and propagates this selection to subsequent non-focal layers.

Image-attention ratios vary substantially across layers. Figure 3 presents this profile at two scales. Panel (a) summarizes the allocation of attention to image tokens across the measured samples: focal layers allocate 42.2% of their attention mass to image tokens, versus 14.3% for non-focal layers. This contrast describes the layers chosen by the selection statistic; a lower ratio does not imply that a layer

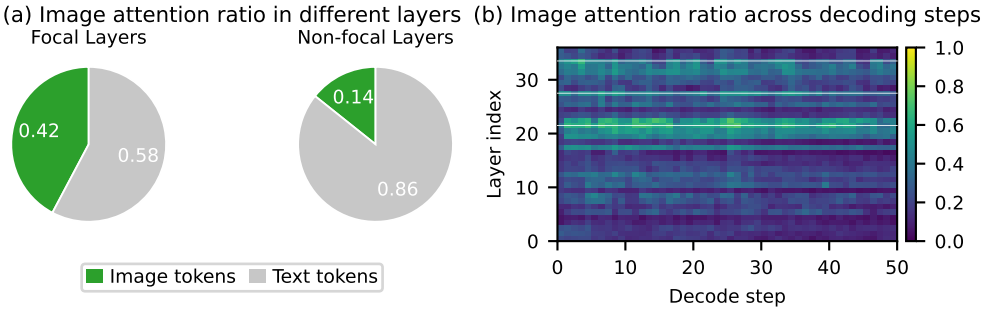

Figure 3: Image attention distribution across layers. (a) Mean image attention ratio for focal vs. non-focal layers, averaged over all OmniDocBench samples. (b) Per-layer image attention heatmap from a single sample over 50 decoding steps; white lines indicate focal layers.

is unimportant to the final transcription. In particular, non-focal layers continue to process all text positions and the selected visual context.

Panel (b) follows one page over 50 decoding steps. The white markers identify focal layers, while the horizontal bands show how the image-attention profile evolves as text is generated. The visible persistence over this window motivates identifying candidate focal layers from a short prefix of decoding. We then test that fixed-layer design through the quality experiments, rather than assuming that a 50-step visualization establishes stability for every long output. Profiling is repeated for each generation batch, so the selected layer set can adapt to a different batch even though it remains fixed within a generation.

The focal layers in our OCR measurements tend to lie in the middle and late stages (Figure 6), unlike the shallow visual attention reported by FastV (Chen et al., 2024) for general image understanding. Layer heterogeneity has also motivated prior pruning methods (Zhang et al., 2025b; Jeddi et al., 2025; Zhuang et al., 2024). Here it motivates periodic full-cache re-examination across depth: selected layers refresh the visual context using their current queries, and the intervening layers reuse those selections. The static activation of OCR heads reported by Baek et al. (2025) offers a related motivation for testing a fixed layer set after short profiling. Neither observation requires the selected image tokens themselves to stay fixed over time.

3.2.1 Focal Layer Selection

During decoding warmup, FastOCR uses full-cache attention to record each layer's image-attention ratio:

$$r_i^{(t)} = \frac{\sum_{j \in \mathcal{I}} \bar{a}_{ij}}{\sum_{j \in \mathcal{I} \cup \mathcal{T}_t} \bar{a}_{ij}}, \tag{1}$$

where $\mathcal{I}$ contains image positions and $\mathcal{T}_t$ all prompt and generated text positions currently in the cache. Profiling averages attention over heads and batch elements. When the configured counter reaches w, layers are ranked by their mean ratio over the preceding $u = w - 1$ forwards, $\hat{r}_i = u^{-1} \sum_{t'=1}^{u} r_i^{(t')}$. This separates a layer's average tendency to attend to the image from fluctuations at an individual decoding step. The implementation targets $\max(1, \lfloor \rho L \rfloor)$ focal layers for $\rho > 0$, subject to the gap constraint below. The selected set is fixed thereafter, so subsequent steps refresh token selections without repeating the layer ranking. Appendix A specifies initialization and the separate prefill-profiling mode.

We enforce a focal gap of g intervening layers by greedily accepting ranked candidates only when $|i - j| > g$ for every already selected layer j. For example, with $g = 1$, selecting layer l_i excludes its immediate neighbors, while l_{i+2} remains eligible. This spaces the points at which the propagated token set is refreshed and can yield fewer layers than the target. It does not aggregate their selections: the implementation always propagates the most recently selected set, not a union of sets across layers.

3.2.2 CROSS-LAYER PROPAGATION

After profiling, layer 0 and the focal layers attend over their full KV caches. All other layers attend over a gathered subset $\mathcal{K}$, containing every current text position and the selected image positions. The subset is represented by token indices: each receiving layer gathers the corresponding entries from its own cache. What is shared across layers is therefore the selection of positions, not the K/V values or attention outputs. Gathering creates a compact dense tensor without modifying the resident cache, allowing the attention kernel to operate on the smaller input while every unselected position remains available for later re-selection.

At a focal layer, head-averaged attention selects the top k image positions and updates $\mathcal{K} \leftarrow \mathcal{T}_t \cup \mathcal{F}_{l_i}^t$. The implementation uses $k = \max(1, \mathrm{round}(\tau N))$ for $\tau > 0$ and $k = 0$ for the zero-retention ablation. Thus

$$\mathbf{o}_i = \begin{cases} \mathrm{Attn}(Q_{l_i}^t, K_{[\mathcal{I}\cup\mathcal{T}_t]}, V_{[\mathcal{I}\cup\mathcal{T}_t]}) & i = 0 \text{ or } l_i \in \mathcal{C}, \\ \mathrm{Attn}(Q_{l_i}^t, K_{[\mathcal{K}]}, V_{[\mathcal{K}]}) & i > 0 \text{ and } l_i \notin \mathcal{C}. \end{cases} \tag{2}$$

Here $K_{[\mathcal{K}]}$ and $V_{[\mathcal{K}]}$ restrict that layer's keys and values to the indicated positions. A focal layer replaces the propagated image set after computing its output; intervening layers reuse that set without re-selecting tokens. For two successive focal layers l_i and l_j, the image positions chosen at l_i guide layers $l_{i+1}, \ldots, l_{j-1}$. Layer l_j then reads its complete cache and may select a different set, including positions omitted by l_i. Selection can thus change across depth as well as across decoding steps, without an irreversible eviction decision.

3.3 CROSS-STEP FIXATION REUSE

Cross-layer propagation leaves an initialization question before the first focal layer: there is no preceding focal selection from the current step to reuse. The previous step provides a candidate. As the model advances through a text line, neighboring output tokens often draw on overlapping visual evidence, motivating a warm start from the most recent deep-layer selection.

At the end of each sparse step, FastOCR saves the image-token set from the deepest focal layer as $\mathcal{F}_{\mathrm{last}}$. After layer 0 computes full attention in the next step, it uses this set to initialize the selection passed to subsequent non-focal layers, provided layer 0 is non-focal and $\mathcal{F}_{\mathrm{last}}$ is available. Otherwise, it initializes the set from its own image-attention top-k. All current text positions are added in either case, including tokens generated since the image set was saved.

CSFR therefore changes the selection propagated beyond layer 0, while leaving layer 0's own full attention intact. The observed temporal continuity motivates this initialization: in the FastOCR traces, roughly two thirds of focal tokens remain selected at the next step and the centroid moves 3.6–3.9 patches (Table 11). Reuse applies only until the first focal layer refreshes the selection; it does not constrain that layer to the previous set. The model can therefore re-examine all visual positions when the reading trajectory moves to another region. Algorithm 1 gives the profiling and sparse phases as mutually exclusive branches.

4 EXPERIMENTS

4.1 EXPERIMENT SETTINGS

Benchmarks. We evaluate our method on OmniDocBench (Ouyang et al., 2025) and olmOCR-Bench (Poznanski et al., 2025), two widely-adopted OCR benchmarks. They cover diverse document types and multiple aspects of recognition, including text, formulas, tables, and reading order, yielding a comprehensive view of OCR model capabilities.

Baselines. We compare against four recent training-free baselines chosen to span the design space rather than to favor our setting: SnapKV (Li et al., 2024) (one-shot KV eviction), DivPrune (Alvar et al., 2025) and CDPruner (Zhang et al., 2025a) (pre-LLM visual token selection by diversity and conditional-DPP objectives), and Quest (Tang et al., 2024) (query-aware page-sparse attention that, like FastOCR, evicts nothing — the fairest contrast to our design). Each baseline's parameter is solved to match FastOCR's nominal attention cost at a fixed reference length; different selection rules

Table 1: Performance comparison on OmniDocBench. Best in bold. Within each block all pruning methods are aligned to the same nominal attention budget at a fixed reference length (Appendix C), shown at two tiers per model, and the FLOPs column reports that budget (Appendix B). Display Formula is scored with CDM, Overall is $\big((1 - \text{Text Edit}) + \text{CDM} + \text{TEDS}/100\big)/3 \times 100$, and Rel. (%) is Overall relative to the corresponding Vanilla model.

Method	FLOPs (G) ↓	Text Block Edit Dist ↓	Formula CDM ↑	Table TEDS ↑	Reading Order Edit Dist ↓	Overall ↑	Rel. (%) ↑
Qwen2.5-VL							
Vanilla	15.10	0.160	0.590	58.95	0.248	67.31	100.0
SnapKV	8.71	0.812	0.138	3.87	0.650	12.19	18.1
DivPrune	8.71	0.327	0.329	27.93	0.352	42.71	63.5
CDPruner	8.71	0.751	0.122	6.46	0.611	14.50	21.5
Quest	8.71	0.408	0.140	11.20	0.457	28.14	41.8
FastOCR (τ=0.25)	8.71	**0.159**	**0.578**	**59.33**	**0.247**	**67.09**	**99.7**
SnapKV	7.00	0.865	0.096	2.04	0.690	8.40	12.5
DivPrune	7.00	0.559	0.174	13.01	0.508	24.82	36.9
CDPruner	7.00	0.888	0.035	1.54	0.720	5.41	8.0
Quest	7.00	0.678	0.055	5.69	0.630	14.46	21.5
FastOCR (τ=0.05)	7.00	**0.172**	**0.590**	**56.86**	**0.255**	**66.20**	**98.3**
dots.ocr							
Vanilla	8.10	0.067	0.828	80.69	0.166	85.59	100.0
SnapKV	4.36	0.739	0.300	20.17	0.503	25.39	29.7
DivPrune	4.36	0.205	0.665	60.40	0.255	68.79	80.4
CDPruner	4.36	0.280	0.597	52.02	0.360	61.23	71.5
Quest	4.36	0.136	0.623	61.40	0.261	70.05	81.8
FastOCR (τ=0.25)	4.36	**0.075**	**0.792**	**78.75**	**0.173**	**83.48**	**97.5**
SnapKV	3.36	0.835	0.201	7.08	0.571	14.56	17.0
DivPrune	3.36	0.529	0.381	28.05	0.435	37.72	44.1
CDPruner	3.36	0.522	0.392	26.19	0.459	37.71	44.1
Quest	3.36	0.364	0.434	37.02	0.376	48.02	56.1
FastOCR (τ=0.05)	3.36	**0.084**	**0.785**	**77.53**	**0.174**	**82.55**	**96.4**

imply different retention ratios. Appendix C gives the alignment equations and what every method actually retains, and reports the exact knob each baseline was run at.

Implementation Details. We primarily evaluate on two models: Qwen2.5-VL (3B) (Bai et al., 2025) and dots.ocr (1.7B) (Li et al., 2025). We also evaluate olmOCR (7B) (Poznanski et al., 2025) and Nanonets-OCR-s (3B) (Nanonets, 2025) to test generalization across OCR models of varying scale and fine-tuning. FastOCR has four hyperparameters: the focal layer ratio ρ, which controls the fraction of layers designated as focal; the focal gap g, which enforces a minimum separation of g layers between any two consecutive focal layers; the kept token ratio τ, which determines the proportion of image tokens retained at each focal layer; and the configured warmup threshold w for profiling and selecting focal layers. Unless otherwise specified, we set $\rho = 0.1$, $g = 1$, $\tau = 0.05$, and $w = 10$. All experiments are conducted on NVIDIA H200 GPUs, with a single full-benchmark evaluation taking roughly four GPU-hours per model–configuration pair.

4.2 Main Results

Results on OmniDocBench. Table 1 reports OmniDocBench under matched nominal reference budgets. FastOCR retains 98.3% of the unpruned accuracy at the low budget on Qwen2.5-VL and 99.7% at the high budget, and 96.4%/97.5% on dots.ocr, while no baseline matches it at either budget on either model. Which baseline comes second is model-dependent — DivPrune on Qwen2.5-VL, the non-evicting Quest on dots.ocr — but the margin to FastOCR widens as the budget tightens in every case. Eviction-based methods also lose table accuracy as the budget tightens: dots.ocr's DivPrune TEDS falls from 60.40 to 28.05, and CDPruner from 52.02 to 26.19. Quest is the most informative comparison, since it likewise destroys nothing: it stays competitive where the underlying model is strong, yet degrades far faster than FastOCR as the budget tightens, preserving local fluency while

Table 2: Efficiency of FastOCR across operating points with $\rho = 0.1$, $\tau = 0.05$, $g = 1$, and configured $w = 10$. Values are ms per decoder forward, averaged over 127 forwards in each of three 128-token trials on one H200. Each cell reports Vanilla → FastOCR with speedup in parentheses. Image tokens are resize targets; FLOPs are nominal reference costs (Appendix B). Exact inputs and measurement variability appear in Appendix D.

Model	Batch	Target Image Tokens	Nominal FLOPs (G)	Attention Latency (ms)	Decoder Latency (ms)
Qwen2.5-VL (3B)	8	4096	15.10 → 7.00	13.95 → 8.40 (1.66×)	37.06 → 32.85 (1.13×)
	12	4096	22.65 → 10.51	20.22 → 9.68 (2.09×)	42.75 → 35.41 (1.21×)
	8	8192	24.76 → 8.57	24.70 → 10.56 (2.34×)	47.03 → 35.22 (1.34×)
olmOCR (7B)	8	4096	26.31 → 15.24	19.74 → 8.22 (2.40×)	38.69 → 28.66 (1.35×)
	12	4096	39.46 → 22.86	29.14 → 9.71 (3.00×)	47.91 → 30.26 (1.58×)
	8	8192	39.46 → 17.33	36.29 → 10.80 (3.36×)	54.96 → 31.80 (1.73×)

Table 3: Generalization across models on OmniDocBench, same scorer and Overall definition as Table 1. FastOCR runs at its default ρ=0.1, τ=0.05, except on olmOCR where a small grid selected ρ=0.2, τ=0.1.

Model	Method	Text Block Edit Dist ↓	Formula CDM ↑	Table TEDS ↑	Reading Order Edit Dist ↓	Overall ↑	Rel. (%) ↑
Qwen2.5-VL (3B)	Vanilla	0.160	0.590	58.95	0.248	67.31	100.0
	FastOCR	0.172	0.590	56.86	0.255	66.20	98.3
dots.ocr (1.7B)	Vanilla	0.067	0.828	80.69	0.166	85.59	100.0
	FastOCR	0.084	0.785	77.53	0.174	82.55	96.4
olmOCR (7B)	Vanilla	0.295	0.406	60.23	0.330	57.13	100.0
	FastOCR	0.315	0.389	58.02	0.345	55.14	96.5
Nanonets-OCR-s (3B)	Vanilla	0.106	0.742	77.71	0.204	80.41	100.0
	FastOCR	0.113	0.748	78.74	0.208	80.75	100.4

losing layout and table structure. Selecting by a generic per-step top-k is therefore not sufficient; what the budget buys depends on whether the selection follows the fixation structure of OCR decoding (Section 3.1). The same trends hold on olmOCR-Bench (Appendix E).

Generalization Across Models. To verify that FastOCR is not tailored to a single model, we evaluate it on four OCR-capable VLMs spanning 1.7B–7B parameters and different fine-tuning recipes (Table 3). Across these four checkpoints, FastOCR retains **96.4%–100.4%** of vanilla performance under the configurations in the caption, supporting transfer across the evaluated scales and fine-tuning recipes. **Efficiency.** Table 2 reports latency across operating points on one H200. The measured gain varies with batch size, input length, and model size, reaching $3.00\times$ on attention and $1.58\times$ per decoder forward on olmOCR (7B) at batch 12 with 4096 image tokens ($3.36\times$ and $1.73\times$ at 8192 tokens), versus $2.09\times$/$1.21\times$ on Qwen2.5-VL (3B) at the same point. The determining quantity is the share of the decoding step spent in attention: the KV caches of concurrent sequences compete for memory bandwidth in the memory-bound decoding regime, so pruning gains leverage precisely where that share is large, and the larger model benefits more. The residual gap between attention speedup and decoder-forward speedup is the FFN and projection work that KV-cache pruning cannot affect.

4.3 Ablation Study

Effect of FGP and CSFR. Table 4 ablates the two key components of FastOCR on Qwen2.5-VL using OmniDocBench. Both ablated variants disable reuse and force layer 0 to re-select image tokens at every sparse step. The variant without both modules sets $\rho = 0.01$, leaving layer 0 as its sole focal layer; the variant without CSFR uses $\rho = 0.1$, reserving one of its three focal slots for layer 0 and selecting the other two by image attention. Their nominal reference costs are 6.25 and 6.75 G, versus 7.00 G for the default method (Appendix C); this component ablation is not a matched-budget comparison. Without both FGP and CSFR, decode-stage pruning scores only 11.45. Adding FGP raises the score to 44.56. This tests the joint effect of focal-layer access, token re-selection, and cross-layer propagation, rather than isolating the layer-ranking criterion alone. The default configuration reaches 66.20 — 98.3% of the unpruned model. This last comparison changes

Table 4: Ablation study on different components of the proposed method (Qwen2.5-VL, OmniDocBench). Best in bold. Scored with the same unified CDM scorer and Overall definition as Table 1.

Method	FGP	CSFR	Text Block Edit Dist ↓	Display Formula CDM ↑	Table TEDS ↑	Reading Order Edit Dist ↓	Overall ↑
Vanilla	–	–	0.160	0.590	58.95	0.248	67.31
FastV	–	–	0.915	0.004	0.92	0.744	3.27
FastOCR w/o both	✗	✗	0.815	0.104	5.48	0.643	11.45
FastOCR w/o CSFR	✓	✗	0.438	0.456	31.89	0.392	44.56
FastOCR	✓	✓	**0.172**	**0.590**	**56.86**	**0.255**	**66.20**

both reuse and the placement of full-attention layers, so it does not isolate CSFR at a fixed access budget. Overall, the full method recovers 98.3% of the unpruned model's accuracy (66.20 vs. 67.31).

Hyperparameter sensitivity. The sweeps in Appendix F show modest variation across focal gaps and the measured positive focal-layer ratios, with a separate low-accuracy boundary at $\rho = 0$. At $\tau = 0$, pruned layers lose direct image-token access, although layer 0 and focal layers remain dense. Prefill-only profiling also lowers accuracy relative to decode profiling. The default configuration balances accuracy and access cost rather than maximizing each panel's score.

Qualitative illustration. Figure 4 illustrates reuse and re-selection on a textbook page. Selected positions overlap across decoding steps while shifting along the text. This example complements the quantitative analysis in Appendix H.

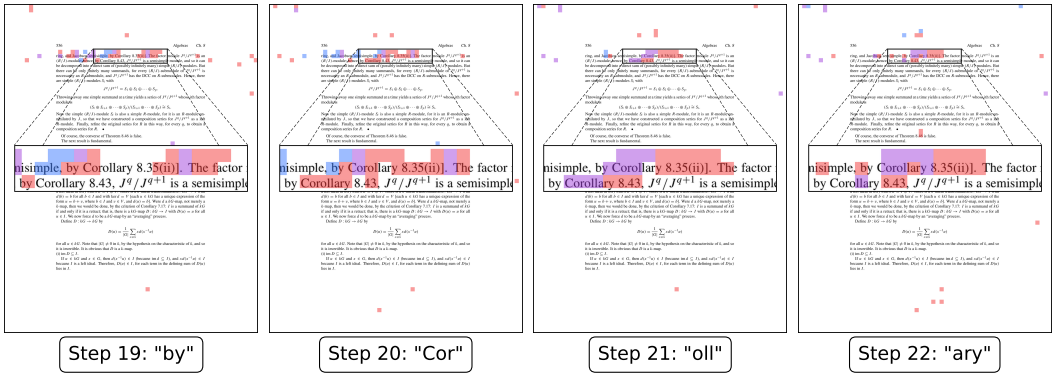

Figure 4: Visualization of dynamic visual fixation across four consecutive decoding steps ($\rho = 0.1$, $\tau = 0.004$). Columns are decoding steps; rows show the full page, a zoomed region, and the generated token. Blue: tokens inherited from the previous step. Red: newly selected tokens. Purple: overlap of the two sets.

5 Conclusion

FastOCR combines Focal-Guided Pruning and Cross-Step Fixation Reuse while keeping every visual cache entry recoverable. Temporally coherent, line-local selected sets occur in FastOCR and at fixed layers of unpruned Qwen2.5-VL. Under nominal reference budgets, FastOCR outperforms four baselines on Qwen2.5-VL and dots.ocr. Qwen2.5-VL retains 98.3% of Vanilla accuracy on OmniDocBench and 97.3% on olmOCR-Bench at the low budget, with 5% image-token access at pruned layers. Latency measurements reach 3.36× attention and 1.73× decoder-forward speedup on olmOCR; these timings exclude the full OCR pipeline. Adaptive focal-layer selection and reduced resident KV storage remain directions for future work.

AI USE STATEMENT

Generative AI tools — LLM-based coding and research assistants — were used extensively in the preparation of this work. We describe below what they were used for and, equally important, how their output was checked.

Tasks with required disclosure. We used generative AI to *implement methods*: the integration of the four comparison baselines (SnapKV, DivPrune, CDPruner, and our Quest variant) into our codebase, the model-specific adapters, the multi-GPU experiment orchestration, and the evaluation and analysis tooling were written with substantial AI assistance. We used it to *design and give feedback on experimental methodology*, including the two matched-budget tiers, the ablation grids, and the calibration runs that verify a benchmark harness reproduces previously reported numbers before any new row is added to a table. We used it to *propose and refine hypotheses* about observed behaviour and to *interpret results*, including the failure-mode and root-cause analyses reported in the appendices. We used it to *clean and reformat data*: rendering the olmOCR-Bench documents to page images, and converting model outputs into the layouts the official scorers expect. AI assistance was also used for the per-step attention FLOPs accounting of Appendix B and for solving each baseline's knob in the budget alignment of Appendix C; these are elementary cost computations rather than formal results, and we verified them against measured token counts.

We did *not* use generative AI to generate synthetic datasets; this work introduces no synthetic data. This work contains no theorems or formal proofs, so assistance with proving mathematical claims is not applicable. We did not use AI for translation or for qualitative or thematic analysis of human-subject data, neither of which arises here.

Tasks with recommended disclosure. We additionally used generative AI to draft and revise parts of this paper, including prose in the experimental sections, table captions, and appendices; to search for, retrieve and summarise related work, which informed the positioning of our contribution against prior work on layer-wise pruning and query-aware sparsity; to edit the manuscript for readability; to suggest hyperparameter settings to sweep; and to format references.

Verification. AI-assisted implementation and analysis were checked in ways designed to catch the specific failure modes such assistance produces. Every pruning baseline had to pass an effectiveness gate before a full run: on a small sample we asserted that the strategy measurably changed the model's behaviour relative to the unpruned run, because a silently inactive hook would otherwise yield unpruned numbers reported under a baseline's name. Where we patched a third-party evaluation harness to fix a crash, we required the patch to be a no-op on inputs that did not crash, and confirmed this by re-scoring an unaffected configuration and checking that every reported digit was unchanged. Adapted model code was checked against reference implementations for output parity. Numbers reported in the tables were traced back to the run artifacts that produced them, and claims an assistant made about our own results were re-derived from those artifacts; this process caught and corrected instances in which an assistant mis-attributed a root cause or carried a stale number forward.

Responsibility. We have reviewed the AI-assisted work reported in this paper. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. DivPrune: Diversity-based visual token pruning for large multimodal models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Ingeol Baek, Hwan Chang, Sunghyun Ryu, and Hwanhee Lee. How do large vision-language models see text in image? unveiling the distinctive role of OCR heads. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. In *International Conference on Learning Representations (ICLR)*, 2024.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

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

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

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 and KV cache compression. *arXiv preprint arXiv:2410.08584*, 2024.

Ahmadreza Jeddi, Negin Baghbanzadeh, Elham Dolatabadi, and Babak Taati. Similarity-aware token pruning: Your VLM but faster. *arXiv preprint arXiv:2503.11549*, 2025.

Geewook Kim, Teakgyu Hong, Moonbin Yim, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Donut: Document understanding transformer without OCR. *arXiv preprint arXiv:2111.15664*, 2021.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yumeng Li, Guang Yang, Hao Liu, Bowen Wang, and Colin Zhang. dots.ocr: Multilingual document layout parsing in a single vision-language model, 2025. URL `https://arxiv.org/abs/2512.02498`.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Nanonets. Nanonets-ocr-s: A document ocr model. `https://huggingface.co/nanonets/Nanonets-OCR-s`, 2025. Qwen2.5-VL-3B fine-tuned for document parsing.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Jake Poznanski, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Aman Rangapur, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. olmocr: Unlocking trillions of tokens in pdfs with vision language models, 2025. URL `https://arxiv.org/abs/2502.18443`.

Keith Rayner. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3):372–422, 1998.

Alessia Saporita, Vittorio Pipoli, Federico Bolelli, Lorenzo Baraldi, Andrea Acquaviva, and Elisa Ficarra. FG-TRACER: Tracing information flow in multimodal large language models in free-form generation. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2026.

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.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference. In *International Conference on Machine Learning (ICML)*, 2024.

Dezhan Tu, Danylo Vashchilenko, Panpan Xu, and Yuzhe Lu. VL-Cache: Sparsity and modality-aware KV cache compression for vision-language model inference acceleration. *arXiv preprint arXiv:2410.23317*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Ben Wan, Yan Feng, Zihan Tang, Weizhe Huang, Yuting Zeng, Jia Wang, and Tongxuan Liu. Rtprune: Reading-twice inspired token pruning for efficient deepseek-ocr inference. *arXiv preprint arXiv:2605.00392*, 2026.

Haoran Wei, Yaofeng Sun, Yukun Li, et al. DeepSeek-OCR: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Qizhe Zhang, Mengzhen Liu, Lichen Li, Ming Lu, Yuan Zhang, Junwen Pan, Qi She, and Shanghang Zhang. Beyond attention or similarity: Maximizing conditional diversity for token pruning in MLLMs. *arXiv preprint arXiv:2506.10967*, 2025a.

Weichen Zhang, Zhui Zhu, Ningbo Li, Shilong Tao, Kebin Liu, and Yunhao Liu. AdaptInfer: Adaptive token pruning for vision-language model inference with dynamical text guidance. *arXiv preprint arXiv:2508.06084*, 2025b.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A. Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *International Conference on Machine Learning (ICML)*, 2025c.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H2O: heavy-hitter oracle for efficient generative inference of large language models. In *NeurIPS*, 2023.

Zhi Zhang, Srishti Yadav, Fengze Han, and Ekaterina Shutova. Cross-modal information flow in multimodal large language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025d.

Jiedong Zhuang, Lu Lu, Ming Dai, Rui Hu, Jian Chen, Qiang Liu, and Haoji Hu. ST3: Accelerating multimodal large language model by spatial-temporal visual token trimming. *arXiv preprint arXiv:2412.20105*, 2024.

A ALGORITHM

Algorithm 1 describes the implemented decoding procedure. Each layer appends the current token's K/V to its full resident cache before attention; only the positions accessed by attention change. All other layer operations are unchanged. The non-image set $\mathcal{T}_t$ includes the prompt and every generated token present in the current forward's cache. The focal layer set $\mathcal{C}$ and profiling counter are shared within a batch, while the selected image sets and $\mathcal{F}_{\text{last}}$ are maintained separately for each example.

Profiling and initialization. A new generation starts with $t = 0$, empty per-layer ratio histories R_i, $\mathcal{C} = \emptyset$, and $\mathcal{F}_{\text{last}} = \perp$ (unavailable). In decode-profiling mode, prefill runs full attention without collecting these histories. The implementation increments t before each decoding forward and selects layers when $t = w$, using the preceding $w - 1$ forwards; thus $w = 10$ means nine full-attention profiling forwards and selection before the tenth. This path requires $w \geq 2$ so that the histories are nonempty. In the separate *prefill-profiling mode*, full attention records each layer's final prompt-query ratio, and $\mathcal{C} \leftarrow \textsc{SelectFocal}(R)$ is computed only after all prefill layers finish. The first decode forward then starts with $\mathcal{C} \neq \emptyset$ but $\mathcal{F}_{\text{last}} = \perp$. This mode is explicitly selected; setting the decode threshold to zero does not enable it automatically.

For profiling, $\bar{a}_{ij}$ is the last query's attention averaged over heads and batch elements, and the recorded ratio is $r_i = \sum_{j \in \mathcal{I}} \bar{a}_{ij} / \sum_{j \in \mathcal{I} \cup \mathcal{T}_t} \bar{a}_{ij}$ (zero if the denominator is zero). Layers are ranked by their mean recorded ratio, with ties resolved by layer index. Selection targets $\max(1, \lfloor \rho L \rfloor)$ layers for $0 < \rho \leq 1$; the gap constraint $|i - j| > g$ can yield fewer. Its condition is vacuously satisfied for an empty set, so at least one layer is selected.

Selection and reuse. After profiling, layer 0 and every focal layer compute full attention. Layer 0 then initializes the set passed to subsequent non-focal layers: it reuses $\mathcal{F}_{\text{last}}$ when available and non-focal, otherwise it selects image tokens from its just-computed attention (lines 16–18). Thus CSFR changes the propagated set, not layer 0's own attention. Focal layers refresh that set, always adding all current non-image positions. Top-k uses head-averaged scores separately for each example and is restricted to $\mathcal{I}$; the implementation uses nearest-integer rounding, with ties to even, and keeps at least one image token when $\tau > 0$. The deepest focal layer saves its image set for the next step.

B COMPUTATIONAL COST ESTIMATION

We report a nominal cost for the matrix multiplications in one decoding step's self-attention sublayers. A multiply–add counts as two FLOPs. Let h be the hidden size, H_q and H_{kv} the query and KV head counts, and $d = h/H_q$ the head dimension. All models for which we report FLOPs use grouped-query attention (GQA), with $h_{kv} = H_{kv} d$.

Projection cost. For a new token $x \in \mathbb{R}^{1 \times h}$, $W_Q, W_O \in \mathbb{R}^{h \times h}$ and $W_K, W_V \in \mathbb{R}^{h \times h_{kv}}$. Thus Q/O cost $4h^2$ and K/V cost $4hh_{kv}$ per layer and batch element, for a total of $4h^2(1 + H_{kv}/H_q)$. The MHA expression $8h^2$ is recovered only when $H_{kv} = H_q$.

Attention cost. Each of the H_q query heads computes $\mathrm{softmax}(QK^\top/\sqrt{d})V$ against its associated KV head. For s_l attended positions, QK and AV each cost $2H_q d s_l = 2hs_l$. Summing projections and attention over batch size b and L layers gives

$$C_{\text{nom}} = b \sum_{l=0}^{L-1} \left[4h^2 \left(1 + \frac{H_{kv}}{H_q} \right) + 4hs_l \right]. \tag{3}$$

Softmax, positional encoding, additional selection scores, top-k, gather, normalization, and FFN costs are outside this nominal model. Their runtime effects are measured separately in the timing experiments.

Reference lengths. Tables 1 and 8 use $b = 8$ and a fixed reference sequence length $S = 4096$. The original alignment script sets $N_{\text{ref}} = \mathrm{round}(0.992S)$ image tokens and $T_{\text{ref}} = S - N_{\text{ref}}$ text tokens: 4063 and 33, respectively. It uses continuous retained lengths, ignoring integer-token and page rounding. Table 2 uses the same reference model at its stated batch size and at S equal to the image-resize target (4096 or 8192). These FLOPs are reference costs, not counts of the actual input or the growing cache in each timed forward. The measured input lengths and repetition protocol are given in Appendix D.

Algorithm 1 FastOCR decoding after full-cache prefill

Require: Layers $l_0, \ldots, l_{L-1}$; image positions $\mathcal{I}$, $N = |\mathcal{I}| \geq 1$; $0 < \rho \leq 1$, $0 \leq \tau \leq 1$, integer $g \geq 0$; $w \geq 2$ in decode-profiling mode
Persistent state: full KV caches, t, R_i, $\mathcal{C}$, $\mathcal{F}_{\text{last}}$; initialized as above

$t \leftarrow t + 1$; update $\mathcal{T}_t$ to include the current token's non-image position
if $\mathcal{C} = \emptyset$ **and** $t = w$ **then**
  $\mathcal{C} \leftarrow$ SELECTFOCAL(R) ▷ before this forward's layer loop
end if
if $\mathcal{C} = \emptyset$ **then** ▷ profiling forward
  for $i = 0, \ldots, L-1$ **do**
    Process l_i with full-cache attention; append its image-attention ratio to R_i
  end for
  return decoder output ▷ no sparse phase in this forward
end if
$k \leftarrow 0$ if $\tau = 0$, otherwise $k \leftarrow \max(1, \text{round}(\tau N))$
for $i = 0, \ldots, L-1$ **do**
  if $i = 0$ **or** $i \in \mathcal{C}$ **then**
    Process l_i with full-cache attention; obtain head-averaged scores a_i per example
    if $i = 0$ **and** $0 \notin \mathcal{C}$ **and** $\mathcal{F}_{\text{last}} \neq \perp$ **then**
      $\mathcal{F}^t_{l_i} \leftarrow \mathcal{F}_{\text{last}}$
    else
      $\mathcal{F}^t_{l_i} \leftarrow \text{TopK}_k(a_i|_{\mathcal{I}})$ ▷ also initializes the first sparse step
    end if
    $\mathcal{K} \leftarrow \mathcal{T}_t \cup \mathcal{F}^t_{l_i}$
    if $i = \max \mathcal{C}$ **then**
      $\mathcal{F}_{\text{last}} \leftarrow \mathcal{F}^t_{l_i}$
    end if
  else
    Process l_i with attention over gathered KV positions $\mathcal{K}$
  end if
end for
return decoder output

function SELECTFOCAL(R) ▷ requires a nonempty history at every layer
  Rank layer indices by $\text{mean}(R_i)$ in descending order
  $\mathcal{C}' \leftarrow \emptyset$; $m \leftarrow \max(1, \lfloor \rho L \rfloor)$
  for each index i in ranked order **do**
    if $|\mathcal{C}'| < m$ **and** $|i - j| > g$ for every $j \in \mathcal{C}'$ **then**
      $\mathcal{C}' \leftarrow \mathcal{C}' \cup \{i\}$
    end if
  end for
  return $\mathcal{C}'$
end function

Table 5: Decoder dimensions used in Equation 3.

Model	h	L	H_q	H_{kv}
Qwen2.5-VL (3B)	2048	36	16	2
dots.ocr (1.7B)	1536	28	12	2
olmOCR (7B)	3584	28	28	4

C NOMINAL BUDGET ALIGNMENT AND PER-METHOD RETENTION RATES

We fix FastOCR's parameters and match each baseline to its nominal attention budget at one reference length. SnapKV exposes a prefix-retention ratio, DivPrune and CDPruner a visual-token ratio, and Quest a page-access ratio. The comparison uses the common cost model of Appendix B; it does not assert equal realized FLOPs or wall-clock time over a full generation.

Table 6: Nominal reference-budget alignment at $b = 8$, $S = 4096$. Qwen2.5-VL: $\bar{s} = 1387.3/665.0$, $8.71/7.00$ G at high/low budgets; dots.ocr: $\bar{s} = 1375.2/649.7$, $4.36/3.36$ G. Rates describe the method-specific selection domain in the equation: visual tokens for DivPrune/CDPruner, prefix tokens for SnapKV, and current-cache pages for Quest. † Quest retains the full cache; its rate specifies access, not eviction.

			Qwen2.5-VL		dots.ocr	
Method	**Class**	**Alignment equation**	high	low	high	low
SnapKV	KV eviction	$r = (\bar{s} - W)/(S - W)$, W=64	0.3282	0.1491	0.3252	0.1453
DivPrune	pre-LLM eviction	$r = (\bar{s} - T_{\text{ref}})/N_{\text{ref}}$	0.3333	0.1556	0.3304	0.1518
CDPruner	pre-LLM eviction	$r = (\bar{s} - T_{\text{ref}})/N_{\text{ref}}$	0.3333	0.1556	0.3304	0.1518
Quest†	no eviction	$r = (\bar{s}L - 2S)/((L-2)S)$	0.2998	0.1131	0.2847	0.0939
FastOCR	no eviction	$\rho = 0.1$, $\tau \in \{0.25, 0.05\}$ (fixed)	$\bar{s}/S = 0.339/0.162$		$\bar{s}/S = 0.336/0.159$	

C.1 Reference Budget

For a given model, the GQA projection term in Equation 3 is shared by all methods. The nominal match is therefore determined by the mean attended length $\bar{s} = L^{-1} \sum_{l=0}^{L-1} s_l$. Correcting the projection cost for GQA changes the reported FLOPs but leaves the original alignment equations and the evaluated baseline parameters unchanged.

The reference input uses $S = 4096$, $N_{\text{ref}} = 4063$ image tokens and $T_{\text{ref}} = 33$ text tokens. The 99.2% image fraction was motivated by a 40-page prompt sample (median 3672 image tokens out of 3702), and describes the initial prompt rather than the entire decoding trajectory. For FastOCR, the reference full-attention layer count is $m = \lfloor \rho L \rfloor + 1$, representing the focal slots plus layer 0; the remaining layers use $T_{\text{ref}} + \tau N_{\text{ref}}$ positions:

$$\bar{s}_{\text{FastOCR}} = \frac{mS + (L - m)(T_{\text{ref}} + \tau N_{\text{ref}})}{L}. \tag{4}$$

Layer 0 actually computes full-cache attention in the implementation. CSFR chooses the image set passed to subsequent non-focal layers after layer 0's output has been computed. The actual full-layer count is $|\{0\} \cup \mathcal{C}|$; the reference count is conservative if layer 0 is itself selected as focal.

At $\rho = 0.1$, Qwen2.5-VL has three focal slots plus layer 0 and 32 pruned layers. Its high/low budgets ($\tau = 0.25/0.05$) give $\bar{s} = 1387.3/665.0$, or $8.71/7.00$ G at $b = 8$, versus 15.10 G unpruned. For dots.ocr, two focal slots plus layer 0 give $\bar{s} = 1375.2/649.7$, or $4.36/3.36$ G, versus 8.10 G unpruned. Table 6 preserves the ratios used in the existing runs.

C.2 Scope of the Match

The equations ignore token/page rounding and selection bookkeeping. SnapKV's W denotes its observation-window length, distinct from FastOCR's warmup parameter w. Quest's first two layers are dense, hence the corresponding term in its equation. Its page scoring, like FastOCR's extra score computation, selection, and gather, is outside the nominal matrix-multiplication budget.

After the prompt, FastOCR, DivPrune, CDPruner, and SnapKV retain all newly generated text tokens. Quest selects a fraction of the current cache pages and only forces its local pages to remain accessible. Their accessed lengths therefore grow differently: matching at S does not match every subsequent step. An evaluation under equal realized whole-generation cost would require a different alignment protocol.

Warmup is also excluded from the reference steady-state budget. For D measured forwards, of which u are dense profiling forwards, a fixed-length approximation is

$$\overline{C} = C_{\text{sparse}} + \frac{\min(u, D)}{D}(C_{\text{dense}} - C_{\text{sparse}}). \tag{5}$$

This overhead depends on output length and need not be below 1%. For the current decode-profiling implementation, the configured threshold $w = 10$ selects layers before forward 10, after nine dense forwards. Changing batch size alone scales the nominal costs linearly; changing reference length or model dimensions changes their relative reduction.

Table 7: Mean ± sample standard deviation over three trials (ms per decoder forward). Each entry is Vanilla → FastOCR; token counts are resize targets.

Model	Batch	Target image tokens	Attention latency	Decoder latency
Qwen2.5-VL-3B	8	4096	$13.95 \pm 0.02 \rightarrow 8.40 \pm 0.05$	$37.06 \pm 0.11 \rightarrow 32.85 \pm 0.15$
	12	4096	$20.22 \pm 0.02 \rightarrow 9.68 \pm 0.09$	$42.75 \pm 0.72 \rightarrow 35.41 \pm 0.61$
	8	8192	$24.70 \pm 0.01 \rightarrow 10.56 \pm 0.02$	$47.03 \pm 0.01 \rightarrow 35.22 \pm 0.06$
olmOCR-2-7B-1025	8	4096	$19.74 \pm 0.02 \rightarrow 8.22 \pm 0.03$	$38.69 \pm 0.22 \rightarrow 28.66 \pm 0.13$
	12	4096	$29.14 \pm 0.02 \rightarrow 9.71 \pm 0.01$	$47.91 \pm 0.19 \rightarrow 30.26 \pm 0.03$
	8	8192	$36.29 \pm 0.07 \rightarrow 10.80 \pm 0.10$	$54.96 \pm 0.09 \rightarrow 31.80 \pm 0.30$

D Implementation and Timing Protocol

Attention implementation. FastOCR keeps the complete KV cache resident. Its token selection is compatible with fused attention kernels (Dao, 2023): the normalized score row needed for selection can be computed alongside the attention output from $Q_{\text{last}}K^\top/\sqrt{d_{\text{head}}}$, masking, and softmax in FP32, followed by conversion to the query dtype. During profiling, every layer supplies image-attention statistics; subsequently, layer 0 and the focal layers supply the selection scores. Layer 0 can inherit the preceding step's selected image positions. Other non-focal layers gather the selected keys and values into compact dense tensors for attention. These copies leave the resident cache intact.

Inputs and configuration. Table 2 uses BF16 `Qwen/Qwen2.5-VL-3B-Instruct` and `allenai/olmOCR-2-7B-1025`, each with its checkpoint's processor, on an H200 GPU. One real document page is resized and replicated across the batch, with the prompt "Extract the text content from this image." The 4096 and 8192 image-token targets produce 4161 and 8058 image tokens, respectively, plus 29 prompt-text tokens per sequence. FastOCR uses $\rho = 0.1$, $\tau = 0.05$, $g = 1$, and configured $w = 10$, with decoding-based profiling.

Profiling and repetitions. Each trial generates 128 tokens autoregressively, using equal minimum and maximum generation lengths, and averages 127 decoder forwards. The configured $w = 10$ counter selects focal layers before the tenth forward: nine profiling forwards contribute to the mean, and the tenth is already sparse. One separate ten-token warmup generation precedes the three measured trials and is excluded. Table 7 reports their means and sample standard deviations; these quantify repeated measurements of this input, not variation across document pages.

E Detailed Results on olmOCR-Bench

Table 8 reports results on olmOCR-Bench, which covers a broader range of document categories than OmniDocBench and scores each category as a pass rate over a finite set of unit tests; possible values are therefore discrete, and identical scores across methods are expected rather than a reporting error. FastOCR retains 97.3% of the unpruned Overall at both budgets on Qwen2.5-VL, and 92.7%/92.5% on dots.ocr, while the strongest baseline reaches only 51.2% and 68.7% at the high budget and 48.0% and 34.1% at the low budget. The per-category results show substantial gaps on content-preservation tasks such as arXiv Math, Long Tiny, Multi-Column, and Table. Headers/Footers instead tests *absence*: specified header and footer snippets should not appear in the linearized output. A high score therefore measures their exclusion, not their recognition. Broadly omitting content can also satisfy an absence test, so this category alone cannot establish transcription completeness or explain a method's failure mechanism. Quest is again the most competitive baseline where the underlying model is strong (dots.ocr at the high budget, 68.7%), and again degrades far faster than FastOCR as the budget tightens, falling to 28.9% at the low budget.

F Hyperparameter Sensitivity and Warmup Ablation

Figure 5 sweeps each of FastOCR's four hyperparameters in isolation. The focal gap g and the positive focal-layer ratios shown vary Overall by about one point; the $\rho = 0$ ablation is a separate low-accuracy boundary, not part of this stable range. At $\tau = 0$, pruned layers have no direct image-token access and Overall falls to 6.89; layer 0 and focal layers still read the full cache. Across the

Table 8: Performance comparison on olmOCR-Bench, against the same four baselines as Table 1. Best among the pruning methods in each block is in bold. For FastOCR, the higher nominal FLOPs block uses τ=0.25 (8.71 G for Qwen2.5-VL; 4.36 G for dots.ocr) and the lower nominal FLOPs block uses τ=0.05 (7.00 G for Qwen2.5-VL; 3.36 G for dots.ocr). FLOPs follow the same cost model as Table 1, recomputed consistently for every entry (Appendix C). Overall is the mean of the eight per-category pass rates. All entries, including the unpruned models, were produced and scored by us under one pipeline.

Method	FLOPs (G) ↓	arXiv Math (%) ↑	Baseline (%) ↑	Headers/Footers (%) ↑	Long Tiny (%) ↑	Multi-Col (%) ↑	Old Scans (%) ↑	Old Scans Math (%) ↑	Table (%) ↑	Overall ↑	Rel. (%) ↑
Qwen2.5-VL											
Vanilla	15.10	26.70	97.30	57.80	68.10	65.70	40.30	28.40	45.00	53.66	100.0
SnapKV	8.71	1.60	97.40	**87.50**	0.20	0.10	14.10	2.60	1.80	25.66	47.8
DivPrune	8.71	1.40	96.50	74.50	1.10	1.90	20.30	5.00	9.60	26.29	49.0
CDPruner	8.71	0.20	81.10	84.60	0.00	0.20	13.30	0.40	1.00	22.60	42.1
Quest	8.71	3.60	74.70	68.80	14.50	11.70	30.40	9.20	6.80	27.46	51.2
FastOCR	8.71	**25.90**	**97.70**	57.80	**66.50**	**65.80**	**39.90**	**22.50**	**41.70**	**52.23**	**97.3**
SnapKV	7.00	0.80	96.80	**93.30**	0.00	0.00	12.90	1.30	0.90	25.75	48.0
DivPrune	7.00	0.10	96.00	87.20	0.00	0.10	15.00	2.00	1.20	25.20	47.0
CDPruner	7.00	0.10	79.80	92.20	0.20	0.00	13.10	0.00	0.40	23.23	43.3
Quest	7.00	0.10	70.70	82.90	0.70	0.50	15.00	2.00	0.40	21.54	40.1
FastOCR	7.00	**26.30**	**97.30**	58.20	**66.70**	**63.80**	**36.90**	**24.50**	**44.00**	**52.21**	**97.3**
dots.ocr											
Vanilla	8.10	77.70	98.30	79.90	85.10	82.70	39.20	69.90	87.00	77.47	100.0
SnapKV	4.36	3.30	80.60	**90.50**	0.90	0.00	14.40	9.00	4.30	25.38	32.8
DivPrune	4.36	7.20	94.30	87.10	0.90	2.00	26.00	14.60	25.10	32.15	41.5
CDPruner	4.36	6.90	94.70	88.30	0.50	1.40	25.10	15.50	19.70	31.51	40.7
Quest	4.36	50.70	85.90	81.20	35.50	38.00	31.70	44.80	58.00	53.23	68.7
FastOCR	4.36	**73.80**	**97.90**	83.40	**64.90**	**63.00**	**36.90**	**68.30**	**86.10**	**71.79**	**92.7**
SnapKV	3.36	1.80	80.60	93.20	0.00	0.00	12.70	3.10	1.00	24.05	31.0
DivPrune	3.36	0.60	92.80	**95.00**	0.20	0.20	15.60	1.30	5.20	26.36	34.0
CDPruner	3.36	0.90	94.10	93.20	0.00	0.10	16.50	2.00	4.60	26.43	34.1
Quest	3.36	4.90	46.10	90.30	2.00	1.60	17.50	7.90	9.00	22.41	28.9
FastOCR	3.36	**74.00**	**98.30**	83.40	**67.40**	**63.60**	**36.30**	**66.40**	**84.20**	**71.70**	**92.5**

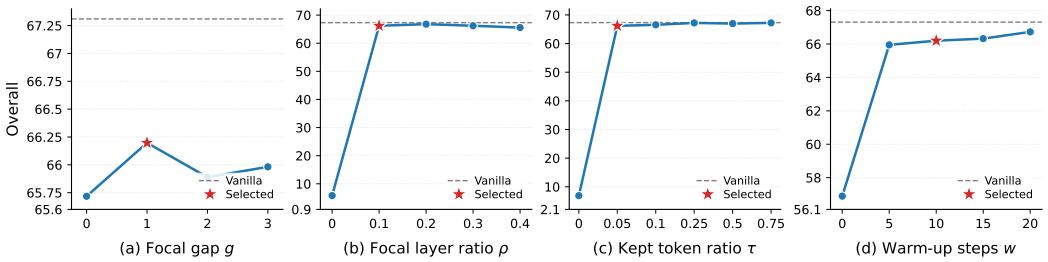

Figure 5: Sensitivity of the OmniDocBench Overall score to FastOCR's four hyperparameters on Qwen2.5-VL. In each panel, one hyperparameter is swept while the remaining three are held at their defaults ($g = 1$, $\rho = 0.1$, $\tau = 0.05$, $w = 10$). The gray dashed line marks the uncompressed Vanilla baseline (67.31), and the red star marks the configuration adopted in our final model—selected to trade a small amount of accuracy for lower FLOPs and more aggressive KV-cache pruning rather than to maximize the Overall score.

measured $\tau \geq 0.05$ settings, Overall rises gradually from 66.20 at $\tau = 0.05$ to 67.22 at $\tau = 0.75$. For warmup (panel (d)), the prefill-profiled variant labeled $w = 0$ scores 56.89, below the 66.20 default. The measured decode-profiling settings $w \geq 5$ score 65.95–66.73 (Table 9). In all four panels, the selected configuration sits close to, though not always at, the per-panel maximum, reflecting our efficiency-driven rather than accuracy-maximizing choice.

Table 9 reports the full ablation on the number of warmup steps w. When $w = 0$ (no warmup), focal layers are selected from the attention distribution observed during the prefill stage rather than the decode stage; the overall score drops to 56.89, confirming that prefill-stage attention is a poor proxy for decoding-time behavior. Introducing even a short warmup ($w = 5$) raises the overall score to 65.95, demonstrating the importance of profiling attention during actual decoding. Performance continues to improve as w increases from 5 to 20, but the marginal gains diminish: the gap between $w = 10$ and $w = 20$ is only 0.53. We adopt $w = 10$ as the default decode-profiling threshold.

Table 9: Ablation on the number of warmup steps w (Qwen2.5-VL, OmniDocBench), scored with the unified CDM scorer of Table 1. $w = 0$: focal layers selected from prefill-stage attention. $w > 0$: decode-profiling threshold w (the counter convention is specified in Appendix A).

Warmup Steps	Text Block Edit Dist ↓	Display Formula CDM ↑	Table TEDS ↑	Reading Order Edit Dist ↓	Overall ↑
$w = 0$ (prefill)	0.256	0.496	46.67	0.353	56.89
$w = 5$	0.179	0.588	56.95	0.260	65.95
$w = 10$	0.172	0.590	56.86	0.255	66.20
$w = 15$	0.167	0.596	56.06	0.254	66.32
$w = 20$	0.167	0.598	57.07	0.254	66.73

G Focal Layer Selection: Signal Choice and Layer Distribution

Focal Layer Selection scores each layer by the image-attention ratio of the *decoding* query during a short warmup (Section 3.2). A natural alternative is to score layers using the attention that *prompt* tokens pay to image tokens during prefill, since prior work on cross-modal information flow shows that visual evidence is progressively integrated into prompt-token representations before generation begins (Zhang et al., 2025d; Saporita et al., 2026). This appendix compares prefill scoring choices and the separate warmup ablation.

Prefill reductions select different layers. We instrument a single prefill pass and compute, for every layer, the image-attention ratio under four reductions of the same attention matrix: the last prompt row (the tail-position signal), the mean over all post-image prompt rows, the best single prompt row, and the mean over all query rows. Table 10 lists the resulting focal-layer sets. The prompt-row reductions rank *early* layers highest (layer 1 for both models), whereas the last-prompt-row proxy selects *middle-to-late* layers; Spearman rank correlation between the prompt-mean and tail-position rankings is only 0.54 (Qwen2.5-VL) and 0.40 (dots.ocr), and the selected sets overlap in at most one layer. All four reductions, including the tail position, use the same prefill pass. This table therefore compares prefill proxies; it does not directly measure the layer ranking obtained after decoding warmup.

Table 10: Focal layers selected by different reductions of the same prefill attention matrix ($\rho = 0.1$, $g = 1$). "Tail position" is the last prompt row. ρ_s is Spearman rank correlation of per-layer scores against the tail-position ranking.

Model	Scoring signal	Focal layers	ρ_s
Qwen2.5-VL	Tail position	21, 27, 33	1.00
	Prompt rows (mean)	1, 12, 21	0.54
	Prompt rows (best row)	1, 12, 21	0.70
	All query rows (mean)	2, 19, 34	0.05
dots.ocr	Tail position	20, 27	1.00
	Prompt rows (mean)	1, 27	0.40
	Prompt rows (best row)	1, 27	0.37
	All query rows (mean)	1, 27	0.18

The decoding-time signal is the one that matters empirically. Table 9 separately compares profiling modes: the variant labeled $w = 0$ selects focal layers from prefill-stage attention and drops the overall score from 66.20 to 56.89, a loss of more than nine points, while as few as five decoding warmup steps recover it. We additionally ran the specific prompt-row variant studied here by forcing the focal set to the layers that signal selects ($\{1, 12, 21\}$ for Qwen2.5-VL) on a 12-sample subset spanning four document types. Measured as character-level agreement with the unpruned transcription of the same page, the reference configuration reaches 0.882 mean agreement versus 0.791 for the prompt-row signal, and is the better of the two on 9 of 12 pages and in *all four* document categories (academic literature 0.843 vs. 0.762, book 0.924 vs. 0.717, exam paper 0.839 vs. 0.806, magazine 0.923 vs.

0.878). The prompt-row signal thus degrades transcription consistently rather than in one outlier category.

Why FastOCR nonetheless preserves the prompt-mediated pathway. It is worth stressing that this design choice concerns only *which layers are profiled*, not which information the model may use. FastOCR leaves the prefill stage completely untouched: attention is unrestricted during prefill and the full KV cache is retained thereafter, so the multi-stage visual-to-prompt integration characterized by Zhang et al. (2025d) takes place exactly as in the unpruned model, and every prompt-token representation that absorbed visual evidence remains available at every subsequent step. Pruning applies only to which image-token keys a *decoding* query attends to, and even then only at non-focal layers. The prompt-mediated pathway is thus intact by construction, which is one reason FastOCR degrades so gracefully at aggressive keep ratios. We further note that Saporita et al. (2026) find fusion patterns to be strongly task-dependent, supporting our decision to profile the behavior of the OCR decoding process directly rather than transferring conclusions drawn from short-answer VQA.

PROPERTIES OF THE SELECTED FOCAL LAYERS

Figure 6 reveals three structural properties of the focal layer set.

Sparsity across the network. Only 10 of 36 layers on Qwen2.5-VL and 7 of 28 layers on dots.ocr are ever identified as focal across all samples, while any individual sample activates only $\lfloor \rho L \rfloor$ of them (7 and 5 at the $\rho = 0.2$ used here). The focal set is therefore far smaller than the theoretical upper bound, leaving room for substantial computational savings.

Stability across samples. Layers 17, 19, 21, 31, and 33 on Qwen2.5-VL, together with layers 16, 20, and 27 on dots.ocr, are selected as focal in 100% of samples. This indicates that focal layers are essentially sample-invariant and can be reliably identified from a small calibration set without per-sample re-identification.

Concentration in middle and late layers. On Qwen2.5-VL, layer 0 is never selected as focal, and on dots.ocr it appears in only 3.5% of samples. This confirms that $l_0 \notin \mathcal{C}$ holds in the vast majority of decoding steps, and Cross-Step Fixation Reuse is therefore broadly applicable.

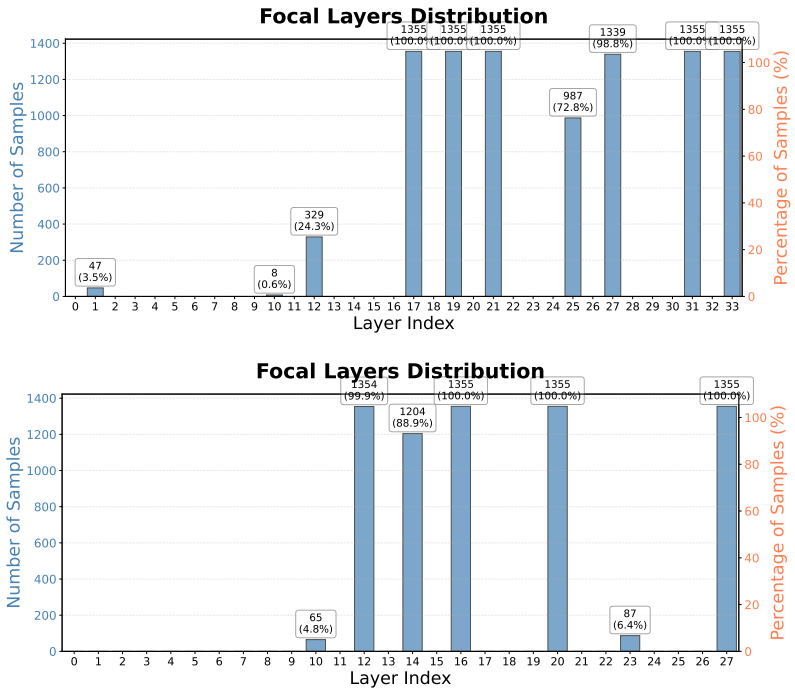

Figure 6: Distribution of focal layers across all 1355 samples in OmniDocBench on Qwen2.5-VL (top) and dots.ocr (bottom) with $\rho = 0.2$. The integer and percentage above each bar denote the number and proportion of samples that select the corresponding layer as a focal layer.

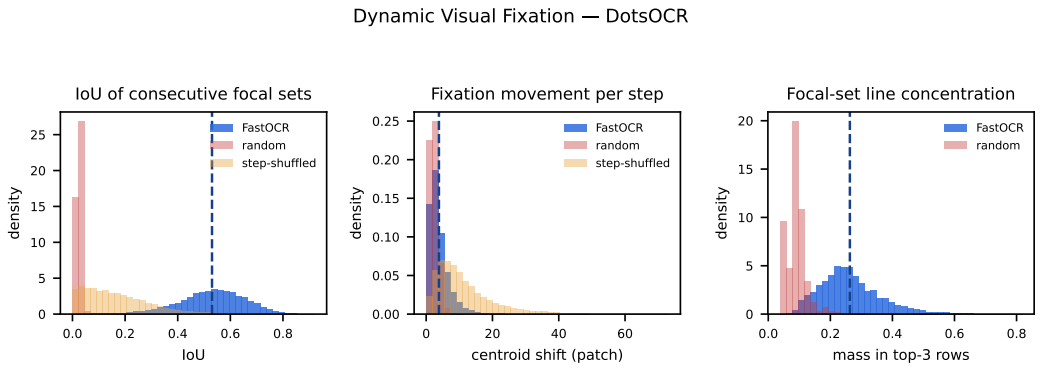

Figure 7: FastOCR distributions underlying Table 11 (dots.ocr). Left: overlap between consecutive focal sets. Middle: centroid displacement between consecutive steps, against the step-shuffled control. Right: fraction of selected tokens in the three most populated patch rows. Per-document-type breakdowns are in Figure 8.

H Quantifying Dynamic Visual Fixation

We quantify Dynamic Visual Fixation (Section 3.1) both during FastOCR decoding and in an unpruned model. We map the focal-token sets $\mathcal{F}_l^t$ to the patch grid and measure consecutive-step set stability, centroid displacement, and per-step spatial concentration. All three measurements use the same analysis procedure, with each recorded set or consecutive pair weighted equally. These are descriptive pooled means; layer-step pairs from the same page are correlated and are not independent samples. The top-3-row statistic is the fraction of selected tokens in the three most populated patch rows; it does not sum attention weights.

FastOCR measurement. We log each freshly computed focal set on dots.ocr and Qwen2.5-VL, with 40 OmniDocBench pages per model spanning all 10 document types and up to 255 decoding steps per page (τ=0.05, ρ=0.1). This yields 18,652 and 25,785 consecutive pairs, respectively. We exclude inherited CSFR sets and pair only adjacent steps at the same layer.

Natural behavior without pruning. For Qwen2.5-VL, we also collect attention during unpruned decoding and record the top 5% of image tokens by head-averaged attention at explicitly fixed, zero-based layers $\{27, 31, 33\}$. These indices are the most frequent fresh-layer set in the original FastOCR traces. The original FastOCR measurement uses focal layers selected by the model through prefill profiling; the natural measurement fixes the three layer indices throughout, independently of the collector's decode warmup. In the natural run, every layer attends to the full cache; the recorded top-token sets are diagnostic and never restrict attention or propagate to other layers or steps. Of 40 sampled pages across all 10 document types, 39 yield usable fixation records and 25,878 consecutive pairs; one page yields no decode-step fixation. Generation is capped at 256 new tokens, as in the FastOCR collection. Thus the two measurements share the token-selection and analysis rules but differ in layer selection and decoding behavior; their similarity is evidence for the phenomenon at the observed layers, rather than a controlled estimate of pruning's effect. The runs generate freely, so output trajectories can differ. The measurements cover the first 255 decode forwards at most and do not establish attention-mass coverage of the selected 5% or fixation statistics later in long outputs.

A raw stability number is uninterpretable without a reference, so we report two controls. The *random* control replaces each focal set with a size-matched uniform sample of image tokens, isolating the effect of set size. The *step-shuffled* control keeps the model's own focal sets but pairs them after randomly permuting step order within each page and layer, testing temporal structure: any gap between it and the consecutive measurement cannot be explained by the model merely preferring certain tokens. Both controls are computed separately for each measurement, matching its token sets and page grids. Table 11 reports the results; Figure 7 shows the FastOCR distributions for dots.ocr.

DVF is present without pruning. At the fixed Qwen2.5-VL layers, natural consecutive sets have IoU 0.511 and retained fraction 0.664, close to FastOCR's 0.502 and 0.656 and above their respective step-shuffled controls. Natural centroid displacement is 3.26 patches versus 9.80 after shuffling; the

Table 11: Dynamic Visual Fixation on OmniDocBench with top-5% image-token sets. Natural behavior is measured without pruning at fixed Qwen2.5-VL layers; FastOCR uses model-selected focal layers (ρ=0.1). Controls use size-matched uniform samples (random) or permuted step order (step-shuffled). In the Qwen2.5-VL panel, paired control values are natural / FastOCR; values equal at the displayed precision appear once. Displacement is compared with step-shuffled sets, since uniform sets have centroids near the page center. The natural run is available only for Qwen2.5-VL.

	Natural	**FastOCR**	**Controls**	
Metric	unpruned		random	step-shuffled
Qwen2.5-VL				
IoU($\mathcal{F}^t, \mathcal{F}^{t+1}$) ↑	0.511	0.502	0.026	0.137 / 0.148
Retained fraction $\lvert \cap \rvert / \lvert \mathcal{F}^t \rvert$ ↑	0.664	0.656	0.050	0.225 / 0.233
Centroid shift (patches) ↓	3.26	3.57	2.27 / 2.28	9.80 / 10.97
Rows occupied / all rows ↓	0.631	0.635	0.951	—
Token fraction in top-3 rows ↑	0.273	0.268	0.089 / 0.087	—
Mean NN distance (patches) ↓	1.68	1.66	2.38	—
Radius of gyration (patches)† ↓	25.9	27.9	30.5 / 32.6	—
dots.ocr				
IoU($\mathcal{F}^t, \mathcal{F}^{t+1}$) ↑	—	0.530	0.026	0.169
Retained fraction $\lvert \cap \rvert / \lvert \mathcal{F}^t \rvert$ ↑	—	0.684	0.050	0.271
Centroid shift (patches) ↓	—	3.85	2.27	11.06
Rows occupied / all rows ↓	—	0.611	0.951	—
Token fraction in top-3 rows ↑	—	0.263	0.089	—
Mean NN distance (patches) ↓	—	1.49	2.38	—
Radius of gyration (patches)† ↓	—	27.6	31.7	—

†Outlier-sensitive; reported for completeness but not used as evidence (see text).

corresponding FastOCR values are 3.57 and 10.97. Row occupancy (0.631 vs. 0.635), top-3-row token fraction (0.273 vs. 0.268), and nearest-neighbor distance (1.68 vs. 1.66) also agree closely. Temporal stability and line-local concentration therefore occur in the unpruned model at these layers, without either pruning or CSFR imposing them. The remaining analysis describes the two FastOCR measurements.

The attended set is stable across steps, and not because of set size. Consecutive focal sets overlap with IoU 0.53 and 0.50, roughly 20× the uniform control. More tellingly, they exceed the step-shuffled control by 3.1× and 3.4×: the stability is a property of *temporal adjacency*, not of the model returning to a globally preferred set of tokens. Two thirds of the tokens attended at one step are attended again at the next.

Fixation drifts gradually. Between consecutive steps the focal centroid moves 3.85 and 3.57 patches, against 11.06 and 10.97 for the same sets paired non-consecutively—a 2.9–3.1× gap that isolates gradual drift from arbitrary jumps. We stress that this must be read against the step-shuffled control: measured against the uniform control, displacement appears *larger* (2.27–2.28), because a uniform sample covers the entire page and its centroid is pinned near the page center regardless of the step. The uniform control is informative about set overlap but structurally unable to speak about movement.

Locality is line-shaped, not blob-shaped. This is where a natural description of the phenomenon fails. The focal set is clearly localized along the vertical axis—it occupies 61–64% of patch rows against 95% for the uniform control, places 26–27% of its selected tokens in just three rows against 8.7–8.9%, and has a smaller mean nearest-neighbor distance—but it is *not* a compact region. Radius of gyration and bounding-box area barely separate it from the uniform control (dots.ocr: 27.6 vs. 31.7; 0.951 vs. 0.997 of the grid), because alongside a dense line-local core the set contains sparse long-range tokens that stretch its extent to the page borders. Describing FastOCR's fixation as attention confined to a small contiguous window would therefore be inaccurate; the correct statement is that attention concentrates on a few text lines while retaining a thin long-range tail. This matches the line-by-line nature of transcription, and it is why our selection operates on token identity rather than on any spatial window.

BREAKDOWN BY DOCUMENT TYPE

The ordering of types is itself consistent across the two models and follows page layout rather than content domain. Single-column, sparsely typeset pages (`note`, `book`) yield the most line-local fixation and the smallest per-step drift, whereas multi-column and irregularly composed pages (`magazine`, `newspaper`) spread the focal set over more rows and produce roughly twice the centroid displacement — the model must jump between columns and inserts, so its reading trajectory is genuinely less smooth. Fixation locality thus degrades gracefully and predictably with layout complexity, which is also the regime in which we would expect a fixed kept-token ratio to be most strained.

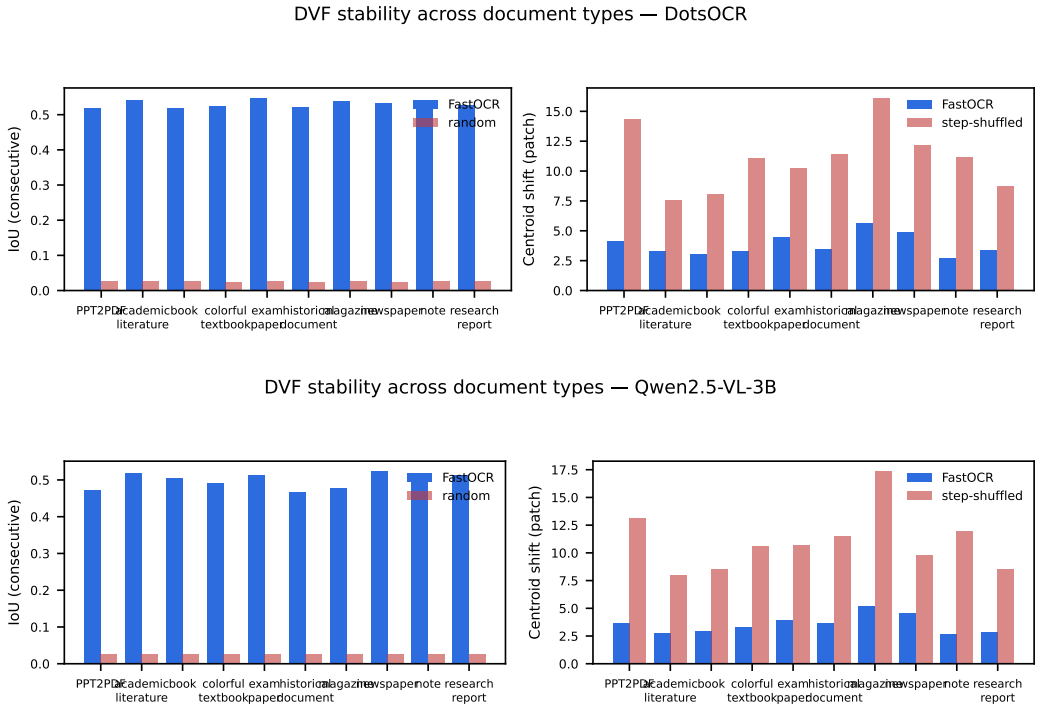

Figure 8: Dynamic Visual Fixation statistics per document type. Top: dots.ocr. Bottom: Qwen2.5-VL. Consecutive-step IoU is shown against the size-matched random control; centroid displacement is shown against the step-shuffled control, which is the appropriate reference for movement (see Section 3.1).

I EFFECT OF PRUNING ON OUTPUT LENGTH AND TERMINATION

OCR metrics are computed on generated text, so a pruning method could in principle improve or damage its score by changing how much text the model produces or when it stops, independently of how well it reads. This appendix checks that directly. We compare each method against the unpruned model on the same pages, pairing outputs by document so that the comparison is per-page rather than distributional.

Table 12 reports the results on OmniDocBench with Qwen2.5-VL. Length is measured in output characters. Repetition is the duplicated fraction of 20-word n-grams, using whitespace tokenization; outputs with fewer than 20 words, including empty outputs, receive zero. This diagnostic can undercount repetition in languages without whitespace-separated words. We count an output as degenerate at a score above 0.5. Per-page length ratios use $\max(1, \text{Vanilla characters})$ as the denominator. Because `num_decode_steps` is recorded per batch rather than per sample, we

Table 12: Output-length and termination behaviour on OmniDocBench (Qwen2.5-VL, cap 4000 new tokens), computed on the 1106 pages completed by all FLOPs-aligned runs. Length ratio is the median per-document ratio of output characters to the unpruned model's, paired by page. Repetition is the duplicated-20-gram fraction (mean); degenerate counts outputs above 0.5.

Method	Median chars	Length ratio	Empty	In capped batch	Repetition	Degenerate
Vanilla	1022	—	5	21.0%	0.026	29
FastOCR	977	1.000	4	2.9%	0.023	25
DivPrune	445	0.525	23	44.6%	0.058	68
CDPruner	74	0.124	133	50.9%	0.078	87
SnapKV	356	0.598	11	88.7%	0.195	220
Quest	245	0.510	8	72.1%	0.150	169

report the share of samples belonging to a batch that reached the token cap, which upper-bounds the individual truncation rate.

The output statistics show both omission and repetitive long tails. DivPrune and CDPruner have median paired length ratios of 0.525 and 0.124, respectively; CDPruner produces 133 empty outputs. These short outputs are consistent with missing content, although length statistics alone do not separate omission from recognition errors.

SnapKV and Quest also have median paired length ratios below one (0.598 and 0.510), while exhibiting stronger repetitive tails. Their repetition scores are 0.195 and 0.150, with 220 and 169 degenerate outputs; their mean lengths (3534 and 2708 characters) exceed their medians (356 and 245). Their high shares of samples in capped batches (88.7% and 72.1%) are consistent with this long-tail behaviour, but do not identify which individual pages failed to terminate.

These results show why OCR accuracy reflects output coverage and repetition as well as character recognition. Variable output lengths also affect total generation time; per-step latency and full-document runtime answer different questions.

J LIMITATIONS

Memory. FastOCR retains the full KV cache so that later steps can re-select any visual token. It therefore does not reduce persistent KV storage or provide the cache-capacity gains of physical eviction. Gathered tensors and score buffers can additionally increase peak memory; retaining the cache does not imply that peak allocation is identical to Vanilla.

Where the speedup applies. Attention and decoder-forward speedups differ because FFN, projections, and normalization remain. The gain depends on model dimensions, batch size, and input length; the nominal FLOPs reduction also changes with model dimensions and reference length. When attention is a small part of the forward, selection and gather costs may outweigh the saved work. The reported timings exclude vision encoding, prefill, and output sampling, and do not measure full-document OCR latency.