PERCEPDRAFT FOR EFFICIENT GENERATIVE PERCEPTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative perception models express different visual perception tasks as one generation process, in which sparse tasks such as detection are decoded autoregressively into structured symbol sequences and dense tasks such as segmentation, depth and surface normals are produced as pixel-level perception maps through iterative denoising. Both generation processes call the full backbone repeatedly, which makes inference expensive. We find that the structural regularity inherent in perception outputs makes their generation predictable from step to step: the serialization pattern of a structured sequence is largely fixed, and the global spatial layout of a dense perception map is essentially formed early in denoising. Based on this observation we propose PercepDraft, which keeps the pretrained backbone frozen and introduces two lightweight drafters, a token drafter and a trajectory drafter. They respectively propose several future tokens, or future denoising velocities with per-offset error estimates used to select a trusted horizon. The backbone then verifies the proposals in one call, so that fewer backbone calls are needed at inference. Experiments confirm the effectiveness of PercepDraft across multiple generative perception models. On SenseNova-Vision, it achieves end-to-end speedups of up to 1.6×, 2.0×, 6.3×, and 5.4× on detection, panoptic segmentation, depth estimation, and surface normal estimation, respectively, with limited changes in task performance.

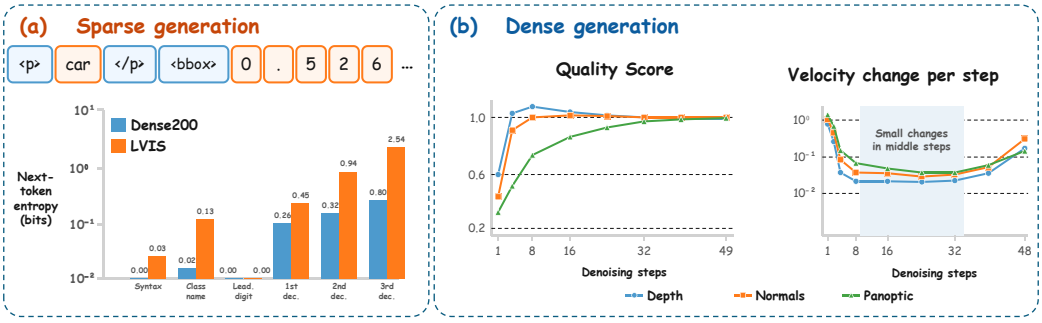


Figure 1: Step-to-step predictability in SenseNova-Vision across structured and dense generation. (a) Next-token entropy by position in detection records on Dense200 and LVIS. (b) Quality of the clean-output estimate after k denoising steps (left) and the change in predicted velocity between adjacent steps (right) for depth, surface normals, and panoptic segmentation.

1 INTRODUCTION

Visual perception covers object detection, image segmentation, depth estimation and surface normal estimation, and underlies the understanding of complex visual scenes. Conventional methods design a separate network structure and prediction head for each task and output the result directly in one forward pass. Generative perception offers a different paradigm: instead of task-specific prediction heads, it expresses visual prediction as a generation process. Sparse tasks such as object detection generate categories and locations as an autoregressive sequence, and dense tasks such as segmentation, depth estimation and surface normal estimation recover pixel-level perception maps through

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.

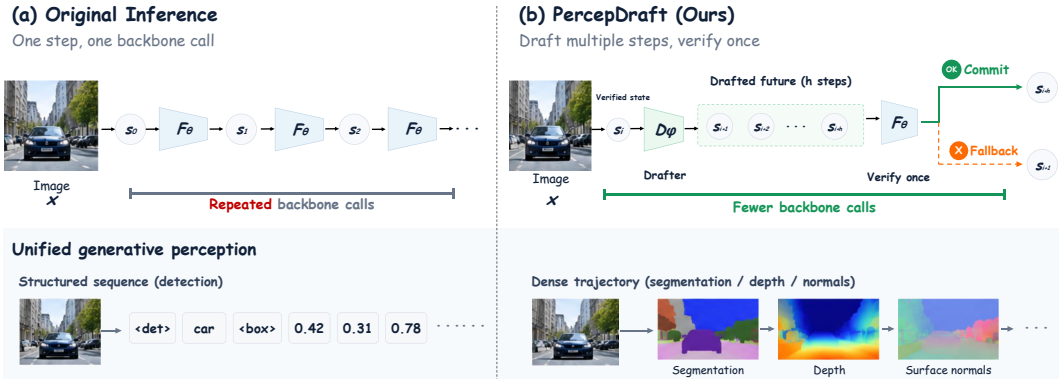


Figure 2: Draft-and-verify inference with PercepDraft. (a) Conventional generative perception advances one state per backbone call. (b) A lightweight drafter proposes h future states, which the frozen backbone verifies in one call. The token branch commits the longest matching prefix plus the backbone's next token; the dense branch accepts the drafted segment or advances by one exact step. Neither branch changes the backbone or output interface.

iterative denoising (Chen et al., 2022; Tang et al., 2025; Ke et al., 2024; Zhao et al., 2025). This unified generative paradigm is becoming an important direction in visual perception and has been validated on a range of generalist models (SenseNova-Vision Team, 2026; Gabeur et al., 2026).

The unified generative paradigm also carries a substantial computational cost. A generative model calls the full backbone repeatedly along the generation process: a sparse output is decoded token by token, each step requiring one full backbone call, and a dense output is denoised over many iterations along its trajectory. The inference cost therefore depends not only on model size but also on the length of the generated output and the number of denoising steps. Reducing the number of backbone calls along the generation trajectory is thus essential for efficient generative perception.

Existing methods for accelerating generation were designed mainly for open-ended text or image generation. For autoregressive generation, speculative decoding lets a lightweight model predict several following tokens that the target model then verifies, which reduces the number of token-by-token decoding steps (Leviathan et al., 2023; Chen et al., 2023); for diffusion generation, existing methods reduce the number of sampling steps, reuse intermediate features or predict future states (Lu et al., 2022; Ma et al., 2024; Liu et al., 2024a; Bajpai et al., 2026). All of them were designed for open-ended generation and do not take into account the structural regularity particular to perception outputs, so transferring them directly to generative perception is often of limited effect.

To design an acceleration strategy suited to perception tasks, we examine the actual inference process of a generative perception model. As Figure 1 shows, in a structured sequence output a large share of the tokens is constrained by the fixed record format and carries low predictive uncertainty, and in dense outputs such as depth and surface normals adjacent denoising steps share a high degree of spatial structure, so that later states can to a considerable extent be predicted from the trajectory so far. Generation in generative perception is therefore markedly predictable from step to step: the next several tokens or denoising steps are not independent of what has been generated but can be anticipated from the existing trajectory.

Based on the observation above we propose PercepDraft, a draft-and-verify framework for generative perception. PercepDraft keeps the pretrained backbone entirely frozen and introduces two lightweight drafters, while the original backbone serves as the verifier (Figure 2). For structured sequence generation, the token drafter uses chained prediction heads to propose several future tokens at once; backbone prefix verification then accepts the longest matching prefix. This reproduces the greedy-decoding rule, apart from rare finite-precision changes at near-tied logits when several query rows are evaluated together. For dense trajectories, the trajectory drafter proposes future denoising velocities and per-offset errors used to select a trusted horizon, calls the backbone only at its landing point, and falls back to an exact step when verification fails. We systematically evaluate PercepDraft across multiple generative perception models on object detection, panoptic segmentation, depth estimation, and surface normal estimation. The results demonstrate its broad applicability across generative backbones: PercepDraft substantially reduces backbone calls while largely pre-

serving task performance, achieving up to 6.3× end-to-end acceleration. Further ablations analyze how the drafting strategy, trusted-horizon selection, and verification rule shape the quality–efficiency trade-off.

In summary, our main contributions are as follows.

- We show that the computational redundancy of generative perception lies in the trajectory of the output rather than inside a single backbone call, and that structured sequence generation and dense trajectories exhibit the same form of step-to-step predictability.
- We propose PercepDraft, a draft-and-verify framework that freezes the backbone and introduces a token drafter and a trajectory drafter to accelerate structured token generation and dense-trajectory generation under verification.
- We validate PercepDraft on several perception tasks and several generative backbones, lowering the inference cost substantially with limited changes in task performance.

2 Related Work

Generative visual perception. Pix2seq casts object detection as language modeling over a sequence of boxes and categories (Chen et al., 2022), and UFO extends that interface to fine-grained perception (Tang et al., 2025). On the dense side Marigold repurposes a diffusion model for monocular depth (Ke et al., 2024) and DICEPTION covers depth, normals and segmentation (Zhao et al., 2025), while generalists such as BAGEL (Deng et al., 2025), SenseNova-Vision (SenseNova-Vision Team, 2026) and Vision Banana (Gabeur et al., 2026) unify understanding and generation. That line of work is concerned with task unification and perception quality; we study the inference inefficiency shared by both generation regimes.

Accelerating generative inference. For autoregressive generation, speculative decoding reduces the number of decoding rounds by proposing several following tokens and verifying them (Leviathan et al., 2023; Chen et al., 2023), and Medusa and EAGLE raise the efficiency of the proposal with extra heads or intermediate features (Cai et al., 2024; Li et al., 2024; Ankner et al., 2024). A second line shortens the context instead of the number of rounds, pruning visual tokens by the attention they receive (Chen et al., 2024; Baek et al., 2026; Yu et al., 2026). For diffusion generation, higher-order solvers reduce the number of sampling steps (Lu et al., 2022), and caching methods reuse or extrapolate the features of neighbouring timesteps (Ma et al., 2024; Liu et al., 2024a; Zou et al., 2025; Liu et al., 2025). Closest to our dense-trajectory branch is FlowCast (Bajpai et al., 2026), which extrapolates the remaining states of a flow-matching trajectory from the current velocity and verifies each of them with its own backbone call. These methods each target one form of open-ended generation and do not exploit the fixed syntax of structured perception records or account for the sensitivity of dense maps to geometric details and instance boundaries. PercepDraft instead targets the redundancy along the output trajectory, using backbone prefix verification for structured sequences and trusted-horizon selection with landing-point verification for dense outputs.

3 PercepDraft

3.1 Overview

PercepDraft is organized around a draft-and-verify framework: the backbone remains the verifier, while one of two lightweight drafters proposes the next part of the already-defined generation trajectory. Both regimes follow the same loop: read a verified state, draft a short continuation, evaluate the backbone once, and commit the verified progress or fall back to one exact step. The two drafters are specialized to their output representations: a token drafter with chained prediction heads for structured records, and a trajectory drafter that predicts future velocities and per-offset errors for dense maps; a trusted-horizon rule selects the draft length from those errors. Nothing in the model's weights, solver, or output format changes.

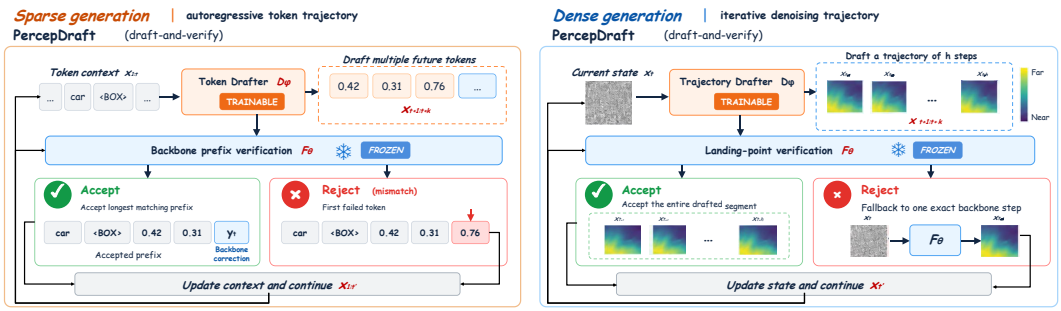


Figure 3: Detailed PercepDraft workflow with a frozen backbone. Left: the token drafter proposes a continuation; backbone prefix verification commits the longest match together with the backbone's next token and discards the first mismatch. Right: the trajectory drafter predicts velocities and per-offset errors used to select a trusted horizon; landing-point verification accepts the drafted segment or falls back to one exact step.

3.2 Problem Formulation

Let c denote the image and task condition and F_θ the frozen backbone; only lightweight drafters with parameters ϕ are trained. An autoregressive record is decoded as $y_i^\star = \arg\max_{w \in V} p_\theta(w \mid c, y_{<i})$ with $p_\theta(\cdot \mid c, y_{<i}) = F_\theta(c, y_{<i})$, one backbone call per token, and a dense output follows the sampler

$$v_i^\star = F_\theta(x_i^\star, t_i; c), \qquad x_{i+1}^\star = \Psi_i(x_i^\star, v_i^\star), \quad 1 = t_0 > \cdots > t_N = 0, \tag{1}$$

with Ψ_i the fixed solver update. The goal is to reduce calls while preserving the reference sequence for tokens, up to finite-precision effects, and the reference task quality for dense outputs.

3.3 Draft-and-Verify Computation

For the dense branch, at a verified state a lightweight drafter proposes a draft of length H, and one backbone call either commits the proposal or falls back to one exact step, so with acceptance probability q the expected progress per backbone call is $\mathbb{E}[A \mid H] = qH + (1-q) = 1 + q(H-1)$: acceleration needs both a long enough draft and a high enough acceptance rate. This treats a draft as accepted or rejected as a whole, which is the dense case; the token branch commits a variable-length prefix. After a dense rejection, the sampler reuses the already-verified anchor velocity v_i to take one exact step, so the fallback itself incurs no additional backbone call.

3.4 Autoregressive Records

Given a prefix p, let the token drafter propose $\hat{y}_{1:K}$. A single batched call evaluates

$$\bar{y}_j = \arg\max_{w \in V} p_\theta(w \mid c, p \,\|\, \hat{y}_{1:j-1}), \qquad j = 1, \ldots, K+1. \tag{2}$$

With $r = \min\{j \le K : \hat{y}_j \neq \bar{y}_j\}$, or $K+1$ when every proposal matches, the decoder emits the matching prefix and $\bar{y}_r$, so one call commits $A_{\text{tok}} = r$ tokens. Under deterministic greedy evaluation this is exact: if the first $j-1$ proposals match, the prefix used for $\bar{y}_j$ is the one sequential decoding would have produced, and induction gives every committed token, including the correction at the first mismatch. In bf16, a batched $K{+}1$-query forward changes the argmax at only 1.0% of Dense200 positions, almost entirely at near-tied coordinate digits.

Token drafter. Let h be the backbone hidden state, u_0 its next-token prediction, and $V_p \subset V$ the restricted record vocabulary. The token drafter uses chained prediction heads:

$$z_k = \mathrm{MLP}_k([h; E(u_{k-1})]), \qquad u_k = \arg\max_{w \in V_p} (W z_k)_w, \qquad k = 1, \ldots, K, \tag{3}$$

with acceptance estimates $q_k = \sigma(a_k^\top \,\mathrm{stopgrad}(z_k) + b_k)$, and drafting stops when $q_k < \eta$. With teacher-forced targets, only the token drafter is optimized:

$$\mathcal{L}_{\text{tok}} = \sum_{k=1}^{K} \left[\mathrm{CE}(W z_k, y_{k+1}^\star) + \lambda_{\text{acc}} \,\mathrm{BCE}\left(q_k, \mathbb{1}[u_k = y_{k+1}^\star]\right) \right]. \tag{4}$$

The restricted vocabulary and conditioning each prediction head on the preceding prediction are well suited to low-entropy syntax tokens, while prefix verification filters residual drafting errors. Prompt/grammar lookup and constant-velocity proposals are training-free baselines.

3.5 DENSE TRAJECTORIES

At a verified anchor (x_i, v_i, t_i), the trajectory drafter uses

$$s_i = \mathrm{Norm}([x_i; v_i; v_i - v_{i-1}; e(t_i)]), \qquad g_\phi(s_i) \mapsto \{\Delta\hat{v}_{i,h}, \hat{e}_{i,h}\}_{h=1}^{H}. \tag{5}$$

It predicts $\hat{v}_{i,h} = v_i + \Delta\hat{v}_{i,h}$ and rolls out the original fine-grained schedule, $\hat{x}_{i,0} = x_i$ and $\hat{x}_{i,h} = \Psi_{i+h-1}(\hat{x}_{i,h-1}, \hat{v}_{i,h-1})$ with $\hat{v}_{i,0} = v_i$. The error head is trained on frozen-backbone trajectories against $e^{\mathrm{ref}}_{i,h} = \|\hat{v}_{i,h} - v^{\mathrm{ref}}_{i+h}\|_2 / (\|v^{\mathrm{ref}}_{i+h}\|_2 + \epsilon)$, the residual of the trajectory drafter's own velocities on a recorded trajectory; this directly supervises the error estimates used for trusted-horizon selection. The trusted horizon is the longest prefix whose predicted errors remain below the threshold, $h_i = \max\{h : \max_{1 \le j \le h} \hat{e}_{i,j} \le \rho\tau\}$, and the backbone is called once at its landing point, giving the residual

$$r_{i,h_i} = \frac{\|F_\theta(\hat{x}_{i,h_i}, t_{i+h_i}; c) - \hat{v}_{i,h_i}\|_2}{\|F_\theta(\hat{x}_{i,h_i}, t_{i+h_i}; c)\|_2 + \epsilon}. \tag{6}$$

The drafted segment is accepted if $r_{i,h_i} \le \tau$; otherwise it is discarded and the sampler takes one exact step with v_i. Landing-point verification compares velocities only at the landing point, so it does not certify the intermediate states within the selected horizon.

Stability rationale. For an L-Lipschitz velocity field the Euler error over a drafted segment is bounded by the drafted velocity residuals weighted by the step sizes and amplified by $\exp(L \sum_j |\Delta_j|)$ (Appendix C.3.1), so the trusted-horizon rule selects regions whose predicted residuals stay small, and verification rejects the states where they do not. Because landing-point verification does not certify every intermediate residual, dense quality is evaluated empirically against same-budget controls in Section 4.2.

Training. Both drafters are trained in one run, with one optimizer and one checkpoint, on the frozen backbone's own outputs: the token drafter by teacher forcing on the backbone's records, and the trajectory drafter on recorded reference trajectories, where the error head's target is the residual of the trajectory drafter's own velocities so that the predicted errors used to select the trusted horizon are calibrated to the draft that is actually used; no task label enters either loss.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate PercepDraft on autoregressive object detection and iterative dense prediction (panoptic segmentation, monocular depth, and surface normals). SenseNova-Vision-7B-MoT supports all four tasks; we further test Marigold (Ke et al., 2025), Rex-Omni (Jiang et al., 2025), and Qwen2.5-VL (Bai et al., 2025) on their supported outputs. Each backbone is frozen and retains its native output format, solver, and decoding rules. Drafters are trained only on outputs from the corresponding frozen backbone, without task labels.

We evaluate detection on LVIS (Gupta et al., 2019), VisDrone (Zhu et al., 2021), Dense200 (SenseNova-Vision Team, 2026) and COCO (Lin et al., 2014); panoptic segmentation on COCO; depth on NYUv2 (Silberman et al., 2012) and KITTI (Geiger et al., 2012); and normals on NYUv2 and ScanNet (Dai et al., 2017), using standard metrics and official evaluators. The Qwen2.5-VL diagnostic scores COCO boxes without class-name matching. Methods within each comparison block share the same data, decoding, and evaluation code; we measure latency on one H100 GPU at batch size one.

We report full-backbone-equivalent calls, weighting partial cache forwards by the executed layer fraction. With classifier-free guidance, each sampler step evaluates two branches, so 49 steps equal 98 calls. Tables report calls, whereas Figure 4 shows sampler steps. Unless noted, tolerances selected on held-out images are 0.4 for depth and normals and 0.2 for panoptic segmentation; the token drafter uses eight chained prediction heads.

Table 1: Acceleration across tasks and backbones. Calls are full-backbone-equivalent evaluations and Tok./call denotes committed tokens per call. Bold and underlined values are best and second-best within each backbone; slash-separated metrics are ranked componentwise.

Detection (F1 at IoU 0.5)

Configuration	LVIS	VisDrone	Dense200	Tok./call↑	LVIS / VisDrone / Dense200 s/img↓	Speedup↑
Rex-Omni-3B	78.1	58.5	**73.6**	1.00	0.76/4.30/13.5	1.00×
+ PercepDraft	**78.6**	**58.8**	73.4	**1.56/1.71/1.62**	**0.57/3.01/7.94**	**1.3/1.4/1.7**×
Qwen2.5-VL-7B (COCO)		**38.0**		1.00	1.84	1.00×
+ Medusa (full vocab.) (Cai et al., 2024)		37.9		3.17	0.78	2.3×
+ PercepDraft		37.9		**3.21**	**0.71**	**2.6**×
SenseNova-Vision-7B	81.8	66.0	**81.3**	1.00	1.91/10.7/30.4	1.00×
+ Medusa (full vocab.)	**82.3**	65.6	79.4	3.12	1.50/8.33/28.7	1.3/1.3/1.1×
+ FastV (Chen et al., 2024)	80.1	**66.8**	77.5	1.00	1.91/9.24/25.1	1.0/1.2/1.2×
+ AgilePruner (Baek et al., 2026)	79.7	66.7	75.9	1.00	2.05/9.39/26.9	0.9/1.1/1.1×
+ PercepDraft	**82.3**	65.6	79.4	**3.54**	**1.25/6.65/23.5**	**1.5/1.6/1.3**×

Panoptic segmentation (COCO)

Configuration	PQ↑	Things↑	Stuff↑	Calls↓	s/img↓	Speedup↑
SenseNova-Vision-7B	**51.4**	**57.6**	42.8	98	12.1	1.00×
+ TeaCache (Liu et al., 2024a)	39.7	47.9	28.6	**31.1**	6.60	1.8×
+ DeepCache (Ma et al., 2024)	51.3	**57.6**	42.8	65.1	9.30	1.3×
+ TaylorSeer (Liu et al., 2025)	49.7	57.0	39.8	65.1	9.21	1.3×
+ FlowCast (Bajpai et al., 2026)	48.1	53.5	40.9	107.7	12.2	1.0×
+ PercepDraft	51.3	57.2	**43.2**	38.7	**5.91**	**2.0**×

Depth and surface normals

Configuration	Depth, AbsRel↓ / δ_1↑ NYUv2	Depth, AbsRel↓ / δ_1↑ KITTI	Normals, err.↓ / 11.25°↑ NYUv2	Normals, err.↓ / 11.25°↑ ScanNet	depth / normals Calls↓	s/img↓	Speedup↑
Marigold	6.02/96.0	12.97/**83.6**	16.98/61.0	15.37/65.8	50	0.88	1.00×
+ PercepDraft	**5.91/96.1**	**12.90**/83.5	**16.54/61.3**	**15.04/66.0**	**5.4/5.1**	**0.16/0.15**	**5.6/5.8**×
SenseNova-Vision-7B	4.05/97.9	6.23/95.0	13.88/65.0	12.69/69.3	98	8.66/8.70	1.00×
+ TeaCache (Liu et al., 2024a)	4.80/97.3	7.53/93.2	26.88/53.3	22.19/59.1	29.5/29.8	3.34/3.31	2.6×
+ DeepCache (Ma et al., 2024)	4.03/98.0	6.14/95.1	**13.77/65.1**	12.60/69.4	65.1	6.13/6.11	1.4×
+ TaylorSeer (Liu et al., 2025)	4.06/97.9	6.38/94.8	14.05/64.6	12.65/69.4	65.1	6.12/6.09	1.4×
+ FlowCast (Bajpai et al., 2026)	3.92/**98.1**	5.63/**96.0**	13.85/65.0	**12.58/69.5**	100/102	8.33/8.47	1.0×
+ PercepDraft	**3.87/98.1**	**5.60/96.0**	**13.77**/64.9	12.62/69.1	**13.4/15.8**	**1.38/1.61**	**6.3/5.4**×

4.2 Main results across tasks and backbones

Table 1 reports quality, calls, and latency across four generative backbones. PercepDraft reduces calls in both generation regimes with limited quality changes, reaching 6.3× end-to-end speedup.

Autoregressive detection. On SenseNova-Vision, PercepDraft raises committed tokens per call from 1.00 to 3.54. Latency falls from 1.91/10.7/30.4 to 1.25/6.65/23.5 seconds on LVIS/VisDrone/Dense200, while F1 changes from 81.8/66.0/81.3 to 82.3/65.6/79.4. Rex-Omni commits 1.6–1.7 tokens per call for 1.3–1.7× speedup. Qwen2.5-VL commits 3.21, reducing COCO latency from 1.84 to 0.71 seconds with F1 changing from 38.0 to 37.9. The gains therefore transfer across record grammars and coordinate encodings.

Dense prediction. On SenseNova-Vision, depth and normals require 13.4 and 15.8 rather than 98 calls, giving 6.3× and 5.4× speedups; NYUv2 AbsRel changes from 4.05 to 3.87 and mean normal error from 13.88° to 13.77°. For panoptic segmentation, calls drop from 98 to 38.7 and latency from 12.1 to 5.91 seconds, with PQ changing from 51.4 to 51.3. On Marigold, PercepDraft reduces 50 calls to 5.4 for depth and 5.1 for normals, with comparable quality and 5.6×/5.8× speedups. These gains span tasks, output spaces, and backbone architectures.

4.3 Comparison with existing acceleration methods

Autoregressive acceleration. Medusa uses the same number of heads as the token drafter but predicts over the full vocabulary. PercepDraft exploits the constrained record vocabulary, chains its heads, and commits the longest matching prefix plus the backbone's next token. At the same 37.9 F1 on Qwen2.5-VL, it raises committed tokens per call from 3.17 to 3.21 and speedup from 2.3× to 2.6×; on SenseNova-Vision, it reaches 3.54 rather than 3.12 tokens per call. FastV and AgilePruner reduce visual-encoding cost but retain token-wise decoding. Consequently, FastV reaches

Table 2: Accuracy comparison with task specialists. Generalist results come from Table 1; specialist results are from their papers. Bold and underlined values are best and second-best per metric.

Detection (F1↑)			
Model	LVIS	VisDrone	Dense200
G-DINO-T (Liu et al., 2024b)	77.3	43.4	25.0
Rex-Omni (Jiang et al., 2025)	78.1	58.5	73.6
SenseNova-Vision	<u>81.8</u>	**66.0**	**81.3**
+ PercepDraft	**82.3**	<u>65.6</u>	<u>79.4</u>

Panoptic segmentation (COCO)			
Model	PQ↑	Things↑	Stuff↑
Mask2Former-L (Cheng et al., 2022)	<u>55.2</u>	**66.0**	40.4
PSALM (Zhang et al., 2024)	**55.9**	–	–
X-SAM (Wang et al., 2026)	54.7	–	–
SenseNova-Vision	51.4	<u>57.6</u>	<u>42.8</u>
+ PercepDraft	51.3	57.2	**43.2**

Monocular depth (AbsRel↓ / δ_1↑) **and surface normals** (Mean err.↓ / 11.25°↑)				
	Depth		Normals	
Model	NYUv2	KITTI	NYUv2	ScanNet
DepthAnything-V2 (Yang et al., 2024)	4.31/97.9	8.06/93.9	–	–
DSINE (Bae & Davison, 2024)	–	–	16.4/59.6	16.2/61.0
MoGe-2 (Wang et al., 2025b)	**3.5**/<u>98.0</u>	**5.5**/**97.7**	14.7/62.3	12.8/68.4
Marigold (Ke et al., 2025)	5.96/96.2	12.85/83.7	16.18/62.0	15.06/65.9
DICEPTION (Zhao et al., 2025)	6.1/96.0	6.9/94.9	18.3/52.9	18.8/53.6
FE2E (Wang et al., 2025a)	4.1/97.7	6.6/<u>96.0</u>	16.2/59.6	13.8/67.2
Lotus-2 (He et al., 2025)	4.1/97.6	6.7/94.5	16.9/59.0	14.2/66.8
Vision Banana (Gabeur et al., 2026)	–	–	17.8/–	15.1/–
SenseNova-Vision	4.05/97.9	6.23/95.0	<u>13.88</u>/**65.0**	<u>12.69</u>/**69.3**
+ PercepDraft	<u>3.87</u>/**98.1**	<u>5.60</u>/<u>96.0</u>	**13.77**/<u>64.9</u>	**12.62**/<u>69.1</u>

only 1.0/1.2/1.2× speedup while losing 1.7 and 3.8 F1 on LVIS and Dense200, respectively. Removing decoding calls thus gives a stronger quality–efficiency trade-off than reducing encoding cost.

Dense-generation acceleration. Caching reduces selected calls but retains the iterative trajectory. DeepCache needs 65.1 calls for 51.3 panoptic PQ, whereas PercepDraft matches that PQ with 38.7; TeaCache uses 31.1 calls but drops to 39.7 PQ. For depth and normals, DeepCache and TaylorSeer use about 65 calls versus 13.4 and 15.8 for PercepDraft at comparable quality. FlowCast evaluates every drafted state, including those later rejected, and thus uses 100–108 calls. PercepDraft instead selects a trusted horizon from predicted per-offset errors and calls the backbone only at the landing point; acceptance commits the intervening states, while rejection falls back to one exact step.

4.4 Comparison with task-specialist models

Table 2 compares accuracy rather than latency because architectures and protocols differ. The accelerated generalist has the lowest mean normal error shown, ranks second to MoGe-2 on depth, and exceeds Rex-Omni by 4–7 detection F1. Specialists remain stronger on panoptic segmentation, whereas PercepDraft serves all four tasks with one frozen backbone and far fewer calls than its original inference procedure.

4.5 Mechanism analysis

Where the computation lies. Repetition along the output trajectory dominates cost. Detection decoding takes 97% of latency at 31 tokens per object, while encoding up to 4060 visual tokens takes under 3%; 49 dense sampler steps take 88% of an 8.8-second run. In Figure 5, skipping two early layers saves 2.3% of computation but raises depth AbsRel by up to 20.8 points. Holding two or four trajectory steps saves 4.1% or 8.2%, with only 0.3 or 0.6 AbsRel change; referring segmentation

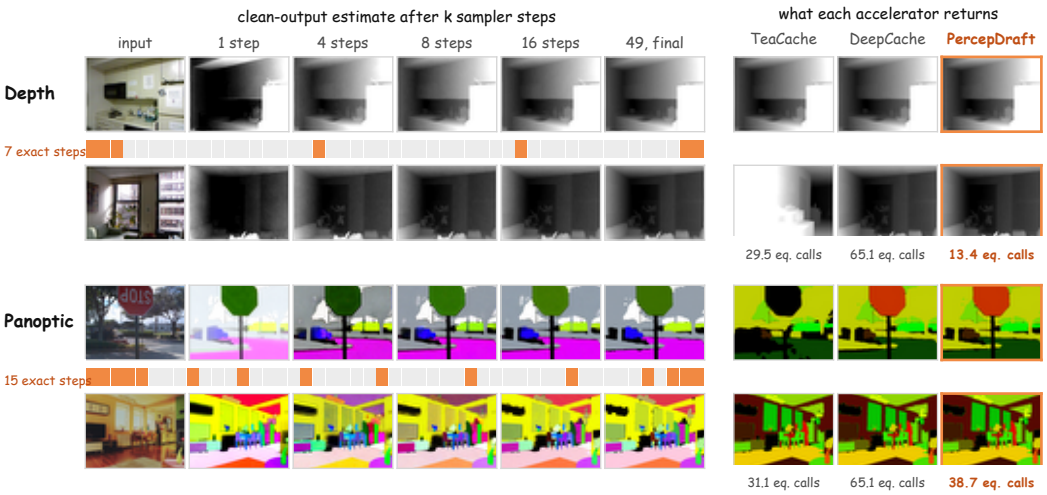


Figure 4: Representative PercepDraft depth and panoptic inference traces. Left: clean-output estimates across sampler steps, with exact evaluations marked. Right: final outputs and equivalent call counts. Depth maps use independent contrast ranges.

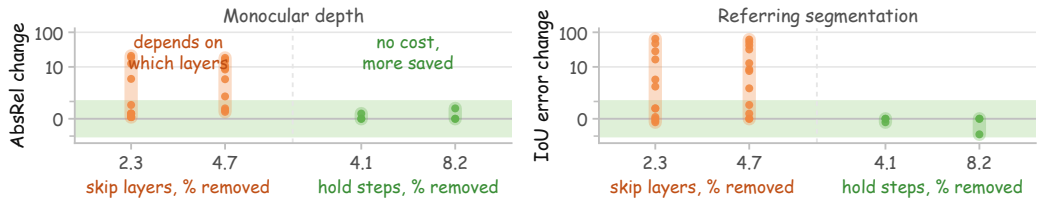


Figure 5: Reducing computation within a backbone call versus along the denoising trajectory. Points skip layers or sampler steps at one stage; the shaded band marks changes within one point of the reference. Positive values denote higher error; the vertical scale is symmetric logarithmic.

shows the same ordering. Thus, removing repeated calls gives a more reliable quality–efficiency trade-off than reducing each call, although individual calls may also contain redundancy.

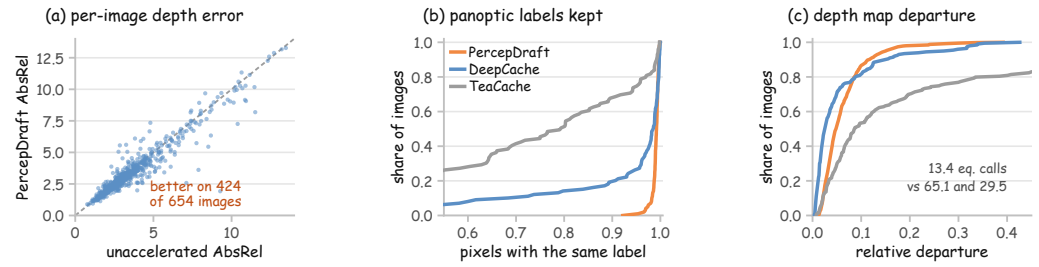


Figure 6: Per-image agreement with reference outputs: (a) NYUv2 depth AbsRel, (b) the fraction of unchanged pixel labels over 100 panoptic maps, and (c) relative depth deviation for methods with similar aggregate accuracy.

How predictability evolves. Dense predictability varies over time: relative velocity change is 74–93% between the first two steps, below 7% mid-trajectory, and 18–27% near the end. Unlike a fixed horizon, the trusted horizon adapts to this variation. For autoregressive records, structural tokens form about half of each record and have low uncertainty, which concentrates in class names and final coordinate digits. Final-digit entropy is 0.80 bit on Dense200 versus 2.54 bits on LVIS,

Table 3: Dense trajectory-drafting ablations over velocity source, horizon rule, and landing-point verification. Calls are backbone evaluations per image; values are not ranked because quality and computation trade off.

	Depth		Normals		Panoptic	
Setting	AbsRel↓	Calls↓	Mean↓	Calls↓	PQ↑	Calls↓
Reference runs						
Unaccelerated SenseNova-Vision-7B	4.05	98	13.88	98	51.4	98
PercepDraft (main result)	3.87	13.4	13.77	15.8	51.3	38.7
Velocity source × horizon rule						
Constant velocity + fixed horizon	3.84	16.2	13.73	40.6	51.3	47.2
Trajectory drafter + fixed horizon	3.93	30.3	13.81	33.8	51.3	33.6
Constant velocity + trusted horizon	3.93	15.6	13.86	18.2	49.2	28.2
Trajectory drafter + trusted horizon	3.85	13.7	13.79	14.9	51.4	29.9
Landing-point verification						
Tighter tolerance (0.2; 0.15 panoptic)	3.89	17.5	13.79	21.1	51.4	43.4
Looser tolerance (0.6; 0.3 panoptic)	3.86	12.0	13.74	14.0	50.1	37.4
Force-accept drafts (h=16)	3.90	12.0	14.17	12.0	28.7	12.0

explaining why gains from the token drafter vary across datasets. In Figure 4, a depth example evaluates only seven of 49 steps exactly, and a 116-token Dense200 record needs 26 calls.

Why verification matters. Figure 7 shows why a single fixed cutoff is unreliable. Depth can exhibit mild late-stage overshooting, but its best cutoff varies by image, whereas panoptic quality benefits from later updates. PercepDraft does not terminate early: it reaches the prescribed terminal time, adapting each span with the trusted horizon and accepting it only after landing-point verification. This handles both behaviors while reducing backbone evaluations. Per-image results further show lower depth error on 424/654 NYUv2 images and median panoptic label agreement of 99.2%, versus 98.5% for DeepCache and 79% for TeaCache. We use $\rho = 0.8$ because the error head slightly underestimates residuals. Landing-point verification checks the endpoint, not every intermediate state; empirically, it captures the dominant failures on these tasks.

4.6 Component ablations

The main row uses the jointly trained checkpoint from Table 1; the velocity–horizon grid uses a separately trained trajectory drafter, so we interpret its trends rather than small error differences. The three components are complementary. With velocities predicted by the trajectory drafter, trusted-horizon selection reduces depth/normal calls from 30.3/33.8 to 13.7/14.9 while changing errors from 3.93/13.81 to 3.85/13.79. At about 30 panoptic calls, the trajectory drafter raises PQ from 49.2 to 51.4 over constant velocity: the trusted horizon controls how far to advance, while the trajectory drafter controls landing accuracy. Landing-point verification prevents error accumulation. Force-accepting length-16 drafts uses 12 calls and keeps depth at 3.90 AbsRel, but raises normal error to 14.17° and collapses panoptic PQ to 28.7. Looser tolerances reduce depth/normal calls with little quality change, but a 0.3 panoptic tolerance lowers PQ from 51.3 to 50.1; 0.2 is the loosest tested value within 0.1 PQ of the reference. Uncertainty-weighted verification and landing-point reintegration do not improve this trade-off.

5 Conclusion

Generative perception repeatedly calls the same backbone along output trajectories that are locally predictable. PercepDraft retains the frozen backbone as the verifier and trains a token drafter and a trajectory drafter solely on backbone outputs. The token drafter proposes future tokens, while the trajectory drafter predicts future velocities and per-offset errors used to select a trusted horizon. With one verification call per draft, PercepDraft removes most backbone calls for dense prediction and substantially reduces autoregressive decoding calls, while largely preserving task performance and achieving up to 6.3× end-to-end speedup across four backbones.

ETHICS STATEMENT

This work accelerates inference of existing perception models and introduces no new data collection. All evaluations use public benchmarks under their own licences and publicly released model weights, and no human subjects or personally identifying data are involved. The method changes how many backbone calls an output costs, not what the model is trained to predict, so the method does not address biases in the underlying backbones; accelerated deployments should re-evaluate those biases. The consistency measurements of Section 4.5 report how far accelerated outputs move from the reference ones, which is the quantity a deployment would need to check. Cheaper dense perception lowers the cost of visual surveillance as well as of its benign uses, a risk this method shares with efficiency work on perception models generally.

REPRODUCIBILITY STATEMENT

Appendices A and B specify the evaluation data and protocols, drafter architectures and parameter counts, optimizer, schedule, hardware, and settings for the additional backbones. Both drafters are trained only on outputs from the frozen backbone, with no task labels. The task-specific tolerances, chunk lengths and horizon rules are reported alongside the runs that use them. The supplementary material contains the implementation of both drafters, the two draft-and-verify loops, the cache baselines and the patches applied to the backbone; code and drafter weights will be released.

THE USE OF LARGE LANGUAGE MODELS (LLMS)

Large language models were used as a general-purpose assistant in this work: for drafting and editing prose, for writing evaluation and plotting scripts, and for summarising related work. They were not used to generate experimental results. Every number reported in this paper comes from a run of the released models on the stated data, and the authors verified the measurements and take responsibility for the content.

REFERENCES

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for Medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.

Gwangbin Bae and Andrew J. Davison. Rethinking inductive biases for surface normal estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Changwoo Baek, Jouwon Song, Sohyeon Kim, and Kyeongbo Kong. AgilePruner: An empirical study of attention and diversity for adaptive visual token pruning in large vision-language models. *arXiv preprint arXiv:2603.01236*, 2026.

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

Divya Jyoti Bajpai, Shubham Agarwal, Apoorv Saxena, Kuldeep Kulkarni, Subrata Mitra, and Manjesh K. Hanawal. FlowCast: Trajectory forecasting for scalable zero-cost speculative flow matching. *arXiv preprint arXiv:2602.01329*, 2026.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning (ICML)*, 2024.

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.

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

Ting Chen, Saurabh Saxena, Lala Li, David J. Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. In *International Conference on Learning Representations (ICLR)*, 2022.

Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Xueqing Deng, Qihang Yu, Peng Wang, Xiaohui Shen, and Liang-Chieh Chen. COCONut: Modernizing COCO segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Valentin Gabeur, Shangbang Long, Songyou Peng, et al. Image generators are generalist vision learners. *arXiv preprint arXiv:2604.20329*, 2026.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Agrim Gupta, Piotr Dollár, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Jing He, Haodong Li, Mingzhi Sheng, and Ying-Cong Chen. Lotus-2: Advancing geometric dense prediction with powerful image generative model. *arXiv preprint arXiv:2512.01030*, 2025.

Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction. *arXiv preprint arXiv:2510.12798*, 2025.

Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Bingxin Ke, Kevin Qu, Tianfu Wang, Nando Metzger, Shengyu Huang, Bo Li, Anton Obukhov, and Konrad Schindler. Marigold: Affordable adaptation of diffusion-based image generators for image analysis. *arXiv preprint arXiv:2505.09358*, 2025.

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

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning (ICML)*, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. *arXiv preprint arXiv:2411.19108*, 2024a.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with TaylorSeers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *European Conference on Computer Vision (ECCV)*, 2024b.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Apoorv Saxena. Prompt lookup decoding. https://github.com/apoorvumang/prompt-lookup-decoding, 2023.

SenseNova-Vision Team. Vision as unified multimodal generation. *arXiv preprint arXiv:2607.06560*, 2026.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *European Conference on Computer Vision (ECCV)*, 2012.

Hao Tang, Chenwei Xie, Haiyang Wang, Xiaoyi Bao, Tingyu Weng, Pandeng Li, et al. UFO: A unified approach to fine-grained visual perception via open-ended language interface. *arXiv preprint arXiv:2503.01342*, 2025.

Hao Wang, Limeng Qiao, Zequn Jie, Zhijian Huang, Chengjian Feng, Qingfang Zheng, Lin Ma, Xiangyuan Lan, and Xiaodan Liang. X-SAM: From segment anything to any segmentation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

JiYuan Wang, Chunyu Lin, Lei Sun, Rongying Liu, Lang Nie, Mingxing Li, Kang Liao, Xiangxiang Chu, and Yao Zhao. From editor to dense geometry estimator. *arXiv preprint arXiv:2509.04338*, 2025a.

Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. MoGe-2: Accurate monocular geometry with metric scale and sharp details. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything V2. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Hanxun Yu, Wentong Li, Xuan Qu, Song Wang, Junbo Chen, and Jianke Zhu. VisionTrim: Unified vision token compression for training-free MLLM acceleration. *arXiv preprint arXiv:2601.22674*, 2026.

Zheng Zhang, YeYao Ma, Enming Zhang, and Xiang Bai. PSALM: Pixelwise segmentation with large multi-modal model. In *European Conference on Computer Vision (ECCV)*, 2024.

Canyu Zhao, Yanlong Sun, Mingyu Liu, Huanyi Zheng, Muzhi Zhu, Zhiyue Zhao, et al. DICEPTION: A generalist diffusion model for visual perceptual tasks. *arXiv preprint arXiv:2502.17157*, 2025.

Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *International Conference on Learning Representations (ICLR)*, 2025.

A Datasets and Evaluation Protocols

A.1 Benchmark Datasets

We evaluate detection on LVIS (Gupta et al., 2019), VisDrone (Zhu et al., 2021), Dense200 (SenseNova-Vision Team, 2026) and COCO (Lin et al., 2014) with the released evaluator at IoU 0.5, and panoptic segmentation on COCO with the official panoptic evaluator. Depth estimation is evaluated on NYUv2 (Silberman et al., 2012) and KITTI (Geiger et al., 2012), while surface normal estimation is evaluated on NYUv2 and ScanNet (Dai et al., 2017). Geometry outputs are aligned and scored with the evaluation code distributed with Marigold. Within every comparison block, all methods use identical data, preprocessing, decoding, and scoring. The separate samples used for runtime, ablation, and diagnostic studies are described with those experiments.

A.2 Evaluation Metrics and Protocols

Table 1 uses the following conventions. Detection is scored as F1 at IoU 0.5. Panoptic segmentation is scored with PQ, Things, and Stuff; the instance list is decoded with prompt lookup for every method in that block. Geometry uses AbsRel and δ_1 for depth, and mean angular error and the fraction of pixels within 11.25° for surface normals. Dense outputs use 49 denoising steps with classifier-free guidance, so the unaccelerated SenseNova-Vision baseline requires 98 backbone calls. Text efficiency is reported as committed tokens per backbone call.

The cache baselines use data-dependent backbone-call counts averaged over the corresponding benchmarks; shallow layer-cache forwards are counted by the fraction of layers executed. The main configuration uses tolerance 0.2 for panoptic segmentation and 0.4 for geometry; alternative settings in the frontier and ablation sweeps are labeled with the corresponding experiment. Wall-clock measurements use the shared optimized runtime described in Section 4.

The measurements of Figure 1 decode the sampler's clean-output estimate, $x_t - t\, v_t$, with the model's own VAE at selected steps of the unchanged 49-step schedule and score it with the evaluators of Section 4; the velocity change is measured on the recorded reference trajectories used to train the trajectory drafter, and the token entropies are read from the backbone's next-token distribution under greedy decoding. The depth-axis probe evaluates all 28 layers. The intervention grid uses the same noise and schedule across 64 images; its detailed stage definitions and oracle analysis are given in Appendix C.3.3.

B Implementation Details

B.1 Backbone-Specific Settings

Marigold (Ke et al., 2025) is a latent diffusion model with fifty DDIM steps as its released schedule, run with a single sample; v1.0 is used for depth and v1.1 for normals, and its trajectory drafter is trained on its own trajectories over 400 training images. Latency is measured with 50 images per set. The constant-velocity draft with the trusted horizon reaches 9.5 calls on depth and 8.1 on normals at the reference quality, against 5.4 and 5.1 for the proposal from the trajectory drafter. Marigold v1.1 normals was trained for few-step inference, and a uniform eight-step schedule already matches its fifty-step reference (16.53 mean error on NYUv2 and 15.07 on ScanNet at 4.6×); the trajectory drafter reaches the same quality in five calls.

Rex-Omni (Jiang et al., 2025) writes each coordinate as a single bin token and its released decoding applies a repetition penalty, which the verifier reproduces exactly; its token drafter is trained on the model's own decoding of two thousand COCO training images, and because its records are an order of magnitude longer on Dense200 than on LVIS, the committed-tokens-per-call metric is measured on a 40-image diagnostic sample for each benchmark. The two draft sources are complementary, lookup on the repetitive records of Dense200 (1.03/1.19/1.38 committed tokens per call on

LVIS/VisDrone/Dense200) and the token drafter on the short records of LVIS (1.56/1.30/1.21), so PercepDraft uses lookup where the prefix repeats an n-gram and the token drafter elsewhere.

Qwen2.5-VL (Bai et al., 2025) is not a detector and is prompted to list every object as JSON, so its F1 is scored against COCO boxes without regard to the category name and is low in absolute terms; its F1 changes from 38.0 to 37.9 under both accelerators.

B.2 Drafter Architecture, Training, and Verification

The PercepDraft token drafter uses a hidden width of 1024 and a shared output layer over the 1718 token ids that occur in the detection and panoptic answers of the training corpus. It has 94M trainable parameters, 1.3% of the backbone, and reuses the backbone's frozen embedding table. The trajectory drafter has 1.6M parameters. The token drafter is trained on official-format detection records from COCO, LVIS, and VisDrone and panoptic instance lists from COCONut-B (Deng et al., 2024) and LVIS, all filtered against the evaluation images. The trajectory drafter is trained on 18k reference trajectories recorded from the backbone on COCO training images, 6k per dense task. The two are trained in the same run for 20k steps with AdamW at a learning rate of 10^{-3} on four H100 GPUs with the frozen backbone in bf16, and the checkpoint of that run is used throughout. Prefix verification uses $K{+}1$ query rows in one batched backbone call; a draft token is accepted only if it equals the backbone's argmax at its position. For denoising, PercepDraft integrates the trajectory drafter's velocities over the trusted horizon; the landing-point velocity is compared with the drafted one by relative ℓ_2 distance over the whole latent, the first two and the last step are always computed exactly, and a rejection is followed by one exact step before PercepDraft resumes.

C Additional Experiments

C.1 Quality and Runtime

C.1.1 Quality–Efficiency Trade-offs

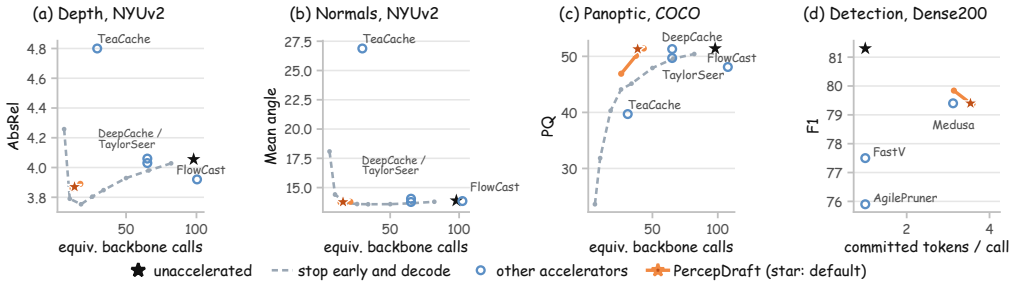


Figure 7: Quality–cost trade-offs for detection and three dense-prediction tasks. Dense-task points vary the landing-point tolerance, while detection points vary the number of chained prediction heads; dashed curves denote early stopping at two backbone calls per denoising step. DeepCache and TaylorSeer coincide on the two geometry panels, where only DeepCache is labelled. All points are measured on the corresponding benchmark data.

C.1.2 Concurrent Throughput

Table 1 measures one image at a time, which is the setting a draft-and-verify method is usually questioned about: a saturated server may have no idle capacity for the lightweight drafters to exploit. We therefore ran the same images with C decoding streams sharing one H100 and measured the throughput of the whole run with the model-loading time removed. Four streams do not fit, since four copies of the backbone exceed the memory of the card, so the comparison is between one and two.

The unaccelerated backbone gains nothing from the second stream, 0.172 to 0.167 images per second on detection and 0.099 to 0.102 on depth, because its calls already keep the device busy. PercepDraft

Table 4: Concurrent throughput (images/s) on one H100. Detection uses 60 LVIS images and depth uses 40 NYUv2 images per stream; comparisons are made within this table.

Configuration	1 stream	2 streams	vs. baseline (2 streams)
SenseNova-Vision-7B, detection	0.172	0.167	–
+ PercepDraft, detection	0.449	0.621	3.7×
SenseNova-Vision-7B, depth	0.099	0.102	–
+ PercepDraft, depth	0.370	0.494	4.8×

gains 38 and 33 percent, so its advantage does not disappear when the device is shared; the remaining gaps in its own timeline, drafter computation and the host-side work between calls, are what the second stream fills.

C.1.3 Per-Image Latency

Table 5 lists the per-image wall-clock of the settings of Table 1, each measured alone on one H100 at batch size one on a fixed 20-image latency sample, with the first image excluded from the mean as warm-up. Because this diagnostic sample differs from the benchmark evaluation, comparisons should be made within this table rather than against the benchmark-wide averages in Table 1. The SenseNova-Vision row is greedy decoding in the same implementation as the accelerated rows. For denoising, the trajectory drafter adds only lightweight vector operations, so the reduction in calls carries over to wall-clock almost fully. For text the saving does not: on Dense200 a single-token step costs 10.2 ms whereas a verification step with nine query rows costs 27.6 ms, so committing 3.6 tokens per call yields 1.3×, and the multi-token rows of Table 1 differ from each other mostly in how many calls they need rather than in the cost of a backbone call. The image context of a 1024-pixel input is several thousand key-value entries, and attention over it for nine queries at once does not take the single-query decode path; a verification kernel specialised for a few queries over a long context would recover much of the difference, and we leave it as engineering.

Table 5: Per-image latency on one H100 at batch size one. Measurements use a fixed 20-image sample per benchmark, with the first image excluded as warm-up.

Detection (s/img)			
Configuration	LVIS	VisDrone	Dense200
SenseNova-Vision	1.91	10.7	30.4
+ Grammar Lookup (Saxena, 2023)	1.59	8.47	26.5
+ Medusa (full-vocab. heads) (Cai et al., 2024)	1.50	8.33	28.7
+ PercepDraft (K=8)	1.25	6.65	23.5
Panoptic segmentation (COCO, s/img)			
Configuration			s/img
SenseNova-Vision			12.1
+ PercepDraft (τ=0.2)			5.6
Geometry (s/img)			
Configuration		NYUv2 depth	NYUv2 normals
SenseNova-Vision		8.66	8.70
+ PercepDraft (τ=0.4)		1.44	1.58

C.2 Ablations and Alternative Designs

C.2.1 Baseline and Intra-Call Controls

Unless "No CFG" is shown, the classifier-free guidance scale is 4.0; the unaccelerated SenseNova-Vision baseline uses its unchanged released schedule.

Visual-token pruning was measured on the detection benchmarks with the attention rule of Chen et al. (2024) at layer two. Keeping three quarters of the patches gives 77.8 F1 on Dense200 at 25.8 seconds per image and keeping half gives 77.5 at 25.1, against 81.3 at 30.4 for the unacceler-

Table 6: Effect of halving the denoising schedule and removing classifier-free guidance on COCO panoptic quality (500 validation images).

Configuration	PQ	Things	Stuff	RQ
SenseNova-Vision	49.9	53.9	44.1	60.9
+ Half schedule	49.1	53.7	42.4	59.7
+ No CFG	46.6	49.8	41.7	57.4
+ Half schedule (No CFG)	46.5	49.7	41.8	57.3

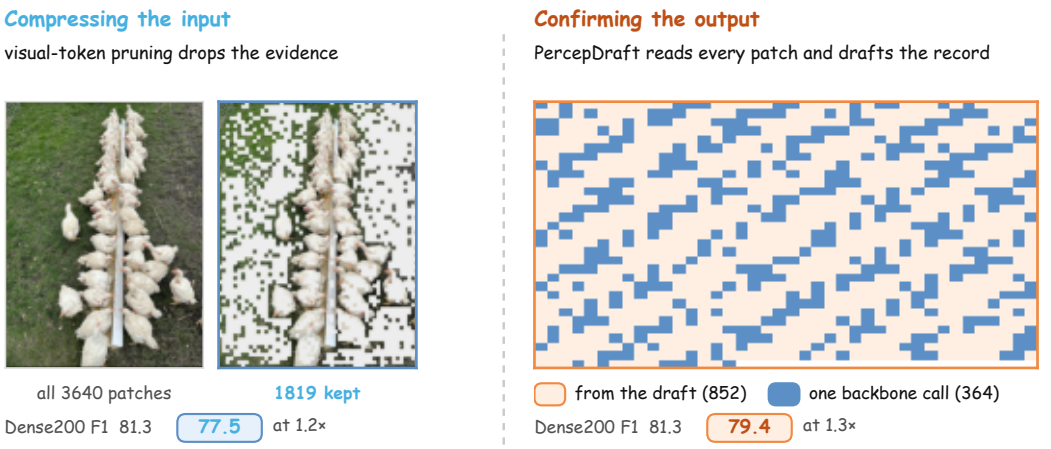


Figure 8: Visual-token pruning (left) and draft-token acceptance (right) for the same Dense200 image. Right-panel cells follow decoding order: 852 of 1,216 tokens are accepted from the draft and 364 are generated by backbone calls.

ated model, so most of the loss appears as soon as any patch is dropped while the wall-clock gain stays near 1.2×. Under PercepDraft the same rule gives 80.4/65.4/76.8 F1 on LVIS, VisDrone and Dense200 at 1.35/6.22/23.5 seconds, against 82.3/65.6/79.4 at 1.25/6.65/23.5 for PercepDraft alone.

Layer skipping and width pruning were probed on the generation expert with agreement with the SenseNova-Vision output as the metric (8 COCO images per task). Skipping the four layers with the highest input–output cosine keeps agreement at 0.97 for depth, 0.975 for normals and 0.89 for panoptic masks; skipping eight layers collapses depth to 0.66 and panoptic to 0.09. Pruning half of the MLP width by Wanda-style importance keeps depth at 0.947 and normals at 0.953 but drops binary masks to 0.809 and panoptic to 0.482. Removing all visual encoder tokens from the generation context leaves depth at 0.982 and normals at 0.988 but drops category segmentation to 0.858.

C.2.2 Velocity Source and Trusted Horizon

Table 7 isolates the two components of dense drafting on SenseNova-Vision: the velocity source, which is either constant velocity or the trajectory drafter's velocities, and the horizon rule, which is either fixed with doubling or selects a trusted horizon from the trajectory drafter's predicted errors. The tolerance is 0.3 for the rows without the trusted horizon and the per-task value for the rows with it. The study uses a trajectory drafter trained alone under the recipe of Section 3; the checkpoint of the joint run used in Table 1 gives 3.87 AbsRel at 13.4 calls on depth, 13.77 degrees at 15.8 calls on normals and 51.3 PQ at 38.7 calls on panoptic segmentation against the last row here, so the separation of the components carries over.

C.2.3 Token-Drafting Accuracy and Output Consistency

The backbone prefix verification rule is exact under greedy decoding, so the accelerated detection output should match the unaccelerated one token for token. In practice the two differ slightly. The reference decoder commits tokens one at a time with a single-query forward; the accelerated verifier evaluates $K+1$ query rows in one forward, which runs the same weights through a different

Table 7: Ablation of the velocity source and horizon rule on SenseNova-Vision. Calls are backbone evaluations per image.

	Depth (AbsRel / δ_1)		Normals (mean / 11.25°)		Panoptic	
Velocity source, horizon rule	NYUv2	Calls	NYUv2	Calls	PQ	Calls
SenseNova-Vision	4.05/97.9	98	13.88/65.0	98	51.4	98
Const. velocity, fixed h	3.84/98.1	16.2	13.73/65.3	40.6	51.3	47.2
Const. velocity, no verification	3.90/98.0	12.0	–	–	–	–
Trajectory drafter, fixed h	3.93/98.1	30.3	13.81/65.0	33.8	51.3	33.6
Const. velocity, trusted horizon	3.93/98.1	15.6	13.86/65.0	18.2	49.2	28.2
Trajectory drafter, trusted horizon (PercepDraft)	3.85/98.1	13.7	13.79/64.9	14.9	51.4	29.9

Table 8: Accepted draft tokens by syntactic class on Dense200 (200 images, $K{=}8$). Rate is acceptance among proposals in each class; share is the class's fraction of all accepted tokens.

			Syntax		Coordinates		Class name
Drafter	Tok./call↑	F1↑	rate	share	rate	share	accept.
Grammar-guided lookup (Saxena, 2023)	3.16	80.6	0.38	0.69	0.16	0.30	0.74
Parallel heads (restricted vocab.) (Cai et al., 2024)	3.46	79.4	0.41	0.65	0.21	0.35	0.03
Medusa (parallel full-vocabulary heads) (Cai et al., 2024)	3.12	79.4	0.36	0.66	0.18	0.34	0.02
PercepDraft	3.54	79.4	0.44	0.65	0.21	0.35	0.03

bf16 GEMM path. Teacher-forcing the reference sequence through the multi-query forward on 21 Dense200 images (18,815 positions) flips the argmax at 1.0% of positions for $K{+}1 \in \{2, 4, 9\}$ alike, almost entirely on coordinate digits (1.4%, template tokens under 0.1%, class names zero), and every flip moves a digit by one at a position where the top-two logits are within bf16 resolution. The same diagnostic on Qwen2.5-VL and Rex-Omni reproduces the pattern: every divergence occurs at a top-two margin of zero and swaps adjacent digits or bins. A flipped digit shifts a box edge by one pixel and, because decoding continues from the altered prefix, subsequent tokens can legitimately differ; this is the mechanism behind the sub-point F1 differences in Table 1. Recomputing near-tie positions with an exact single-query step recovers most of the gap, from 79.3 to 80.4 F1 for prompt lookup on Dense200, at about 15% of the throughput gain; we report the unguarded numbers because the guard trades speed for a numerical property that greedy decoding at batch size one does not itself guarantee across kernels.

C.2.4 Alternative Verification Designs

Richer verification rules were tried on panoptic segmentation, but none improves on the tolerance rule. Weighting the relative error with an auxiliary per-token uncertainty head restores 51.5 PQ but rejects long segments, requiring 80 calls at the default safety factor versus 30 for the tighter tolerance. Retraining the error head for that weighted check cuts rejections from 11 to 1.7 per image but still requires 44 calls at 51.4 PQ. Re-integrating an accepted segment with the verified velocity costs one to five PQ regardless of how the correction is distributed. Recomputing only the most uncertain tokens reaches 51.4 PQ at 31 calls with a fifth of the tokens, which the tighter tolerance already provides. Capping the horizon at six steps reaches the same frontier from the other side, 51.4 PQ at 27.8 calls, while raising the safety factor from 0.8 to 0.9 only adds rejections. Giving the trajectory drafter the input image's VAE latent as an extra feature leaves depth and normal errors unchanged at the reported precision and moves panoptic quality by 0.2 PQ, so the latent is not used by the main drafter.

C.2.5 FlowCast Sensitivity Analysis

The FlowCast rows of Table 1 use our tolerance of 0.3. Table 9 varies its threshold and chunk length on the same 100 panoptic images so that the comparison is not read as a choice of threshold that favors us. A tight threshold recovers the reference quality and pays for it with 120 backbone calls per image, 22% more than the trajectory it accelerates, because eleven chunks per image are rejected after their states have all been evaluated; loosening the threshold lowers the call count toward the

reference but never below it, while the accepted drafts increasingly damage the output; halving the chunk length reduces the waste per rejection and keeps quality, at 105 calls. No setting of FlowCast uses fewer calls than the unaccelerated reference trajectory on this backbone, because every drafted state is still evaluated.

Two caveats. FlowCast verifies its drafted states in one batched forward while ours verifies them serially and stops at the first failure, so the call counts above are favourable to it, but its wall-clock entry in Table 1 is our serial implementation rather than a measurement of the released system; batching shortens the serial depth rather than the work, since one call on 4096 latent tokens already runs near the arithmetic peak of the device. Its threshold is an MSE on the state while ours is a relative ℓ_2 residual on the velocity, which is why the sweep varies it over a wide range instead of quoting one setting.

Table 9: FlowCast sensitivity on COCO panoptic segmentation (100 diagnostic images). The reference uses 98 backbone calls and reaches 51.4 PQ.

k	ϵ	PQ↑	Calls↓	Rejections / image
16	0.15	51.5	120.0	11.0
16	0.30	48.1	107.7	4.8
16	0.50	42.0	102.4	2.2
8	0.30	51.0	104.8	3.4

C.3 Mechanism Diagnostics

C.3.1 Stability Bound for Drafted Segments

Assume the velocity field is L-Lipschitz in x and let $\delta_j = \|\hat{v}_j - F_\theta(\hat{x}_j, t_j; c)\|_2$ be the residual of the drafted velocity at step j. The Euler error obeys

$$\epsilon_{j+1} \le (1 + L|\Delta_j|)\epsilon_j + |\Delta_j|\delta_j, \qquad \epsilon_j = \|\hat{x}_j - x_j^\star\|_2, \tag{7}$$

and hence

$$\epsilon_{i+h} \le \exp\left(L \sum_{j=i}^{i+h-1} |\Delta_j|\right)\left(\epsilon_i + \sum_{j=i}^{i+h-1} |\Delta_j|\delta_j\right). \tag{8}$$

The reference here is the exact trajectory restarted from the same anchor, for which $\epsilon_i = 0$; measured against the original trajectory the drafted segment inherits whatever error the state already carries. The trusted-horizon rule selects regions whose predicted residuals remain small, while verification rejects inconsistent states, which addresses fixed stale velocity without replacing the original fine-grained schedule by uncontrolled large steps.

C.3.2 Computation Profile and Predictability

This subsection provides the profiling and predictability measurements that motivate trajectory-level verification. They are not required to reproduce the main comparisons, but clarify where computation is spent and why reducing the number of generation steps is more reliable than uniformly reducing the computation within a backbone call.

This instrumented profile is separate from the optimized latency benchmark. In particular, its panoptic row includes autoregressive instance-list generation, whereas the main panoptic comparison uses the same prompt lookup for every method. The times are therefore not compared across tables; only the within-profile stage shares are used here.

Across ten equal record-position bins, mean next-token entropy ranges from 0.16 to 0.19 bit on Dense200 and from 0.36 to 0.51 bit on LVIS, so the classwise pattern is not driven by a single part of the record.

C.3.3 Intervention and Routing Analysis

For each budget of Table 12 we compare the best single intervention with a per-sample oracle that picks, for every image, the intervention with the lowest error at that budget. On depth the oracle

Table 10: SenseNova-Vision latency profile by task on one H100 at batch size one.

Task	Total	Dominant stage	Share
Detection (per box)	0.75 s	token-by-token decoding, 31 tokens per box	97%
Detection prefill	0.08–0.23 s	image encoding, up to 4060 ViT tokens	<3%
Binary mask / depth / normal	8.8 s	49 denoising steps, two backbone calls each	88%
Panoptic segmentation	25.1 s	instance list 15.8 s, denoising 8.3 s	63% + 33%

Table 11: Predictability measurements underlying Figure 1: decoded quality across denoising steps, relative velocity change, and token statistics by syntactic class. Sample counts are 200 NYUv2 images, 100 COCO images, 100 trajectories per task, and 20 Dense200/LVIS images; $H<0.05$ denotes the near-deterministic token share.

Clean-output estimate after k denoising steps									
k	1	2	4	8	16	24	32	40	49
Depth, AbsRel ↓	8.03	7.21	3.90	3.74	3.84	3.92	3.97	4.02	4.05
Normals, mean angular error ↓	33.76	35.63	15.38	13.75	13.58	13.58	13.65	13.77	13.88
Panoptic, PQ ↑	24.9	14.2	30.3	39.2	45.3	47.8	49.2	50.3	51.4
Velocity change from step k to $k+1$ (%)									
k	1	2	4	8	16	24	32	40	48
Depth	73.7	29.2	3.9	3.1	2.8	2.8	3.0	3.8	21.4
Normals	93.4	52.6	8.0	4.0	3.5	3.5	3.8	4.8	26.5
Panoptic	86.6	70.2	17.8	6.9	3.8	3.3	3.5	4.3	17.6

Detection tokens by syntactic position						
	Dense200			LVIS		
Token class	Share	Entropy	$H<0.05$	Share	Entropy	$H<0.05$
Syntax	48%	0.00	100%	46%	0.03	95%
Class name	0%	0.02	97%	9%	0.13	94%
Leading digit	13%	0.00	100%	11%	0.00	100%
1st decimal	13%	0.26	69%	11%	0.45	46%
2nd decimal	13%	0.31	57%	11%	0.94	10%
3rd decimal	13%	0.80	22%	11%	2.54	0%

improves the mean error by at most 0.5 AbsRel points over the best fixed choice (four budgets between 0.02 and 0.13 points, the largest at the 2.3% budget where seven candidates compete), and on RefCOCOg by at most 0.5 IoU points except at the two budgets where eight layer-skip candidates compete and the oracle can exploit per-image noise (2.7 and 5.5 points). The relative L_2 change of the final latent caused by an intervention has a correlation with the change of the task error between -0.2 and $+0.3$ for almost every configuration; the exceptions are early-stage holds on RefCOCOg, where the correlation is negative. A router that watches how much the latent moves therefore cannot tell a harmful intervention from a harmless one.

The depth-routing probe gives panoptic quality 17.0 on its 100-image diagnostic sample at seventy percent capacity and 1.8 at fifty percent capacity. At batch size one, the arithmetic saved by routing does not shorten text decoding because the backbone call is memory-bandwidth bound.

C.3.4 Landing-Point Verification

Section 4.5 disables the landing-point verification decision on depth by setting the tolerance to ten, so that every drafted segment is kept regardless of what the backbone reports at its landing point. Table 13 gives the full measurement. For each image we record the relative velocity error the backbone measured at every landing point and the per-image departure of the resulting depth map from the 98-call reference, then rank-correlate the two.

The three tasks span the useful range of the score. On depth the measured errors stay below 0.2 on average, no tested horizon harms the output, and the correlation with per-image departure is absent because there is nothing to correlate with. On normals the errors reach 0.3 at k=16, the output begins to degrade, and the per-image split shows the degradation is carried entirely by the quartile of images whose worst drafted segment scored highest, while the remaining images end slightly better

Table 12: Layer-skipping and step-holding interventions at early (S1), middle (S2), and late (S3) trajectory stages. Saved is compute reduction; entries are task-metric changes from the unaccelerated baseline.

Intervention details			*Depth (NYUv2)*			*Referring segmentation (RefCOCOg, Mao et al., 2016)*		
Setting	Location	Saved (%)	S1	S2	S3	S1	S2	S3
Skip 2 layers	4–5	2.3%	+20.8	+18.6	+12.4	+64.7	+47.5	+28.3
	11–12	2.3%	+0.1	+0.1	+0.1	+2.7	−0.2	−0.1
	18–19	2.3%	+4.5	+0.8	+0.3	+16.4	+0.6	−0.0
	24–25	2.3%	+0.3	+0.1	+0.3	+4.3	+0.6	+0.1
Skip 4 layers	3–6	4.7%	+16.4	+10.9	+8.4	+61.5	+8.5	+2.4
	10–13	4.7%	+18.2	+0.6	+0.4	+12.8	+0.3	−0.0
	17–20	4.7%	+9.2	+1.4	+0.5	+51.5	+0.8	+0.0
	23–26	4.7%	+9.3	+4.4	+14.9	+32.0	+7.6	+41.6
Hold 2 steps	trajectory	4.1%	+0.3	−0.0	−0.0	−0.2	−0.0	−0.0
Hold 4 steps	trajectory	8.2%	+0.6	−0.0	−0.0	−0.9	−0.0	−0.0

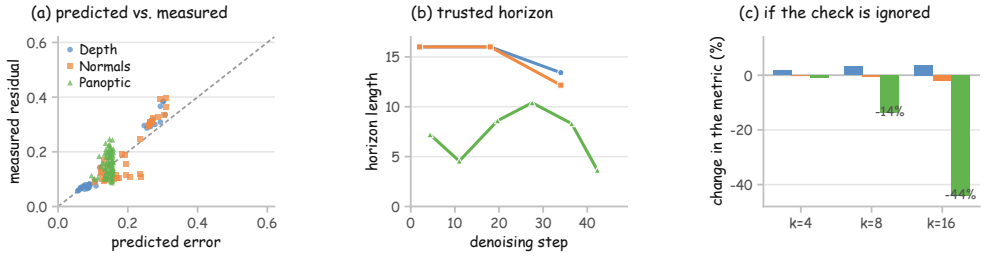


Figure 9: Landing-point verification diagnostics on 12 images per task. (a) Predicted draft error versus the backbone-measured residual at the same landing point; the dashed line denotes equality. (b) Trusted horizon along the trajectory. (c) Task-metric change when fixed-horizon drafts are force-accepted while retaining the verification call.

Table 13: Force-accepted fixed-horizon drafts on SenseNova-Vision. Δ is the task-metric change from the reference, and ρ is the Spearman correlation between the landing-point residual and per-image metric change; the final columns split images into low- and high-error quartiles.

Task	k	Calls↓	Mean residual	Δ metric	ρ	Δ Q1	Δ Q4
Depth (AbsRel × 100)	4	30	0.074	−0.07	−0.01	−0.07	−0.06
	8	18	0.114	−0.13	−0.00	−0.09	−0.14
	16	12	0.181	−0.15	+0.03	−0.12	−0.26
Normals (mean angle, deg)	4	30	0.103	−0.00	−0.00	−0.09	+0.29
	8	18	0.167	+0.06	+0.16	−0.12	+0.70
	16	12	0.290	+0.29	+0.36	−0.05	+1.37
Panoptic (PQ)	4	30	0.187	−0.4	–	–	–
	8	18	0.350	−7.0	–	–	–
	16	12	0.461	−22.7	–	–	–

than the reference. On panoptic segmentation the errors reach 0.46 and the output collapses; PQ is a dataset-level metric, so we report the aggregate only, and it tracks the mean error monotonically. The rule discriminates where discrimination is needed. This diagnostic is not used to choose a common threshold; the main experiments use the task-specific tolerances stated in Section 4.

C.3.5 Coordinate Acceptance by Digit Position

Coordinates in the detection record are written as fractions of the image side with three decimals, one token per digit. The backbone's next-token entropy is close to one bit at the third decimal, so its distribution there is nearly flat. That bounds how peaked the distribution is, not how predictable the argmax is from another model's features, so the numbers below are what our drafters reach rather than a limit on any drafter. Table 14 reports acceptance separately for the leading zero and for each decimal. Under the backbone's own greedy prediction these tokens commit at layers 26,

23, 25 and 27 in that order, against layer 22 for the record template. The leading zero is syntax in all but name and is accepted 44% of the time; the first decimal, which places the box coarsely, is accepted only 14% of the time, the second 20%, and the third 11%, close to what the backbone's own entropy allows. Lookup, which copies digits from earlier records, reaches 11, 7 and 7%, and the parallel prediction heads have the same profile as the token drafter's chained prediction heads. The coarse position of the next box is therefore not read off the hidden state at the current position either, and the finest digit is where both lookup and the token drafter lose most of their acceptance. Relaxing verification to accept a drafted third decimal within ± 1 of the backbone's predicted digit raises its acceptance to 0.25 and the committed tokens per call from 3.54 to 4.08; on Dense200 the F1 changes by +0.1 (79.5 versus 79.4), on LVIS it drops by 0.6 and on VisDrone, whose objects are small enough for a thousandth of the image side to matter, by 1.2, so the tolerance follows object size and we keep the exact rule in the main results.

Table 14: Acceptance rates of drafted coordinate tokens by digit position on Dense200 (200 images, $K=8$).

Drafter	leading zero	1st decimal	2nd decimal	3rd decimal
Grammar-guided lookup (Saxena, 2023)	0.41	0.11	0.07	0.07
Parallel heads, restricted vocab. (Cai et al., 2024)	0.40	0.15	0.20	0.10
PercepDraft	0.44	0.14	0.20	0.11
PercepDraft, third decimal within one	0.51	0.16	0.22	0.25