Ablate-to-Validate: Are Vision-Language Models Really Using Continuous Thought Tokens?

Tianyi Zhang* Mahtab Bigverdi* Ranjay Krishna
University of Washington
{tzhang26, mahtab, ranjay}@cs.washington.edu
*Equal contribution.

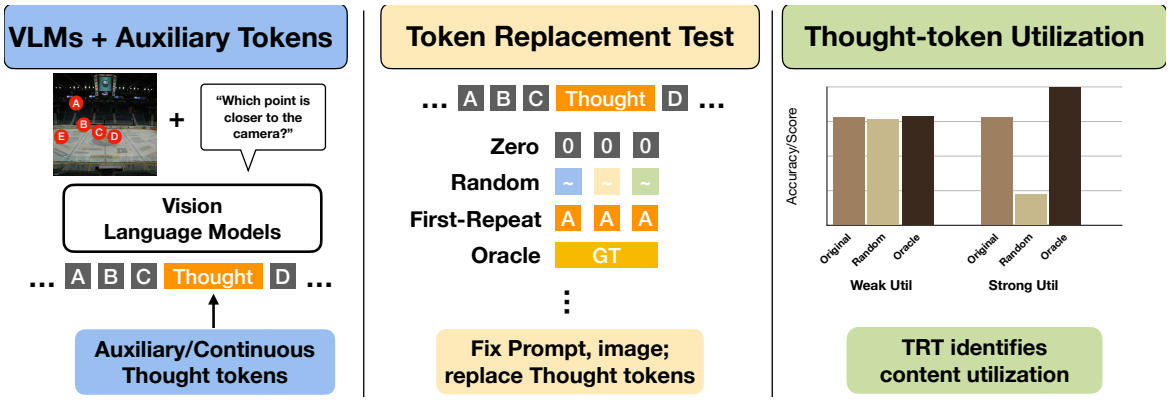

Figure 1. Overview of the Token Replacement Test (TRT). TRT replaces the intermediate thought-token span while fixing the prompt, image, token budget, and decoding procedure. Comparing identity, random, first-repeat, and oracle replacements separates content utilization from span-position and token-budget effects.

Abstract

Vision language models (VLMs) are increasingly augmented with continuous or latent non-textual tokens intended to support "visual thinking." Despite the improved task accuracy, this alone does not show that models actually use these tokens for reasoning. Gains may instead arise from confounds such as added context length, special-token anchoring, or training-time regularization. We formalize a diagnostic principle, Ablate-to-Validate, for testing whether latent-token content is genuinely utilized. We instantiate this principle as the Token Replacement Test (TRT), a standardized suite of content-replacement ablations. TRT measures whether performance depends on the information carried by latent tokens rather than their mere presence. It probes (1) span-position bias through zero and random replacement, (2) disentangles token-budget from token-diversity effects through first-repeat and count-matched variants, and (3) evaluates information content using oracle or ground-truth token injection together with distribution-matched random baselines. As a controlled testbed, we study relative depth reasoning, where continuous depth embeddings can be inserted at explicit span positions under a fixed token budget. We train and prove LLaVA and Qwen2.5-VL models both to predict and to consume these tokens, and show that TRT applies across heterogeneous latent-token model backbones. Concretely, we cover two trained backbones (LLaVA-13B, Qwen2.5-VL-3B) with continuous and discrete depth spans across three frozen visual encoders (SigLIP2, CLIP, DINOv2) and multiple token budgets, and additionally apply TRT to three off-the-shelf visual-thinking systems (Mirage, Mull-Tokens, CoVT) evaluated on BLINK, VSP, and CV-Bench. Our results show that accuracy gains can be a misleading proxy for latent-token reasoning: across multiple model backbones, types of continuous visual tokens, and compute budgets, VLMs retain most of the improvement even when latent-token content is corrupted or replaced, revealing a persistent gap between "having a latent channel" and actually using it as an information bottleneck. By separating true content utilization from alternative explanations, TRT provides a simple and standardized way to evaluate continuous thought tokens in vision-language models, and we recommend reporting such diagnostics as standard practice. Code will be released at

the project page.

1. introduction

Vision-language models (VLMs) first rose to prominence on tasks where success largely reduces to conditioning on visual features and producing short-form text: captioning, visual question answering, and referring expression comprehension [2, 9, 26]. In these regimes, the dominant bottlenecks are perceptual coverage and alignment. In other words, is the model seeing the right thing and attaching the right words. For such tasks, scaling data, architectures, and instruction tuning tends to yield steady improvements.

But computer vision did not begin as "language-conditioned pattern recognition." Early vision work treated images as projections of underlying structure, and sought intermediate representations that make geometry and objecthood explicit [6, 18, 19]. That original framing is resurfacing in modern evaluations. Recent benchmarks increasingly target settings where perception is necessary but not sufficient, and *multi-step visual reasoning* becomes the limiting factor: expert-level multimodal understanding [35], mathematical reasoning grounded in images [16], and integrated capability tests that combine perception with knowledge, spatial reasoning, and computation [34]. In parallel, the language modeling community has shown that explicitly eliciting intermediate structure, most notably via *chain-of-thought* [28], can substantially improve performance on hard problems [10, 28]. Together, these trends raise a basic question for multimodal systems: what should intermediate structure look like when plain text is an awkward—or lossy—interface for visual computation?

A natural answer is to augment VLMs with the ability to reason beyond language, using intermediate visual representations. Ideally, they would can emit and later consume such intermediate representations as part of their reasoning process. One family of such representations uses *discrete* visual tokens; for instance, "perception tokens" encode intermediate artifacts like depth maps or bounding boxes, and conditions reasoning on those tokens [4]. A second family uses continuous or latent non-textual tokens, often framed as *continuous* visual thoughts, that the model produces and later reads [21, 23, 24, 32]. Across both families, these visual token mechanisms routinely report accuracy gains. However, accuracy gains can reflect incidental effects rather than information in the tokens: a fixed insertion position can become a reliable cue, a larger effective context/capacity from extra tokens can provide more computational strength, or auxiliary objectives for learning visual tokens might regularize training.

This paper asks a more pointed version of the question: *when auxiliary tokens help, are gains driven by the information in the tokens, or by incidental position and budget effects?* We propose **Ablate-to-Validate**, an empirical principle for answering this. If a model truly relies on auxiliary-token information, then systematically perturbing token values, while holding the prompt, token budget, image features, and decoding procedure fixed, should induce predictable performance changes. We operationalize this principle with the **Token Replacement Test (TRT)**, our standardized inference-time replacement protocol that isolates (1) *span-position* bias via zero and distribution-matched random replacement, (2) *token-budget versus token-diversity* effects via first-repeat and count-matched variants when token counts differ, and (3) genuine information utilization via oracle or *ground-truth injection* compared against matched random baselines (Figure 1).

To make these interventions precise, we develop a controlled testbed based on questions that require depth reasoning. Depth provides a continuous, geometrically grounded signal that is both meaningful for visual computation and amenable to clean counterfactual manipulations: we can zero or randomize embeddings, and we can define a clear upper bound by injecting ground-truth (oracle) depth embeddings. Concretely, we insert an explicit depth-token span into the language stream and train models to both predict and consume continuous depth embeddings at those positions. We implement this pipeline for multiple VLM backbones (LLaVA [14] and Qwen2.5-VL [3]), enforcing a fixed visual token budget across experiments. We also compare against a discrete depth baseline [4] that quantizes depth into a VQVAE codebook and trains via cross-entropy under the same token budget. Finally, to show TRT's broader applicability, we apply the same ablate-to-validate lens to additional off-the-shelf VLMs: Mirage [32], Mull-Tokens [23], and Chain-of-Visual-Thought (CoVT) [21] with visual token mechanisms.

Empirically, TRT reveals a recurring content–utility gap: across many configurations, continuous visual tokens function more like a fixed interface than an information-rich reasoning channel. Across multiple model backbones, types of continuous visual tokens, and compute budgets, replacing the model's predicted tokens with zeros or distribution-matched random vectors produces little to no degradation, and injecting oracle tokens yields only marginal gains in some cases. This pattern indicates that models can profit from the presence and placement of a visual-token span—its budget, its anchoring, and its training-time effects—without reliably decoding the span's semantic content as a bottleneck for the final decision. TRT provides a minimal, standardized validation step to separate these cases from settings where auxiliary token content is genuinely used, and we recommend reporting TRT-style diagnostics as standard practice when introducing new latent-token mechanisms.

2. Related work

We situate our work amongst the growing body of VLM algorithms that inject the capability of models to reason beyond language.

How VLMs process visual content. Early VLMs such as Flamingo [1] showed that inserting cross-modal attention into a (largely) frozen LLM can yield strong few-shot transfer, while later open models such as LLaVA [14] and Qwen2.5-VL [3] popularized simpler projection-based designs that map visual features into the LLM embedding space. These architectures achieve strong performance on many perception-heavy benchmarks, and scaling data and model sizes have often yielded steady improvements.

As evaluation has shifted toward multi-step reasoning regimes, research has explored whether explicitly eliciting intermediate structure improves multimodal performance. Multiple efforts have tried to include intermediate rationales or structured traces to support visual–linguistic inference [15, 36]. However, serializing geometric or perceptual computation into text can be awkward and lossy: viewpoint changes, occlusions, and metric comparisons are often more naturally expressed in a visual or continuous representation than in natural language. This motivates approaches that augment VLMs with visual token streams intended to serve as intermediate reasoning substrates.

Spatial and depth reasoning benchmarks as testbeds. A large body of work diagnoses where VLMs fail on spatial reasoning, revealing brittleness even for seemingly basic predicates. SpatialSense [30], VSR [13], and What'sUp [8] use adversarial or minimal-pair designs to reduce language priors and expose persistent failures in left/right and above/below reasoning. Recent benchmarks extend this to 3D and viewpoint sensitivity: 3DSRBench [17] shows that models break under modest viewpoint changes, while ViewSpatial-Bench [12] highlights a perspective gap between camera-centered and human-centered viewpoints. For multi-image/video settings, VSI-Bench [29] and MMSI-Bench [31] test whether models can maintain a consistent spatial map over time and viewpoints, and MindCube [33] targets spatial mental modeling from limited views. Related benchmark methodology work further emphasizes that shortcuts remain pervasive and motivates debiasing through iterative filtering [5].

These benchmarks are invaluable for locating failure modes, but they typically do not isolate *how* a model internally computes spatial conclusions. In particular, they do not provide standardized diagnostics for whether a proposed intermediate mechanism (e.g., visual token streams) is genuinely *used as information* versus merely acting as a training scaffold. Our work complements these evaluations by introducing a controlled, intervention-based diagnostic (TRT) and instantiating it in a depth reasoning testbed where counterfactual token replacements and oracle injections are well-defined. Our method can be expanded to also support spatial benchmarks.

Discrete visual tokens for reasoning. One line of work introduces explicit intermediate artifacts as tokens and conditions subsequent reasoning on them. Perception tokens [4] exemplify this approach by encoding intermediate representations such as depth maps or bounding boxes into discrete tokens that the model can generate and then consume during inference. More broadly, discrete tokenizations provide a natural interface to autoregressive decoding and allow straightforward accounting of token budgets.

Latent and continuous visual tokens. Motivated by the hypothesis that continuous representations better preserve perceptual structure than discretized alternatives, several recent methods augment VLMs with latent or continuous token streams that lie outside the standard vocabulary. MetaMorph [24] introduces interleaved continuous visual tokens in the generation stream and trains models for multimodal understanding and generation. Chain-of-Visual-Thought (CoVT) [21] distills information from multiple vision experts into a compact continuous span that the VLM predicts before generating answers. Mirage [32] frames latent tokens as "machine mental imagery," recasting hidden states into a latent visual channel through a two-stage training paradigm. Mull-Tokens [23] generalizes this idea to modality-agnostic latent thinking tokens, and Latent Implicit Visual Reasoning [11] trains models to discover visual reasoning tokens without explicit supervision.

Across these works, visual token mechanisms often report improvements on a variety of perception and reasoning benchmarks. However, accuracy gains alone do not establish that a model relies on *visual token content*. Improvements can arise from confounds such as fixed span placement (a reliable marker), changes in effective capacity via token budget, or visual objectives that regularize training. Our work directly targets this gap: we propose the Token Replacement Test (TRT), a standardized content-replacement diagnostic that separates (i) span-position bias, (ii) token-budget vs. token-diversity effects, and (iii) genuine information utilization via oracle injection baselines.

3. Method

3.1. Preliminaries: visual tokens in VLMs

Autoregressive VLMs generate an output sequence conditioned on an image I and a textual prompt x. We write the textual output as $y_{1:T}$ with $y_t \in \mathcal{V}$, and the next-token distribution

$$p(y_t \mid y_{<t}, x, I) = \mathrm{Softmax}(Wh_t), \tag{1}$$

where $h_t \in \mathbb{R}^H$ is the hidden state and W is the output projection.

A growing class of methods augments this process with an *visual token span* intended to act as an intermediate representation. We denote the visual span by $u_{1:K}$ where K is the visual token budget. Depending on the method, the visual tokens may be

$$u_i \in \begin{cases} \mathcal{V}_u & \text{(discrete visual tokens)} \\ \mathbb{R}^D & \text{(continuous/latent visual tokens).} \end{cases} \tag{2}$$

Operationally, the model first produces (or is given) a visual span at designated positions, and then generates downstream text conditioned on that span. This creates a natural distinction between **prediction** and **utilization**: a model may learn to *predict* visual tokens (e.g., via a visual loss) without *using their content* when producing the final answer. Since most prior work reports only end-task accuracy, it remains unclear whether visual tokens are acting as an information-bearing bottleneck or merely as a scaffold (e.g., a fixed marker, extra budget, or regularization).

Our goal is to test *utilization*: does changing the *values* of $u_{1:K}$ (while keeping everything else fixed) change downstream performance in the way we would expect if the model were relying on visual content?

3.2. Ablate-to-Validate Principle

We propose **Ablate-to-Validate**: if a model genuinely relies on the information encoded in a visual span, then systematically perturbing that span at inference should induce predictable changes in the final output quality. We instantiate this principle as the **Token Replacement Test (TRT)**, a standardized suite of *content-replacement interventions* applied to the visual span $u_{1:K}$ while holding fixed the prompt x, image features from I, the token budget K, and the decoding procedure.

3.3. Token Replacement Test (TRT)

We treat the visual span as an explicit object in the forward pass. Let $\mathrm{Inject}(x, u_{1:K})$ denote the augmented input sequence (or internal stream) in which the visual span is consumed. TRT evaluates a model by running the same example under a collection of replacements $u_{1:K} \leftarrow \tilde{u}_{1:K}$ and measuring the change in task performance.

Replacement interventions. TRT uses the following replacements, designed to isolate distinct confounds:

- **Identity re-injection.** $\tilde{u}_{1:K} \leftarrow u_{1:K}$, verifying that the interception and replacement pathway is correct.
- **Zero replacement.** $\tilde{u}_{1:K} \leftarrow 0$ (or a designated null token for discrete spans), testing sensitivity to removing visual content.
- **Random replacement.** $\tilde{u}_{1:K} \leftarrow \epsilon$, with ϵ sampled i.i.d. (e.g., Gaussian for continuous spans), testing whether performance depends on the specific predicted content.
- **Distribution-matched random.** $\tilde{u}_{1:K} \leftarrow \epsilon$ sampled to match the empirical distribution of the model's predicted visual tokens (or ground-truth tokens when available), controlling for scale and marginal statistics.
- **First-repeat.** $\tilde{u}_i \leftarrow u_1$ for all i, preserving budget and span placement while removing token diversity.
- **Count-matched variants.** When comparing methods whose native visual spans differ in length, we apply the same replacement while matching the number of intervened tokens to the number used by the corresponding identity run for that example.[1] This separates "more tokens" from "better tokens."
- **Oracle / ground-truth injection.** When a supervised visual target exists, $\tilde{u}_{1:K} \leftarrow u^{\star}_{1:K}$ provides an interpretable upper bound and diagnoses whether the model *can* exploit a high-quality visual signal.

Interpreting TRT. If a model uses visual *content*, then zero/random replacement should measurably hurt, and oracle injection should measurably help. If performance is largely unchanged under these interventions, then the visual mechanism is likely not acting as an information bottleneck; gains may instead come from span placement, budget expansion, or training-time regularization. First-repeat and count-matched variants further separate "token diversity" effects from "token budget" effects.

3.4. Controlled testbed: depth-span counterfactuals

To apply TRT cleanly, we require a setting where (i) visual content is well-defined, (ii) counterfactual replacements are meaningful, and (iii) oracle visual signals are available. We choose **depth reasoning** as a controlled testbed because depth provides a continuous, geometrically grounded representation that supports clean interventions (zeroing or randomizing embeddings) and admits a strong upper bound via ground-truth depth injection.

Depth span markup and fixed budget. We insert an explicit depth span into the response using boundary tokens `<DEPTH_START>` and `<DEPTH_END>`. At the text level, we include a single placeholder token `<DEPTH_TOKEN>` to keep responses readable; internally, the model expands this placeholder into a fixed-length span of K visual tokens. This enforces a strictly matched visual budget across training, inference, and all TRT replacements in our controlled depth setting. For discrete depth spans, we use a

[1] Some methods do not enforce a fixed auxiliary-token budget during generation. For example, in Mirage [32] the number of auxiliary tokens can vary across examples and, consequently, across ablations. In these cases, we match the number of ablated tokens to the number generated in the identity condition for the same example, and force generation of the end marker once that matched count is reached.

uniform budget of $K = 100$ tokens across methods, following Aurora [4]; for continuous depth spans, K is held fixed within each method and across all TRT ablations for that method. Off-the-shelf methods whose native mechanisms do not expose a fixed auxiliary-token budget are evaluated using the count-matching protocol above rather than by forcing a shared global K.

Depth targets. For each example, we obtain a depth map (ground-truth when available, otherwise from a fixed estimator in our controlled pipeline) and convert it into a token-aligned target representation under two instantiations:

- **Continuous depth tokens:** real-valued embeddings in a frozen feature space [23, 24, 32].
- **Discrete depth tokens:** quantized codebook indices trained with cross-entropy [4].

This lets us study both discrete and continuous depth representations under aligned span placement, matched supervision locations, and controlled span length within each method family. We therefore isolate the effect of representation as much as possible, while noting that the comparison is not a literal one-to-one budget equivalence across discrete and continuous formulations: discrete variants expand the output vocabulary, whereas continuous variants inject real-valued vectors and may differ in feature dimension.

3.5. Continuous depth spans

In the continuous variant, depth is represented as $z \in \mathbb{R}^{K \times D}$, a sequence of K depth embeddings extracted in a frozen encoder space of dimension D. We support multiple frozen encoders (e.g., SigLIP2 [25], DINOv2 [20], and CLIP [22]) to test how the target feature geometry affects utilization. When an encoder yields a different patch count than K, we interpolate patch embeddings (using 2D interpolation when preserving a grid is appropriate) to obtain exactly K targets.

Injection. At training time, we learn a **depth projector** $P : \mathbb{R}^D \to \mathbb{R}^H$ that maps each depth embedding into the language hidden size. During the forward pass, the depth span positions are replaced by $P(z)$.

Prediction. To train the model to produce depth content at the depth span, we learn a **depth head** $H_d : \mathbb{R}^H \to \mathbb{R}^D$ that maps language hidden states back into the frozen depth feature space. Let $\hat{z} = H_d(h)$ denote the predicted depth embeddings at the depth span positions. We supervise depth with a regression-style objective (e.g., MSE or cosine loss) applied only on the depth span.

Training objective. We jointly optimize language modeling and depth prediction:

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda_{\text{depth}} \cdot \mathcal{L}_{\text{depth}}, \tag{3}$$

where $\mathcal{L}_{\text{depth}}$ is computed only at depth span positions and λ_{depth} balances the two objectives.

3.6. Discrete depth spans

In the discrete baseline, depth is quantized into a codebook and represented as token IDs $d_{1:K} \in \{1, \ldots, |\mathcal{C}|\}$. Training reduces to cross-entropy over discrete depth tokens, and inference can optionally enforce constrained decoding inside the depth span to ensure only valid depth tokens appear. This baseline isolates whether any gains in depth reasoning come from (i) the presence of a visual span and supervision, or (ii) operating in a continuous latent feature space.

3.7. Backbones: LLaVA and Qwen2.5-VL

We implement the depth-span interface and TRT instrumentation in two widely used open VLM families.

LLaVA. LLaVA [14] couples a vision encoder to an LLM via a learned multimodal projector. We extend the sequence construction by expanding each `<DEPTH_TOKEN>` placeholder into K depth positions and injecting $P(z)$ at those positions. Attention masks, position IDs, and loss masks are updated to reflect the expanded span. Depth supervision is applied via H_d only on the depth positions, while text positions are trained with the standard next-token loss.

Qwen2.5-VL. Qwen2.5-VL [3] provides a native multimodal transformer with strong localization behavior. We integrate the same depth projector/head interface into Qwen's embedding path: depth tensors are provided alongside token IDs, depth positions are masked out of the text loss, and $\mathcal{L}_{\text{depth}}$ is computed analogously to the LLaVA variant.

3.8. TRT on off-the-shelf VLMs

While depth spans provide a controlled environment for matched token budgets and oracle injection, TRT is intended as a general diagnostic. To demonstrate breadth, we apply TRT to additional "visual thinking" models that expose visual token streams. For each model, we intercept the visual tokens produced during the intermediate phase and replace them with the TRT interventions immediately before they are consumed for downstream decoding.

- **Mirage** [32] interleaves latent visual tokens with text by recasting hidden states as a latent visual channel. We apply TRT by replacing these continuous latent vectors at the designated latent positions. Because Mirage does not enforce a fixed auxiliary-token budget during generation, we use the count-matched protocol described above:

for each example, the number of intervened tokens is matched to the number generated in the corresponding identity run, and generation is forced to emit the end marker once that matched count is reached. This lets us test whether performance depends on latent content rather than merely on the presence or length of the latent span.

- **Mull-Tokens** [23] introduces modality-agnostic latent tokens intended to store intermediate information. We apply TRT by intervening on the Mull-token segment to test whether downstream performance depends on the content of these latent tokens rather than simply on the existence of an auxiliary latent channel.
- **CoVT** [21] trains a VLM to predict a compact span of continuous visual-thought tokens distilled from lightweight vision experts. We apply TRT by replacing the predicted continuous span immediately before it is consumed by the decoder.

Across these heterogeneous mechanisms, TRT provides a common lens for testing the same core hypotheses—span-position bias, token-budget vs. token-diversity effects, and genuine information utilization—without requiring architectural changes.

Relation to MetaMorph [24] and "visual prediction" objectives. Our continuous depth span is conceptually aligned with the broader idea of training an LLM to predict continuous visual features in addition to text. MetaMorph's Visual Predictive Instruction Tuning (VPiT) [24] adds a visual head that maps language hidden states into a vision feature space and trains with a combined language + visual objective. We differ in three key respects: (i) our targets are depth-specific embeddings derived from depth maps in a frozen encoder space, (ii) supervision is localized to explicit depth spans under a strictly fixed budget K, enabling controlled counterfactual replacement, and (iii) we explicitly test *utilization* using TRT rather than relying on accuracy alone. Since the training data and model checkpoints are not available publicly, at the time of this work, we do not include MetaMorph in our empirical evaluation.

4. Experiments

Our experiments ask a targeted question: *when continuous "thought" tokens improve accuracy, do models actually use their content?* We answer this by (i) building a controlled depth-span testbed with aligned span placement and controlled budget protocols for **continuous** and **discrete** auxiliary tokens, and (ii) applying the **Token Replacement Test (TRT)** to probe whether performance depends on auxiliary *content* rather than incidental span effects.

4.1. Task and evaluation protocol

Depth reasoning benchmark. We evaluate relative depth reasoning on a HardBLINK-style [4] setup derived from BLINK [7]. Each example overlays $N \in \{3, 4, 5\}$ labeled points and asks which point is closest to the camera. We use three subsets (3/4/5 points), each with 124 images (372 total). To reduce multiple-choice artifacts, the model outputs a brief rationale and then a single final label; we deterministically parse the final label, treating unparsable answers as incorrect.

Table 1. **Depth reasoning on HardBLINK (avg. accuracy across 3/4/5 points).** All methods use the same span placement, but not the same token budget across representation families. The discrete baseline uses a fixed depth span of $K = 100$ tokens. "Continuous (best)" reports the best continuous result for each backbone over the budgets in Table 2.

Backbone	No-aux	Discrete ($K = 100$)	Continuous (best)
LLaVA-13B	76.68	**77.69**	74.46
Qwen2.5-VL-3B	58.87	**71.24**	68.55

What we vary. We vary backbone (**LLaVA-13B**, **Qwen2.5-VL-3B**), frozen depth feature space (**SigLIP2**, **CLIP**, **DINOv2**, **VQ-VAE**[2]), and the continuous token budget $K \in \{4, 16, 64, \text{full}\}$. For the discrete baseline, we use a fixed budget of $K = 100$ depth tokens, following the protocol described in the method section. Depth supports strict interventions (zero/random/oracle replacement) and also admits a discrete-token baseline with the same span placement and supervision interface, though not the same nominal K as the continuous runs.

4.2. Training and implementation details

We train depth-span models on ADE20K (19,279 images) for a fixed horizon. The depth encoder is frozen; we train the depth projector P and depth head H_d, along with a small set of language parameters (LoRA for LLaVA; standard fine-tuning knobs for Qwen2.5-VL as described in the previous section). Unless stated otherwise, we optimize $\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}}$. For the continuous setting reported in the main text, $\mathcal{L}_{\text{depth}}$ is an MSE loss or a cosine loss applied only on the depth span. For the discrete setting, $\mathcal{L}_{\text{depth}}$ is token-level cross-entropy over the target codebook indices on the depth span. Full hyperparameters and compute are provided in Appendix B.1.

4.3. Main results: continuous vs. discrete

We compare **No-aux**, **Discrete**, and **Continuous** variants. **No-aux** is the plain VQA baseline: the model is trained to output only the final answer, without any auxiliary depth span or auxiliary reasoning tokens. **Discrete** predicts codebook depth tokens with cross-entropy and uses a fixed budget of $K = 100$. **Continuous** regresses depth embeddings in a frozen feature space, with K varied as in Ta-

[2] see Appendix A.1

Table 2. **Continuous depth spans across encoders and token budgets.** Accuracy is averaged over the 3/4/5-point subsets.

Backbone	Encoder	$K=4$	$K=16$	$K=64$	full
LLaVA-13B	SigLIP2	73.39	72.58	**74.46**	71.77
	CLIP	72.31	73.12	73.39	67.20
	DINOv2	66.94	63.71	61.56	58.87
Qwen2.5-VL-3B	SigLIP2	54.57	61.02	55.11	63.98
	CLIP	64.25	68.28	55.38	67.20
	DINOv2	**68.55**	63.98	61.47	64.25

ble 2. Accordingly, Table 1 should be read as a comparison across representation families under their respective controlled budget protocols, rather than as a literal same-K comparison between discrete and continuous in every row.

Auxiliary tokens can outperform No-aux, but gains depend on representation and backbone; in our controlled depth-reasoning setting, discrete tokens consistently match or outperform their continuous counterparts (Table 1).

Gains depend on backbone and representation. As summarized in Table 1, auxiliary tokens can outperform No-aux, but the gains depend on both backbone and representation. For LLaVA-13B, the discrete model improves modestly over No-aux (77.69 vs. 76.68), whereas the best continuous configuration reaches 74.46. For Qwen2.5-VL-3B, the strongest continuous configuration reaches 68.55 at K=4 (Table 2), and the discrete result reported in Table 1 is 71.24. In other words, the Qwen results do not show a clear advantage for continuous tokens over the discrete baseline.

Visual tokens help—but continuous is not consistently better.

Visual backbone choice matters—but longer visual spans are not better.

Across backbones, the frozen feature space is a major driver (SigLIP/CLIP often outperform DINO), and very long spans can be counterproductive—"more latent tokens" does not reliably mean "more useful information."

4.4. Do models actually use span content?

Main accuracy gains do not imply reliance on span *content*. We apply TRT to the strongest continuous configurations (LLaVA: SigLIP2, $K = 64$; Qwen: DINOv2, $K = 4$) by replacing depth-span embeddings at inference while holding fixed prompt, image features, span placement, and decoding. We also run the analogous TRT for the discrete baseline by replacing token IDs (with constrained decoding for validity). The corresponding continuous and discrete results are reported in Tables 3 and 4, respectively.

Continuous spans are robust to content corruption; discrete spans are not.

For continuous spans, random and first-repeat replacements produce only modest changes, and oracle injection yields limited headroom—or even slightly worse performance. For LLaVA-13B, oracle injection improves average accuracy only marginally over self-predicted tokens (74.46 → 74.73). For Qwen2.5-VL-3B, replacing the model's predicted depth span with oracle embeddings slightly lowers accuracy (68.55 → 67.74). This pattern is consistent with weak reliance on fine-grained continuous span content. By contrast, the discrete setting shows substantially stronger dependence on token content. For Qwen2.5-VL-3B, oracle replacement produces a large improvement (71.24 → 80.64, +9.40), while random replacement causes a severe drop (71.24 → 51.34, -19.90), and constant/zero tokens also degrade performance significantly (to 58.87). These effects are much larger than those observed in the continuous setting, where analogous perturbations typically change accuracy by only 1–2 points. Taken together, these results indicate that discrete depth tokens act as a meaningful information-bearing bottleneck, whereas continuous spans are comparatively insensitive to the correctness of their content and are more consistent with reliance on span presence, placement, or coarse distributional structure.

Table 3. **TRT for continuous depth spans.** Replacements are applied only to the injected depth vectors before the language backbone consumes them.

Backbone	Replacement	3-pt	4-pt	5-pt	Avg
LLaVA-13B (SigLIP2, $K=64$)	Identity (self)	78.23	73.39	71.77	74.46
	Oracle (GT)	78.23	73.39	72.58	**74.73**
	Random	77.42	72.58	67.74	72.58
	First-repeat	78.23	73.39	71.77	74.46
Qwen2.5-VL-3B (DINOv2, $K=4$)	Identity (self)	71.77	70.97	62.90	**68.55**
	Oracle (GT)	71.77	70.97	60.48	67.74
	Random	70.97	69.35	61.29	67.20
	First-repeat	71.77	71.77	60.48	68.01

Table 4. **TRT for discrete depth tokens.** Random replacement corrupts the symbolic codebook indices and induces larger drops than in the continuous setting.

Backbone	Replacement	3-pt	4-pt	5-pt	Avg
LLaVA-13B (discrete)	Identity (self)	83.06	74.19	75.81	77.69
	Oracle (GT)	85.48	75.81	75.00	**78.76**
	Random	73.39	66.94	66.13	68.82
	Constant / zero	78.23	71.77	70.97	73.66
Qwen2.5-VL-3B (discrete)	Identity (self)	72.58	75.00	66.13	71.24
	Oracle (GT)	83.06	84.68	74.19	**80.64**
	Random	58.06	52.42	43.55	51.34
	Constant / zero	66.94	62.90	46.77	58.87

4.5. What about other "visual thinking" methods?

Finally, we apply TRT to three auxiliary-token mechanisms (Mirage [32], Mull-Tokens [23], and CoVT [21]) by intercepting their intermediate token streams and applying the same replacement suite immediately before consumption (Sec. 3.8). The corresponding results are reported in Tables 5, 6, and 7.

Mirage [32] interleaves latent visual tokens with text during decoding by reusing hidden states as compact "visual

Table 5. Mirage reproduction on Spatial Planning (SP) and HardBLINK depth (accuracy in %). For HardBLINK, "Total" reflects the evaluated subset for these runs.

Replacement	Spatial Planning (SP)		HardBLINK (Depth)				
	Acc (%)	Correct / Total	Overall	3-pt	4-pt	5-pt	Correct / Total
Identity (self)	**76.25**	**305 / 400**	26.08	38.71	22.58	16.94	**97 / 372**
First-repeat	75.75	303 / 400	26.61	37.90	25.00	16.94	99 / 372
Oracle (GT)	75.00	300 / 400	25.00	30.65	**26.61**	17.74	93 / 372
Oracle (GT, count-matched)	**76.25**	**305 / 400**	–	–	–	–	–
Random	74.75	299 / 400	18.82	18.55	20.16	17.74	70 / 372
Random (GT dist)	74.75	299 / 400	14.52	13.71	17.74	12.10	54 / 372
Random (model dist)	**76.25**	**305 / 400**	22.85	29.84	19.35	**19.35**	85 / 372
Zero	51.50	206 / 400	8.06	8.87	8.06	7.26	30 / 372

Table 6. Mull-Tokens TRT-style latent replacements on BLINK and SAT.

Replacement	Qwen2.5-VL-Mull		Qwen2.5-VL-MullGRPO	
	BLINK (mean acc)	SAT (mean acc)	BLINK (mean acc)	SAT (mean acc)
Identity (self)	63.71	76.33	**64.57**	**77.00**
First-repeat	63.71	76.33	64.57	77.00
Random	**63.86**	76.33	63.14	76.33
Random (same dist)	63.29	**77.00**	63.43	**77.00**
Zero	63.43	76.33	64.00	76.00

Table 7. CoVT TRT-style depth-token replacements on CV-Bench 2D/3D (overall).

Replacement	CV-Bench-2D (Overall)	CV-Bench-3D (Overall)
Identity (self)	**76.26**	**80.83**
First-repeat	76.26	80.83
Random	21.39	19.00
Random (same dist)	74.65	79.17
Zero	58.97	69.00

thoughts." We reproduce Mirage's Spatial Planning (SP) evaluation and test its depth reasoning on HardBLINK using depth maps as helper images. As shown in Table 5, zeroing latents moderately degrades SP (76.25% $\rightarrow$ 51.50%) but nearly collapses HardBLINK depth (26.08% $\rightarrow$ 8.06%), indicating much higher sensitivity for relative depth. Reproduction details for the depth-helper setup are provided in Appendix F.1.

Mull-Tokens [23] adds latent tokens for intermediate states. First-repeat matches baseline, and random latents induce only small changes (Table 6), consistent with token *presence* dominating token *diversity*.

CoVT [21] distills continuous "thinking" tokens from vision experts. Identity/first-repeat match baseline, while pure random noise collapses performance; distribution-matched random largely recovers it (Table 7), indicating reliance on distributional statistics more than fine-grained token content.

Content of visual tokens matters less than their presence in many settings.

4.6. Summary of TRT hypotheses

Three TRT signatures indicate that an auxiliary-token gain is not driven by content:

(1) span-position bias (zero/random preserves the gain)
(2) budget confound (first-repeat suffices)
(3) content unused (oracle adds no headroom)

Across our depth testbed and off-the-shelf systems, we see strong evidence for (1)–(2) in many configurations and weaker evidence for (3), motivating intervention-based validation alongside accuracy improvements.

5. Discussion

Our results suggest a simple but important takeaway: an intermediate token channel can *exist* and even correlate with higher accuracy without becoming a strict causal bottleneck for the final prediction. This mirrors a pattern observed in the chain-of-thought (CoT) literature: LLMs often retain strong performance even when the provided CoT contains incorrect steps, irrelevant detours, or intentionally corrupted rationales [27, 37]. In other words, intermediate "reasoning traces" can function as training or prompting scaffolds without being faithfully *consumed* as information at inference time.

Importantly, this conclusion is representation-dependent. In the discrete setting, we observe substantially stronger sensitivity to token content: replacing predicted tokens with oracle depth tokens yields large gains, while random or constant replacements cause severe degradation. This indicates that discrete auxiliary tokens can act as meaningful information-bearing bottlenecks. In contrast, the continuous setting exhibits much weaker sensitivity to content, suggesting that continuous spans are often used through coarse or structural signals rather than as fine-grained intermediate representations.

For continuous tokens, this robustness may arise from training and architectural effects. Modern continuous-token methods use multi-stage pipelines and auxiliary objectives that can improve representations or calibration even if the model does not ultimately use the span as an information-bearing channel at test time. When the visual span is always present at a consistent location, the model may also treat it as a structural cue without decoding its fine-grained content.

A complementary explanation is an information-bottleneck mismatch. Continuous "visual thought" spaces are typically high-dimensional, noisy, or redundant relative to the downstream decision. For depth-style tasks, the answer often depends on a few coarse comparisons rather than reconstructing a full field. As a result, the model may exploit marginal statistics or positional regularities of the span instead of the intended semantics, producing the TRT signature we observe: limited gains from oracle injection and only modest degradation under random or zero replacement.

Auxiliary-token evaluation therefore requires more than end-task accuracy: methods should be tested by perturbing *content* while holding span placement and budget fixed.

Acknowledgments

This work was partially supported by Samsung and CoCoSys.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. 2022. 3

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2425–2433, 2015. 2

[3] 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. 2, 3, 5

[4] Mahtab Bigverdi, Zelun Luo, Cheng-Yu Hsieh, Ethan Shen, Dongping Chen, Linda G Shapiro, and Ranjay Krishna. Perception tokens enhance visual reasoning in multimodal language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 3836–3845, 2025. 2, 3, 5, 6, 11

[5] Ellis Brown, Jihan Yang, Shusheng Yang, Rob Fergus, and Saining Xie. Benchmark designers should "train on the test set" to expose exploitable non-visual shortcuts. *arXiv preprint arXiv:2511.04655*, 2025. 3

[6] PHILOSO EPHY DO CT OR OF. *MACHINE PERCEPTION OF THREE-DIMENSIONAL, SO LIDS*. PhD thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 1961. 2

[7] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. 2024. 6

[8] Amita Kamath, Jack Hessel, and Kai-Wei Chang. What's "up" with vision-language models? Investigating their struggle with spatial reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. 3

[9] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. ReferItGame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 787–798, 2014. 2

[10] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022. 2

[11] Kelvin Li, Chuyi Shang, Leonid Karlinsky, Rogerio Feris, Trevor Darrell, and Roei Herzig. Latent implicit visual reasoning. *arXiv preprint arXiv:2512.21218*, 2025. 3

[12] Linnan Li, Xiaoyu Chen, Peng Chen, et al. ViewSpatial-Bench: Evaluating multi-perspective spatial understanding of vision-language models. *arXiv preprint arXiv:2505.21500*, 2025. 3

[13] Fangyu Liu, Guy Emerson, and Nigel Collier. Visual spatial reasoning. *arXiv preprint arXiv:2205.00363*, 2022. 3

[14] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, 2023. 2, 3, 5

[15] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. 2022. 3

[16] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024. 2

[17] Wufei Ma, Haoyu Chen, Guofeng Zhang, Yu-Cheng Chou, Jieneng Chen, Celso M. de Melo, and Alan Yuille. 3DSR-Bench: A comprehensive 3D spatial reasoning benchmark. 2025. 3

[18] David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010. 2

[19] Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 479(480): 104, 1969. 2

[20] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. 2023. 5

[21] Yiming Qin, Bomin Wei, Jiaxin Ge, Konstantinos Kallidromitis, Stephanie Fu, Trevor Darrell, and XuDong Wang. Chain-of-visual-thought: Teaching vlms to see and think better with continuous visual tokens. 2025. 2, 3, 6, 7, 8, 11

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. 2021. 5

[23] Arijit Ray, Ahmed Abdelkader, Chengzhi Mao, Bryan A Plummer, Kate Saenko, Ranjay Krishna, Leonidas Guibas, and Wen-Sheng Chu. Mull-tokens: Modality-agnostic latent thinking. *arXiv preprint arXiv:2512.10941*, 2025. 2, 3, 5, 6, 7, 8, 11

[24] Shengbang Tong, David Fan, Jiachen Zhu, Yunyang Xiong, Xinlei Chen, Koustuv Sinha, Michael Rabbat, Yann LeCun,

Saining Xie, and Zhuang Liu. Metamorph: Multimodal understanding and generation via instruction tuning. *arXiv preprint arXiv:2412.14164*, 2024. 2, 3, 5, 6, 11

[25] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Henaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. 2025. 5

[26] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2015. 2

[27] Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717–2739, Toronto, Canada, 2023. Association for Computational Linguistics. 8

[28] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022. 2

[29] Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. *arXiv preprint arXiv:2412.14171*, 2025. 3

[30] Kaiyu Yang, Olga Russakovsky, and Jia Deng. SpatialSense: An adversarially crowdsourced benchmark for spatial relation recognition. 2019. 3

[31] Sihan Yang, Runsen Xu, Yiman Xie, et al. MMSI-Bench: A benchmark for multi-image spatial intelligence. *arXiv preprint arXiv:2505.23764*, 2025. 3

[32] Zeyuan Yang, Xueyang Yu, Delin Chen, Maohao Shen, and Chuang Gan. Machine mental imagery: Empower multimodal reasoning with latent visual tokens. 2025. 2, 3, 4, 5, 7, 11

[33] Baiqiao Yin, Qineng Wang, Pingyue Zhang, et al. Spatial mental modeling from limited views. *arXiv preprint arXiv:2506.21458*, 2025. 3

[34] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. MM-vet: Evaluating large multimodal models for integrated capabilities. In *Proceedings of the 41st International Conference on Machine Learning*, pages 57730–57754. PMLR, 2024. 2

[35] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9556–9567, 2024. 2

[36] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. 2023. 3

[37] Zhanke Zhou, Rong Tao, Jianing Zhu, Yiwen Luo, Zengmao Wang, and Bo Han. Can language models perform robust reasoning in chain-of-thought prompting with noisy rationales? *arXiv preprint arXiv:2410.23856*, 2024. 8

A. Additional Results

A.1. HardBlink additional runs (full blocks)

In Table 8, we present the comprehensive HardBlink average accuracy results. We evaluate performance across different continuous depth encoders (CLIP, SigLIP, DINO , and VQ-VAE), target sequence lengths (Full, 64, 16, 4), and various ablation settings, including learning rate variations and intermediate layer supervision.

Table 8. **HardBlink average accuracy (%).** Performance across architectures, continuous depth encoders, sequence lengths, and ablation settings (learning rate and intermediate layers).

Method	Configuration	LLaVA 13B	Qwen2.5 3B
Baselines	Base model	32.00	54.30
	No aux token	76.68	58.87
	Discrete tokens	**77.69**	**71.24**
Continuous (CLIP)	Full	67.20	67.20
	64	73.39	55.38
	16	73.12	68.28
	4	72.31	64.25
Continuous (SigLIP)	Full	71.77	63.98
	64	**74.46**	55.11
	16	72.58	61.02
	4	73.39	54.57
Continuous (DINO)	Full	58.87	64.25
	64	61.56	61.47
	16	63.71	63.98
	4	66.94	**68.55**
Continuous (VQ-VAE)	100	70.97	-
SigLIP (64) LRs	No depth loss	72.85	-
	50 ep., LR 2×10^{-4}	24.73	-
	50 ep., LR 1×10^{-5}	64.00	-
	50 ep., LR 5×10^{-5}	66.67	-
SigLIP (16) Layers	Layer 9 (5 ep.)	71.51	-
	Layer 19 (5 ep.)	68.01	-
	Layer 29 (5 ep.)	69.09	-
	Layer -1 (5 ep.)	70.70	-
	Layer 19 (10 ep.)	72.04	-

A.2. Scope and method coverage

TRT is a task-agnostic, inference-time diagnostic that tests whether visual reasoning is causally used by a VLM. We apply TRT across multiple tasks, summarized in Table 9: three off-the-shelf continuous-token methods, two backbone families, six benchmarks spanning spatial, perceptual, and 2D/3D understanding, and four distinct training paradigms. Across all of these, identity, random, oracle, and first-repeat replacements yield near-identical accuracy, supporting the conclusion that visual continuous tokens are not causally used at inference. This establishes TRT as a general protocol rather than a task- or model-specific probe. We use depth reasoning as a stress test: VLMs reliably fail at it without explicit grounding, it is interpretable, has ground truth, and admits clean counterfactuals (zero, random, distribution-matched, first-repeat, oracle) under matched span placement and token budget, allowing us to isolate the effect of information content from position, budget, and regularization confounds. We do not propose a depth-specific model nor claim discrete tokens are categorically better; the discrete-depth setup is an interpretable diagnostic baseline.

Table 9. **Task/backbone/training coverage.** L13B=LLaVA-13B; Q3B/Q7B=Qwen2.5-VL-3B/7B. Training stages verified against the corresponding papers.

Method	Backbone	Training	Eval. task
Aurora discrete [4]	L13B/Q3B	VQ-VAE + multi-task SFT	HardBLINK
Ours, depth (cont.)	L13B/Q3B	1-stage joint LM+depth	HardBLINK
Mirage [32]	Q7B	2-stage: latent+text → text-only	VSP/SAT/COMT
Mull-Tokens [23]	Q7B	2-stage SFT: anchored → free-form	BLINK/SAT
Mull-Tokens-GRPO [23]	Q7B	2-stage SFT + Stage 3 GRPO	BLINK/SAT
CoVT [21]	Q7B/L13B	4-stage curriculum	CV-Bench

Table 10. **Setup verification** (Paper baseline → method vs. Ours). Aurora baselines: LLaVA-13B base / fine-tuned-without-tokens. Mirage baseline: Direct-SFT (Qwen2.5-VL-7B). Mull-Tokens / CoVT baselines: Qwen2.5-VL-7B base. MetaMorph baseline / method: VQA-only vs. Cosine-Sim VPiT ablation on LLaMA-3 8B (their Table 3); no public checkpoint, "Ours" not reproduced.

Method	Task	Paper (base. → method)	Ours
Aurora discrete [4]	HardBLINK 3/4/5	34.1 / 50.8 → 60.7	77.69
Mirage [32]	VSP-SP	72.0 → 76.0	76.25
Mull-Tokens [23]	BLINK / SAT	55.33 / 59.00 → 66.80 / 77.66	63.71 / 76.33
Mull-Tokens-GRPO [23]	BLINK / SAT	55.33 / 59.00 → 67.13 / 77.00	64.57 / 77.00
CoVT [21]	CV-Bench	74.5 → 80.0	78.55
MetaMorph [24]	MMBench	73.1 → 73.8	—

A.3. Baseline reproduction verification

Our reimplementations (Sec. 4.3) reproduce each method's published numbers within tolerance. Table 10 verifies our setup against the paper-reported baseline and method numbers. Aurora-discrete is LLaVA-13B-based; Mirage and Mull-Tokens / Mull-Tokens-GRPO use Qwen2.5-VL-7B; CoVT covers both LLaVA-13B and Qwen2.5-VL-7B. Identity rows match within tolerance except for Aurora-discrete, which retrains on 20k ADE depth examples for 10 epochs (vs. the paper's curriculum mix), explaining the higher reproduced accuracy. CoVT "Ours" averages CV-Bench-2D (76.26) and 3D (80.83). MetaMorph is architecturally similar to our continuous model but has no public checkpoint, so its paper numbers are listed in Table 10 for reference only.

A.4. KV-cache ablation

TRT operates as an input-embedding-level intervention: it replaces tokens at the *input-embedding* level, before any forward pass, so K/V projections at those positions and every downstream self-attention over them already operate on the replaced states. KV-mediated transfer through the cache is therefore fully exercised by the existing TRT protocol:

Table 11. **KV-cache ablation on HardBLINK** (Qwen2.5-VL-3B, DINOv2, $K = 4$, 372 queries). Content insensitivity holds with and without caching across the depth span and answer, ruling out the KV cache as the source of the result.

Regime	Identity	Random	Oracle	First-repeat
KV-cached	68.55	68.01	68.55	68.82
KV-off	69.62	69.62	69.35	69.62

if visual reasoning carried meaningful content through the cache, random or oracle replacement would propagate and change predictions. The fact that they do not is itself the finding.

To further rule out the KV cache as the source of content insensitivity, we ran a dedicated KV-off ablation on Qwen2.5-VL-3B with DINOv2 at $K = 4$ on HardBLINK (372 queries). In the KV-off regime, the cache is used through the prefix, then disabled (full recompute) over the depth span and answer, with bit-identical prefixes verified. As shown in Table 11, content insensitivity persists in both regimes: identity, random, oracle, and first-repeat replacements all yield near-identical accuracy whether the cache is on or off. This rules out the KV cache as the channel through which TRT's null result could be artefactually produced.

B. Additional Implementation Details

This appendix section consolidates both the hyperparameter summary and the repository-specific launcher/configuration details used in our runs. In a few cases, multiple configurations coexist because the repository contains both the primary reported settings and additional ablation or stage-wise training recipes.

B.1. Training hyperparameters (expanded)

LLaVA continuous (reported 10-epoch setting). Vision tower frozen; LoRA on the LLM ($r = 128$, $\alpha = 256$); max length 2048; AdamW with cosine LR; BF16+TF32; per-device batch size 16 on 8 GPUs (effective batch size 128); weight decay 0. Component learning rates: base/LoRA LR 2×10^{-4}; multimodal projector LR 2×10^{-5}; depth projector/head LR 5×10^{-5}; depth loss type cosine with normalization enabled; $\lambda_{\text{depth}} = 0$ for the reported continuous setting. The choice of the learning rate is highly sensitive; as shown in the SigLIP (64) LR ablations in Table 8, using a higher LR of 2×10^{-4} leads to catastrophic collapse, whereas 5×10^{-5} stabilizes training.

The "No depth loss" ablation in Table 8 removes auxiliary depth supervision entirely: the depth-related parameters remain randomly initialized and are not trained with the depth objective, allowing us to measure what happens when the depth channel is present architecturally but receives no explicit supervision.

LLaVA discrete (reported 10-epoch setting). Vision tower frozen; LoRA on the LLM; max length 2048; AdamW with cosine LR; BF16+TF32; effective batch size 128. Each image is represented with $K = 100$ discrete depth tokens, where K denotes the discrete depth-token budget. The depth projector/head are linear with depth LR 1×10^{-5}; $\lambda_{\text{depth}} = 1.0$.

Qwen2.5-VL-3B (reported 10-epoch setting). Vision encoder frozen; LLM + visual MLP + embeddings are fine-tuned; AdamW with cosine LR; BF16; warmup ratio 0.03; effective batch size 128; max length 4096. The base learning rate is 2×10^{-4}. Depth loss type MSE with $\lambda_{\text{depth}} = 1.0$; depth head/projector are linear and weight-tied ($W_{\text{head}} = W_{\text{proj}}^{\top}$).

B.2. Evaluation datasets and sizes (expanded)

HardBlink. Three subsets (3/4/5 point) with 124 questions each (372 total), long-format question JSONLs.

Mirage SPC reproduction. `vsp-spatial-planning` with 400 test samples and 1000 train samples.

Mull. BLINK (700 samples) and SAT (300 samples), greedy decoding.

CoVT. CV-Bench-2D (ADE20K+COCO) and CV-Bench-3D (Depth+Distance), evaluated via VLMEvalKit.

Training-set sizes used locally. The training dataset size used in our runs have 19,279 examples

C. Data and Evaluation Protocol

C.1. ADE point sampling generation

The point-marked images used for training and evaluation are generated dynamically from the ADE20K dataset. The generation pipeline scans the `ADE_depth` directory, matches each depth map to its corresponding RGB image, resizes both to 336×336 pixels, and samples 3 to 5 points per image.

To ensure task difficulty and spatial diversity, the sampling process enforces several constraints:

- Points are sampled within bounded coordinates ($x \in [10, 324]$, $y \in [95, 224]$).
- Points with a depth value of 0 are rejected.
- The minimum spatial distance between any two points must be at least 20 pixels.
- The minimum absolute depth difference between points must be at least 20.

- The generator uses up to 10,000 sampling attempts per image to satisfy these constraints.

Once valid points are found, they are sorted by depth, but their visual labels (A through E) are randomly shuffled so that alphabetical order does not correlate with depth order. The correct answer is defined as the label attached to the maximum depth value. Our final mixed-depth dataset contains 19,279 images, distributed as 6,736 (3-point), 6,562 (4-point), and 5,981 (5-point) examples.

C.2. Training data formats and splits

Our training data is formatted into JSON files depending on the architectural path:

- **Continuous Long (`mixed_depth_long.json`):** Used for Qwen continuous training. It contains 19,279 examples pairing the ADE images with a long-form rationale. The depth target is represented by a single `<DEPTH_TOKEN>` bounded by `<DEPTH_START>` and `<DEPTH_END>`. The `embedding` key points to the corresponding precomputed `.npy` file.
- **Continuous Short (`mixed_depth_short.json`):** Contains the same 19,279 examples, but the assistant's target response is strictly the final multiple-choice letter (e.g., `(C)`).
- **Discrete Mixed-Depth Long (`mixed_depth_long.json`):** Contains the same rationale format as the continuous version, but the depth span is populated with $K = 100$ discrete depth tokens per image (e.g., `<DEPTH_36><DEPTH_64>...`), where K is the discrete token budget.

C.3. Prompting protocol

We use two prompt variants for HardBLINK. The short prompt is:

> *"Multiple points are circled... Which point is the closest to the camera?"*

The long prompt appends a step-by-step scaffold that explicitly references the point coordinates, the depth map, and the rule that higher depth-map values indicate points closer to the camera.

C.4. Answer extraction

Predictions are evaluated using `eval_answers.py`. Ground-truth labels are extracted from `(A)`-style answers. The parser also supports several long-form answer patterns, including forms such as *"the answer is that point X is closer ... "*. Predictions that cannot be parsed are counted as incorrect rather than dropped.

D. Depth Targets and TRT Mechanics

D.1. Continuous depth encoders

The continuous depth encoders listed in `encoder_config.json` are:

- `google/siglip2-large-patch16-256` with $K = 256$, $D = 1024$
- `facebook/dinov2-base` with $K = 256$, $D = 768$
- `openai/clip-vit-large-patch14-336` with $K = 576$, $D = 1024$
- `ait-vqvae-continuous-100x512` with $K = 100$, $D = 512$. For this model, we extract continuous codebook embeddings from the same VQ-VAE used for discrete token generation, obtaining dense per-image feature representations for each input view.

Configuration names ending in `_interploate_N` override the token horizon so that the target sequence has exactly $K = N$ tokens.

D.2. Continuous target resizing

Continuous targets are resized to exactly K tokens before supervision. For square token grids, we use bilinear interpolation. For non-square token layouts, we use one-dimensional linear interpolation. As demonstrated in Table 8, the optimal target sequence length K varies by encoder. For instance, SigLIP achieves peak performance on LLaVA at $K = 64$ (74.46%), while DINO performs best under extreme compression at $K = 4$ (68.55% on Qwen2.5), suggesting differing spatial densities in their respective feature maps.

D.3. TRT-style ablations

In continuous LLaVA mode, TRT-style ablations are

$$\{\texttt{gt}, \texttt{random}, \texttt{zero}, \texttt{random\_gt\_dist}, \texttt{model}, \texttt{first\_repeat}\}.$$

Here, `random_gt_dist` denotes distribution-matched random replacement: for each sample, replacement vectors are sampled from a Gaussian $\mathcal{N}(\mu, \sigma)$ using the mean and standard deviation of that sample's ground-truth depth-token embeddings.

In continuous Qwen mode, the same ablation family is implemented through `set_depth_ablation(...)` together with constrained generation of exactly K `<DEPTH_TOKEN>` tokens after `<DEPTH_START>`.

E. Backbone-Specific Implementation

E.1. LLaVA continuous path

In continuous LLaVA, image features are inserted at image-token positions and a single textual `<DEPTH_TOKEN>` placeholder is expanded into K depth embeddings. All depth positions are masked from the cross-entropy loss. Continuous depth loss is applied autoregressively, using hidden state $t - 1$ to predict depth target t.

E.2. Qwen continuous path

In continuous Qwen2.5-VL, preprocessing expands <DEPTH_TOKEN> into K repeated tokens, masks all depth-token labels from the cross-entropy loss, and applies depth loss using hidden state $t-1 \rightarrow$ depth target t. Position IDs follow Qwen2.5-VL's multimodal RoPE path.

During decoding, a `ContinuousDepthLogitsProcessor` forces exactly K <DEPTH_TOKEN> tokens after <DEPTH_START>, followed by one <DEPTH_END> token.

E.3. Intermediate layer depth supervision

In addition to standard training, we tested whether applying the auxiliary depth loss to different intermediate transformer layers, rather than strictly the final decoder output, would change downstream performance. This allows us to evaluate the representation of spatial depth at varying levels of model abstraction.

This mechanism is controlled via the `depth_loss_layer` argument (defaulting to `-1` for the final hidden state). Setting this argument to a specific intermediate block k forces the model's forward pass to retain all hidden states (`output_hidden_states=True`) and routes `hidden_states[k+1]` to the depth loss computation, bypassing the rest of the decoder blocks for that specific objective.

Once the specified layer's state is extracted, the supervision behaves identically to the default setting: the loss is applied autoregressively such that the hidden state at position $t-1$ predicts the target depth vector at position t. The projection is handled via either a tied linear map or a separate `depth_head`, and compared against the target using MSE, cosine, or softmax loss.

This intermediate layer-selection pathway is implemented symmetrically in both the LLaVA and Qwen2.5-VL architectures. Our evaluated layer-ablation checkpoints (e.g., `depthlayer_9`, `depthlayer_19`, `depthlayer_29`, and the default `depthlayer_m1`) correspond directly to these targeted transformer blocks. The quantitative impact of supervising these intermediate layers is reported in Table 8 in Section A; among the evaluated intermediate-layer configurations, supervising Layer 19 for 10 epochs gives the best result (72.04%).

E.4. Discrete decoding constraints

In discrete LLaVA decoding, custom `LogitsProcessors` force either ground-truth depth tokens, random depth tokens, or an all-`DEPTH_0` sequence inside the depth span, then force <DEPTH_END>.

F. Off-the-Shelf Model Reproduction

F.1. Mirage

Our Mirage reproduction uses `Qwen/Qwen2.5-VL-7B-Instruct` with checkpoint `Miiche/vsp_spatial_planning_direct_sft`. The HardBLINK adapter uses the question text verbatim as the user prompt, appends the image through the processor chat template, and injects latent replacements inside Mirage's modified generation loop after <|latent_start|>.

Mirage GT mode uses a helper depth image if present, runs it through `model.visual(...)`, and compresses the resulting sequence to the fixed latent count (default `latent_size=4`) by averaging groups. The count-matched GT variant uses a two-pass protocol: first generate once, count latent tokens, then recompress GT latents to that count and regenerate.

Mirage training data and latent collation. Mirage's depth training utilizes a dataset of 19,279 examples (`train_depth_long.jsonl`) for both stages of training. In the JSONL schema, the `text_input` contains the point-localization question and step-by-step reasoning prompt, while the `text_output` contains the rationale and final answer. The `image_output` key points to the ADE depth map.

Mechanically, Mirage does not store latent tokens directly in the JSONL. Instead, during data collation, the raw row is converted into a standard user/assistant turn. The assistant's output image slot is dynamically converted into a latent span using <|latent_start|> and <|latent_end|>, which is then expanded into a fixed sequence of <|latent_pad|> tokens. Our default Mirage training scripts use a sequence length of `latent_size=4`.

F.2. Mull-Tokens

Our Mull-Tokens reproduction uses `array/Qwen2.5-VL-Mull` and `array/Qwen2.5-VL-MullGRPO`. The prompt template appends an assistant message containing

```
"<think>" + "<|latent_pad|>"*20 +
"</think>".
```

Answers are extracted from <answer>...</answer>.

Mull ablations replace both prefill <|latent_pad|> embeddings and autoregressive latent embeddings after <|latent_start|>. The *same-dist random* setting samples each replacement vector independently from $\mathcal{N}(\mu_i, \sigma_i)$, where μ_i and σ_i are computed from that vector's own statistics. GT and GT-dist modes require an external helper-image directory.

F.3. CoVT

Our CoVT reproduction uses four VLMEvalKit-wrapped checkpoints under `Wakals/`: `CoVT-7B-depth`, `CoVT-7B-seg_depth_dino`, `CoVT-7B-seg_depth_dino_edge`, and `CoVT-LLaVA-13B-depth`.

At evaluation time, ablations replace the anchor-pad token embeddings associated with tokens such as `<|sam_pad|>`, `<|dino_pad|>`, and `<|depth_pad|>`. Supported modes are

$$\{\texttt{zero}, \texttt{random}, \texttt{same}, \texttt{random\_dist}, \texttt{first\_repeat}\}.$$

GT and count-matched modes are not implemented in the CoVT evaluation wrappers.

G. Compute and Resources

The compute settings used in our experiments are summarized below: 1 node on partition `gpu-l40s`, 8 GPUs, 40 CPUs per task, and 300 GB RAM. 10 epochs of continuous training takes approximately 10 wall-clock hours.