RIPPLEVLM: STRUCTURED VISUAL EXTRACTION VIA BLOCK DIFFUSION AND PROJECTOR ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal extraction needs JSON that software can parse and values that match the page. Diffusion vision language models are well suited to that contract: tokens that are already known can be written into the response and held fixed, while unknown positions are filled by denoising, with bidirectional context inside each block. For a fixed-key record the whole object skeleton is known before generation. Template-Anchored Diffusion Decoding (TADD) therefore places field names and delimiters in a response template, denoises only the bounded value spans, and serializes those values into JSON. The returned object contains the requested keys because it was assembled from the template, and a generation cap cannot cut the object off mid-structure. Diffusion-Aligned Visual Projection (DAVP) adapts only the visual projector to this template, yielding models we call RippleVLM. On multiple datasets, RippleVLM-8B returns valid JSON with every required key in 2.50 s and raises content token F1 from 40.3% to 55.0%. A autoregressive model of same size baseline with Outlines reaches 58.4% F1 but passes the same JSON check at 38%, in 4.62 s under a 256-token cap. As the schema grows from 2 to 8 fields, TADD stays at 100% JSON validity, including on DiffusionGemma-26B. These results show that fixed-template diffusion decoding produces parseable JSON for bounded extraction while leaving content accuracy as a learned problem.

1 INTRODUCTION

Document extraction connects visual recognition to software that expects a specific output format. It has two distinct requirements: the returned record must satisfy an interface, and its values must reflect the document. A valid JSON object can contain incorrect text, and an incomplete object can contain accurate text, so structure and content must be measured separately.

Autoregressive (AR) VLMs emit structured responses token by token. Grammar-constrained decoding restricts each continuation to a target language (Willard & Louf, 2023; Dong et al., 2024; Geng et al., 2023), but the model still generates the structure, and a finite generation cap can interrupt a valid prefix before the object closes. Diffusion language models (Austin et al., 2021; Sahoo et al., 2024; Nie et al., 2025) change this: known parts of a response can be written before generation and unknown positions filled by denoising. Recent work enforces regular and context-free constraints during diffusion sampling (Suresh et al., 2025; Mündler et al., 2025). For fixed-key extraction a simpler observation applies: the entire structure is known in advance, so it need not be sampled at all.

We study this interface in block-diffusion VLMs, which denoise bidirectionally within a block and process blocks causally (Arriola et al., 2025; Chen et al., 2026). In Template-Anchored Diffusion Decoding (TADD), an application supplies field names and per-field token budgets; the decoder receives a template whose structural positions are fixed and whose value positions are masked; after denoising, a serializer assembles the values under the requested keys. This separates a deterministic output contract from the learned extraction problem. The contract is deliberately narrow: fixed-key objects with string values, not arbitrary JSON Schema.

An off-the-shelf diffusion VLM is not trained for this input–output arrangement. Diffusion-Aligned Visual Projection (DAVP) therefore updates only the visual projector, freezing the visual encoder and language backbone, and trains it to predict value tokens from the image and the fixed template. We call the adapted models RippleVLM.

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.

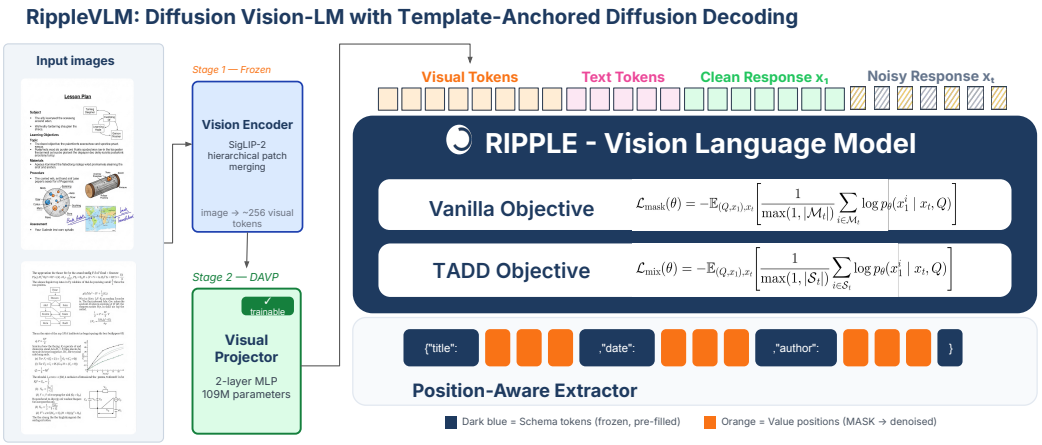

Figure 1: **RippleVLM overview.** TADD fixes structural tokens and denoises value spans; DAVP trains the visual projector while freezing the remaining model. Position-based extraction supplies the values for JSON serialization. The schematic illustrates the architecture and masking objectives; checkpoint-specific dimensions vary, and Equation 1 specifies the implemented training loss.

Our contributions are threefold. (i) We specify a template-based decoding interface for diffusion VLMs and state exactly which structural property it guarantees and which it does not. (ii) We evaluate projector-only adaptation on five Bard-VL variants and a canvas-diffusion model with paired document-cluster uncertainty: the gain is large and resolved at 8B (+14.8 F1) and unresolved elsewhere. (iii) We characterize where the interface helps and where it does not: AR completion under a length cap, value-budget saturation, schema size, and regressions on general benchmarks.

2 Related Work

Structured generation. Grammar-constrained decoding masks invalid continuations of an AR model: Outlines compiles constraints to finite-state machines (Willard & Louf, 2023), XGrammar integrates efficient grammar execution with serving (Dong et al., 2024), and Geng et al. (2023) apply grammar constraints to structured NLP tasks without finetuning. JSONSchemaBench evaluates such frameworks for efficiency, coverage, and output quality (Geng et al., 2025). For diffusion LLMs, DINGO enforces regular-expression constraints with a distribution-preserving dynamic program (Suresh et al., 2025), Mündler et al. (2025) extend constrained decoding to context-free grammars, and Cardei et al. (2025) integrate differentiable constraint optimization into discrete diffusion sampling. These methods act on the sampling process and support broad constraint classes. TADD addresses the narrower fixed-key case, where structural tokens are written into the template and never sampled, and pairs it with visual-projector adaptation. We do not assume that grammar execution dominates AR cost, and our capped Outlines baseline is not an estimate of the best achievable AR performance.

Diffusion language and vision–language models. Discrete and masked diffusion models (Austin et al., 2021; Sahoo et al., 2024) now scale to large language models (Nie et al., 2025); block diffusion interpolates between AR and global diffusion (Arriola et al., 2025), and KV caching with parallel decoding reduces inference cost (Wu et al., 2025). LLaDA-V (You et al., 2025), MMaDA (Yang et al., 2025a), and Dream-VL (Ye et al., 2025) extend diffusion generation to multimodal understanding. Dimple uses structural priors to control response format and length (Yu et al., 2025). BARD converts AR VLMs into block-diffusion VLMs through progressive block merging and distillation (Chen et al., 2026); SDAR and SDAR-VL study related blockwise designs (Cheng et al., 2025a;b); DiffusionGemma denoises a 256-token canvas (DiffusionGemma Team et al., 2026).

Document understanding and visual adaptation. OCR-free document models map page images directly to structured text (Kim et al., 2021; Lee et al., 2022), and layout-aware pretraining combines text, layout, and image signals (Huang et al., 2022). General VLMs such as Qwen3-VL (Bai et al.,

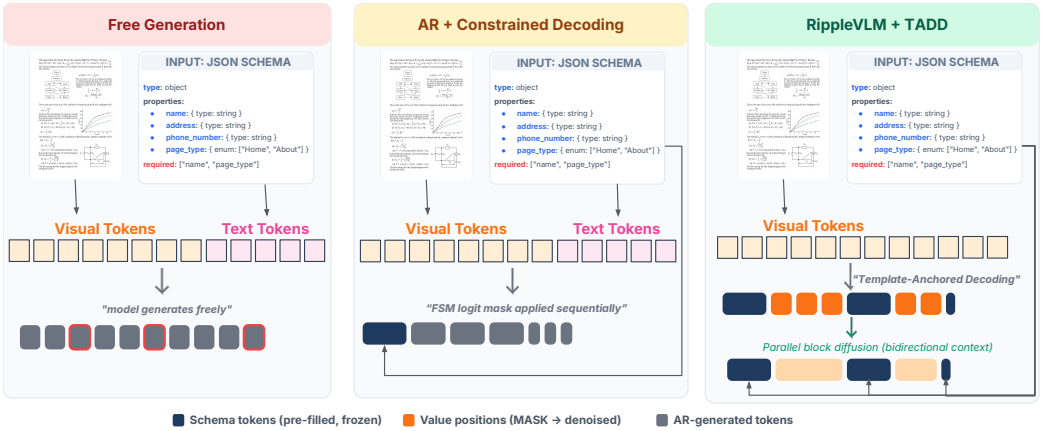

Figure 2: **Three interfaces for structured extraction.** Free generation predicts the entire response; constrained AR generation restricts each sequential continuation; TADD supplies structural tokens before denoising the value spans. The schema is illustrative: the evaluated implementation supports fixed keys with string values, rather than all types shown in the diagram.

2025), whose architecture Bard-VL inherits, now perform such tasks through prompting. Freezing most of a VLM reduces the optimized parameters but does not by itself preserve capabilities: a changed projector alters every visual input to the language model. We therefore report VLMEvalKit results (Duan et al., 2024) alongside extraction, including regressions.

3 Method

Figure 1 summarizes the visual interface and training scope; Figure 2 compares the decoding interfaces. We first define the output contract, then describe decoding and projector adaptation.

3.1 Task and Output Contract

An application specifies a fixed-key string schema $\mathcal{K} = \{k_1, \ldots, k_K\}$ and positive token budgets $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_K)$. Given image I, the model predicts strings $\hat{v}_1, \ldots, \hat{v}_K$. We evaluate the returned JSON object separately from the text content of these values. Dates and numbers are represented as strings in the current implementation; type validation, nullable fields, and variable-cardinality arrays are outside the evaluated interface.

The template has $n = s + \sum_j \ell_j$ tokens, where s is the tokenizer-dependent number of structural tokens. Let $\mathcal{S}$ denote the structural *positions* and $\mathcal{V}_j$ the contiguous value positions associated with field j. This partition is determined during template construction, not by token identity: the same token can occur in a key and in a value. Every $i \in \mathcal{S}$ is initialized to a known token, and every $i \in \bigcup_j \mathcal{V}_j$ is initialized to a mask.

3.2 Template-Anchored Diffusion Decoding

TADD partitions the template into $M = \lceil n/B \rceil$ blocks of size B. Within the active block, masked positions are predicted with bidirectional attention; preceding blocks provide a causal context. Structural positions remain fixed. The Bard-VL implementation suppresses a static set of JSON-breaking tokens, EOS, and the mask token at masked positions, then commits sufficiently confident predictions, falling back to a scheduled number of highest-confidence predictions. A block can finish before its maximum denoising-step count when no masks remain. A finalized block is added to the KV cache before the next block is processed. Algorithm 1 states the procedure in full.

Serialization is part of the method. After decoding, each $\mathcal{V}_j$ is read by position, decoded into a string, and stored under k_j. The implementation serializes this dictionary as JSON. Thus the output

Algorithm 1 Template-Anchored Diffusion Decoding (TADD) with block diffusion. Successful template construction and value extraction are preconditions for the structural invariant.

Require: Image I; schema $\{(k_j, \ell_j)\}$; block size B; denoising steps D

Encode $\hat{\mathbf{V}} \leftarrow g_p(f_v(I))$ ▷ frozen visual encoder; DAVP-trained projector g_p
Build template: place concrete tokens for $\{k_j : \underbrace{[\mathrm{M}] \cdots [\mathrm{M}]}_{\ell_j}\}$ in the response slot
$\mathcal{M} \leftarrow \{i : \text{response}[i] = [\mathrm{M}]\}$ ▷ value positions only
Prefill prompt + visual tokens into the KV cache
for each response block $\mathbf{b}_k$ (left to right) **do**
    Initialise $\mathbf{b}_k^{(D)} \leftarrow \text{template}[k]$ ▷ schema tokens set; values are M
    for $d = D, D{-}1, \ldots, 1$ **do**
        $\boldsymbol{\ell} \leftarrow f_\theta(\mathbf{b}_k^{(d)}, \mathbf{b}_{<k}, \hat{\mathbf{V}})$ ▷ bidirectional within block, causal across
        Commit positions in $\mathcal{M} \cap \mathbf{b}_k$ above confidence τ
    end for
    Append $\mathbf{b}_k$ to the KV cache
end for
return $\{x_i : i \in \mathcal{M}\}$ ▷ value tokens; schema tokens unchanged

shown to the evaluator is a serialized object, not necessarily the raw token sequence. This distinction matters when comparing validity with a baseline evaluated on its raw generated text.

Conditional structural invariant. If template construction yields exactly one value span per requested key, decoding and extraction complete successfully, and all values are strings accepted by the serializer, the returned object is valid JSON containing those keys. The argument follows directly from dictionary construction and serialization. It does not establish that a value is nonempty, correctly bound, supported by the image, or valid under a richer schema. It also does not imply that every implementation run succeeds: the saved B32 variant has structural failures (Section 5.2).

Scope of conditioning. Known field names are available where the block attention mask permits access. A value can attend to keys in its own block and preceding blocks; it cannot attend to all future schema tokens across causal block boundaries. AR generation can also condition on field names in its generated prefix. The distinction is the arrangement of fixed and unknown positions, not the presence or absence of schema context.

3.3 Diffusion-Aligned Visual Projection

Let f_v be the frozen visual encoder, g_ϕ the trainable projector, and f_θ the frozen language backbone. Training pairs are converted to heading/body targets and placed in the template's value slots, truncating to the slot budget. The Bard-VL trainer presents *all* value slots as masks. It minimizes cross-entropy at those positions, ignoring EOS-padded targets:

$$\mathcal{L}_{\mathrm{DAVP}}(\phi) = -\mathbb{E}_{(I,y)} \frac{1}{|\mathcal{V}^+|} \sum_{i \in \mathcal{V}^+} \log p_{\theta,\phi}\big(y_i \mid T_{\mathcal{S}}, [\mathrm{M}]_{\mathcal{V}}, I; A_B\big), \tag{1}$$

where $\mathcal{V}^+$ excludes EOS-padded targets and A_B is the block-causal attention mask. The Bard-VL implementation uses shifted response logits; the canvas trainer uses its backbone-specific output alignment. This is a supervised fully masked value-prediction objective, not a diffusion ELBO sampled over random mask rates. During training, preceding response blocks also contain masked values, whereas inference caches their predictions. Sharing the schema layout therefore does not eliminate every training–inference mismatch.

Only the visual merger and DeepStack merger parameters are optimized for Bard-VL; DiffusionGemma uses its visual embedding projection. Projector-only optimization is a restricted adaptation strategy, not proof that all extraction errors originate in the projector. We use the term "alignment" to describe adaptation to the task and decoding interface, rather than a measured geometric property of the representation.

3.4 COMPUTE AND LENGTH CONTROL

With D_b denoising evaluations in block b, the approximate runtime is

$$T_{\mathrm{TADD}} = T_{\mathrm{preprocess}} + T_{\mathrm{prefill}} + \sum_{b=1}^{M} \left(D_b \, c_{\mathrm{denoise}}(B, L_b) + c_{\mathrm{cache}}(B, L_b) \right) + T_{\mathrm{serialize}}, \quad (2)$$

where L_b is the cached context length. Our implementation performs an additional cache-update forward pass after a completed block, except when its stopping condition is reached. The useful comparison with n autoregressive decoding steps is therefore the actual number and cost of model evaluations. For fixed B and D, both methods remain linear in output length in a simplified step-count model; a block forward pass is not assumed to cost the same as a single-token pass.

Prefilled structure saves predicting known tokens, but those positions can still incur attention and cache costs. Static token suppression also has a cost. DAVP changes weights without changing the inference graph, yet its effect on confidence can change the number of denoising iterations. Component-level timings and matched-quality comparisons are needed to attribute a measured speed difference to these mechanisms.

4 EXPERIMENTAL PROTOCOL

Evaluation data. Pages come from PDFA (Pixparse, 2024). The primary cohort contains 50 pages shared by all Bard-VL, Qwen3-VL, and Dream-VL runs, selected by sorting the local image paths and shuffling with seed 42. The DiffusionGemma runs use 50 different pages with zero overlap with the primary cohort, so we report them separately. Evaluation pages are drawn from the same local PDFA cache that supplies projector-training pages, and the trainer applies no page-level hold-out; Section 6 discusses the consequences.

Configurations. The main TADD setting uses two string fields, heading and body, with $\ell = 96$ tokens per field. Bard-VL uses at most 32 denoising steps per block and confidence threshold 0.5. B8/B16/B32 denote distilled 4B variants with the stated inference block sizes; the non-distilled 4B and 8B variants use $B = 32$ in the evaluation code. Qwen3-VL uses greedy decoding and a 256-token generation cap. Bard-VL free generation is capped at 128 tokens. Thus free-generation and TADD rows are different operating points, and the comparison is not matched for generated length. Checkpoint identifiers and source-file mappings appear in Appendix B.

Training data. The projector trainers combine PDFA OCR pages with CoSyn synthetic chart and diagram images (Yang et al., 2025b) obtained from FineVision (Wiedmann et al., 2025), using AdamW, cosine scheduling, weight decay 0.01, and bfloat16. Appendix B lists the training configuration and the executable defaults.

Metrics. *JSON + keys* is the fraction of outputs that pass the structural check: parse the output directly, or parse a brace-delimited object found by a regular-expression fallback, and test for all requested keys. It is not full JSON Schema validation and does not reject empty values or additional keys. *Content token F1* is set F1 between lowercased whitespace tokens of extracted values and full-page OCR text. If parsing fails, the scorer uses the output text itself. Consequently F1 can be nonzero for invalid records; it ignores order, repetition, and field binding. *Latency* is wall-clock time per evaluation call, including input preparation inside the timer and excluding model loading, measured on a single NVIDIA RTX 6000 GPU unless stated otherwise.

Aggregation and uncertainty. Every extraction mean is recomputed from per-image rows. For DAVP, image IDs are matched before computing differences. We report 95% percentile intervals from 10,000 paired document-cluster bootstrap resamples (seed 42; 48 source-document IDs in each 50-page cohort). These intervals describe variation within the observed sample; they do not account for training seeds, split contamination, or run-to-run timing variation. General VLM results are benchmark-level aggregates without per-example uncertainty.

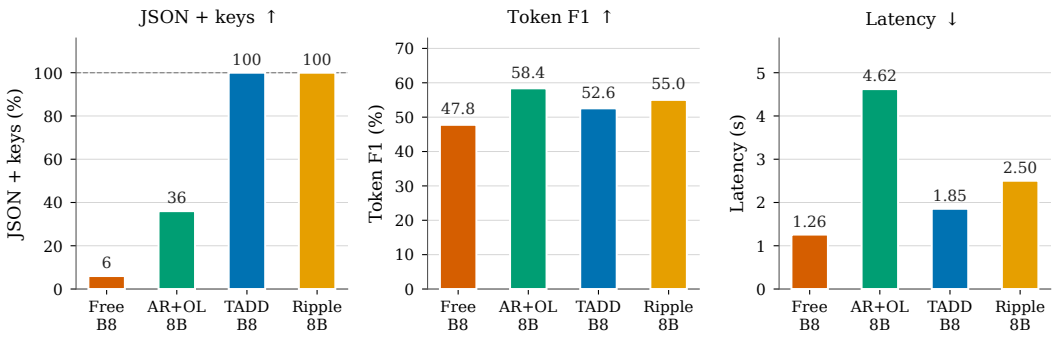

Figure 3: **Structured extraction on the primary 50-page cohort.** Free generation and TADD use Bard-VL-4B-B8; AR+OL denotes Qwen3-VL-8B with Outlines; Ripple denotes RippleVLM-8B. The three panels report structural validity, content F1, and recorded latency. AR uses a 256-token cap; TADD allocates 96 tokens per field. Full configurations appear in Table 4.

Table 1: Extraction results, recomputed from saved per-image rows ($N = 50$ per cohort). DiffusionGemma uses a separate cohort. TADD rows use $K = 2$, $\ell = 96$. The 8B RippleVLM uses $B = 32$. All quantities are means; no latency normalization is applied.

System	JSON + keys (%)	F1 (%)	Latency (s)
Qwen3-VL-8B, free	28	61.8	4.93
Qwen3-VL-8B, Outlines	36	58.4	4.62
Bard-VL-8B, TADD	100	40.3	1.90
RippleVLM-8B, TADD	100	55.0	2.50
Bard-VL-4B-B8, TADD	100	52.6	1.85
RippleVLM-4B-B8, TADD	100	53.9	2.02
RippleVLM-4B-B16, TADD	100	51.2	1.63
Dream-VL-7B, TADD	100	28.0	50.24
Separate DiffusionGemma cohort (26B-A4B)			
DiffusionGemma, free	14	58.9	0.97
DiffusionGemma, TADD	100	40.6	3.48
RippleVLM-DG-26B	100	41.6	3.40

5 Results

5.1 Structure, content, and runtime are separate outcomes

Table 1 and Figure 3 show the main trade-off. RippleVLM-8B reaches 55.0% content F1 and passes the structural check on 50/50 pages. Qwen3-VL-8B with Outlines has higher F1, 58.4%, but only 18/50 returned records pass the check under the configured cap. RippleVLM-8B takes 2.50 s versus 4.62 s for Outlines, a ratio of 1.85×. The F1 gap is 3.4 points, not a large accuracy advantage: the scorer credits overlapping words even in invalid JSON. Among Outlines outputs, valid and invalid records average 68.4% and 52.8% F1, respectively. These are descriptive subsets, not matched comparisons. The 4B-B16 model offers another operating point: 51.2% F1 in 1.63 s with 50/50 structurally valid records. No single system dominates all metrics. Figure 4 places these operating points in the broader accuracy–latency comparison; Table 4 reports all configurations.

Completion under a finite token cap. Outlines succeeds on 18/50 records under the 256-token cap. Passing records have shorter reference texts (149 versus 349 words on average), consistent with truncation, but raw outputs and termination reasons are unavailable. The scorer accepts empty values and extra keys, so these cannot explain failures. Larger caps and matched serialization policies are needed to isolate completion limits from decoding effects.

Global diffusion comparison. Dream-VL TADD records 28.0% F1 in 50.24 s on the same image IDs. Figure 13 shows this cross-architecture comparison. The ratio to RippleVLM-8B is approxi-

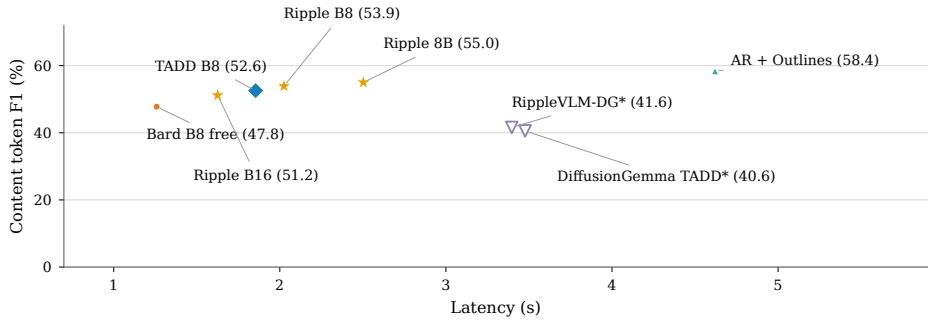

Figure 4: **Content F1 versus recorded latency.** The zero-based F1 axis shows a 3.4-point gap between AR+Outlines and RippleVLM-8B. F1 credits overlapping text even when JSON is invalid: structural validity is 36% for AR+Outlines and 100% for the TADD points shown. Hollow purple markers (*) denote DiffusionGemma and RippleVLM-DG on a separate cohort, so their positions do not establish a cross-backbone ranking. Labels give F1 (%).

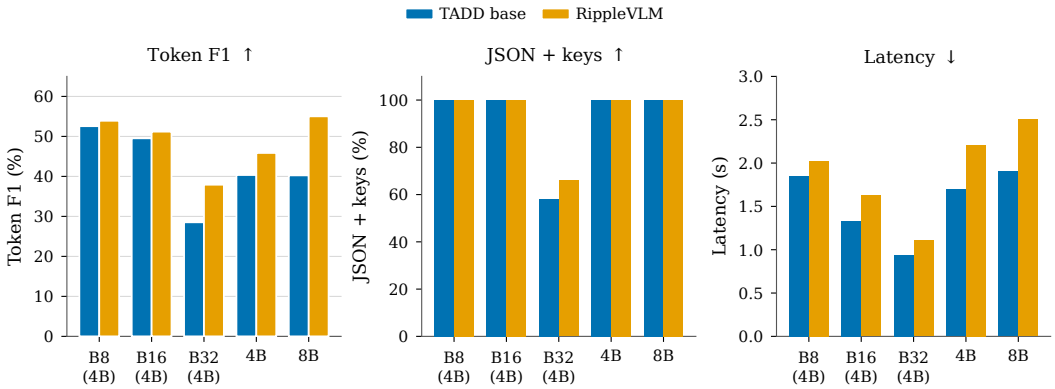

Figure 5: **DAVP across five Bard-VL variants.** Grouped bars show base TADD and projector-adapted results for content, structure, and runtime. Accuracy point estimates increase, while recorded latency also increases; paired uncertainty appears in Figure 8.

mately 20.1×, but it combines different backbones, samplers, and implementations. It is descriptive evidence about these runs, not a controlled estimate of the causal effect of block diffusion.

5.2 Projector adaptation improves extraction most clearly at 8B

DAVP raises 8B content F1 from 40.3% to 55.0%, a 14.8-point gain using unrounded records. Its paired interval is [8.4, 21.3] points. Figure 5 compares all three metrics across variants. The other variants have positive point estimates, but their intervals include zero (Table 2; Figure 8). Thus the evidence supports a substantial 8B improvement more strongly than a universal benefit across backbones. The experiment changes both task exposure and the trainable projector; without a matched alternative training objective, it does not isolate diffusion-specific alignment from document adaptation.

Sensitivity of the distilled B32 checkpoint. Two baseline runs of this checkpoint disagree. The run used for every B32 comparison passes the structural check on 58% of pages (66% after DAVP); a second run passes on 100% with 46.8% F1, which would reverse the apparent adaptation gain. We report both and make no claim of universal 100% validity; the stored records do not identify the cause.

Transfer to a canvas model. On its separate cohort, DiffusionGemma TADD achieves 100% JSON + keys, 40.6% F1, and 3.48 s. DAVP yields 41.6% F1 in 3.40 s with the same validity rate. The

Table 2: Paired DAVP comparison at $\ell = 96$. Gain and interval are percentage points of content F1; OK is the adapted model's JSON + keys rate. DG uses its separate cohort. Section 5.2 discusses a second B32 (distilled) baseline run.

Backbone / block	Base	DAVP	Gain	95% interval	OK (%)
4B / B8	52.6	53.9	+1.4	[-2.8, 5.8]	100
4B / B16	49.5	51.2	+1.7	[-3.5, 7.3]	100
4B / B32	28.5	37.9	+9.4	[-1.4, 20.6]	66
4B / B32 (non-distilled)	40.3	45.9	+5.5	[-1.9, 12.7]	100
8B / B32	40.3	55.0	+14.8	[8.4, 21.3]	100
DG-26B / canvas 256	40.6	41.6	+1.0	[-3.9, 6.3]	100

gain interval is $[-3.9, 6.3]$ points. This establishes an implementation of the interface on a second architecture, but not a statistically resolved accuracy benefit. Its free-generation F1 is 58.9% at 14% structural validity and 0.97 s: TADD can improve structure while reducing content accuracy and increasing runtime. The different image cohort precludes ranking these values directly against the Bard-VL rows.

5.3 Value budgets and multi-field extraction

At $\ell = 28$, base Bard-VL-4B-B8 reaches 31.5% F1 in 0.68 s; 96 gives 52.6% in 1.67 s and 128 gives 52.1% in 2.13 s (Appendix Figure 7), with 50/50 structurally valid records at every budget. Figure 12 plots runtime against allocated value tokens. Among the observed settings 96 is preferred, but this is not a general optimum. A repeat run at 96 tokens, used in Table 1, gives identical F1 in 1.85 s, indicating the run-to-run timing variance.

Schema size is harder to evaluate. With a 4B checkpoint at $\ell = 28$ (Figure 10; Table 8), TADD stays at 100% structural validity for $K = 2, 4, 6, 8$ with 26.9, 17.2, 18.2, and 19.2% F1, whereas free generation fails the check on every page at $K \geq 4$. On the DiffusionGemma cohort at $\ell = 96$ (Table 9), TADD also stays at 100%; content F1 at $K = 4, 6, 8$ is 17.2, 20.2, and 23.7% for the base model and 14.6, 19.6, and 22.9% after DAVP. Because the requested keys change with K while the reference remains full-page OCR, these values reflect schema-size mechanics rather than field-level accuracy, and DAVP does not improve them for $K>2$.

5.4 General vision–language benchmarks

Table 3 reports seven general vision–language benchmarks: MMMU (Yue et al., 2023), MMMU-Pro (Yue et al., 2024), MME (Fu et al., 2023), RealWorldQA (xAI, 2024), MMStar (Chen et al., 2024), AI2D (Kembhavi et al., 2016), and ChartQA (Masry et al., 2022). The baselines are Qwen3-VL (Bai et al., 2025), InternVL3.5 (Wang et al., 2025), LLaDA-V (You et al., 2025), Dream-VL (Ye et al., 2025), LaViDa (Li et al., 2025), SDAR-VL (Cheng et al., 2025b), MMaDA (Yang et al., 2025a), Dimple (Yu et al., 2025), and Bard-VL (Chen et al., 2026).

RippleVLM-4B-B16 reports 76.3 ChartQA and 2410 MME, and RippleVLM-4B-B8 reports 83.7 AI2D. RippleVLM-8B improves ChartQA over the reported Bard-VL value, 61.0 to 70.4, while lowering MMMU from 59.4 to 51.4, so specialization must be weighed against extraction gains. Appendix D reports a 13-benchmark VLMEvalKit suite, and Figure 9 shows its 8B differences. Because baseline protocols differ, we make no state-of-the-art claim on these benchmarks.

6 Limitations

Evaluation pages are sampled from the same PDFA cache that supplies projector-training pages, without an explicit hold-out, so some evaluation pages may have been seen during DAVP training. Such overlap can inflate the DAVP gains; the base-model and AR results do not involve projector training. A document-disjoint re-evaluation is required before the adaptation gains can be read as generalization.

The 50-page cohorts limit precision, and document-cluster bootstrapping captures neither training-seed nor repeated-timing variation. Assuming known keys and bounded strings leaves variable

Table 3: **General vision–language comparison.** Seven general-purpose benchmarks across autoregressive VLMs, diffusion VLMs, and RippleVLM. These are reported results under differing protocols, not a matched comparison. Higher is better; dashes denote unavailable scores.

Model	MMMU	Pro	MME	RWQA	MMStar	AI2D	ChartQA
Autoregressive VLMs							
Qwen3-VL-4B	54.2	38.0	2079	63.0	70.0	83.5	87.0
Qwen3-VL-8B	60.6	43.1	2214	69.1	73.7	85.7	88.2
InternVL3.5-4B	58.1	42.8	2163	68.3	73.5	83.0	83.4
InternVL3.5-8B	64.2	51.5	2356	72.5	77.3	87.4	89.7
Diffusion VLMs							
LLaDA-V-8B	42.1	26.4	1510	55.0	50.8	72.2	32.0
Dream-VL-7B	51.8	—	—	—	—	69.5	0.0
LaViDa-8B	56.8	41.6	2035	65.6	68.0	82.4	67.3
SDAR-VL-8B	53.4	38.2	1859	62.3	66.1	80.8	58.1
MMaDA-8B	41.4	24.0	1472	52.6	49.3	71.6	29.3
Dimple-VL-7B	49.3	—	—	—	—	68.7	53.2
Bard-VL-8B	59.4	43.2	2257	71.2	74.5	83.5	61.0
RippleVLM (ours)							
RippleVLM-8B	51.4	42.1	2302	68.5	67.3	81.9	70.4
RippleVLM-4B-B8	53.1	44.6	2376	68.9	72.4	83.7	70.8
RippleVLM-4B-B16	55.1	43.8	2410	69.5	75.6	83.1	76.3

Pro denotes MMMU-Pro; RWQA denotes RealWorldQA. Baseline rows are reported values from their sources, not re-evaluated here. DiffusionGemma publishes only MMMU-Pro (66.0 without and 54.3 with thinking; DiffusionGemma Team et al., 2026, Table 3) and RippleVLM-DG has not been evaluated on these benchmarks, so both are omitted.

arrays, optional fields, typed validation, and abstention unevaluated, and set F1 misses repetition and incorrect field assignments, which field-annotated references would expose.

The cross-model latency ratios do not isolate TADD's causal effect: a controlled comparison would need matched backbones, image resolution, output quality, serving code, and serialization policies, and component timings. Broad-benchmark regressions similarly limit generalization; mixed-task training, an AR-style projector objective, and unseen schemas would help separate document specialization from diffusion alignment. Neither medical-image nor handwritten-form generalization is supported.

7 Conclusion

RippleVLM pairs fixed-template diffusion decoding with visual-projector adaptation for bounded structured extraction, yielding a 14.8-point 8B content-F1 gain and a configuration-dependent accuracy–latency trade-off against capped AR decoding. Structural validity follows from the interface, whereas grounded field accuracy remains learned, so the two require separate evaluation.

Reproducibility Statement

The accompanying source includes the inference and training implementations, per-image evaluation records, and a deterministic paper builder that generates every table and plot from those records with image-ID matching, per-row mean checks, and a SHA-256 source manifest. Appendix B gives checkpoint identifiers, training settings, and the mapping from results to source files; Appendix F describes a constraint-processor benchmark with output-equivalence checks.

AI Usage Disclosure

An AI assistant helped edit the exposition, check consistency between code and reported results, write figure and table generation code, verify citation metadata, and launch the constraint-processor benchmark (Appendix F) and the DiffusionGemma $K{=}4$–8 evaluations (Table 9). The authors take full responsibility for all technical claims and the final manuscript.

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. In *International Conference on Learning Representations (ICLR)*, 2025. Oral.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, et al. Structured denoising diffusion models in discrete state-spaces. *arXiv preprint arXiv:2107.03006*, 2021.

Shuai Bai et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Michael Cardei, Jacob K Christopher, Thomas Hartvigsen, Bhavya Kailkhura, et al. Constrained discrete diffusion. *arXiv preprint arXiv:2503.09790*, 2025.

Baoyou Chen, Hanchen Xia, Peng Tu, Haojun Shi, Liwei Zhang, Yuxuan Yao, Weihao Yuan, and Siyu Zhu. BARD: Bridging AutoRegressive and diffusion vision-language models via highly efficient progressive block merging and stage-wise distillation. *arXiv preprint arXiv:2604.16514*, 2026.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, et al. Are we on the right way for evaluating large vision-language models? *arXiv preprint arXiv:2403.20330*, 2024.

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

Shuang Cheng, Yuhua Jiang, Zineng Zhou, Dawei Liu, Wang Tao, Linfeng Zhang, Biqing Qi, and Bowen Zhou. SDAR-VL: Stable and efficient block-wise diffusion for vision-language understanding. *arXiv preprint arXiv:2512.14068*, 2025b.

DiffusionGemma Team, Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, Tianlin Liu, Ivan Lobov, Ofir Nabati, João Gabriel Oliveira, Nicolas Perez-Nieves, Nastasia Prutianova, Bobak Shahriari, Jean Tarbouriech, et al. DiffusionGemma technical report. Technical report, Google DeepMind, 2026. arXiv:2608.00146.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. XGrammar: Flexible and efficient structured generation engine for large language models. *arXiv preprint arXiv:2411.15100*, 2024.

Haodong Duan, Xinyu Fang, Junming Yang, Xiangyu Zhao, Zerun Ma, Yuxuan Qiao, et al. VLMEvalKit: An open-source toolkit for evaluating large multi-modality models. *arXiv preprint arXiv:2407.11691*, 2024. Version 5, 2026.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, et al. MME: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. *arXiv preprint arXiv:2305.13971*, 2023.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, et al. JSONSchemaBench: A rigorous benchmark of structured outputs for language models. *arXiv preprint arXiv:2501.10868*, 2025.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, et al. HallusionBench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. *arXiv preprint arXiv:2310.14566*, 2023.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, et al. LayoutLMv3: Pre-training for document AI with unified text and image masking. *arXiv preprint arXiv:2204.08387*, 2022.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, et al. A diagram is worth a dozen images. *arXiv preprint arXiv:1603.07396*, 2016.

Geewook Kim, Teakgyu Hong, Moonbin Yim, Jeongyeon Nam, et al. OCR-free document understanding transformer. *arXiv preprint arXiv:2111.15664*, 2021.

Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, et al. Pix2Struct: Screenshot parsing as pretraining for visual language understanding. *arXiv preprint arXiv:2210.03347*, 2022.

Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. LaViDa: A large diffusion language model for multimodal understanding. *arXiv preprint arXiv:2505.16839*, 2025.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, et al. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, et al. MMBench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*, 2023a.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, et al. OCRBench: On the hidden mystery of ocr in large multimodal models. *arXiv preprint arXiv:2305.07895*, 2023b.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, et al. Learn to explain: Multimodal reasoning via thought chains for science question answering. *arXiv preprint arXiv:2209.09513*, 2022.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, et al. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, et al. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. *arXiv preprint arXiv:2007.00398*, 2020.

Minesh Mathew, Viraj Bagal, Rubèn Pérez Tito, Dimosthenis Karatzas, et al. InfographicVQA. *arXiv preprint arXiv:2104.12756*, 2021.

Niels Mündler, Jasper Dekoninck, and Martin Vechev. Constrained decoding of diffusion LLMs with context-free grammars. *arXiv preprint arXiv:2508.10111*, 2025.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, et al. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Pixparse. PDFA-eng-WDS. `https://huggingface.co/datasets/pixparse/pdfa-eng-wds`, 2024.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, et al. Simple and effective masked diffusion language models. *arXiv preprint arXiv:2406.07524*, 2024.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, et al. Towards VQA models that can read. *arXiv preprint arXiv:1904.08920*, 2019.

Tarun Suresh, Debangshu Banerjee, Shubham Ugare, Sasa Misailovic, et al. DINGO: Constrained inference for diffusion LLMs. *arXiv preprint arXiv:2505.23061*, 2025.

Weiyun Wang et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

Luis Wiedmann, Orr Zohar, Amir Mahla, Xiaohan Wang, et al. FineVision: Open data is all you need. *arXiv preprint arXiv:2510.17269*, 2025.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, et al. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. *arXiv preprint arXiv:2505.22618*, 2025.

xAI. RealWorldQA. https://huggingface.co/datasets/xai-org/RealworldQA, 2024. Released with Grok-1.5 Vision.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. MMaDA: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025a.

Yue Yang, Ajay Patel, Matt Deitke, Tanmay Gupta, et al. Scaling text-rich image understanding via code-guided synthetic multimodal data generation. *arXiv preprint arXiv:2502.14846*, 2025b.

Jiacheng Ye, Shansan Gong, Jiahui Gao, Junming Fan, Shuang Wu, Wei Bi, Haoli Bai, Lifeng Shang, and Lingpeng Kong. Dream-VL & Dream-VLA: Open vision-language and vision-language-action models with diffusion language model backbone. *arXiv preprint arXiv:2512.22615*, 2025.

Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. LLaDA-V: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*, 2025.

Runpeng Yu, Xinyin Ma, and Xinchao Wang. Dimple: Discrete diffusion multimodal large language model with parallel decoding. *arXiv preprint arXiv:2505.16990*, 2025.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, et al. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. *arXiv preprint arXiv:2311.16502*, 2023.

Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, et al. MMMU-Pro: A more robust multi-discipline multimodal understanding benchmark. *arXiv preprint arXiv:2409.02813*, 2024.

A Full Structured Extraction Results

Table 4 reports all primary-cohort configurations, including both AR sizes, free-generation baselines, all five Bard-VL TADD variants, and all five RippleVLM variants. Table 5 reports the separate DiffusionGemma cohort. Recorded latencies are used directly.

Table 4: Full extraction results on the common 50-page cohort. TADD uses two fields and 96 tokens per field. Free-generation and AR caps follow Section 4; B32 TADD is the baseline run discussed in Section 5.2.

System	JSON + keys (%)	F1 (%)	Latency (s)
Qwen3-VL-4B, free	30	59.1	3.70
Qwen3-VL-4B, Outlines	32	59.7	3.77
Qwen3-VL-8B, free	28	61.8	4.93
Qwen3-VL-8B, Outlines	36	58.4	4.62
Dream-VL-7B, free	46	26.7	13.68
Dream-VL-7B, TADD	100	28.0	50.24
Bard-VL-4B-B8, free	6	47.8	1.26
Bard-VL-4B-B8, TADD	100	52.6	1.85
Bard-VL-4B-B16, TADD	100	49.5	1.33
Bard-VL-4B-B32, TADD	58	28.5	0.94
Bard-VL-4B, TADD	100	40.3	1.70
Bard-VL-8B, TADD	100	40.3	1.90
RippleVLM-4B-B8	100	53.9	2.02
RippleVLM-4B-B16	100	51.2	1.63
RippleVLM-4B-B32	66	37.9	1.11
RippleVLM-4B	100	45.9	2.20
RippleVLM-8B	100	55.0	2.50

Table 5: Canvas-diffusion extraction on the separate 50-page DiffusionGemma cohort.

System	JSON + keys (%)	F1 (%)	Latency (s)
DiffusionGemma, free	14	58.9	0.97
DiffusionGemma, TADD	100	40.6	3.48
RippleVLM-DG-26B	100	41.6	3.40

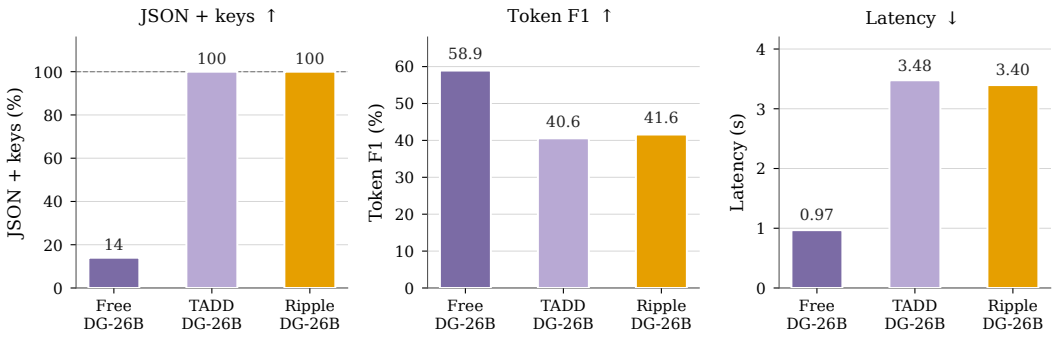

Figure 6: **DiffusionGemma extraction on its separate cohort.** Free generation, TADD, and RippleVLM-DG-26B trade off structure, content, and latency. These 50 image IDs differ from the primary cohort.

B Reproduction Details

Checkpoint initialization. Bard-VL variants come from the fudan-generative-ai organization. The distilled checkpoints are Bard-VL-B8-Mask-4B-Distil-Instruct, Bard-VL-B16-Mask-4B-Distil-

Instruct, and Bard-VL-B32-Mask-4B-Distil-Instruct. The non-distilled checkpoints are Bard-VL-B4-Mask-4B-Instruct and Bard-VL-B4-Mask-8B-Instruct; despite "B4" in those names, our evaluator sets their inference block size to 32. DAVP variants initialize from the corresponding backbone and load separate projector weights. The other backbones are Dream-VL-7B, DiffusionGemma-26B-A4B, and Qwen3-VL-8B-Instruct with its 4B counterpart.

Trainable scope and targets. For Bard-VL, training updates the visual merger and the DeepStack merger; other parameters are frozen. For DiffusionGemma, the trainer updates the visual embedding projection. The Bard-VL dataset helper splits OCR text at the first blank line, uses up to 60 whitespace words from the first segment as the heading and up to 60 from the remainder as the body, and duplicates the first segment when no remainder exists. Targets are then tokenized, truncated to their value slots, and EOS-padded. This heuristic is not manually annotated field supervision. The trainable parameter count depends on the loaded checkpoint and should be measured for each released variant.

Table 6: Training-script defaults. Command-line overrides can change these per run; the configuration used for the reported checkpoints is given in the text.

Setting	Bard-VL trainer	DiffusionGemma trainer
Learning rate	10^{-4}	5×10^{-5}
Warmup steps	100	200
Gradient accumulation	4	8
Requested steps	2,000	2,000
Seed	42	42
Weight decay	0.01	0.01

Training configuration. Projector training uses a 168K-example mixture (40K PDFA pages and 128K CoSyn images), learning rate 5×10^{-5}, 200 warmup steps, gradient accumulation 8, and 5,000 steps for B8 versus 2,000 for the other models. The Bard-VL trainer caps processor image area at 512×512, whereas the AR evaluation path permits up to 1024×1024, so image preprocessing is an additional uncontrolled factor in latency comparisons. Extraction runs use an NVIDIA RTX 6000; the DiffusionGemma $K \geq 4$ runs and the component benchmark (Appendix F) use an NVIDIA RTX PRO 6000 Blackwell Workstation Edition (96 GB).

Primary extraction sources. The deterministic builder selects the B8 budget-96 rerun and the B16, B32, 4B, and 8B second budget-96 files for the base-versus-DAVP table. Adapted rows use the corresponding projector-adapted files. Selection is fixed in source rather than chosen for the largest gain. The budget sweep uses the B8 files at budgets 28, 96, and 128. A generated manifest records each source filename, hash, and paired-effect interval. No input record is overwritten.

Table 7: Recorded TADD latency before and after DAVP. Unmodified per-image means; differences do not isolate adaptation's causal runtime effect.

Backbone	Base (s)	DAVP (s)
4B / B8	1.85	2.02
4B / B16	1.33	1.63
4B / B32	0.94	1.11
4B / B32 (non-distilled)	1.70	2.20
8B / B32	1.90	2.50
DG-26B / canvas 256	3.48	3.40

Artifact checks. For every primary extraction source, the builder verifies row count, unique image IDs, and agreement between per-row means and stored aggregates. It checks image-ID equality within paired comparisons and confirms that the DiffusionGemma cohort is separate. These checks validate arithmetic and pairing; they cannot validate how predictions were obtained. The broad-benchmark JSON files contain aggregates only, so they receive no equivalent per-example verification.

C Multi-Field and Budget Results

Table 8: Multi-field evaluation of a 4B Bard-VL checkpoint at fixed budget $\ell = 28$, $N = 50$. F1 compares values to full-page text and is not a field-accuracy metric.

	Free generation			TADD		
Fields	OK (%)	F1 (%)	Time (s)	OK (%)	F1 (%)	Time (s)
2	6	27.9	0.92	100	26.9	0.61
4	0	8.3	0.28	100	17.2	0.89
6	0	9.6	0.40	100	18.2	1.24
8	0	11.5	0.94	100	19.2	3.05

Table 9: DiffusionGemma multi-field TADD on the canvas cohort (N=50, ℓ=96). K=2 rows are the runs in Table 5; $K \geq 4$ rows were run later on an RTX PRO 6000 Blackwell, so latencies are not comparable across K. Keys follow Table 8; F1 is page-text overlap. At $K \geq 4$ free generation often emits the requested keys while F1 remains low.

		Free generation			TADD		
Model	K	OK (%)	F1 (%)	Time (s)	OK (%)	F1 (%)	Time (s)
DiffusionGemma-26B	2	14	58.9	0.97	100	40.6	3.48
DiffusionGemma-26B	4	100	8.3	0.44	100	17.2	2.72
DiffusionGemma-26B	6	96	24.4	0.70	100	20.2	2.63
DiffusionGemma-26B	8	90	25.8	0.92	100	23.7	2.68
RippleVLM-DG	2	14	58.4	1.04	100	41.6	3.40
RippleVLM-DG	4	100	8.4	0.44	100	14.6	2.70
RippleVLM-DG	6	98	23.6	0.67	100	19.6	2.80
RippleVLM-DG	8	92	25.6	0.90	100	22.9	2.75

For $K = 2$ the keys are heading and body. For $K = 4$ they are title, date, author, and organization. $K = 6$ adds venue and abstract; $K = 8$ adds keywords and summary. These differing content requests make direct F1 comparisons across K ambiguous even at a matched token budget, and no manually annotated field values are available. A separate 500-image heading/body run records 100% TADD structural validity and 28.1% F1, versus 4.2% validity for free generation.

Table 10: Base Bard-VL-4B-B8 budget sweep underlying Figure 7. A 192-token run failed on every page (0% validity and F1) and is excluded as a failed run rather than evidence about long budgets.

Tokens / field	JSON + keys (%)	F1 (%)	Latency (s)
28	100	31.5	0.68
96	100	52.6	1.67
128	100	52.1	2.13

D General Vision–Language Evaluation

Tables 11–12 report VLMEvalKit aggregates; missing results are not extrapolated. The document-oriented suite covers OCRBench (Liu et al., 2023b), DocVQA (Mathew et al., 2020), TextVQA (Singh et al., 2019), ChartQA (Masry et al., 2022), and InfoVQA (Mathew et al., 2021); the general suite covers MMBench (Liu et al., 2023a), MME (Fu et al., 2023), POPE (Li et al., 2023), ScienceQA (Lu et al., 2022), AI2D (Kembhavi et al., 2016), MMMU (Yue et al., 2023), MathVista (Lu et al., 2023), and HallusionBench (Guan et al., 2023). They include the B8/B16/B32 variants. Unreported values are denoted by n/a. Native metric scales differ across columns; no cross-benchmark average is reported. OCRBench is on its original 1,000-point scale; DocVQA and InfoVQA use ANLS, TextVQA accuracy, and ChartQA relaxed accuracy.

The distilled 4B-B16 variant reaches 76.3 on ChartQA and 55.1 on MMMU, while 4B-B32 reaches 77.5 on DocVQA. These variants therefore deserve consideration alongside the 8B model, rather than presenting model size as a monotonic improvement. The corresponding base 4B benchmark

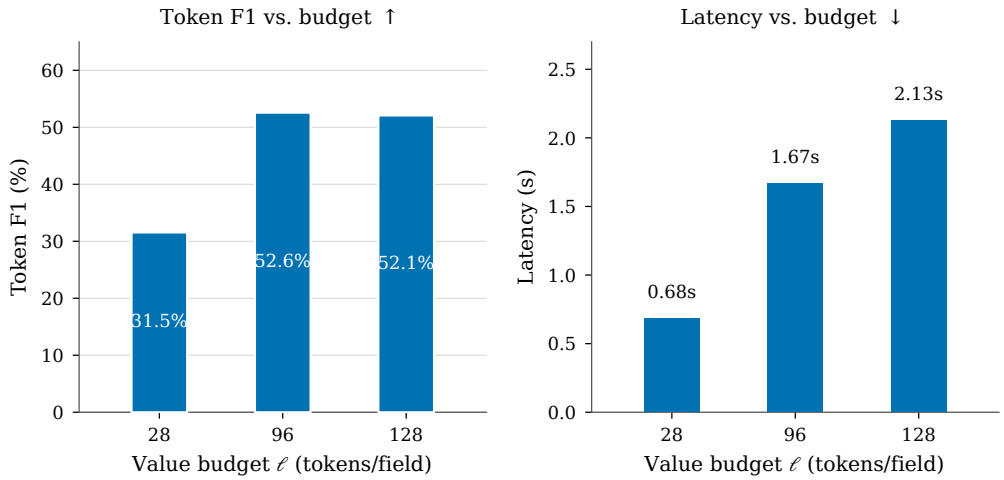

Figure 7: Value-budget sweep for the base Bard-VL-4B-B8 model ($K = 2$, $N = 50$). Bars show the three measured operating points. Moving from 28 to 96 tokens per field improves F1 substantially; 128 offers no point-estimate gain.

Table 11: Document-oriented benchmark summaries. Entries use each benchmark's native metric.

Model	OCRBench	DocVQA	TextVQA	ChartQA	InfoVQA
Bard-VL-8b	140.0	85.4	61.5	61.0	62.4
Ripple-8b	484.0	73.6	58.5	70.4	62.5
Ripple-b8	506.0	66.9	56.5	70.8	50.7
Ripple-b16	552.0	75.9	58.9	76.3	58.8
Ripple-b32	559.0	77.5	58.8	73.7	61.3
Ripple-4b	312.0	48.8	16.9	11.5	32.5
Dream-VL-7b	150.0	29.9	0.0	0.0	9.3

Table 12: General benchmark summaries. MME is the stored total; POPE is F1; Hall. is Hallusion-Bench aAcc. Other columns use their stored accuracy metric.

Model	MMBench	MME	POPE	ScienceQA	AI2D	MMMU	MathVista	Hall.
Bard-VL-8b	89.0	n/a	88.7	91.0	83.5	59.4	48.5	53.0
Ripple-8b	78.4	2301.7	89.8	89.7	81.9	51.4	n/a	58.8
Ripple-b8	88.1	2375.9	89.1	87.7	83.7	53.1	n/a	64.1
Ripple-b16	86.5	2410.0	88.9	88.1	83.1	55.1	n/a	68.3
Ripple-b32	87.9	2314.9	89.5	90.2	83.5	52.2	n/a	59.2
Ripple-4b	79.1	2144.2	85.9	89.3	73.3	48.4	n/a	56.7
Dream-VL-7b	80.5	n/a	74.8	82.0	69.5	51.8	52.5	64.5

summaries are unavailable, so these cross-variant values do not isolate a DAVP effect. The 8B comparison is available, but its losses on DocVQA, MMBench, and MMMU preclude a blanket claim of preserved general capability.

E Additional Ablations and Visual Analysis

The main paper reports extraction (Table 1), the latency trade-off (Figure 4), and paired DAVP effects (Table 2). This appendix gives the complementary block-size, multi-field, diffusion-type, structural-validity, and combined ablation plots. All figures use observed records; the latency analysis connects measured budgets without extrapolation.

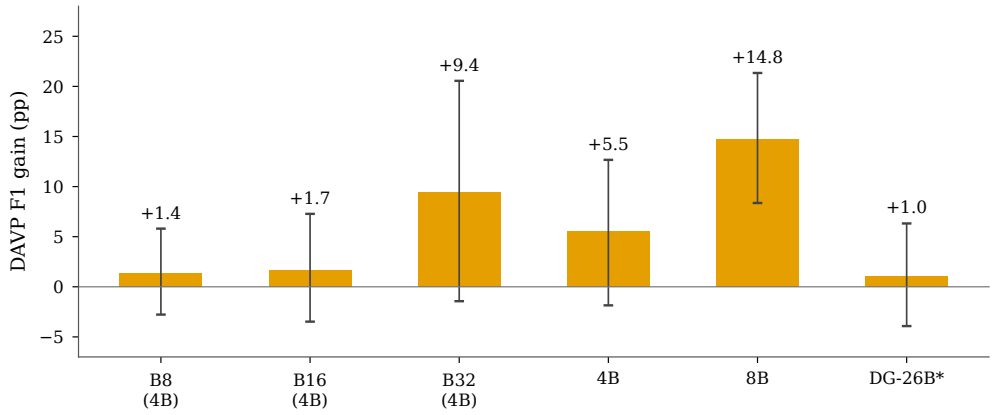

Figure 8: DAVP F1 gains with paired 95% document-cluster bootstrap intervals. The horizontal line marks no change; DG-26B* uses its separate cohort. Positive point estimates alone are not treated as conclusive improvements; only the 8B interval excludes zero in this analysis.

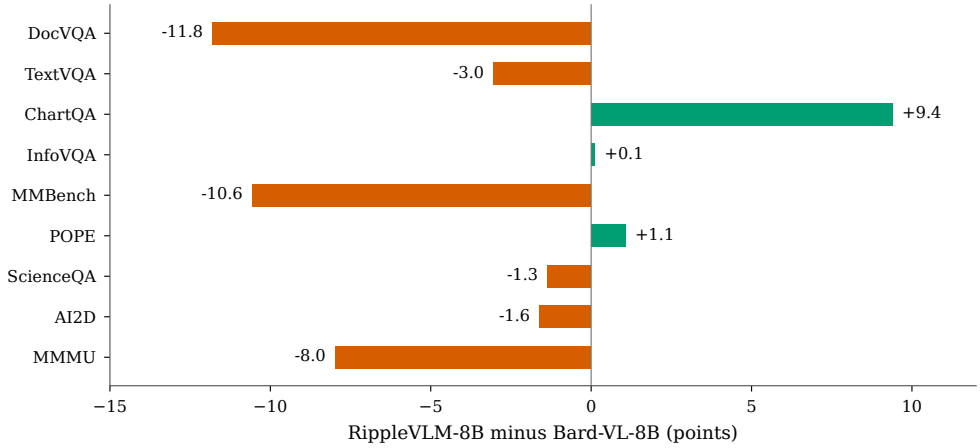

Figure 9: General-benchmark differences between RippleVLM-8B and Bard-VL-8B. Each benchmark uses its own native metric. Gains in chart understanding coexist with document and reasoning regressions; these aggregate summaries do not support a claim that projector-only training preserves all capabilities.

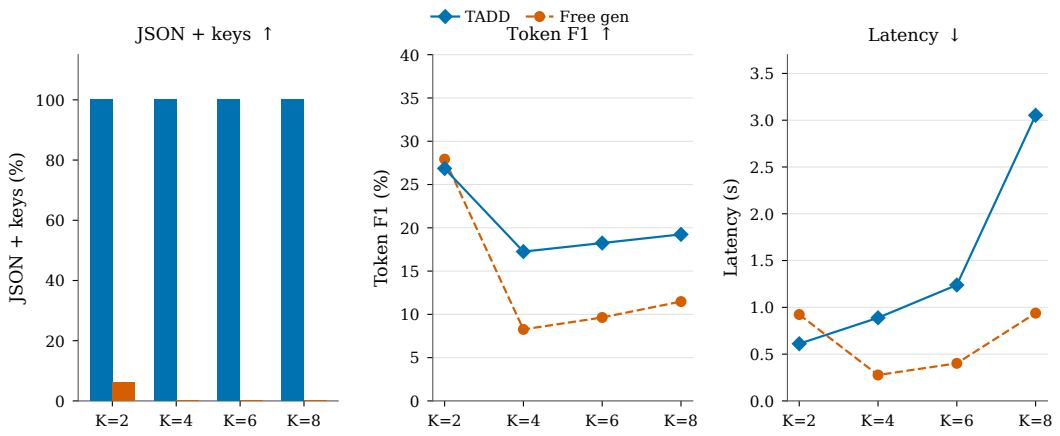

Figure 10: **Multi-field extraction with a 4B checkpoint at $\ell = 28$.** TADD returns valid records at every tested schema size; free-generation validity falls to zero at $K \geq 4$. Both methods use the same cohort. Page-text F1 measures content coverage, not field accuracy; requested content changes with K.

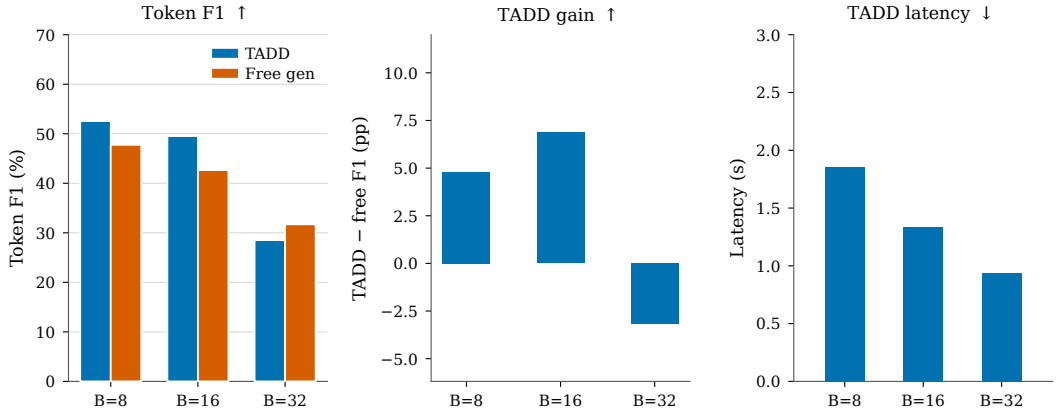

Figure 11: **Distilled block-size comparison at budget 96.** The three panels compare free-generation and TADD F1, their difference, and TADD latency. B32 uses the baseline run discussed in Section 5.2, whose TADD F1 is lower than free generation. Block size identifies a separately trained checkpoint. Table 13 additionally reports projector-adapted results.

Table 13: Base TADD and DAVP block-size results complementing Figure 11.

	Base TADD			DAVP		
Block	OK (%)	F1 (%)	Time (s)	OK (%)	F1 (%)	Time (s)
8	100	52.6	1.85	100	53.9	2.02
16	100	49.5	1.33	100	51.2	1.63
32	58	28.5	0.94	66	37.9	1.11

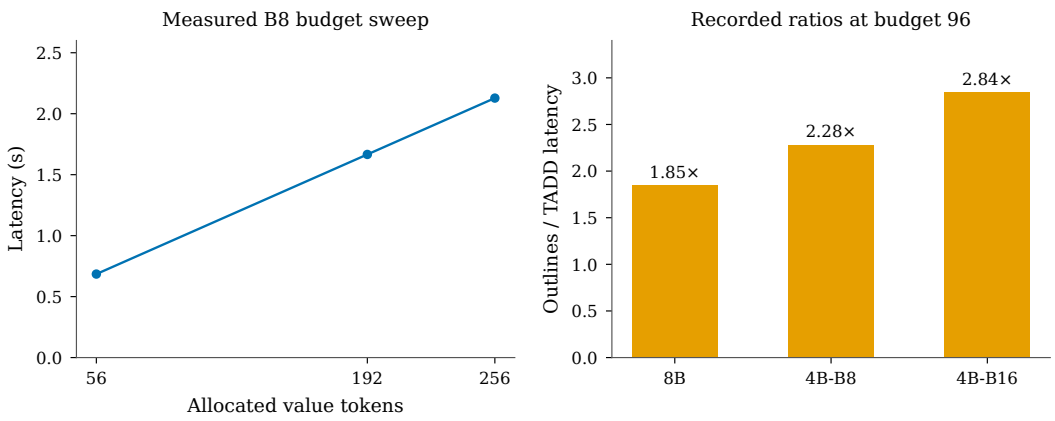

Figure 12: **Measured runtime analysis.** Left: base Bard-VL-4B-B8 latency at three allocated value budgets; allocations are not observed generated lengths. Right: recorded Outlines/TADD latency ratios at budget 96, with different caps and models. These are configuration comparisons, not matched-quality speedups or fitted scaling laws.

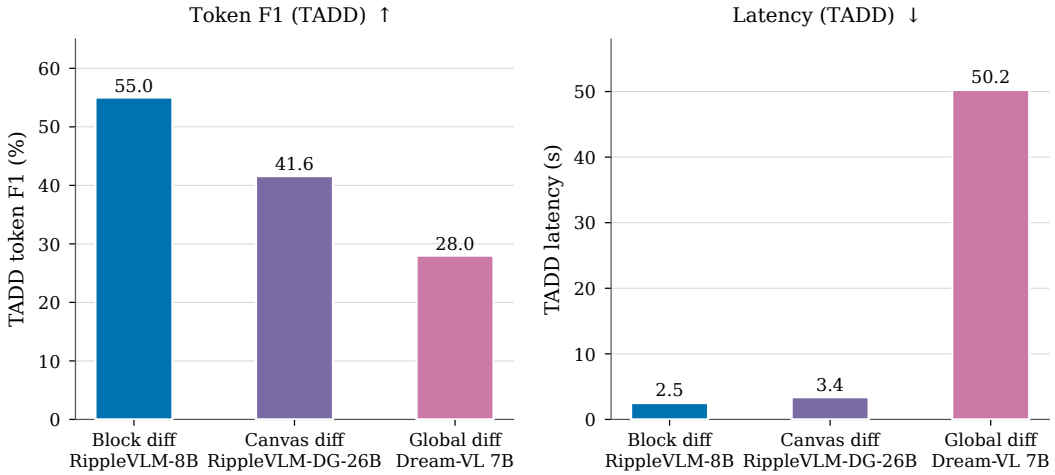

Figure 13: **Diffusion backbones under TADD.** RippleVLM-8B and Dream-VL share the primary cohort; RippleVLM-DG-26B uses the separate canvas cohort. Each reaches 100% JSON + keys. The primary-cohort latency ratio is $20.1\times$, but backbones and samplers differ. The canvas row demonstrates a separate operating point and cannot be ranked as a matched-cohort accuracy comparison.

Table 14: Recorded latency ratios relative to Qwen3-VL-8B with Outlines. All rows use the primary cohort; output policies differ.

System	Latency (s)	Outlines / system	OK (%)
Qwen3-VL-8B, Outlines	4.62	1.00	36
RippleVLM-8B	2.50	1.85	100
RippleVLM-4B-B8	2.02	2.28	100
RippleVLM-4B-B16	1.63	2.84	100
Dream-VL-7B, TADD	50.24	0.09	100

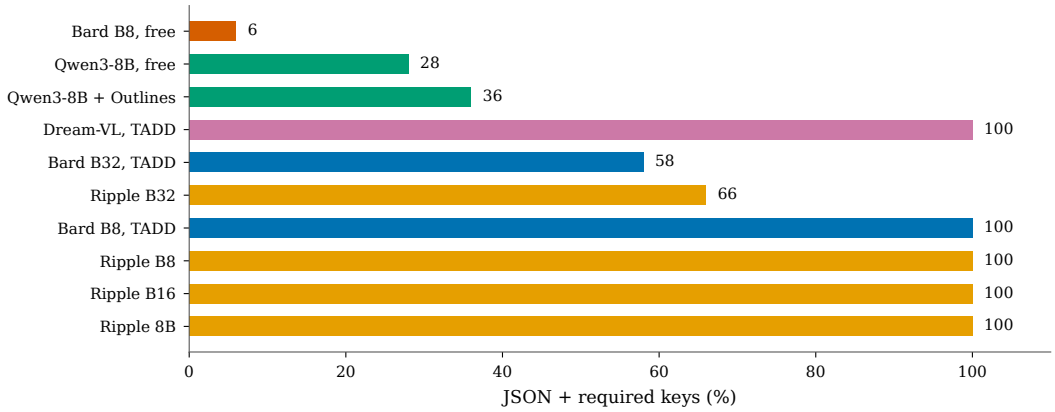

Figure 14: **Structural validity across decoding configurations.** The plot includes the B32 failures alongside successful TADD configurations. Validity is the evaluator's JSON + required-keys check; it does not measure content correctness or arbitrary JSON Schema compliance.

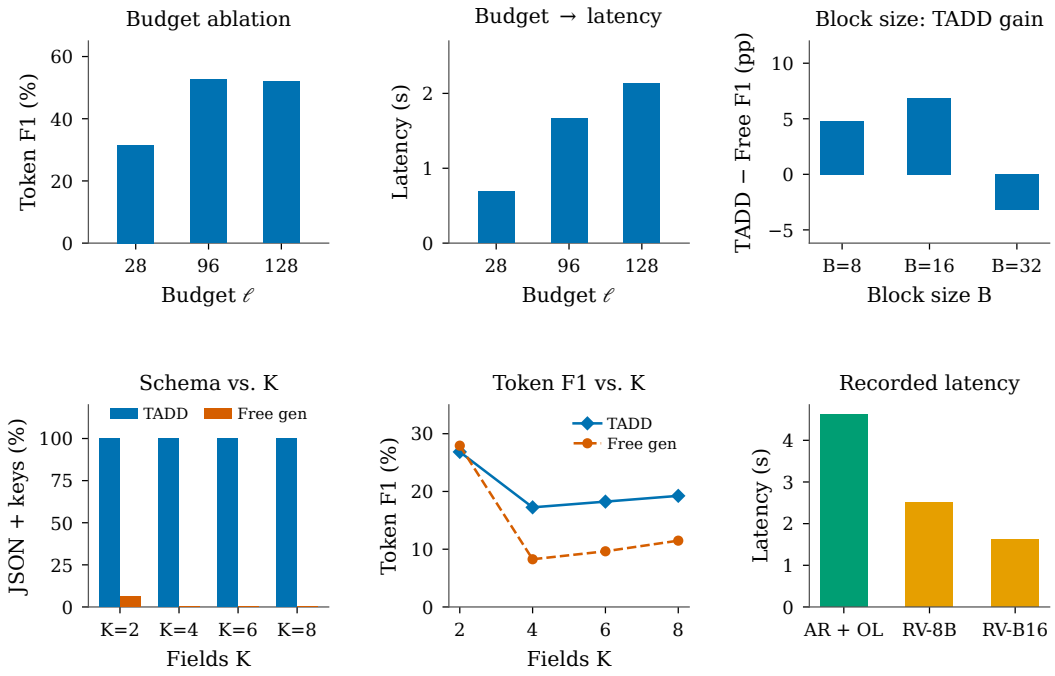

Figure 15: **Six-panel ablation overview.** Budget and block-size panels use the 50-page primary cohort; field-count panels use the 4B checkpoint at budget 28. Matched canvas multi-field TADD at $\ell = 96$ is Table 9. The final panel compares recorded runtime under the main settings. These panels summarize distinct experiments rather than a single factorial study.

F Constraint-Processor Benchmark on RTX PRO 6000

Motivation and implementation. The DiffusionGemma TADD processor suppresses blocked vocabulary entries at masked template positions. The reference implementation iterates over blocked token IDs and performs one indexed assignment per ID. We replace this loop with a single indexed assignment over the Cartesian product of masked positions and blocked IDs, caching the indices for a

fixed template, device, and vocabulary size. This changes the implementation of token suppression without changing its rule or the denoising schedule.

Protocol. We benchmark both implementations on an NVIDIA RTX PRO 6000 Blackwell Workstation Edition (96 GB), using PyTorch 2.8.0+cu128 and bfloat16 logits. The synthetic score tensor has shape $1 \times 256 \times 262{,}144$, with 192 masked positions. We sample fixed sets of 32, 256, and 1,024 blocked IDs (seed 42) to measure sensitivity to constraint-set size; these are not tokenizer-derived blocklists. After five warmups per implementation, we collect 30 timings per condition, randomly interleaving the two implementations and synchronizing CUDA before and after each call. Timing includes host dispatch and device execution but excludes index construction, score allocation, and model inference. Other GPU services retain approximately 69 GB of memory during the run; the device is not reserved exclusively for this experiment.

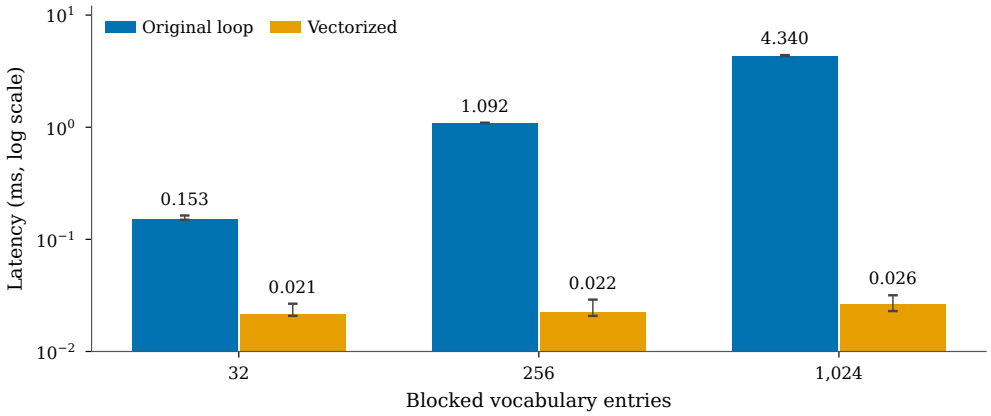

Figure 16: Measured constraint-processor latency on RTX PRO 6000 Blackwell. Bars are medians of 30 synchronized calls; whiskers span the 10th–90th percentiles. The logarithmic axis separates the two implementations. Inputs and blocked-token sets are synthetic. These ratios describe token suppression only, not end-to-end extraction speed.

Results and scope. Median latency decreases for all three tested sizes: 32 blocked IDs: 0.153 to 0.021 ms ($7.1\times$); 256 blocked IDs: 1.092 to 0.022 ms ($48.7\times$); 1,024 blocked IDs: 4.340 to 0.026 ms ($164.2\times$). Exact tensor equality holds for every benchmark condition and for checks with an empty blocklist, no masked positions, and an out-of-vocabulary blocked ID. The reduction reflects fewer host-dispatched indexing operations. Overall extraction latency also includes vision encoding, prefill, denoising, and serialization, so the component ratios cannot be applied to the full model. Measuring an end-to-end gain requires repeated model inference with the same checkpoint and inputs; main-text extraction latencies use the reference implementation.

Reproduction. The constraint-processor benchmark loads the production processor directly from source without importing the model stack. Its saved record lists individual timings, tensor dimensions, software versions, source hashes, and the GPU process snapshot. The benchmark requires CUDA and does not download model weights.