DIFFUSION DRAFTS, AR VERIFIES: ACCELERATING DOCUMENT OCR WITH SELF-SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive OCR vision–language models accurately convert document images into text and structured markup, but require one sequential decoding step per output token, limiting inference speed. Unlike open-ended text generation, OCR outputs are strongly grounded in the input image, making diffusion-based parallel generation promising. However, when several tokens are predicted in one diffusion step, each is predicted before the others are known. Committing them directly can therefore introduce errors. We therefore introduce GRAVITYOCR, a parameter-shared AR–block-diffusion model jointly trained for parallel drafting and causal AR verification. Verifying drafts before commitment lets the model commit multiple output tokens per round without a separate drafting network. The causal AR path also enables GRPO with sequence- and structure-level OCR rewards, avoiding diffusion-trajectory likelihood estimation while updating the shared drafter parameters. On OmniDocBench v1.6, AR-path GRPO improves the Overall score from 94.92 to 95.16 without reducing diffusion drafting efficiency, while the final model remains close to the original GLM-OCR score of 95.48. In an SGLang serving deployment, GRAVITYOCR commits an average of 9.7 output tokens per forward pass and achieves a **3.94× decode-only speedup** on region crops and a **1.32× end-to-end page-processing speedup** over AR decoding.

1 INTRODUCTION

Generative OCR vision–language models have emerged as a leading approach to document understanding, converting document images into serialized sequences of plain text and structured markup such as HTML and LaTeX. Most of these models generate their outputs autoregressively, requiring

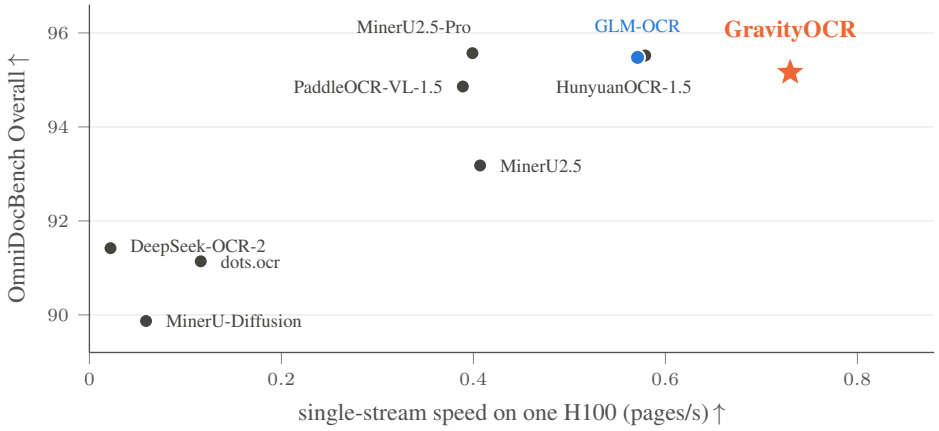


Figure 1: **Accuracy–speed landscape of document OCR systems.** OmniDocBench v1.6 Overall vs. single-stream page rate, with the same evaluation set used across systems for each metric. GravityOCR is the fastest system measured while maintaining top-tier accuracy.

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.

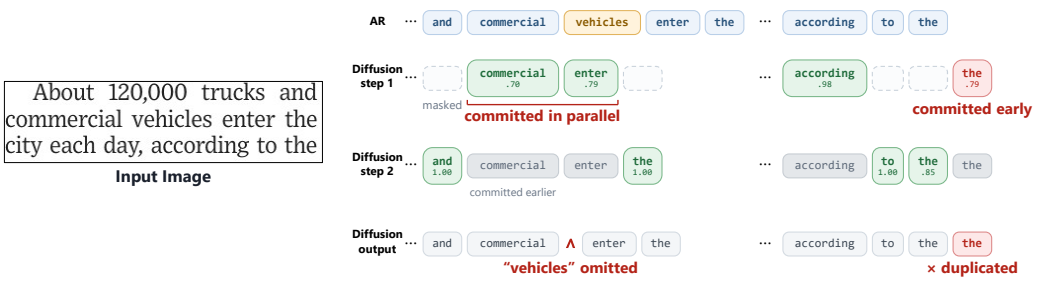


Figure 2: **Failure case of confidence-based parallel commitment.** At τ_c=0.7, commercial and enter are committed together while the intervening position remains unresolved, ultimately omitting vehicles. Similarly, the final the is committed before the two positions to its left are resolved, producing the repeated span to the the. The causal AR output from the same checkpoint is shown for comparison.

one sequential decoding step for each output token. This sequential dependency directly affects latency and throughput when OCR models are deployed at scale. Recent OCR systems have therefore explored faster generation through multi-token prediction and speculative decoding (Leviathan et al., 2023; Chen et al., 2023; Duan et al., 2026; Li et al., 2026). In parallel, diffusion-based OCR models have begun to generate multiple output tokens simultaneously (Dong et al., 2026; Man et al., 2026).

OCR is particularly amenable to parallel generation because many output positions are directly constrained by the input image, unlike in open-ended text generation (Man et al., 2026; Dong et al., 2026). Masked diffusion provides a natural mechanism for exploiting this property: multiple output positions can be recovered together rather than generated strictly one at a time (Austin et al., 2021; Sahoo et al., 2024). This makes diffusion-based decoding a promising direction for reducing the inference cost of OCR.

Visual grounding, however, does not eliminate sequential dependencies among OCR output tokens. Text must preserve its reading order, while tags and delimiters in tables and formulas must remain structurally consistent. Under causal AR decoding, each token is predicted after the preceding tokens have become available. In a parallel diffusion step, by contrast, positions predicted together cannot condition on the tokens simultaneously selected at the other positions. Directly committing these predictions can therefore produce tokens that are individually plausible under the partially resolved block but inconsistent in the completed output. Figure 2 illustrates this failure mode under confidence-based parallel decoding (Wu et al., 2025b).

Motivated by this failure mode, we introduce GRAVITYOCR, a parameter-shared AR–diffusion model inspired by recent AR–diffusion hybrids and diffusion-based drafting methods (Fu et al., 2026; Liu et al., 2025b; Wu et al., 2026a; Chen et al., 2026). GRAVITYOCR uses block diffusion (Arriola et al., 2025) to propose an entire token block in one forward pass and causal AR decoding to verify the proposal before commitment. The causal path commits only the longest draft prefix consistent with its token-by-token predictions. With top-1 verification, the resulting self-speculative decoder reproduces the causal AR greedy output exactly (Appendix C.3 measures the agreement under bf16 serving kernels). Successful drafts advance decoding by multiple tokens, with the first draft–verifier disagreement determining the accepted draft length and, consequently, the attainable acceleration. Because both prediction paths share parameters, our method requires neither a separate drafting network nor an auxiliary prediction head.

We instantiate GRAVITYOCR by adapting GLM-OCR (Duan et al., 2026). We then further train the model with GRPO through the causal AR path, which determines the final output, using sequence- and structure-level OCR rewards. We show that this improves the shared model without requiring likelihood estimation over diffusion trajectories. On OmniDocBench v1.6, our final model retains 99.7% of the original GLM-OCR model's Overall score (95.16 vs. 95.48), while self-speculative decoding substantially reduces the number of sequential decoding steps (Figure 1).

Our contributions are summarized as follows:

- We show that a pretrained causal AR OCR model can be adapted into a parameter-shared block-diffusion drafter and causal AR verifier, enabling faster decoding with minimal accuracy degradation and without requiring a separate drafting network.
- We implement full-backbone self-speculative decoding in SGLang (Zheng et al., 2024), achieving 9.7 tokens per forward (TPF), where both diffusion drafting and causal verification forwards are counted. This yields a $3.94\times$ speedup in token generation and a $1.32\times$ speedup in end-to-end page processing over causal AR decoding.
- We compare direct block-diffusion and self-speculative decoding on OCR, characterizing their accuracy–parallelism trade-off. We further show that GRPO applied only through the causal AR path improves the OmniDocBench Overall score from 94.92 to 95.16 while preserving the efficiency of the shared diffusion drafter.

2 PRELIMINARIES

Task Formulation. Given an image I and task prompt c, the target token sequence $\mathbf{y} = (y_1, \ldots, y_N)$ encodes plain text, an HTML table, or a LaTeX formula. The causal decoder p_θ^{AR} is trained by minimizing

$$\mathcal{L}_{\mathrm{AR}} = -\log p_\theta^{\mathrm{AR}}(\mathbf{y} \mid I, c) = -\sum_{i=1}^{N} \log p_\theta^{\mathrm{AR}}(y_i \mid I, c, \mathbf{y}_{<i}). \tag{1}$$

At inference, causal AR decoding generates one token per step, so an output of length N requires N sequential decoding steps.

Block Diffusion. Masked diffusion reconstructs multiple masked positions in parallel (Austin et al., 2021; Nie et al., 2025; Ye et al., 2025). Block diffusion partitions $\mathbf{y}$ into consecutive blocks $\mathbf{y}^{(b)}$ of at most B tokens (Arriola et al., 2025; Wu et al., 2025a). It generates the blocks from left to right: for block b, the diffusion decoder p_θ^{diff} conditions on all preceding completed blocks $\mathbf{y}^{(<b)}$ and uses bidirectional attention within the current block. At noise level t, the current block is represented by $\mathbf{y}_t^{(b)}$, where positions $\mathcal{M}_t^{(b)}$ are replaced by the mask token $[\mathtt{M}]$.

The corresponding denoising objective is

$$\mathcal{L}_{\mathrm{diff}} = -\mathbb{E}_{b,t,\mathcal{M}_t^{(b)}} \left[w(t) \sum_{j \in \mathcal{M}_t^{(b)}} \log p_\theta^{\mathrm{diff}} \left(y_j^{(b)} \mid I, c, \mathbf{y}^{(<b)}, \mathbf{y}_t^{(b)} \right) \right], \tag{2}$$

where j indexes token positions within block b, and $w(t)$ denotes the timestep-dependent loss weight.

At inference, the current block is initialized with B mask tokens. At each denoising step, the model predicts the remaining masked positions in parallel; a selection rule, such as confidence thresholding, determines which predictions to unmask and retain (Sahoo et al., 2024; Wu et al., 2025b). Once a block is complete, its cached prefix states are reused for subsequent blocks. Further details are given in Appendix C.1.

Speculative Decoding. Speculative decoding accelerates AR generation by allowing a drafter to propose multiple future tokens, which the target AR model evaluates in parallel (Leviathan et al., 2023; Chen et al., 2023). Under greedy decoding, consecutive draft tokens are accepted while they match the target model's predictions. Under stochastic decoding, the standard acceptance–rejection rule preserves the target distribution. Our shared-model realization is described in Section 3.2.

3 METHOD

Our goal is to reduce the decoding cost of OCR while retaining the recognition accuracy of causal AR decoding. Motivated by the failure case in Figure 2, we separate parallel drafting from token commitment. A block-diffusion path proposes multiple tokens in parallel, while a causal AR path checks the proposals against its next-token predictions and determines which tokens can be committed. Each verification prediction conditions on the preceding tokens in the proposed sequence, allowing the model to account for dependencies that are unresolved during parallel drafting. We implement both

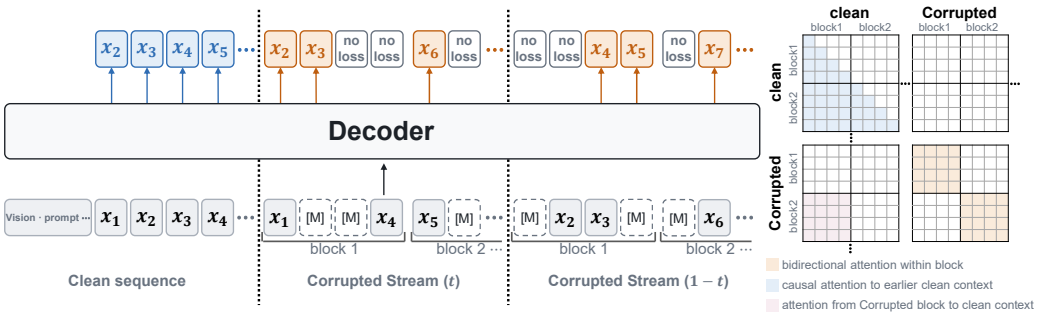


Figure 3: **Joint AR–diffusion training in a single forward pass.** The clean response stream provides causal next-token supervision, while two complementary corrupted streams with masking ratios t and $1 - t$ provide denoising supervision over disjoint masked targets. All streams share the visual and prompt context and backbone. The right panel contrasts token-level causal attention in the clean stream with causal cross-block and bidirectional within-block attention in the corrupted streams.

paths within a shared model. Sections 3.1 to 3.3 describe joint AR–diffusion training, self-speculative decoding, and reinforcement learning through the causal AR path, respectively.

3.1 Learning a Shared Block-Diffusion Drafter and AR Verifier

Joint AR–Diffusion Training. We adapt a pretrained AR OCR model to support block-diffusion drafting alongside causal AR verification. Both paths share the vision encoder, language decoder, and LM head. The only architectural addition is a learned embedding for the mask token [M]; no separate drafting network or auxiliary prediction head is introduced. To support parallel drafting, we train the model to reconstruct masked tokens in corrupted responses through denoising supervision. We also retain causal next-token supervision on clean responses to train the AR path used for verification. As shown in Figure 3, each training forward processes one clean response stream and two complementary corrupted streams, all conditioned on the same uncorrupted image I and prompt c. For each block, we sample $t \sim \mathcal{U}(0, 1)$ and independently mask tokens with probability t in one stream and the complementary positions in the other, following Fast-dLLM v2 (Wu et al., 2025a). Every response token thus receives denoising supervision in exactly one corrupted stream, while the clean stream supplies causal next-token supervision. Image and prompt tokens are neither masked nor supervised. The clean stream uses causal attention, so each target token y_i is predicted using only the image I, task prompt c, and preceding tokens $\mathbf{y}_{<i}$. Within each corrupted stream, tokens in block b can attend to all positions in the same corrupted block. They can also attend to the preceding clean response blocks $\mathbf{y}^{(<b)}$, but not to the current or future clean blocks. Both clean and corrupted streams predict y_i from the logit at position $i - 1$. In corrupted streams, denoising supervision applies only to masked targets.

Training Objective. The losses from the clean and corrupted response streams are combined as

$$\mathcal{L} = \mathcal{L}_{\text{AR}} + \lambda \mathcal{L}_{\text{diff}}, \tag{3}$$

where λ controls the relative weight of the diffusion objective. We use equally weighted, per-token averaged AR and diffusion losses; see Appendix A for details.

3.2 Self-Speculative Decoding

In direct block-diffusion decoding, token commitment is based on the diffusion model's confidence. An incorrect token can therefore be committed before the block is fully resolved; once fixed, it may lead to omissions, repetitions, or structural errors. We avoid this failure mode by separating parallel drafting from token commitment. The block-diffusion path proposes a token block in parallel, while the causal AR path verifies the draft and determines which tokens enter the output. The verifier accepts only the longest draft prefix that matches its next-token predictions, preserving causal AR decoding while allowing successful drafts to advance generation by multiple tokens. Figure 4 illustrates the resulting procedure.

Each round feeds the last committed token x_0 followed by B mask tokens to the shared decoder. In a single forward pass, the decoder predicts the next token a_0 and a block of draft tokens $d_{1:B}$. We

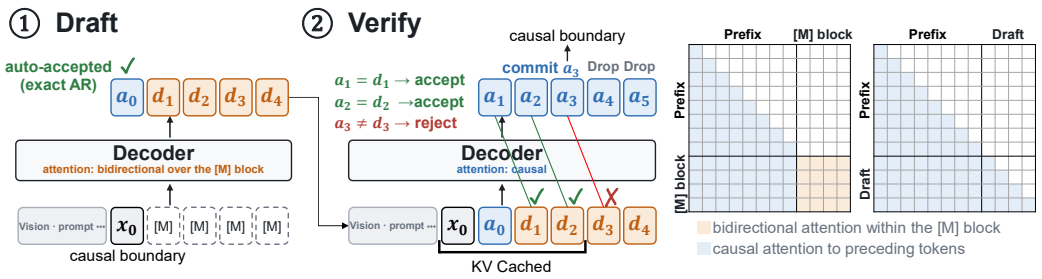


Figure 4: **Self-speculative decoding.** Given a causal boundary token x_0 and B masks, the shared decoder produces the causal prediction a_0, accepted by construction, and parallel drafts $d_{1:B}$ in one forward pass, without iterative unmasking or confidence-based selection. Masks attend bidirectionally within the block and causally to the cached prefix. Causal verification of $a_0, d_{1:B}$ yields $a_{1:B+1}$ and commits the longest AR-consistent draft prefix plus the next AR token while building the accepted prefix's KV cache; rejected states are discarded. The final AR token becomes the next boundary and receives its KV state in the following round.

commit a_0 directly because its prediction is conditioned only on the committed prefix via causal attention. For verification, we run a second forward pass over $[a_0, d_{1:B}]$ with causal attention. We then commit the longest draft prefix $d_{1:A}$ matching the verifier's predictions $a_{1:A}$, followed by its next prediction a_{A+1}, which becomes the next round's boundary token.

3.3 RL ON THE AR PATH

The joint training objective in Section 3.1 provides token-level supervision for both causal AR prediction and block-diffusion drafting. However, OCR quality is evaluated on complete outputs using sequence- and structure-level criteria that are not directly optimized by token-level likelihood training. We therefore further optimize the jointly trained model using task-aware OCR rewards.

For multi-step masked diffusion, evaluating the sequence likelihood requires marginalizing over unmasking trajectories, so existing RL methods approximate it or apply policy gradients along the denoising trajectory (Zhao et al., 2025; Zhan, 2025). In GRAVITYOCR, however, the causal AR verifier determines the output of self-speculative decoding. We can therefore optimize the AR path using its exact autoregressive likelihood and apply standard GRPO directly. We fine-tune the full model through this path, updating the diffusion drafter through shared parameters without a diffusion-specific objective.

Task-Aware OCR Rewards. For each sampled output, we select the reward according to its OCR task. For plain-text recognition, we use normalized edit similarity, $1 - \mathrm{NED}$, where NED is the normalized edit distance. For HTML table recognition, we combine structure-only TEDS (Zhong et al., 2020) with a cell-content similarity and penalize malformed or degenerate markup. For formula recognition, we use the edit similarity of canonicalized LaTeX, scaled down when the output fails well-formedness checks, as a rendering-free proxy for CDM (Wang et al., 2024b). All rewards lie in $[0, 1]$; exact definitions and coefficients are given in Appendix B.

GRPO on the Causal AR Path. For each input, we sample a group of outputs from the jointly trained model's causal AR path and score them with the corresponding task-aware reward. GRPO (Shao et al., 2024) computes relative advantages within each group and updates the model using causal token-level log-probabilities. Because RL updates through the causal AR path are applied to the shared parameters, they can also change the block-diffusion drafter and its agreement with the verifier. As shown in Section 4.4, GRPO improves OCR accuracy while tokens per forward remain essentially unchanged, indicating that drafter–verifier agreement is preserved after RL.

4 EXPERIMENTS

4.1 SETUP

Model and Training. We adapt the vision encoder and language decoder of GLM-OCR (Duan et al., 2026), retaining its PP-DocLayout-V3 layout detector (Sun et al., 2025) and output assembly pipeline.

Table 1: **Main results on OmniDocBench v1.6 and page-processing speed.** Text and Order measure normalized edit distance over character sequences and text-block reading order, respectively; TEDS is tree-edit-distance similarity for tables, and CDM is character detection matching for formulas. Overall averages the three recognition axes. Page speed includes each system's layout stage. All results are our measurements; speed protocols are in Appendix D. †These modes use extra draft parameters; TPF counts tokens accepted per base-model verification, excluding drafting computation.

		Quality					Speed		
Model	Decode	Text↓	TEDS↑	CDM↑	Order↓	Overall↑	TPF↑	tok/s↑	pages/s↑
External parsers									
MinerU2.5	AR	0.045	0.881	0.959	0.130	93.18	1.0	559	0.407
MinerU2.5-Pro	AR	0.037	0.935	0.969	0.124	95.57	1.0	545	0.399
MinerU-Diffusion	diff	0.073	0.853	0.916	0.154	89.87	5.2	79	0.059
PaddleOCR-VL-1.5	AR	0.042	0.919	0.969	0.129	94.86	1.0	741	0.389
dots.ocr	AR	0.048	0.865	0.917	0.140	91.14	1.0	149	0.116
DeepSeek-OCR-2	AR	0.049	0.859	0.933	0.144	91.42	1.0	24	0.022
HunyuanOCR-1.5	AR	0.036	0.952	0.949	0.125	95.52	1.0	390	0.313
HunyuanOCR-1.5	DFlash	0.036	0.952	0.949	0.125	95.52	9.9†	720	0.579
GLM-OCR (base)	AR	0.040	0.934	0.970	0.141	95.48	1.0	807	0.571
GLM-OCR (base)	MTP	0.040	0.934	0.970	0.141	95.48	3.7†	667	0.472
Ours									
GravityOCR	AR	0.040	0.928	0.967	0.141	95.16	1.0	794	0.554
GravityOCR	**self-spec**	0.040	0.928	0.967	0.141	95.16	**9.7**	**1,047**	**0.730**

We use $B = 32$ for training and inference. We train on 10.8M predominantly English region crops with a 60/20/20 text/table/formula mixture, using ground-truth annotations and targets generated by the base GLM-OCR. We jointly train the model for 40,000 steps and then apply GRPO for 500 steps. More details are in Appendices A and B.

Evaluation. We report OmniDocBench v1.6 quality on all 1,651 pages using the official metrics and aggregation (Ouyang et al., 2025; Wang et al., 2026). We also evaluate the PubTabNet validation set with full and structure-only TEDS (Zhong et al., 2020), and UniMER-Test with CDM (Wang et al., 2024a;b). We compare the original GLM-OCR under AR and MTP decoding with GRAVITYOCR under AR, direct block-diffusion, and self-speculative decoding. External systems in Tables 1 and 5 are evaluated using their released implementations (Niu et al., 2025; Wang et al., 2026; Dong et al., 2026; Cui et al., 2026; rednote hilab, 2025; Wei et al., 2026; Li et al., 2026).

Efficiency Measurement. We evaluate efficiency in terms of decoding parallelism and inference speed. Tokens per forward (TPF) is the total number of committed output tokens divided by the number of decoding forward passes, counting drafting and verification separately and excluding vision encoding and prompt prefill. We also report the average number of draft tokens accepted by the verifier per round. Since our training data are predominantly English, we measure inference speed on English OmniDocBench pages and their region crops, reporting throughput in tokens per second and pages per second. The speed measurement setup for each experiment is described in Appendix D.

4.2 MAIN RESULTS

OmniDocBench. Table 1 compares GRAVITYOCR with the original GLM-OCR model and representative document parsing systems on OmniDocBench v1.6, reporting recognition quality and page-processing speed side by side. GRAVITYOCR achieves an Overall score of 95.16, compared with 95.48 for the original GLM-OCR, retaining its document parsing quality within 0.32 points. Relative to the base model, TEDS and CDM drop only slightly. Apart from the original GLM-OCR, the Overall score of GRAVITYOCR is surpassed only by MinerU2.5-Pro and HunyuanOCR-1.5 among the evaluated parsers.

Page Processing Speed. The speed columns of Table 1 are measured on a 100-page subset of OmniDocBench, using a single H100 with one page in flight. Each model is evaluated through its released page-processing pipeline, with layout analysis, region recognition, and output assembly

Table 2: **AR vs. self-speculative decoding.** Decode-only and end-to-end throughput under Transformers and SGLang.

		tok/s↑		
Backend	Decode	decode-only	end-to-end	speedup
Transformers	AR	53	51	—
Transformers	self-spec	293	161	5.53× / 3.20×
SGLang	AR	777	486	—
SGLang	self-spec	**3057**	**844**	3.94× / 1.74×

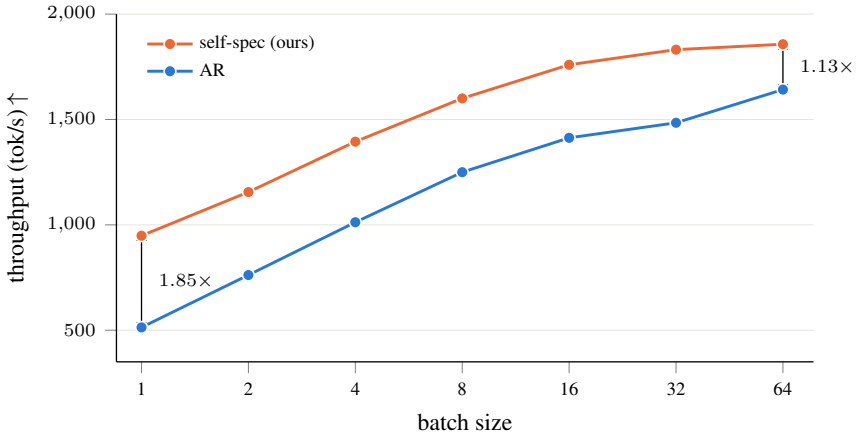


Figure 5: **Throughput vs. batch size.** The speedup narrows with batch size: speculation leads by 1.85× at batch 1 and by 1.13× at batch 64. Protocol in Appendix D.

included in the wall-clock measurement. With self-speculative decoding, GRAVITYOCR commits an average of 9.7 output tokens per forward and achieves the highest processing rates in the table: 1,047 tok/s and 0.730 pages/s. This corresponds to a 1.32× page-processing speedup over causal AR decoding with the same checkpoint. In contrast, the original GLM-OCR's built-in MTP mode remains slower than its causal AR mode despite advancing 3.7 tokens per forward.

4.3 DECODING EFFICIENCY

Inference Backend and Measurement Scope. Table 2 compares AR and self-speculative decoding using the same checkpoint in Hugging Face Transformers (Wolf et al., 2020) and SGLang (Zheng et al., 2024). Decode-only timing covers token generation; end-to-end timing also includes vision encoding and prompt prefill. Our self-speculative decoding achieves decode-only speedups of 5.53× in Transformers and 3.94× in SGLang, providing substantial gains even with SGLang's faster AR baseline. By committing multiple tokens per draft–verification round, it reduces the number of sequential decoding forward passes. Each speculative forward, however, processes more token positions than a one-token AR forward. The resulting speedup therefore depends on the backend's per-forward execution cost, not only on the number of tokens committed. Even with vision encoding and prompt prefill included, self-speculative decoding achieves speedups of 3.20× and 1.74×, respectively.

Throughput Scaling with Batch Size. Batching can improve GPU utilization for causal AR decoding, potentially narrowing the relative advantage of self-speculation. Figure 5 evaluates SGLang throughput under increasing numbers of concurrently processed region inputs, including vision encoding, prompt prefill, and token generation. In this sweep, measured before RL, self-speculative decoding leads by 1.85× at batch size one. As the batch size increases, batching exposes more parallel work from the AR decoder to the GPU, improving its utilization and narrowing the advantage of self-speculation. At batch size 64, AR decoding reaches 1,642 tok/s and self-speculative decoding reaches 1,857 tok/s. Although the advantage narrows at higher batch sizes, self-speculative decoding still achieves a 1.13× higher throughput. These results show that self-speculative decoding

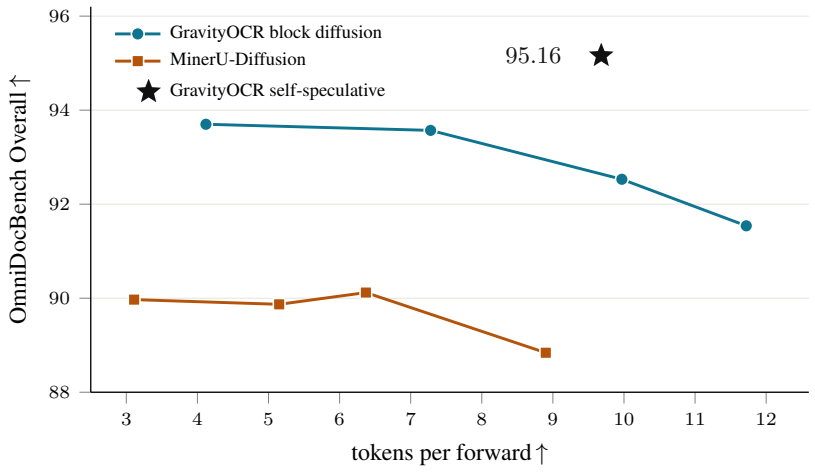


Figure 6: **Quality and parallelism of diffusion decoding.** Curves sweep each system's confidence threshold τ_c. Direct block-diffusion TPF counts forwards that commit tokens, excluding cache writes; self-speculative TPF counts both drafting and verification (Appendix D).

provides its largest throughput gains in low-concurrency settings, where sequential AR decoding leaves more GPU capacity underutilized.

Direct Block-Diffusion vs. Self-Speculative Decoding. Figure 6 tests whether causal verification avoids the accuracy loss from directly committing confidence-selected diffusion predictions. Across the confidence sweep, greater parallelism reduces direct block-diffusion accuracy in both GRAVITYOCR and MinerU-Diffusion, although GRAVITYOCR remains more accurate throughout. At approximately 10 tokens per forward, direct block diffusion achieves 92.53 Overall, whereas self-speculative decoding reaches 95.16 at 9.7 tokens per forward. Because both GRAVITYOCR modes use the same checkpoint, their accuracy gap reflects the decoding procedure rather than different learned parameters.

4.4 EFFECT OF AR-PATH TRAINING

Effect of the AR Loss. We evaluate the effect of the causal AR loss on self-speculative decoding using two 10k-step checkpoints that share the training setup except for the AR loss weight. Even without this loss, the model can perform causal verification because it is initialized from a pretrained AR model. However, verification follows the adapted model's causal predictions, whose accuracy can change as diffusion training updates the shared parameters. Removing AR supervision lowers the OmniDocBench Overall score from 95.02 to 93.64, while TPF remains similar: 6.75 with the AR loss and 6.96 without it. The AR objective therefore helps preserve recognition quality during diffusion adaptation with little change in drafting efficiency.

GRPO on the AR Path. We evaluate whether AR-path GRPO improves OCR quality while preserving diffusion drafting efficiency. Table 3 compares direct diffusion and self-speculative decoding before and after GRPO. Under self-speculative decoding, GRPO improves the OmniDocBench Overall score from 94.92 to 95.16, while TPF remains essentially unchanged at 9.61 and 9.68, respectively. Direct diffusion also retains similar or improved accuracy and similar TPF across the evaluated confidence thresholds, despite using no diffusion-specific objective during GRPO. These results show that AR-path GRPO improves OCR quality while preserving drafting efficiency.

5 RELATED WORK

Document Parsing and Parallel OCR. Generative OCR systems recover text and markup from document images (Blecher et al., 2023; Poznanski et al., 2025; Duan et al., 2026; Wang et al., 2026; Lu et al., 2026; Cui et al., 2025; rednote hilab, 2025; Wei et al., 2025). Recent work also targets decoding cost: GLM-OCR uses an MTP branch, HunyuanOCR-1.5 employs a separate DFlash

Table 3: **GRPO on the AR path.** OmniDocBench v1.6 Overall and TPF for direct block-diffusion and self-speculative decoding before and after GRPO.

		Overall$^{\uparrow}$			TPF$^{\uparrow}$	
Decode	τ_c	before	after	Δ	before	after
Block diffusion	0.7	86.17	86.06	-0.11	14.85	15.43
Block diffusion	0.9	91.55	91.54	-0.01	11.89	11.72
Block diffusion	0.95	89.62	**92.53**	$\mathbf{+2.91}$	10.20	9.97
Block diffusion	0.99	93.23	93.57	$+0.34$	7.18	7.28
Self-spec	—	94.92	**95.16**	$\mathbf{+0.24}$	9.61	9.68

drafter, and HSD verifies drafts at region and page levels (Duan et al., 2026; Li et al., 2026; Chen et al., 2026; Liao et al., 2026). DODO and MinerU-Diffusion instead directly generate OCR outputs with block diffusion (Man et al., 2026; Dong et al., 2026). We use parallel diffusion proposals while retaining causal verification, addressing the dependency-sensitive failures of direct commitment.

Diffusion and AR–Diffusion Hybrids. Masked diffusion recovers text through iterative denoising (Austin et al., 2021; Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025). Block Diffusion and Fast-dLLM make this process more compatible with caching (Arriola et al., 2025; Wu et al., 2025b); Fast-dLLM v2 and SDAR study efficient adaptation from pretrained AR models (Wu et al., 2025a; Cheng et al., 2025). WeDLM uses causal attention and topological token reordering for parallel prediction (Liu et al., 2025a). Closest to our shared-model design, Fast-dVLM jointly trains causal and denoising objectives for VLMs, Nemotron-Labs-Diffusion supports AR, diffusion, and self-speculative modes, and TiDAR integrates diffusion drafting with AR generation through structured attention (Wu et al., 2026a; Fu et al., 2026; Liu et al., 2025b). We adapt this shared-model approach to region-level OCR and evaluate its recognition quality, serving speed, and compatibility with AR-path GRPO.

Diffusion Drafters and Verification. Speculative decoding verifies multiple proposed tokens in one AR forward (Leviathan et al., 2023; Chen et al., 2023). Speculative diffusion decoding and DFlash train separate diffusion drafters; DiffuSpec repurposes a pretrained diffusion model, while SpecDiff-2 improves drafter–verifier agreement (Christopher et al., 2025; Chen et al., 2026; Li et al., 2025; Sandler et al., 2025). Further work aligns drafts with left-to-right verification through position-aware objectives or targeted repair (Wu et al., 2026b; Whalen et al., 2026; Yang & Li, 2026; Liu et al., 2026). Our drafter and verifier share all model parameters, avoiding a separate draft network.

Reinforcement Learning for Diffusion Language Models. Diffusion RL methods approximate trajectory-dependent likelihoods or derive step-level policy gradients (Zhao et al., 2025; Zhan, 2025); self-speculation has also been used to accelerate RL rollouts (Kim et al., 2026). Because our AR verifier determines the output, we apply GRPO (Shao et al., 2024) using its exact causal likelihood and update the shared diffusion drafter without a diffusion-specific objective.

6 Conclusion

We present GravityOCR, a parameter-shared framework that combines block-diffusion drafting with causal AR verification to accelerate document OCR. Joint AR–diffusion training enables a single model to draft token blocks in parallel and verify them before commitment, preserving its AR output while allowing multiple tokens to be committed per round. We further apply GRPO through the causal AR path using sequence- and structure-level OCR rewards, avoiding diffusion-trajectory likelihood estimation while updating the shared drafter and verifier parameters. On OmniDocBench v1.6, this improves the Overall score from 94.92 to 95.16, close to the original GLM-OCR score of 95.48, while preserving diffusion drafting efficiency. In SGLang, GravityOCR achieves a $3.94\times$ decode-only speedup on region crops and a $1.32\times$ end-to-end page-processing speedup over AR decoding. These results show that parameter-shared drafting and verification can greatly accelerate document OCR while largely retaining the base model's recognition capabilities, without a separate drafter network.

AI USE STATEMENT

In this work, we used generative AI tools (large-language-model coding and writing assistants) for drafting and editing manuscript text under the authors' direction, for writing and refactoring the training, evaluation, measurement and plotting scripts, and for producing documentation. We also used the base GLM-OCR model to generate training targets, as described in Appendix A. We have not used generative AI tools to generate the research idea or the method, to design the experiments, or to produce, select, or alter any experimental result. We have reviewed all AI-assisted work: AI-drafted text was edited and fact-checked by the authors against the experimental records, and AI-written code was run and its outputs verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies the decoding efficiency of a document OCR model and does not involve human subjects or personally identifiable data collected by the authors. Training data consist of publicly available document datasets, used under their respective licenses, together with transcriptions produced by the publicly released GLM-OCR model; evaluation uses the public OmniDocBench benchmark under its official protocol. The released model inherits the permissive license of its base model. Document OCR systems can be applied to documents whose owners have not consented to their processing; our contribution does not change what such systems can read, only how fast an existing model decodes, and we release it for the same document-parsing uses as the base model. We are not aware of conflicts of interest beyond the authors' affiliations.

REPRODUCIBILITY STATEMENT

The model is initialized from a public checkpoint and trained on the data described in Section 4.1, with every training hyper-parameter listed in Appendix A (Table 4) and the reinforcement-learning recipe in Appendix B. The two decoding algorithms are specified in Section 3.2 and, with the exact acceptance rule and cache handling, in Appendix C; Appendix C.3 reports the measured agreement between the self-speculative and autoregressive outputs. OmniDocBench results use the official OmniDocBench scoring code and aggregation (Section 4.1); all speed numbers state their page set, hardware, batch size and measurement boundary (Section 4.3, Appendix D), and the fixed-seed subsampling of the training streams is given in Appendix A. Code, model checkpoints, the serving implementation, and the evaluation scripts will be released.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. International Conference on Learning Representations, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. Advances in Neural Information Processing Systems, 2021.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. arXiv preprint arXiv:2308.13418, 2023.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. arXiv preprint arXiv:2302.01318, 2023.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. arXiv preprint arXiv:2602.06036, 2026.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. arXiv preprint arXiv:2510.06303, 2025.

Jacob K. Christopher, Brian R. Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. Speculative diffusion decoding: Accelerating language generation through diffusion. In NAACL, 2025. arXiv:2408.05636.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, et al. PaddleOCR-VL: Boosting multilingual document parsing via a 0.9b ultra-compact vision-language model. arXiv preprint arXiv:2510.14528, 2025.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, Yue Zhang, Yubo Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. Paddleocr-vl-1.5: Towards a multi-task 0.9b VLM for robust in-the-wild document parsing. arXiv preprint arXiv:2601.21957, 2026.

Hejun Dong, Junbo Niu, Bin Wang, Weijun Zeng, Wentao Zhang, and Conghui He. Mineru-diffusion: Rethinking document OCR as inverse rendering via diffusion decoding. arXiv preprint arXiv:2603.22458, 2026.

Shuaiqi Duan, Yadong Xue, Weihan Wang, Zhe Su, Huan Liu, et al. GLM-OCR technical report. arXiv preprint arXiv:2603.10910, 2026.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, et al. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. arXiv preprint arXiv:2607.05722, 2026.

Adam W. Harley, Alex Ufkes, and Konstantinos G. Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In International Conference on Document Analysis and Recognition, 2015.

Minseo Kim, Minjae Lee, Seunghyuk Oh, Kevin Galim, Donghoon Kim, Coleman Hooper, Harman Singh, Amir Gholami, Hyung Il Koo, and Wonjun Kang. Efficientrollout: System-aware self-speculative decoding for rl rollouts. arXiv preprint arXiv:2606.18967, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles (SOSP), 2023. arXiv:2309.06180.

Hugo Laurençon, Andrés Marafioti, Victor Sanh, and Léo Tronchon. Building and better understanding vision-language models: insights and future directions. arXiv preprint arXiv:2408.12637, 2024.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In International Conference on Machine Learning, 2023.

Gengluo Li, Xingyu Wan, Shangpin Peng, Weinong Wang, Hao Feng, Yongkun Du, Binghong Wu, Zheng Ruan, et al. Hunyuanocr-1.5: Making lightweight OCR VLMs faster and better. arXiv preprint arXiv:2607.04884, 2026.

Guanghao Li, Zhihui Fu, Min Fang, Qibin Zhao, Ming Tang, Chun Yuan, and Jun Wang. Diffuspec: Unlocking diffusion language models for speculative decoding. arXiv preprint arXiv:2510.02358, 2025.

Wenhui Liao, Hongliang Li, Pengyu Xie, Xinyu Cai, Yufan Shen, Yi Xin, Qi Qin, Shenglong Ye, et al. Hsd: Training-free acceleration for document parsing vision-language models with hierarchical speculative decoding. arXiv preprint arXiv:2602.12957, 2026.

Aiwei Liu, Minghua He, Shaoxun Zeng, Sijun Zhang, Linhao Zhang, Chuhan Wu, Wei Jia, Yuan Liu, Xiao Zhou, and Jie Zhou. WeDLM: Reconciling diffusion language models with standard causal attention for fast inference. arXiv preprint arXiv:2512.22737, 2025a.

Aofan Liu, Jingxiang Meng, Fangxin Liu, and Yongbiao Chen. Cure: Local uncertainty repair for block-parallel speculative decoding. arXiv preprint arXiv:2608.00531, 2026.

Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. TiDAR: Think in diffusion, talk in autoregression. arXiv preprint arXiv:2511.08923, 2025b.

Shiyin Lu, Yinglun Li, Yu Xia, Yuhui Chen, An-Yang Ji, et al. Ovisocr2 technical report. arXiv preprint arXiv:2607.13639, 2026.

Sean Man, Gilad Deutch, Roy Ganz, Roi Ronen, Shahar Tsiper, Shai Mazor, and Niv Nayman. Dodo: Discrete OCR diffusion models. arXiv preprint arXiv:2602.16872, 2026.

Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter Staar. Tableformer: Table structure understanding with transformers. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. arXiv preprint arXiv:2502.09992, 2025.

Junbo Niu, Zheng Liu, Zhuangcheng Gu, Bin Wang, Linke Ouyang, et al. Mineru2.5: A decoupled vision-language model for efficient high-resolution document parsing. arXiv preprint arXiv:2509.22186, 2025.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, et al. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations. In CVPR, 2025.

Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter Staar. Doclaynet: A large human-annotated dataset for document-layout segmentation. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022.

Jake Poznanski, Aman Rangapur, Jon Borchardt, Jason Dunkelberger, Regan Huff, et al. olmocr: Unlocking trillions of tokens in pdfs with vision language models. arXiv preprint arXiv:2502.18443, 2025.

rednote hilab. dots.ocr: Multilingual document layout parsing in a single vision-language model. https://github.com/rednote-hilab/dots.ocr, 2025.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. Advances in Neural Information Processing Systems, 2024.

Jameson Sandler, Jacob K. Christopher, Thomas Hartvigsen, and Ferdinando Fioretto. Specdiff-2: Scaling diffusion drafter alignment for faster speculative decoding. arXiv preprint arXiv:2511.00606, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.

Brandon Smock, Rohith Pesala, and Robin Abraham. Pubtables-1m: Towards comprehensive table extraction from unstructured documents. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

Ting Sun, Cheng Cui, Yuning Du, and Yi Liu. Pp-doclayout: A unified document layout detection model to accelerate large-scale data construction. arXiv preprint arXiv:2503.17213, 2025.

Bin Wang, Zhuangcheng Gu, Guang Liang, Chao Xu, Bo Zhang, Botian Shi, and Conghui He. Unimernet: A universal network for real-world mathematical expression recognition. arXiv preprint arXiv:2404.15254, 2024a.

Bin Wang, Fan Wu, Linke Ouyang, Zhuangcheng Gu, Rui Zhang, Renqiu Xia, Bo Zhang, and Conghui He. Image over text: Transforming formula recognition evaluation with character detection matching. arXiv preprint arXiv:2409.03643, 2024b.

Bin Wang, Tianyao He, Linke Ouyang, Fan Wu, Zhiyuan Zhao, et al. Mineru2.5-pro: Pushing the limits of data-centric document parsing at scale. arXiv preprint arXiv:2604.04771, 2026.

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

Haoran Wei, Yaofeng Sun, and Yukun Li. DeepSeek-OCR 2: Visual causal flow. arXiv preprint arXiv:2601.20552, 2026.

Lexington Whalen, Yuki Ito, and Ryo Sakamoto. Teaching diffusion to speculate left-to-right. arXiv preprint arXiv:2606.11552, 2026.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45. Association for Computational Linguistics, 2020.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient Block-Diffusion LLM. arXiv preprint arXiv:2509.26328, 2025a.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. arXiv preprint arXiv:2505.22618, 2025b.

Chengyue Wu, Shiyi Lan, Yonggan Fu, Sensen Gao, Jin Wang, Jincheng Yu, Jose M. Alvarez, Pavlo Molchanov, Ping Luo, Song Han, Ligeng Zhu, and Enze Xie. Fast-dvlm: Efficient Block-Diffusion VLM via direct conversion from autoregressive VLM. arXiv preprint arXiv:2604.06832, 2026a.

Tianyu Wu, Yu Yao, Zhenting Qi, Han Zheng, Zhuohan Wang, Haoran Ma, Lawrence Liao, Himabindu Lakkaraju, Ju Li, and Yilun Du. D-pace: Dynamic position-aware cross-entropy for parallel speculative drafting. arXiv preprint arXiv:2605.18810, 2026b.

Renqiu Xia, Song Mao, Xiangchao Yan, Hongbin Zhou, Bo Zhang, Haoyang Peng, Jiahao Pi, Daocheng Fu, Wenjie Wu, Hancheng Ye, Shiyang Wang, Junchi Ye, Bin Wang, Aojun Zhou, Zhen Chen, Qi Zhang, Fu Wang, Chao Chen, Lei Bai, Hongsheng Li, Junchi Yan, and Yu Qiao. Docgenome: An open large-scale scientific document benchmark for training and testing multi-modal large language models. arXiv preprint arXiv:2406.11633, 2024.

Tianjian Yang and Meng Li. Spec-auf: Accept-until-fail training under train-inference misalignment for masked block drafters. arXiv preprint arXiv:2607.01893, 2026.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. arXiv preprint arXiv:2508.15487, 2025.

Anthony Zhan. Simple policy gradients for reasoning with diffusion language models. arXiv preprint arXiv:2510.04019, 2025.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. arXiv preprint arXiv:2504.12216, 2025.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, et al. SGLang: Efficient execution of structured language model programs. In Advances in Neural Information Processing Systems, 2024. arXiv:2312.07104.

Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context. In IEEE/CVF Winter Conference on Applications of Computer Vision, 2021.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. Image-based table recognition: Data, model, and evaluation. In European Conference on Computer Vision, 2020.

Table 4: **Training configuration.**

Hardware	16×H100 (2 nodes)
Steps	40,000 (warmup 500, linear decay to 0)
Optimizer	AdamW (β_1=0.9, β_2=0.999), no weight decay
Peak LR	2×10^{-5} decoder, 2×10^{-6} vision encoder
Precision / parallelism	bf16, ZeRO stage 2, gradient clipping 1.0
Batch	1 packed row/GPU × 16 GPUs × accumulation 4
Packing budget	10,240 forward tokens/row (response cap 3,072)
Block size / mask schedule	B=32, $t \sim \mathcal{U}(0, 1)$
Loss weights	$0.5\,\mathcal{L}_{\text{diff}} + 0.5\,\mathcal{L}_{\text{AR}}$

A Training Details

Table 4 lists the training configuration. All parameters are trained, including the vision encoder, which uses $0.1\times$ the decoder learning rate; the only architectural addition is the [M] embedding, a new vocabulary row initialized by mean-resizing, i.e. drawn from a normal distribution fitted to the existing embedding statistics. The joint loss is normalized so that its coefficients sum to one, $0.5\,\mathcal{L}_{\text{diff}} + 0.5\,\mathcal{L}_{\text{AR}}$ for $\lambda = 1$, leaving learning-rate semantics unchanged.

Training data. The training pool contains 12.3M region-level examples assembled from predominantly public data, including DocGenome (Xia et al., 2024), Docmatix (Laurençon et al., 2024), PubTables-1M (Smock et al., 2022), FinTabNet (Zheng et al., 2021), SynthTabNet (Nassar et al., 2022), PubTabNet (Zhong et al., 2020), RVL-CDIP (Harley et al., 2015), DocLayNet (Pfitzmann et al., 2022), and the training split of UniMER (Wang et al., 2024a). Each example is a layout region cropped from a full page at native resolution, paired with a text target. For full-page sources, regions are detected with the same PP-DocLayout-V3 detector used at inference; table- and formula-only sources are used as whole crops. We use ground-truth annotations provided by the source datasets for table crops when available; all other training targets are generated by the base GLM-OCR. The pool consists of three streams—page text (6.49M regions), tables (2.90M), and formulas (2.89M)—and is predominantly English. To realize a fixed 60/20/20 stream ratio by example count, the table and formula streams are each subsampled once with a fixed seed to 2.16M examples, giving a 10.8M training set that is globally shuffled. We additionally hold out 1,000 pages (4,614 region crops) as an internal validation set. PubTabNet is used through its training split only.

Token count and loss averaging. We train for 40,000 steps, processing $\sim$26B forward tokens in total; this count includes vision tokens and the three response streams of each example. In implementation, we average the AR loss in Equation (1) over supervised clean-stream tokens and the diffusion loss in Equation (2) over masked targets pooled across both corrupted streams, using $w(t) = 1$. We normalize the combined loss by $1 + \lambda$ and use $\lambda = 1$, giving each objective weight 0.5, except in the no-AR-loss ablation in Section 4.4.

Shared-prefix packing. Because the three response streams of Figure 3 share one image and prompt, a training row costs $\text{prompt} + \text{vision} + 3 \cdot \text{response}$ forward tokens, and the packer fills each row against this cost rather than the raw sequence length—the same consideration behind Fast-dVLM's vision-efficient concatenation (Wu et al., 2026a) and Nemotron's asymmetric dual stream (Fu et al., 2026). Rows are filled first-fit with carry-over: a sample that does not fit starts the next batch, and typical rows pack 21–30 document samples.

EOS block fill. Each response is padded with exactly B EOS tokens during training, so the model learns to fill the tail block with an EOS run instead of leaking length information through block alignment. The AR loss counts only the first EOS of each run, while the diffusion streams supervise all of them; without this restriction the AR path over-predicts EOS and Overall drops by more than four points.

B RL DETAILS

Optimization. We apply GRPO with full fine-tuning of the entire model (no LoRA), starting from the jointly trained 40k checkpoint. Each step draws 24 prompts with 28 rollouts per prompt (672 completions); advantages are standardized within each group, the loss is token-level, and clipping is asymmetric (ε_{low}=0.2, $\varepsilon_{\text{high}}$=0.28). A KL penalty with β=10^{-3} anchors the policy to a frozen copy of the initial checkpoint. Training uses a learning rate of 3×10^{-6} with cosine decay, 8-bit AdamW, bf16, and gradient clipping 1.0; checkpoints are saved every 250 steps, and we report step 500 (about 1.6 epochs over the prompt pool). Truncated completions are masked out of the loss. No diffusion-specific objective is applied during GRPO; the drafter changes only through the shared parameters updated by the AR-path loss. On the 8,922 English OmniDocBench crops, tokens per forward are 9.61 before RL and 9.68 at step 500.

Rollouts. Rollouts are sampled from the causal AR policy at temperature 1.0 without nucleus truncation, with a maximum completion length of 8,448 tokens, using a colocated vLLM engine (Kwon et al., 2023). Policy log-probabilities are recomputed by the training framework through the same causal factorization, with token-level importance-sampling correction between the rollout and training kernels.

Rewards. All rewards lie in $[0, 1]$; CDM is not used as a reward. Both the prediction and the reference are first normalized with the same markup normalization the official OmniDocBench scorer applies, so the reward cannot be improved by markup conventions the evaluation ignores. Plain text is scored by the normalized Levenshtein similarity of the two normalized strings. For tables, let TEDS_s be the structure-only TEDS and s_{cell} the normalized Levenshtein similarity of the concatenated cell strings in reading order; the reward is

$$
\begin{aligned}
r_{\text{table}} &= \text{clip}_{[0,1]}\big((0.45\,\text{TEDS}_s + 0.55\,s_{\text{cell}})\,(1-p_{\text{row}})\,(1-p_{\text{loop}})\big),\\
p_{\text{row}} &= 0.35\,\min(1, \text{over}/d) + 0.35\,\min(1, \text{under}/d), \qquad d = \max(4,\, n_{\text{gt}}),\\
p_{\text{loop}} &= \begin{cases} 0 & f = 0,\\ \min(1,\ 0.30 + 0.70\,f) & f > 0, \end{cases}
\end{aligned}
$$

where over and under are the row-count excess and shortfall relative to the n_{gt} reference rows, and f is the fraction of predicted rows that are excess repeats of a row key beyond $\max(3,\, n_{\text{gt}}(\text{key}) + 1)$, zero if no row repeats that often. The result is then raised toward one by an exact-grid bonus, $r \leftarrow r + b\,(1-r)$, where b is 0.05 if the predicted row count matches the reference plus 0.05 if the predicted column count matches. An unclosed `<table>` scores zero (the evaluator drops such tables), and a table that cannot be parsed scores 0.15 times the plain edit similarity. For formulas, the reward is $r_{\text{formula}} = \text{sim}\big(\text{canon}(\hat{y}), \text{canon}(y)\big) \cdot 0.3^{\,v}$, where sim is normalized Levenshtein similarity and canon strips math delimiters, maps `\dfrac` and `\tfrac` to `\frac`, and removes `\left`/`\right`, spacing commands, whitespace, and braces; v counts the well-formedness checks (`\left`/`\right` balance, brace balance, `\begin`/`\end` environments, and unescaped `$` parity) failed by the prediction but not by the reference. Every reward is finally multiplied by a degeneration guard $g = \min\big(\rho_8,\ \max(0,\ 1 - (L_{\text{run}} - 20)/200)\big)$, where ρ_8 is the distinct 8-gram ratio, computed only for outputs of at least 16 whitespace-separated tokens and taken as one otherwise, and L_{run} is the longest single-character run.

Prompt pool. The pool contains 7,723 region crops from public document datasets of the same kind as the training pool; it contains no benchmark images, and the PubTabNet portion uses its training split only. Its composition is 92% tables, 6% text, and 2% formulas. References are the source datasets' original annotations for the table and formula crops (94% of the pool) and teacher transcriptions for the text crops. Candidate prompts were screened by sampling eight rollouts each and discarding prompts whose reward variance is zero (their group advantage vanishes under GRPO), and the remainder was mined toward hard and long tables; the median reference length is about 1,900 characters.

C Decoding Details and Equality Testing

C.1 Direct Block-Diffusion Decoding

The completed prefix is represented by causal KV states, and generation proceeds block by block. Each block is initialized with B [M] tokens; under the shifted convention, the prediction for the block's first position comes from the final position of the preceding block.

At each denoising step, one forward pass predicts all remaining masked positions in parallel, reading the distribution for position p from the logit at position $p-1$. Every prediction whose token probability exceeds the confidence threshold τ_c is revealed; if none exceeds it, the single most confident position is revealed instead. Revealed tokens remain fixed in later steps.

Once the block is complete, one additional forward pass writes its KV states into the cache. This pass extends the cache only and does not verify or replace any token. Generation stops at the block containing the EOS token, and the output is truncated at the first EOS.

C.2 Self-Speculative Decoding and KV-Cache Update

One self-speculative round of the serving implementation proceeds as follows; Figure 4 illustrates the same round.

1. **State.** The KV cache holds causal states for every committed token except the boundary token x_0, the last committed token, which was produced as the verifier's prediction in the preceding round and has not yet been fed to the model as input (the initial boundary is the first token generated during prefill).
2. **Draft forward.** A window is formed from x_0 followed by B [M] tokens. The boundary position attends causally to the cache; the masked positions attend causally to the cache and bidirectionally within the window. One forward pass yields, under the shifted alignment, a_0 from the boundary logit, which is identical to the causal AR prediction, and the draft $d_{1:B}$ from the mask logits. No confidence threshold is applied: the one-shot draft proposes all positions unconditionally.
3. **Verify forward.** The sequence $a_0, d_{1:B}$ is processed under token-level causal attention in a single forward pass, attending to the cache and to x_0, whose causal KV state was written by the draft forward, and producing the AR predictions $a_{1:B+1}$. The verification pass reuses the draft's KV slots for the mask positions and overwrites their bidirectional states with causal ones, so the cache that later rounds read is always causal.
4. **Accept.** $A \leq B$ is the length of the longest prefix with $d_j = a_j$. The round commits a_0, $d_{1:A}$, and a_{A+1}; the committed tokens are the verifier's own predictions at every position.
5. **Cache update.** The causal KV states of x_0, a_0 and $d_{1:A}$ are retained; the states of the rejected suffix are freed. No bidirectional draft state survives the round.
6. **Next round.** a_{A+1} becomes the next boundary token and obtains its KV state in the next draft forward. If $A = B$, the draft was fully accepted and a_{A+1} is the bonus prediction following the entire block. If any committed token completes a stop condition, the committed run is truncated at that token and the remainder of the round is discarded.

A round therefore commits between two tokens (a_0 and a_1) and $B{+}2$ tokens (a_0, the full draft, and the bonus token).

C.3 AR-Equivalence and Equality Testing

Argument. Under top-1 decoding, every committed token is the verifier's argmax prediction given the committed prefix: a_0 is computed under strictly causal attention at the boundary, each accepted draft token satisfies $d_j = a_j$ by the acceptance rule, and a_{A+1} is the verifier's own prediction. By induction over rounds, the committed sequence equals the sequence that standalone greedy AR decoding produces from the same checkpoint, in exact arithmetic.

Empirical check. We compare the decoded output strings of the two decoders under identical checkpoint, prompts, region crops, and greedy decoding: the final GRPO checkpoint, both decoders served with SGLang, block size $B{=}32$, repetition penalty 1.0, and no output cap. On the full English OmniDocBench crop set (8,922 crops), the two outputs are identical on 96.6% of crops. The remaining differences are consistent with near-tie argmax flips under bf16 kernels: the AR and verification forwards use different attention kernels, and at the first divergent position the median top-two logit margin is 0.14 nats. Re-evaluating the 305 first divergences in fp32, the fp32 argmax agrees with the AR token in 228 cases and with the self-speculative token in 77, and 65 of these positions are ties with a margin below 0.05 nats.

D Speed-Measurement Protocol and Profiles

What each speed table measures. All speed tables and Figure 5 use one H100, full resolution, no output cap (natural EOS), and are run solo (no co-tenant job on the node).

Speed columns of Table 1 — page-level, single stream. The unit is one OmniDocBench page and pages are submitted **one at a time** ($C{=}1$). The wall clock covers the **entire pipeline including layout analysis** (112–146 ms/page for the GLM-family systems, measured live inside the wall) plus region cropping, recognition and markdown assembly. Systems differ in what happens inside one page, and we keep each system's own design: region-pipeline models (ours, GLM-OCR) detect $\sim$18 regions and fire those recognition requests concurrently, MinerU runs its own two-step pipeline with a much heavier layout stage ($\sim$1.3 s/page), while full-page models (HunyuanOCR, dots.ocr, DeepSeek-OCR-2) answer the page in a single request and have no layout stage. Each competitor runs on its own official inference stack: HunyuanOCR-1.5 on vLLM (Kwon et al., 2023) with the vendor recipe, MinerU2.5 and MinerU2.5-Pro through their native pipeline over a vLLM engine, dots.ocr on a vLLM server, PaddleOCR-VL-1.5 on the PaddleX generative-AI server with a vLLM backend, DeepSeek-OCR-2 and MinerU-Diffusion through their native Transformers and diffusion pipelines, and the GLM-OCR MTP mode on vLLM; the GLM-OCR AR baseline and our model are served with SGLang.

Concurrency regimes. Same client, pages and layout-included boundary as Table 1, measured before RL; the **only** change is how the $\sim$18 region requests of a page are issued: strictly sequentially ($C{=}1$: self-spec $1.54\times$ over AR, 0.420 vs 0.273 pages/s) or all at once ($C{=}18$: $1.34\times$, 0.781 vs 0.581). This isolates how much of the speculative advantage comes from an otherwise idle GPU.

Table 2 — crop-level, batch 1, no layout. The unit is a single ground-truth region crop from the English OmniDocBench set (8,922 crops), fed directly to the model, so **layout is not involved at all** and there is no page assembly. Here end-to-end is the full request wall (vision encoding + prefill + decode); decode-only counts only the time spent generating output tokens, excluding the per-crop vision encoding and prompt prefill.

Figure 5 — crop-level throughput with varying batch size. Throughput is measured before RL on layout-detected crops from 755 English OmniDocBench pages. Layout is run once beforehand and excluded from timing. We measure the time to process all crops, including vision encoding, prompt prefill, and decoding, with a full client queue ($C = \max(64,\ 4 \times \mathrm{bs})$) and the server's running-request cap set to the plotted batch size. Table 2 instead uses crops defined by the dataset's annotated region boundaries, submitted one at a time, and the final GRPO checkpoint.

Forward-count accounting. For GLM-OCR MTP and DFlash, TPF counts base-model verification forwards and excludes auxiliary drafting computation. For GravityOCR self-speculation, it counts both full-model draft and verification forwards, excluding prefill. For direct block-diffusion decoding, it counts only forwards that commit at least one token, excluding prefill and the cache-write forward performed after each completed 32-token block. This direct-diffusion convention matches the step-count accounting used for MinerU-Diffusion.

E Additional Results

Table and Formula Recognition. Beyond OmniDocBench, Table 5 evaluates GravityOCR on PubTabNet and UniMER, dedicated benchmarks for table and formula recognition, respectively.

Table 5: **Table and formula recognition.** PubTabNet table recognition on the full validation set (9,115 tables) and UniMER formula recognition, all rows measured by us under one pipeline per benchmark, with each system's own task prompt and the same output normalisation.

Model	PubTabNet TEDS$^{\uparrow}$	TEDS-struct$^{\uparrow}$	UniMER CDM$^{\uparrow}$
MinerU2.5	0.854	0.909	0.927
MinerU2.5-Pro	0.866	0.913	0.956
MinerU-Diffusion	0.734	0.858	0.952
PaddleOCR-VL-1.5	0.828	0.900	0.956
dots.ocr	**0.893**	**0.935**	0.910
DeepSeek-OCR-2	0.849	0.896	0.863
HunyuanOCR-1.5	0.852	0.903	0.942
GLM-OCR (base, AR)	0.803	0.858	**0.963**
GravityOCR	0.871	0.916	0.962

Table 6: **Parallelism and speed by content type.** Accepted tokens is the number of draft tokens accepted per round. TPF divides committed output tokens by all draft and verification forwards pooled within each content type.

		mean output	accepted		tok/s$^{\uparrow}$		
Type	crops	tokens	tokens$^{\uparrow}$	TPF$^{\uparrow}$	AR	self-spec	speedup
Text	7,019	73	15.0	8.5	419	647	1.54×
Formula	1,660	95	18.9	10.5	520	947	1.82×
Table	243	872	**25.0**	**13.5**	732	**2,462**	**3.36**×
All	8,922	99	17.4	9.7	486	844	1.74×

TEDS evaluates both table structure and cell content, whereas TEDS-struct isolates the recovered table structure. Compared with the original GLM-OCR, GRAVITYOCR improves TEDS from 0.803 to 0.871 and TEDS-struct from 0.858 to 0.916. On UniMER, GRAVITYOCR achieves a CDM score of 0.962, closely matching the original GLM-OCR score of 0.963 and exceeding all evaluated external systems.

Taken together, these results show that our framework largely preserves the recognition performance of the base model across the evaluated benchmarks while enabling block-diffusion drafting and causal AR verification.

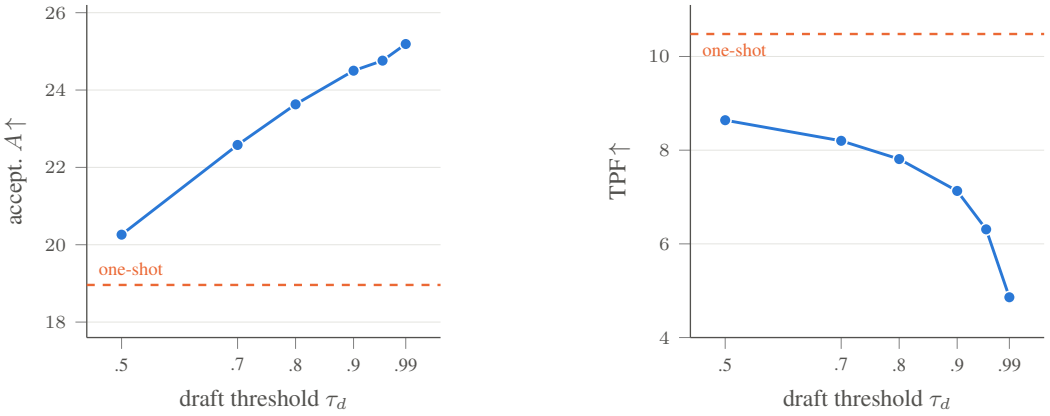
Decoding Efficiency by Output Type. Table 6 shows that table regions benefit more from self-speculative decoding than text regions. They accept 25.0 of the 32 drafted tokens per round, compared with 15.0 for text, and achieve speedups of $3.36\times$ and $1.54\times$, respectively. Table outputs are also longer, averaging 872 tokens compared with 73 for text, which reduces the relative cost of vision encoding and prompt prefill. Their higher draft acceptance may reflect the stronger syntactic constraints of structured outputs.

Speedup by Output Length. Table 7 breaks the crop-level comparison of Table 2 down by output length. Both acceptance and speedup rise monotonically with length, from $1.43\times$ below 128 output tokens to $3.68\times$ above 1,024. Two effects compound. Longer outputs spend more of their wall-clock time in decoding, which is the only stage speculation accelerates, so the fixed per-crop vision and prefill cost stops diluting the gain. At the same time, acceptance itself grows with length, from 14.8 of 32 drafted tokens per round below 128 output tokens to 23.9 above 1,024—long regions are dominated by structured, syntax-constrained spans that the one-shot draft predicts well—so each verification also commits more tokens. Four fifths of OmniDocBench crops fall in the shortest bin, which is why the crop-averaged speedup of $1.74\times$ sits well below the longest bin. This length dependence may partly explain the larger speedups observed for table regions in Table 6.

Draft Schedule. Figure 7 sweeps the number of denoising steps used to build the draft, controlled by the draft-side confidence threshold τ_d, measured on 400 English OmniDocBench crops. Refining

Table 7: **Parallelism and speed by output length.** AR and self-speculative decoding results across output-length ranges. n is the number of crops in each range. Throughput includes vision encoding, prompt prefill, and decoding; speedup is relative to AR.

Output length	n	accepted tokens$^{\uparrow}$	TPF$^{\uparrow}$	AR tok/s	self-spec tok/s	speedup
<128	7,257	14.8	8.4	357	510	1.43×
128–511	1,496	18.1	10.1	583	1,176	2.02×
512–1023	92	18.4	10.2	737	2,033	2.76×
≥ 1024	77	**23.9**	**12.9**	761	**2,805**	**3.68**×



(a) Accepted draft tokens per round

(b) Tokens per forward

Figure 7: **Multi-step drafting versus one-shot drafting.** Instead of filling the masked block in a single forward pass, the draft is refined over several denoising steps: at each step the positions whose confidence exceeds the threshold τ_d are fixed and the rest are re-drafted, and the finished draft is then verified using the same causal AR acceptance rule. (a) A higher threshold spends more steps on the draft and more of its tokens are accepted per round. (b) Every extra step is another forward pass, and the tokens it adds do not pay for it: tokens per forward falls monotonically, and one-shot drafting (dashed) commits the fewest tokens per round yet delivers the highest tokens per forward.

the draft over more steps raises the accepted length per round (Figure 7(a)) but costs additional draft forwards, and the extra forwards cost more than the tokens they buy: tokens per forward falls monotonically as the draft is refined (Figure 7(b)). One-shot drafting is therefore the operating point we deploy.

Direct-Diffusion Quality. Table 3 also reports the diagnostic direct-diffusion mode of Appendix C.1 on the official protocol at four confidence thresholds. After GRPO, quality rises monotonically with τ_c while parallelism falls; even the best direct-diffusion operating point trails self-speculative decoding on both axes simultaneously.

Mask Schedule. Table 8 compares the uniform partial-masking objective against an all-mask variant matched in every other training setting, evaluated at the same step counts on the self-speculative path. The all-mask model is behind on quality at five of six checkpoints and commits fewer tokens per forward, indicating that the partial-masking objective, not merely exposure to masked positions, is what makes the draft strong.

The τ_c=0.95 Gap After GRPO. In Table 3, direct diffusion gains +2.91 Overall at τ_c=0.95 while every other threshold moves by less than 0.4. The pre-GRPO model degrades three times as many formulas at exactly that setting—116 versus 37 expressions scoring zero CDM out of 2,352—which is also why its accuracy is non-monotone in τ_c while the post-GRPO curve is not.

Table 8: **All-mask training does not produce a better draft.** The all-mask model commits fewer tokens per forward at every checkpoint, and its accuracy is behind at five of six.

tokens per forward$^{\uparrow}$ (self-speculative)	2k	4k	5k	6k	8k	10k
All-mask	4.04	4.82	5.54	5.94	6.45	6.66
Uniform t (ours)	**4.64**	**5.30**	**6.04**	**6.00**	**6.78**	**6.75**
OmniDocBench Overall$^{\uparrow}$	2k	4k	5k	6k	8k	10k
All-mask	94.49	94.59	94.75	**94.66**	94.76	94.86
Uniform t (ours)	**94.71**	**94.74**	**94.76**	94.55	**94.92**	**95.02**

F Qualitative Examples

Figures 8 to 10 show self-speculative decoding with the final GRPO checkpoint on OmniDocBench crops of the three region types. For each crop we show the raw serialized output with every character shaded by the decoding round that committed it; one round is one bidirectional draft forward over the masked block followed by one causal verify forward, so the number of shaded segments is the number of forward pairs that produced the output. A red tick marks a position where the verifier rejected the draft and committed its own token instead. Every round commits a_0, the accepted draft prefix and the verifier's own next token, so each round advances by at least two tokens; <EOS> marks the end-of-sequence token, shaded with the round that emitted it. Each figure carries a legend of the rounds it contains, and each block is headed by its number of rounds and tokens. For all crops shown, the self-speculative output is byte-identical to autoregressive greedy decoding of the same weights.

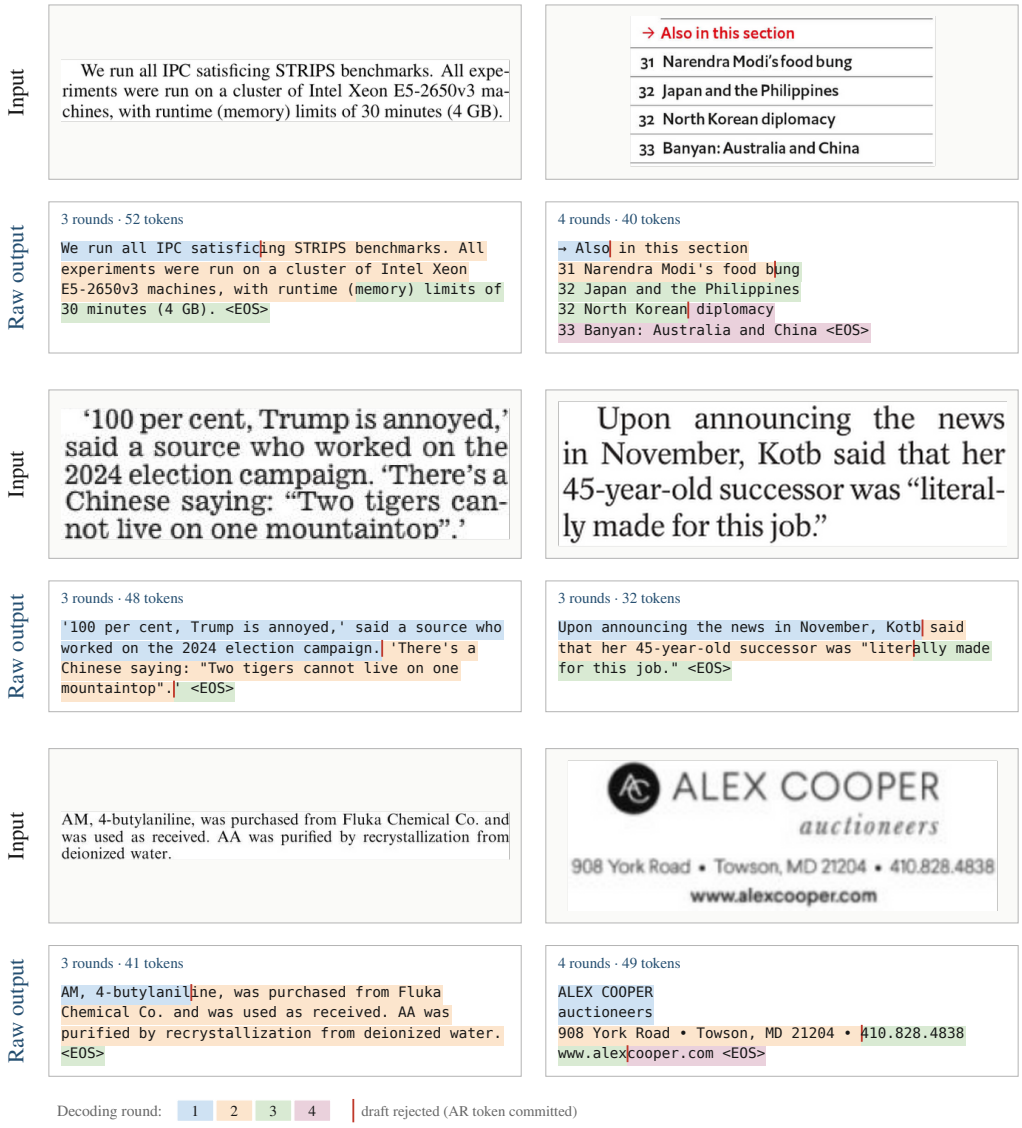


Figure 8: **Qualitative results on text regions.** Self-speculative decoding on OmniDocBench. Each block shows the input crop (top) and the raw output (bottom); each color denotes a different decoding round, a red tick marks a rejected draft token, and <EOS> is the end-of-sequence token.

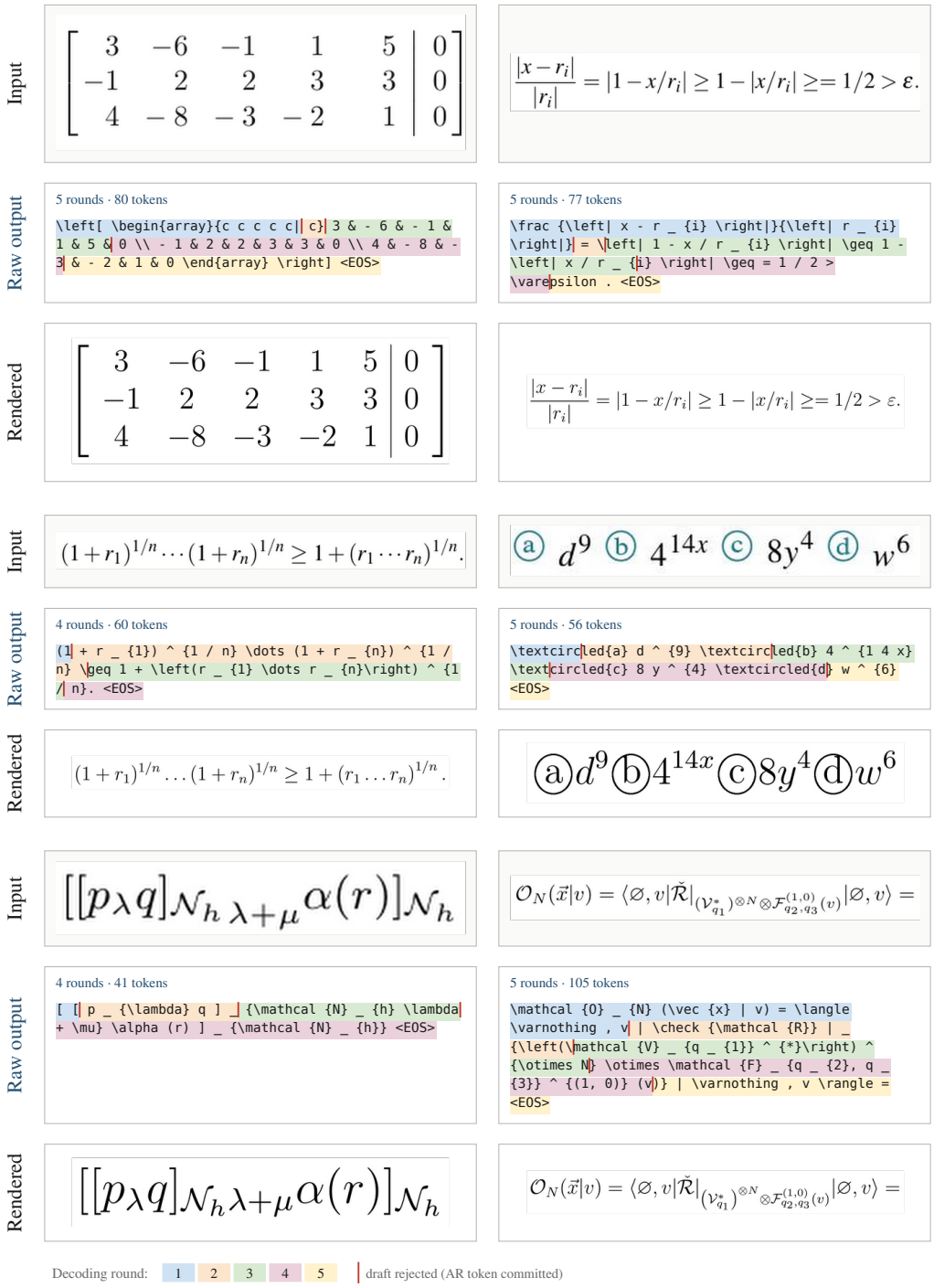


Figure 9: **Qualitative results on formula regions.** Self-speculative decoding on OmniDocBench. Each block shows the input crop, the LaTeX output and its rendering; each color denotes a different decoding round, a red tick marks a rejected draft token, and <EOS> is the end-of-sequence token.

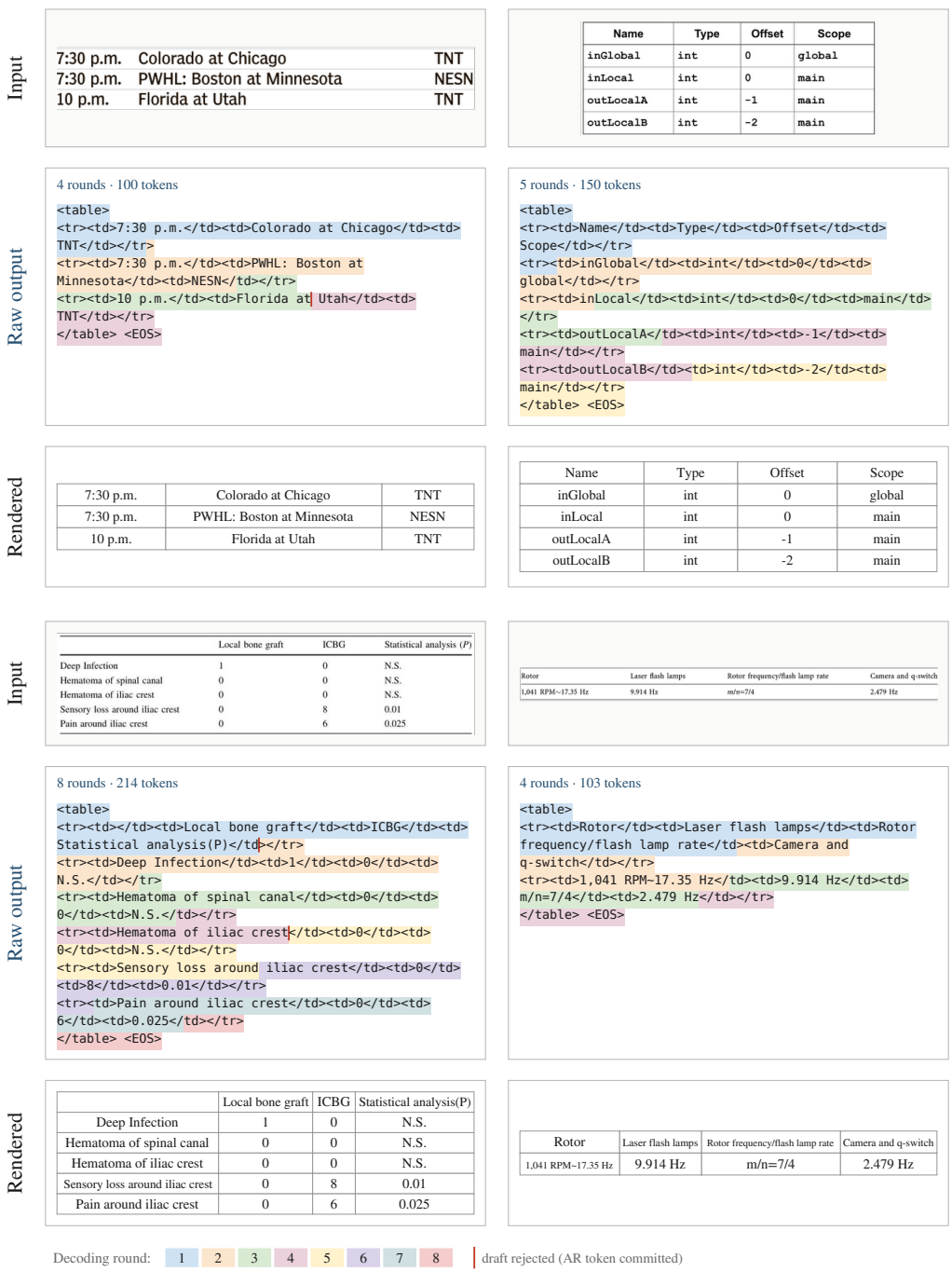


Figure 10: **Qualitative results on table regions.** Self-speculative decoding on OmniDocBench. Each block shows the input crop, the HTML output and its rendering; each color denotes a different decoding round, a red tick marks a rejected draft token, and <EOS> is the end-of-sequence token.