Multimodal Latent Reasoning via Predictive Embeddings

Ashutosh Adhikari & Mirella Lapata
School of Informatics
University of Edinburgh
United Kingdom
ashutosh.adhikari@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

Tool-augmented multimodal reasoning enables visual language models (VLMs) to improve perception by interacting with external tools (e.g., cropping, depth estimation). However, such approaches incur substantial inference overhead, require specialized supervision, and are prone to erroneous tool calls. We propose PEARL (**P**redictive **E**mbedding **A**lignment for **R**easoning in **L**atent space), a JEPA-inspired framework that learns from expert tool-use trajectories entirely in the latent space, eliminating the need for explicit tool invocation at inference time. Unlike reconstruction-based latent reasoning methods, which autoregressively generate latent tokens and suffer from training–inference mismatch and limited support for multi-step tool use, PEARL directly learns predictive embeddings from multimodal trajectories while preserving the standard vision-language generation pipeline: it is model-agnostic, simple to train, and naturally supports trajectories with multiple tool calls. Experiments across multiple perception benchmarks show that PEARL matches or outperforms standard supervised fine-tuning and reconstruction-based latent reasoning approaches. Furthermore, we provide empirical evidence that reconstruction-based methods primarily learn embeddings rather than image edits in latent space, motivating predictive embedding learning as a more principled alternative. We will release the code at https://github.com/Ashutosh-Adhikari/pearl.

1 Introduction

Recent work in multimodal reasoning has explored augmenting vision-language models (VLMs) with external tools (e.g., cropping, object detection, depth estimation) to improve grounded reasoning (Su et al., 2024; Wu et al., 2025b; Su et al., 2025). By enabling models to iteratively manipulate visual inputs, these approaches allow VLMs to "think with images" rather than relying solely on textual reasoning.

While interacting with expert tools to edit images is an effective strategy for grounding LLMs in visual context (Yue et al., 2024; Hao et al., 2025), tool-based approaches introduce practical and conceptual challenges. First, invoking external tools incurs substantial inference-time latency and compute overhead (Nichols et al., 2025; Wu et al., 2025a). Second, learning to correctly select and parameterize tools requires specialized supervision (Wu et al., 2025b; Su et al., 2024), and even after such training, erroneous calls waste inference compute and pollute the context with irrelevant information. Third, most approaches assume a homogeneous tool set, whereas handling the diverse tools exposed by complex MCP servers requires advanced planning and instruction-following (Wu et al., 2025a).

A promising direction replaces explicit tool use with latent reasoning, where models operate in a continuous embedding space instead of generating discrete intermediate outputs (Hao et al., 2024; Tan et al., 2025; Gozeten et al., 2026). Prior work has explored *reconstruction-based* latent reasoning, in which models autoregressively generate latent tokens intended to represent intermediate visual transformations (Li et al., 2025; Yang et al., 2025; Gu et al., 2025). Borrowing from Coconut (Hao et al., 2024), these methods supervise continuous latent tokens with a reconstruction objective against the outputs of visual tools, while

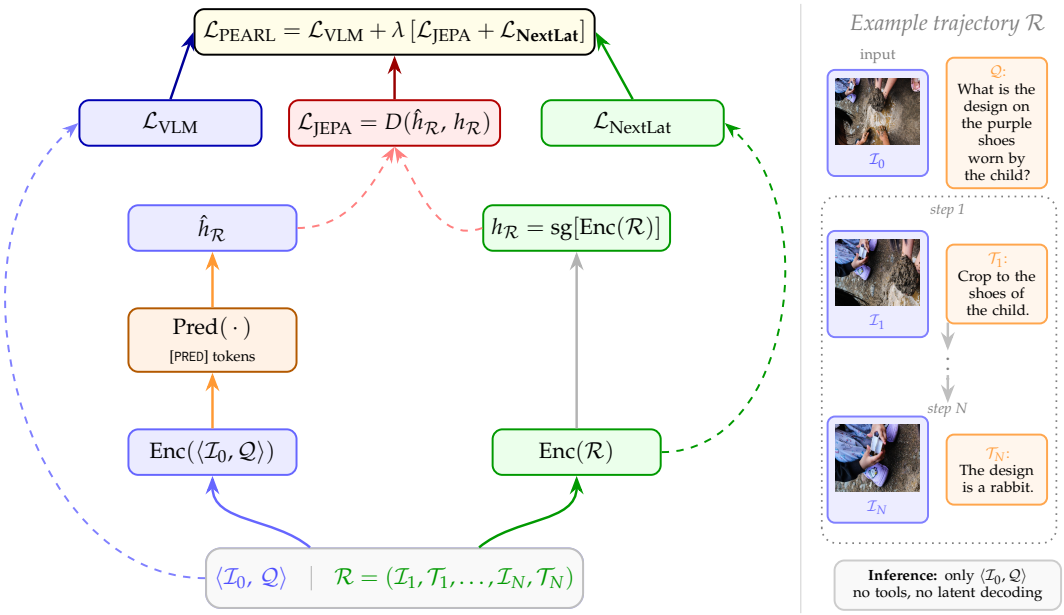


Figure 1: PEARL architecture. *Left:* Solid arrows denote forward-pass dataflow; dashed arrows denote which components contribute to each loss (no forward pass). During training, two independent forward passes encode $\langle \mathcal{I}_0, \mathcal{Q} \rangle$ and the expert trajectory $\mathcal{R}$. A tied-weights predictor maps the input encoding to the trajectory latent space. $\mathcal{L}_{\text{JEPA}}$ aligns the predicted embedding $\hat{h}_{\mathcal{R}}$ with the stop-gradient target $h_{\mathcal{R}}$; $\mathcal{L}_{\text{VLM}}$ preserves autoregressive text generation; $\mathcal{L}_{\text{NextLat}}$ regularises hidden states to act as belief states. *Right:* An example trajectory of sequential visual edits $(\mathcal{I}_1, \ldots, \mathcal{I}_N)$ interleaved with reasoning text $(\mathcal{T}_1, \ldots, \mathcal{T}_N)$. At inferences PEARL uses only $\langle \mathcal{I}_0, \mathcal{Q} \rangle$: it invokes no tools and decodes no latent tokens, retaining the standard VLM pipeline.

preserving the standard transformer architecture (see Figure 6 for an overview). Despite their appeal, these approaches suffer from two fundamental limitations. First, they exhibit a training–inference mismatch: during training, models are supervised with as many latent tokens as there are image patch tokens in the tool output, yet at inference only a small, fixed number of latent tokens is decoded, often without improving, and sometimes degrading, performance (Yang et al., 2025; Li et al., 2025). Second, they are typically confined to single-step transformations, failing to support multi-step reasoning over sequences of tool operations. These observations suggest that reconstruction-based methods primarily learn useful *embeddings* rather than genuinely simulating visual transformations in latent space.

In this work, we propose **PEARL** (**P**redictive **E**mbedding **A**lignment for **R**easoning in **L**atent space), a JEPA-inspired framework that learns predictive representations from expert tool-use trajectories. Rather than autoregressively generating latent tokens, PEARL predicts trajectory embeddings from an image–question pair (see Figure 1, right), allowing the model to internalize the effects of tool use without explicit tool invocation. The framework operates entirely in latent space during training, avoids training–inference mismatch, and supports multi-step reasoning over trajectories with multiple tool calls. We instantiate PEARL by jointly optimizing a standard vision–language generation objective with a predictive embedding objective over interleaved multimodal trajectories (see Figure 1, left), enabling the model to retain its text generation capabilities while learning both the *effects* and *sequencing* of task-relevant transformations.

We evaluate PEARL across a range of multimodal reasoning benchmarks, including settings with single and multiple tool calls. PEARL consistently matches or outperforms supervised fine-tuning and reconstruction-based latent reasoning approaches. Moreover, our analysis shows that reconstruction-based methods primarily learn embeddings rather than performing genuine latent "imagination", supporting predictive embedding learning as a more principled alternative.

2 Related Work

Tool-augmented Multi-modal Reasoning. A prominent line of work augments vision-language models with external visual tools to improve grounded reasoning (Su et al., 2024; Huang et al., 2025b; Zheng et al., 2025). These approaches enable models to iteratively manipulate images through operations such as cropping, object detection, or spatial transformations, effectively allowing them to "think with images" rather than relying solely on textual reasoning. More advanced systems further integrate specialized tools such as depth estimation or multi-step visual editing pipelines, often interleaving tool execution with chain-of-thought reasoning. To determine when and how to invoke tools, early methods rely on supervised fine-tuning with expert trajectories (Su et al., 2025; Chung et al., 2026) while more recent approaches leverage reinforcement learning to acquire tool-use policies and support multi-step reasoning (Zheng et al., 2025; Su et al., 2024; Geng et al., 2026).

Despite their effectiveness, these methods introduce significant practical challenges: tool invocation incurs substantial inference-time overhead, requires specialized supervision for correct tool selection and parameterization, and remains brittle to errors that can propagate through the reasoning process. Furthermore, many approaches assume a fixed or homogeneous tool set, limiting scalability to diverse or dynamically evolving tool environments. We instead explore latent visual reasoning, where the model internalizes the effects of tool use directly in representation space, eliminating the need for explicit tool invocation at inference time.

Reconstruction-based Latent Reasoning. Existing work on latent reasoning in multimodal models autoregressively generates latent tokens under a reconstruction objective, aiming to "imagine" intermediate image edits. Concretely, these latent tokens are trained to reconstruct tool outputs, borrowing from approaches such as Coconut (Hao et al., 2024) in the text domain. This requires models to switch between latent and discrete tokens at inference, a complication that PEARL avoids. However, this line of work is largely confined to single tool calls within a reasoning trajectory (Li et al., 2025; Yang et al., 2025; Gu et al., 2025). COVT (Qin et al., 2025) extends this setting to multiple tool calls, but applies a fixed sequence of operations regardless of the input query. This design imposes two key limitations. First, COVT avoids dynamic planning by restricting tool calls to parameter-free operations, precluding actions such as cropping that require query-specific arguments. Second, all tool calls are applied directly to the original image $\mathcal{I}_0$ rather than to the outputs of preceding operations, resulting in a shallow tree of independent branches rather than a chain of dependent steps.

In contrast, PEARL avoids autoregressive latent generation by predicting a single embedding of the full expert trajectory, naturally supporting multi-step tool use as the prediction target encodes the entire trajectory $\mathcal{R}$ rather than a single transformation.

Joint Embedding Predictive Architectures for Language Models. The Joint Embedding Predictive Architecture (JEPA) is a self-supervised learning framework that trains a model to predict the embedding of one view of the data from another, rather than reconstructing raw inputs. It has shown promise as a pre-training objective for multimodal models with V-JEPA2 (Assran et al., 2025) and VL-JEPA (Chen et al., 2026) achieving competitive performance; however, both require substantial pre-training compute to match the performance of current state-of-the-art models. In the textual domain, LLM-JEPA (Huang et al., 2025a) adapts JEPA to fine-tune off-the-shelf language models, avoiding the need to train from scratch. Concretely, LLM-JEPA minimises a distance between the encoded representations of paired text and code, encouraging the model to develop modality-agnostic representations conducive to text-to-code generation. Similarly, V-JEPA2 aligns embeddings from video frames and robotic actions with predicted future frames for video model pre-training.

PEARL adapts the JEPA objective to fine-tune an off-the-shelf VLM from expert multimodal tool-use trajectories, treating the image-question pair and the full reasoning trajectory as two views of the same problem. This allows PEARL to internalize the effect of sequential visual tool use in the latent space, without incurring the prohibitive compute of pre-training or departing from the standard image-text-to-text inference pipeline.

3 PEARL: Predictive Latent Reasoning

3.1 Problem Formulation and Overview

We consider a training setting where each example consists of an image-question pair $\mathbf{x} = \langle \mathcal{I}_0, \mathcal{Q} \rangle$ and an expert multimodal reasoning trajectory

$$\mathcal{R} = (\mathcal{I}_1, \mathcal{T}_1, \mathcal{I}_2, \mathcal{T}_2, \ldots, \mathcal{I}_N, \mathcal{T}_N),$$

where each $\mathcal{I}_i$ is an intermediate image produced by an expert visual tool (e.g., crop, highlight, spatial transformation) and each $\mathcal{T}_i$ is the associated reasoning text, with the final step containing the answer to $\mathcal{Q}$ (see Figure 1).

Our goal is to train a VLM that benefits from such tool-use trajectories *without* invoking tools at inference time. In contrast to prior reconstruction-based approaches, which autoregressively generate latent tokens intended to reconstruct intermediate visual edits (Li et al., 2025; Yang et al., 2025; Gu et al., 2025), PEARL (**P**redictive **E**mbedding **A**lignment for **R**easoning in **L**atent space) directly predicts a latent representation of the full trajectory from the original image-question pair, preserving the standard VLM inference pipeline while internalizing information from tool-based reasoning during training.

Concretely, PEARL encodes $\mathbf{x}$ and $\mathcal{R}$ independently and trains a lightweight predictor to anticipate the trajectory embedding from the input alone. Intuitively, the predictor asks: given only the image and question, can the model anticipate what the expert tool-use trajectory would look like in latent space? This design has three advantages. First, it avoids explicit tool invocation at inference. Second, it avoids the training–inference mismatch of reconstruction-based methods, where models are trained with many latent tokens but decode only a small fixed number at test time. Third, because the prediction target encodes the *entire* multimodal trajectory rather than a single image edit, PEARL naturally supports multiple tool calls.

3.2 Trajectory Encoding

We instantiate both $\mathrm{Enc}(\mathbf{x})$ and $\mathrm{Enc}(\mathcal{R})$ using the hidden states of an off-the-shelf autoregressive VLM, serializing the two views as follows (see also Figure 1):

- the **input view** consists of the original image-question pair $\langle \mathcal{I}_0, \mathcal{Q} \rangle$;
- the **trajectory view** consists of the interleaved sequence $(\mathcal{I}_1, \mathcal{T}_1, \ldots, \mathcal{I}_N, \mathcal{T}_N)$.

For each view, we run a forward pass through the VLM and take the final hidden state of the last token as its sequence representation, following prior work on JEPA-style fine-tuning of decoder-only language models (Huang et al., 2025a):

$$h_{\mathbf{x}} = \mathrm{Enc}(\mathbf{x}), \qquad h_{\mathcal{R}} = \mathrm{Enc}(\mathcal{R}).$$

Using separate forward passes avoids cross-view information leakage and keeps the method architecture-agnostic, at the cost of additional training-time compute. This overhead applies only during training; inference remains identical to standard VLM decoding. A lightweight predictor network then takes the input encoding $h_{\mathbf{x}}$ and produces a predicted version of the trajectory embedding.

3.3 Latent Trajectory Predictor

To map the input representation $h_{\mathbf{x}}$ to the trajectory latent space, we use a predictor built from the VLM itself. Following prior tied-weights JEPA formulations (Huang et al., 2025a), we append K learnable special tokens `[PRED]` to the serialized input $\mathbf{x}$, and define the predicted trajectory representation as the hidden state of the final predictor token:

$$\hat{h}_{\mathcal{R}} = \mathrm{Pred}(h_{\mathbf{x}}).$$

Intuitively, the predictor allows the model to perform additional nonlinear computation over the image-question representation before producing the target latent (see visualization in Figure 1). When $K = 0$, the predictor reduces to the identity map. In practice, using predictor tokens lets us reuse the VLM's existing self-attention stack rather than introducing a separate MLP or auxiliary transformer, thereby keeping the method simple and parameter-efficient.

3.4 Predictive Embedding Objective

Our central training signal is a JEPA-style predictive embedding loss that aligns the predicted latent $\hat{h}_{\mathcal{R}}$ with the encoded expert trajectory $h_{\mathcal{R}}$. We define

$$\mathcal{L}_{\text{JEPA}} = D\Big(\hat{h}_{\mathcal{R}}, \text{sg}[h_{\mathcal{R}}]\Big), \tag{1}$$

where $D(\cdot,\cdot)$ is a distance function and $\text{sg}[\cdot]$ denotes stop-gradient. In our experiments, we use SmoothL1 loss for D.

This objective encourages the model to learn a compact representation of the *effect* of expert tool use and multimodal reasoning, rather than explicitly reconstructing intermediate image edits. In this sense, PEARL learns predictive trajectory embeddings rather than latent image generation (see Figure 1).

3.5 Next-Latent Prediction

A potential limitation of using the final hidden state of a decoder as a sequence representation is that it may not reliably summarize all relevant preceding context. To encourage hidden states to behave as predictive summary states, we add a next-latent prediction objective inspired by recent work on latent dynamics in transformers (Teoh et al., 2025).

Let h_t denote the hidden state at time step t within the serialized trajectory. A lightweight latent predictor is trained to forecast future hidden states $\hat{h}_{t+i}$ from the current state h_t, for a prediction horizon d. We optimize

$$\mathcal{L}_{\text{NextLat}} = \mathbb{E}_t \left[\frac{1}{d} \sum_{i=1}^{d} \text{SmoothL1Loss}\left(\text{sg}[h_{t+i}], \hat{h}_{t+i}\right) \right]. \tag{2}$$

This objective encourages hidden states to be informative about future trajectory evolution, making them better suited for sequence-level latent alignment. Specifically, Teoh et al. (2025) show that optimizing the hidden state transitions as in Equation (2) causes them to converge to *belief states*, which Kaelbling et al. (1998) define as sufficient statistics of the past history. We view this term as a regularizer that improves the quality of the learned latent representations, rather than as a separate reasoning mechanism.

3.6 Autoregressive Generation Objective

In addition to latent alignment, we retain the standard VLM training objective over the textual portions of the expert trajectory. Given the interleaved multimodal context, the model is trained to autoregressively predict each token in the textual segments $\mathcal{T}_1, \ldots, \mathcal{T}_N$:

$$\mathcal{L}_{\text{VLM}} = -\sum_{n=1}^{N} \sum_{t=1}^{|\mathcal{T}_n|} \log p_\theta\Big(\mathcal{T}_n^{(t)} \mid \mathcal{I}_0, \mathcal{Q}, \mathcal{I}_1, \mathcal{T}_1, \ldots, \mathcal{I}_n, \mathcal{T}_n^{(<t)}\Big). \tag{3}$$

Here $\mathcal{T}_n^{(t)}$ denotes the t-th token of the n-th reasoning step, $\mathcal{T}_n^{(<t)}$ denotes all preceding tokens within that step, and the conditioning context includes all prior image-text pairs $(\mathcal{I}_1, \mathcal{T}_1, \ldots, \mathcal{I}_{n-1}, \mathcal{T}_{n-1})$ as well as the original input $\langle \mathcal{I}_0, \mathcal{Q} \rangle$. This term ensures that PEARL preserves the VLM's standard text generation capability, which is necessary for producing final answers at test time.

3.7 Training Objective

We jointly optimize the autoregressive generation objective, the predictive embedding objective, and the next-latent regularizer:

$$\mathcal{L}_{\text{PEARL}} = \mathcal{L}_{\text{VLM}} + \lambda \left[\mathcal{L}_{\text{JEPA}} + \mathcal{L}_{\text{NextLat}}\right], \tag{4}$$

where λ jointly controls the contribution of both latent objectives relative to the generation loss, reflecting the view that $\mathcal{L}_{\text{JEPA}}$ and $\mathcal{L}_{\text{NextLat}}$ together constitute a single latent learning signal (see Figure 1).

These three terms play complementary roles. $\mathcal{L}_{\text{VLM}}$ preserves the model's ability to generate answers in the discrete token space. $\mathcal{L}_{\text{JEPA}}$ teaches the model to predict a latent representation of expert multimodal reasoning from the original image-question pair. $\mathcal{L}_{\text{NextLat}}$ encourages hidden states to act as belief states (i.e., sufficient summaries of past context), making them more informative encoding targets for $\mathcal{L}_{\text{JEPA}}$.

3.8 Inference

At inference time, PEARL requires only the original image-question pair $\langle \mathcal{I}_0, \mathcal{Q} \rangle$, and answers using the standard generation pipeline of the underlying VLM. It does *not* invoke external tools, does *not* generate intermediate edited images, does *not* use `[PRED]` tokens, and does *not* autoregressively decode latent reasoning tokens. The cost of learning from tool use is shifted entirely to training time, preserving simple and efficient inference.

4 Experimental Setting

Training Regimes. To demonstrate the effectiveness of PEARL at learning from expert tool-use trajectories, we finetune models across three settings: (i) single-type, single tool call per trajectory; (ii) multiple-type, single tool call per trajectory; and (iii) single-type, multiple tool calls per trajectory. We leave the multiple-type, multiple tool call setting to future work, as no open-source training data currently exists for this combination.

For setting (i), we use the data from LVR (Li et al., 2025), which provides regions of interest used to crop the input image $\mathcal{I}_0$, forming the trajectory $\mathcal{R}$. For setting (ii), we use the ThinkMorph dataset (Gu et al., 2025), which contains four equal-sized subsets corresponding to different tool types: bounding boxes over regions of interest, highlights over charts, jigsaw puzzle reconstructions, and spatial navigation paths over maze images. For setting (iii), we use the PixelReasoner dataset (Su et al., 2024), where each trajectory $\mathcal{R}$ contains up to three sequential crops of $\mathcal{I}_0$. Dataset statistics and examples are provided in Appendix B.

Evaluation Benchmarks. Following previous work (e.g., Li et al. 2025) we evaluate PEARL on a suite of perception intensive visual question answering (VQA; Antol et al. 2015) benchmarks. These include V* (Wu & Xie, 2023), which tests models' ability to perform visual search for objects and their attributes (V*$_{DA}$) and to identify relative positions of objects (V*$_{RP}$). We further evaluate on five subsets of the Blink benchmark (Fu et al., 2025): Counting, IQ, Jigsaw, Spatial Relation, and Relative Reflectance. Finally, we include MMVP (Tong et al., 2024), which probes perceptual robustness using image pairs that CLIP treats as similar despite clear visual differences. All benchmarks are formulated as multiple-choice tasks, enabling straightforward answer parsing (see Appendix B for details).

Comparison Models. Our primary comparisons use Qwen2.5-VL-7B-Instruct (Team, 2025), enabling direct head-to-head evaluation against all reconstruction-based baselines. To further demonstrate PEARL's model-agnostic nature, we also report results for the smaller Qwen2.5-VL-3B-Instruct variant and the 4B variant of Qwen3-VL (Bai et al., 2025).

For the single-type, single tool call setting, we compare against LVR (Li et al., 2025), using their released HuggingFace checkpoint, with 4 latent tokens (aka 4 steps) which the authors note yields the best overall quality. We also compare against CoVT (Qin et al., 2025), reporting results directly from the original paper. LVR achieves the strongest performance among reconstruction-based latent reasoning methods (see Figure 6 for an illustration).

For the multiple-type, single tool call setting, we compare PEARL against a LoRA-finetuned variant trained on the ThinkMorph data (Gu et al., 2025). We do not compare against the original ThinkMorph model, as it relies on explicit intermediate image generation at inference, making it incomparable with latent reasoning methods.[1] For the single-type, multiple tool call setting, we compare directly against PixelReasoner's (2024) released model, which invokes tools explicitly at inference.

[1]ThinkMorph uses unified models as opposed to image-text-to-text models as in our work. Furthermore, for models trained on ThinkMorph data, we allow generation up to 256 tokens since the training data is verbose.

Model	V^*	V^*_{DA}	V^*_{RP}	MMVP	Counting	IQ	Jigsaw	Rel. Ref	Spatial Rel
No fine-tuning									
Qwen2.5-VL-7B-Instruct	78.5	81.7	73.7	66.7	66.7	26.0	52.0	38.8	87.4
Single-type, single tool call									
CoVT (Qin et al., 2025)	78.0	—	—	58.7	—	—	—	—	—
LVR (Li et al., 2025) (4 steps)	80.1	85.2	73.7	72.0	68.3	26.0	51.3	**41.0**	89.5
SFT (LVR data)	79.1	82.6	73.7	65.7	67.5	26.7	45.3	33.6	88.8
PEARL (LVR data)	**81.5**	**86.1**	**74.5**	**73.5**	68.3	**28.2**	**53.1**	39.6	89.5
Multiple-type, single tool call									
SFT (ThinkMorph data)	79.1	80.0	77.6	71.0	38.3	16.7	22.0	38.8	60.1
PEARL (ThinkMorph data)	**82.2**	**80.9**	**84.2**	**75.3**	**65.0**	**26.0**	**53.3**	**46.3**	**88.8**
Single-type, multiple tool calls									
PixelReasoner (Su et al., 2024)	**80.1**	**81.7**	**77.6**	67.0	66.7	25.3	52.7	**42.5**	88.1
PEARL (PixelReasoner data)	79.1	**81.7**	75.0	**70.0**	**70.0**	**28.7**	**53.3**	40.3	**89.5**

Table 1: Results for Qwen2.5-VL-7B-Instruct across all training settings. PEARL requires no tool calls at inference time. **Bold** denotes best result per block.

PEARL is finetuned with LoRA (Hu et al., 2021) adapters (rank $r = 64$ and $\alpha = 128$). Across settings, we include a LoRA SFT baseline trained on the same data as PEARL and the instruction-tuned model without fine-tuning as a zero-shot baseline. Hyperparameter settings for PEARL are provided in Appendix A.

5 Results

5.1 How Does PEARL Compare to Reconstruction-based Methods?

Table 1 compares PEARL against various Qwen2.5-VL-7B-Instruct baselines and comparison systems (across three training settings). As can be seen, PEARL consistently matches or outperforms its respective baselines while requiring no tool calls at inference, an advantage none of the reconstruction-based or tool-augmented methods share.

Single-type, single tool call. PEARL trained on LVR data outperforms both the SFT baseline and LVR (Li et al., 2025) (4 steps) on V^* (81.5 vs. 79.1 and 80.1) and MMVP (73.5 vs. 65.7 and 72.0), while matching LVR on Spatial Rel (89.5) and improving on Jigsaw (53.1 vs. 51.3). Notably, LVR finetunes the entire decoder whereas PEARL uses only LoRA adapters, making these gains more parameter-efficient. CoVT (Qin et al., 2025) underperforms even the zero-shot baseline on MMVP (58.7 vs. 66.7), suggesting its fixed-sequence design is poorly suited to this benchmark.

Multiple-type, single tool call. The ThinkMorph results are the most striking in the table. PEARL outperforms the SFT baseline comprehensively across the datasets – by over 3 points on V^* (82.2 vs. 79.1) and achieves the highest accuracy on MMVP across the settings for Qwen2.5-7B (75.3 vs. 71.0). The SFT baseline collapses under the heterogeneity of four qualitatively different tool types as seen for the Blink test sets, whereas PEARL's trajectory-level embedding target is agnostic to tool type, explaining its robustness across the full set of benchmarks.

Single-type, multiple tool calls. PEARL is competitive with PixelReasoner (Su et al., 2024), which explicitly invokes tools at inference time. PEARL outperforms it on MMVP (70.0 vs. 67.0), Counting (70.0 vs. 66.7), Jigsaw (53.3 vs. 52.7), IQ (28.7 vs. 25.3), and Spatial Rel (89.5 vs. 88.1), while PixelReasoner leads on V^*_{RP} (77.6 vs. 75.0) and Rel. Ref (42.5 vs. 40.3). The fact that PEARL matches an inference-time tool-use system while operating as a standard image-to-text model demonstrates that the tool-use signal can be effectively internalized during training through predictive embedding alignment.

Ablations in Appendix A confirm that encouraging hidden states to act as belief states meaningfully improves the quality of the learned trajectory embeddings. Figure 5 provides further support: t-SNE visualizations show that PEARL induces coherent clusters that align

Model	V*	V^*_{DA}	V^*_{RP}	MMVP	Counting	IQ	Jigsaw	Rel. Ref	Spatial Rel
Qwen2.5-VL-3B-Instruct									
No fine-tuning	56.0	53.0	60.5	59.3	65.8	26.0	45.3	**44.8**	80.4
LVR (Li et al., 2025) (4 steps)	64.9	69.6	60.5	54.7	—	29.3	**52.7**	—	—
PEARL (LVR data)	**73.8**	**82.6**	60.5	**68.7**	66.7	29.3	51.3	41.8	**84.6**
PEARL (PixelReasoner data)	69.6	78.3	56.6	63.7	**67.5**	**30.7**	49.3	43.3	81.8
PEARL (ThinkMorph data)	62.8	69.6	52.6	60.0	61.7	29.3	43.3	35.1	78.3
Qwen3-VL-4B-Instruct									
No fine-tuning	81.2	**86.1**	73.7	75.7	65.8	24.0	68.0	62.7	83.9
PEARL (LVR data)	**81.7**	85.2	**76.3**	**80.0**	67.5	**27.3**	70.0	53.7	85.3
PEARL (PixelReasoner data)	79.6	82.6	75.0	77.3	**70.8**	25.3	68.0	57.5	**87.4**
PEARL (ThinkMorph data)	75.4	81.7	65.8	76.3	66.7	26.7	**75.3**	**66.4**	81.8

Table 2: Results for smaller model variants (Qwen2.5-VL-3B-Instruct and Qwen3-VL-4B-Instruct) across three training regimes. **Bold** denotes best result per block.

the two views ($\langle \mathcal{I}_0, \mathcal{Q} \rangle$ and $\mathcal{R}$) across tasks, whereas SFT produces fragmented clusters, confirming that predictive embedding alignment learns more structured representations.

5.2 What is the Effect of Training Regime on PEARL?

Although no single training regime dominates uniformly across all benchmarks, clear patterns emerge. The LVR regime (single-type, single tool call) is strongest on *visual search* tasks, yielding the highest scores on V* (81.5) and V^*_{DA} (86.1). The PixelReasoner regime (single-type, multiple tool calls) performs best on tasks requiring *counting and spatial reasoning*, leading on Counting (70.0) and matching the best result on Spatial Rel (89.5). The ThinkMorph regime (multiple-type, single tool call) stands out on *perceptual robustness* benchmarks, leading clearly on MMVP (75.3) and Rel. Ref (46.3), suggesting that exposure to diverse tool types improves fine-grained perceptual discrimination. Taken together, these results indicate that the three regimes are complementary rather than competing.A natural direction for future work is a combined training strategy that draws on all three regimes simultaneously, which we would expect to yield stronger across-the-board performance.

5.3 Does PEARL Generalise Across Model Sizes and Architectures?

Table 2 reports results for smaller model variants, demonstrating that PEARL's gains are not specific to the 7B scale or to the Qwen2.5 architecture.

PEARL trained on LVR data substantially outperforms the 3B LVR baseline across nearly all benchmarks — most strikingly on V* (73.8 vs. 64.9) and MMVP (68.7 vs. 54.7) — despite using only LoRA adapters. This mirrors the pattern observed at 7B and confirms that predictive embedding learning scales down gracefully. The PixelReasoner-trained variant performs slightly lower overall but remains competitive on Counting and Rel. Ref, consistent with the regime-specific patterns observed in Table 1.

The zero-shot Qwen3-VL-4B baseline is already strong, particularly on Jigsaw (68.0) and Rel. Ref (62.7), which substantially exceed the corresponding 7B zero-shot scores, reflecting the stronger perceptual capabilities of the Qwen3 architecture. PEARL trained on LVR data improves further on V* (81.7 vs. 81.2) and MMVP (80.0 vs. 75.7), while the PixelReasoner-trained variant gains on Spatial Rel (87.4 vs. 83.9). Both variants show some regression on Rel. Ref relative to the zero-shot baseline, which we leave to future investigation.

For the 3B variant, PEARL trained on ThinkMorph underperforms the LVR-trained variant across most benchmarks (V*: 62.8 vs. 73.8; MMVP: 60.0 vs. 68.7). This is likely due to two compounding factors: the diverse tool-type signal may require greater model capacity, and ThinkMorph's verbose, open-ended reasoning steps introduce a training-test format mismatch that smaller models struggle to overcome when producing multiple-choice answers. By contrast, the 4B Qwen3-VL variant benefits strongly from the ThinkMorph regime, achieving 75.3 on Jigsaw (vs. 68.0 zero-shot) and 66.4 on Rel. Ref (vs. 62.7 zero-shot), surpass-

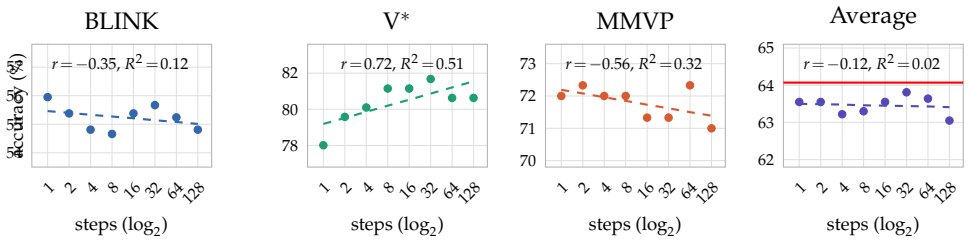


Figure 2: Correlation between the number of reasoning steps and accuracy across BLINK (n=697), V*(n=191), MMVP (n=300), and on average (n=1,188). The x-axis uses a $\log_2$ scale; dashed lines show the log-linear trend. Near-zero r and R^2 values confirm that embedding quality is stable across reasoning steps. The red line is PEARL trained on LVR.

ing both the LVR- and PixelReasoner-trained variants on these benchmarks — consistent with the 7B finding that diverse tool exposure improves perceptual discrimination.

Across both a smaller and a newer model architecture, PEARL consistently matches or improves over its respective fine-tuning baselines without any architecture-specific modifications, confirming its model-agnostic nature. The complementary strengths of the three training regimes identified at 7B (visual search (LVR), spatial and counting abilities (PixelReasoner), and perceptual robustness (ThinkMorph)) generalise across scales, although the ThinkMorph gains appear sensitive to base model capacity. Furthermore, we also report PEARL's sensitivity to LoRA ranks in Appendix A.4.

5.4 Do Reconstruction-based Methods Actually "Imagine" Images?

A central motivation for PEARL is the observation that reconstruction-based latent reasoning methods (Li et al., 2025; Yang et al., 2025; Qin et al., 2025) may not be doing what they claim. These methods assume that autoregressively generating latent tokens allows a model to "imagine" intermediate image edits in the latent space, and that more tokens should correspond to a more complete imagined transformation. Figure 3 shows that over 75% of the edited images used to supervise LVR contain more than 8 latent tokens during training, as a direct consequence of the token count scaling with the number of image patch tokens in each example, yet LVR fixes this to just 4 or 8 tokens at inference. Figure 2 further reveals that model quality does not improve as the number of latent tokens increases, and in some cases slightly degrades, with a weakly negative correlation across BLINK and MMVP. In fact, using just 1 or 2 latent tokens achieves parity with much higher token counts. This training–inference mismatch, combined with the insensitivity of performance to token count, suggests that reconstruction-based methods are not genuinely simulating visual transformations in latent space. Instead, they appear to learn useful *embeddings*: compact representations that improve answer quality regardless of how many latent tokens are decoded. This finding directly motivates PEARL: if reconstruction-based methods are learning embeddings anyway, it is more principled to learn these directly via a predictive objective, without the added complexity of autoregressive latent generation and the practical burden of switching between continuous and discrete tokens at inference.

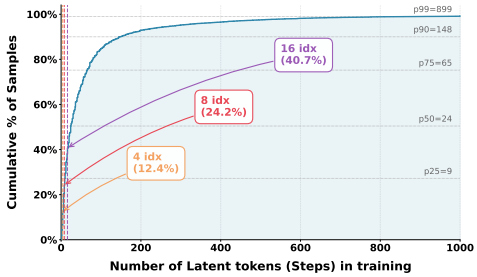
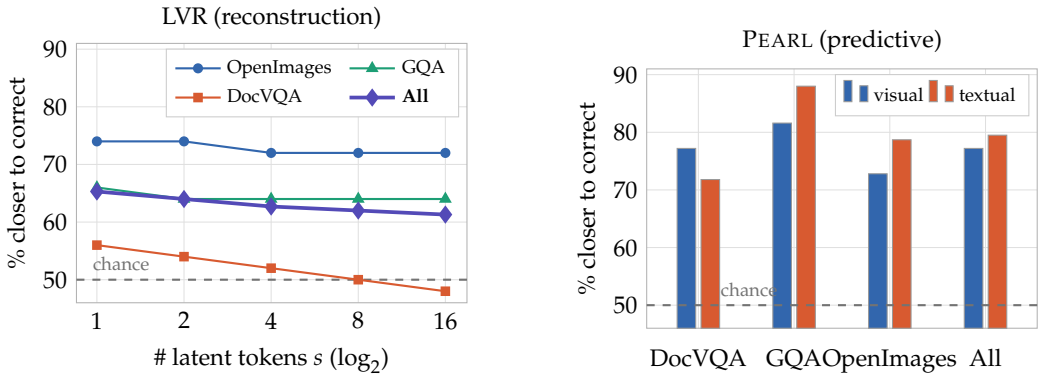


Figure 3: Cumulative distribution function (CDF) of the number of latent tokens per training example (x-axis) over a sample of ∼19k examples used to train LVR.

5.5 Do PEARL's Embeddings Internalize Tool Calls Meaningfully?

Reconstruction-based methods (Li et al., 2025; Yang et al., 2025; Qin et al., 2025) assume that generating more latent tokens yields a more complete imagined image edit. We find this does not hold. Over 75% of the images supervising LVR use more than 8 latent tokens at



(a) LVR latent tokens vs. ViT patch embeddings, across s tokens.

(b) Proportion of PEARL predicted embedding closer to correct trajectories.

Figure 4: Both panels report the fraction of examples whose learned representation is closer (in MSE) to the correct than to an incorrect/corrupted target; (a) LVR's reconstruction latents stay near chance and *decline* as more latent tokens s are decoded, with DocVQA reaching chance by s=8. (b) PEARL's predicted embeddings are significantly better than chance ($p < 10^{-18}$, Wilcoxon signed-rank) under both visual (bounding-box) and textual (reasoning-text) corruption. The two probes use different corruption targets.

training, yet only 4–8 are decoded at inference (Figure 3), and accuracy does not improve—and sometimes degrades—as the token count grows, with 1–2 tokens matching much higher counts (Figure 2). This suggests these methods learn a fixed useful *embedding* rather than progressively imagining edits. We test this directly by probing whether a representation is closer (in MSE) to the correct expert trajectory than to a corrupted one (Figure 4). For PEARL, we corrupt either the bounding boxes in the reasoning trace (the *visual* setting) or the reasoning text (the *textual* setting), and find that the predicted embeddings remain well above chance in both cases. The bounding-box corruption is the more demanding test, since it perturbs only a spatial region within the same image, and the fact that PEARL still discriminates the correct trajectory indicates that it internalizes the spatial effects of tool use rather than relying on dataset-level shortcuts. When we apply the same probe to LVR's latents, which is a fair comparison because LVR is trained to match tool-output patch embeddings under an MSE objective, we find the opposite: its representations sit near chance and in fact decline as s grows, reaching chance on DocVQA by $s = 8$.

Together with Figure 2, these results confirm that reconstruction latents encode a fixed embedding signal that does not grow richer with more tokens, whereas PEARL's predicted embeddings capture to a greater extent the visual and textual content of tool use.

6 Conclusion

We presented PEARL, a JEPA-inspired framework that learns from expert tool-use trajectories in the latent space without requiring explicit tool invocation at inference. Rather than reconstructing intermediate image edits autoregressively, PEARL directly predicts a trajectory-level embedding from the image-question pair, preserving the standard VLM inference pipeline. Across three training regimes and multiple perception benchmarks, PEARL consistently matches or outperforms reconstruction-based methods and SFT baselines using only LoRA adapters, with gains that generalise across model sizes and architectures.

Our analysis further challenges the premise of reconstruction-based latent reasoning: performance is largely insensitive to the number of latent tokens decoded at inference, suggesting these methods learn useful embeddings rather than genuinely imagining image edits. A natural direction for future work is a combined training strategy that draws on all three regimes simultaneously, as well as extending PEARL to settings with diverse, multi-step tool use and explicit latent planning at inference (see Appendix C for further discussion).

Ethics Statement

This work presents PEARL, a framework for training vision-language models to internalize the effects of visual tool use in the latent space. We discuss the ethical considerations most relevant to this research.

Intended Use and Misuse. PEARL is designed to improve the efficiency and accuracy of multimodal reasoning in VLMs for perception-intensive tasks. As with any method that improves the capability of language models, there is potential for misuse in applications that generate misleading visual interpretations or automate harmful decision-making. We encourage practitioners to apply appropriate safeguards when deploying systems built on this work in high-stakes settings.

Data and Bias. Our experiments rely on publicly available datasets (LVR, ThinkMorph, PixelReasoner) and pre-trained models (Qwen2.5-VL, Qwen3-VL). Any biases present in these data sources or base models may be inherited or amplified by PEARL. We did not conduct a systematic bias audit and caution against deployment in sensitive domains without further evaluation.

Environmental Cost. Training PEARL requires two forward passes per example, roughly doubling compute relative to standard fine-tuning. All experiments were conducted on H100 and H200 GPUs. We partially mitigate this cost by using LoRA adapters rather than full fine-tuning, and by training for a limited number of steps or epochs per regime.

Broader Impact. By eliminating the need for explicit tool invocation at inference, PEARL reduces the latency and resource cost of deploying tool-augmented VLMs, which may make capable multimodal reasoning more accessible. We release model weights and code to support reproducibility and further research.

Acknowledgments

We thank the anonymous reviewers and area chair for their feedback. We also thank Alex Gurung, Hakan Bilen and Alessandro Suglia for valuable discussions through the course of the project. This work was supported by the Edinburgh International Data Facility (EIDF) and the Data-Driven Innovation Programme at the University of Edinburgh. Access to EIDF was facilitated through the University of Edinburgh's Generative AI Laboratory GAIL Fellow scheme. Lapata gratefully acknowledges the support of the UK Engineering and Physical Sciences Research Council (grant EP/W002876/1).

References

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*, pp. 2425–2433, 2015.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025. URL https://arxiv.org/abs/2506.09985.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang,

Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Delong Chen, Mustafa Shukor, Theo Moutakanni, Willy Chung, Jade Yu, Tejaswi Kasarla, Yejin Bang, Allen Bolourchi, Yann LeCun, and Pascale Fung. Vl-jepa: Joint embedding predictive architecture for vision-language, 2026. URL https://arxiv.org/abs/2512.10942.

Jiwan Chung, Junhyeok Kim, Siyeol Kim, Jaeyoung Lee, Min Soo Kim, and Youngjae Yu. v1: Learning to point visual tokens for multimodal grounded reasoning, 2026. URL https://arxiv.org/abs/2505.18842.

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. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 148–166, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73337-6.

Xinyu Geng, Peng Xia, Zhen Zhang, Xinyu Wang, Qiuchen Wang, Ruixue Ding, Chenxi Wang, Jialong Wu, Kuan Li, Yida Zhao, Huifeng Yin, Yong Jiang, Pengjun Xie, Fei Huang, Huaxiu Yao, Yi R. Fung, and Jingren Zhou. Webwatcher: Breaking new frontiers of vision-language deep research agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=8jsaazdAb3.

Halil Alperen Gozeten, Muhammed Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=sTPKDKn5ig.

Jiawei Gu, Yunzhuo Hao, Huichen Will Wang, Linjie Li, Michael Qizhe Shieh, Yejin Choi, Ranjay Krishna, and Yu Cheng. Thinkmorph: Emergent properties in multimodal interleaved chain-of-thought reasoning, 2025. URL https://arxiv.org/abs/2510.27492.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2024. URL https://arxiv.org/abs/2412.06769.

Yunzhuo Hao, Jiawei Gu, Huichen Will Wang, Linjie Li, Zhengyuan Yang, Lijuan Wang, and Yu Cheng. Can mllms reason in multimodality? emma: An enhanced multimodal reasoning benchmark. *arXiv preprint arXiv:2501.05444*, 2025.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

Hai Huang, Yann LeCun, and Randall Balestriero. Llm-jepa: Large language models meet joint embedding predictive architectures, 2025a. URL https://arxiv.org/abs/2509.14252.

Zeyi Huang, Yuyang Ji, Anirudh Sundara Rajan, Zefan Cai, Wen Xiao, Junjie Hu, and Yong Jae Lee. Visualtoolagent (VisTA): A reinforcement learning framework for visual tool selection, 2025b. URL https://arxiv.org/abs/2505.20289.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(98)00023-X. URL https://www.sciencedirect.com/science/article/pii/S000437029800023X.

Bangzheng Li, Ximeng Sun, Jiang Liu, Ze Wang, Jialian Wu, Xiaodong Yu, Hao Chen, Emad Barsoum, Muhao Chen, and Zicheng Liu. Latent visual reasoning, 2025. URL https://arxiv.org/abs/2509.24251.

Daniel Nichols, Prajwal Singhania, Charles Jekel, Abhinav Bhatele, and Harshitha Menon. Optimizing agentic language model inference via speculative tool calls, 2025. URL https://arxiv.org/abs/2512.15834.

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. *arXiv preprint arXiv:2511.19418*, 2025.

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. Visual cot: Unleashing chain-of-thought reasoning in multi-modal language models, 2024.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. *arXiv preprint arXiv:2505.15966*, 2024.

Zhao Yu Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, and Yu Cheng. Openthinkimg: Learning to think with images via visual tool reinforcement learning. *ArXiv*, abs/2505.08617, 2025. URL https://api.semanticscholar.org/CorpusID:278534476.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=AQsko3PPUe.

Qwen Team. Qwen2.5-vl, January 2025. URL https://qwenlm.github.io/blog/qwen2.5-vl/.

Jayden Teoh, Manan Tomar, Kwangjun Ahn, Edward S. Hu, Pratyusha Sharma, Riashat Islam, Alex Lamb, and John Langford. Next-latent prediction transformers learn compact world models, 2025. URL https://arxiv.org/abs/2511.05963.

Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9568–9578, June 2024.

Bin Wu, Edgar Meij, and Emine Yilmaz. A joint optimization framework for enhancing efficiency of tool utilization in LLM agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22361–22373, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1149. URL https://aclanthology.org/2025.findings-acl.1149/.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use. 2025b. URL https://api.semanticscholar.org/CorpusID:278904571.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. *arXiv preprint arXiv:2312.14135*, 2023.

Zeyuan Yang, Xueyang Yu, Delin Chen, Maohao Shen, and Chuang Gan. Machine mental imagery: Empower multimodal reasoning with latent visual tokens, 2025. URL https://arxiv.org/abs/2506.17218.

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 CVPR*, 2024.

Jiyang Zheng, Jialiang Shen, Yu Yao, Min Wang, Yang Yang, Dadong Wang, and Tongliang Liu. Chain-of-focus prompting: Leveraging sequential visual cues to prompt large autoregressive vision models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=noidywkBba.

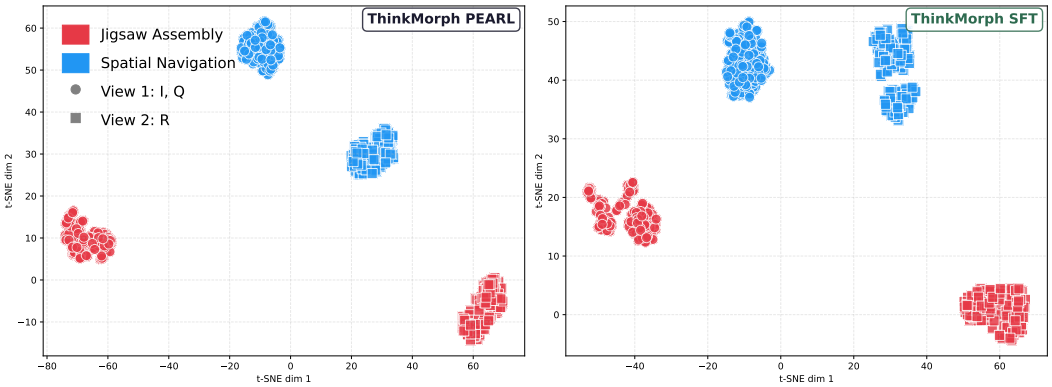


Figure 5: T-SNE visualization of the views I_0, Q and R across tasks for Qwen2.5-VL-7B-Instruct trained with PEARL on the left, compared with simple fine-tuning with next-token prediction on the right.

A Additional Results and Hyperparameter Settings

A.1 Hyperparameter Settings

We set λ in Equation (4) to 0.2 and the number of `[PRED]` tokens to 4 across all training settings.

All experiments are conducted on either 4 NVIDIA H200 or 6 NVIDIA H100 GPUs. We train on LVR data for 2,500 steps, on PixelReasoner data for 4 epochs, and on ThinkMorph data for 1 epoch, selecting the best checkpoint based on validation loss. On H200s, we use a per-device batch size of 4 with gradient accumulation of 4; on H100s, we reduce the per-device batch size to 2 to fit within memory. For all runs, LoRA adapters are configured with rank $r = 64$ and $\alpha = 128$.

At inference, we constrain model outputs to the option letter using a maximum of 4 tokens, enabling straightforward answer parsing and ensuring fair, direct comparison with LVR (Li et al., 2025).

A.2 Embedding Visualisation

Figure 5 visualizes t-SNE projections of the embeddings learned by PEARL and a LoRA SFT baseline, both trained on ThinkMorph data. For PEARL, the two views, i.e., $\langle \mathcal{I}_0, \mathcal{Q} \rangle$ and $\mathcal{R}$, form coherent, well-separated clusters that align across tasks (Jigsaw and Spatial Navigation), indicating that the predictive embedding objective encourages the model to develop shared, task-discriminative representations of the input and trajectory. By contrast, the SFT baseline produces fragmented clusters in which the two views are not consistently aligned, suggesting that next-token prediction alone does not induce the same degree of structured latent organisation. This qualitative difference is consistent with PEARL's quantitative gains and supports the view that the JEPA objective encourages more semantically meaningful representations than standard fine-tuning.

A.3 Ablation for Loss Terms

In this section, we show that using next latent predictions for training the hidden states h_t to be the *belief states* for representing views, aids model quality.

Table 1 ablates the contribution of $\mathcal{L}_{\text{NextLat}}$ by comparing PEARL against a variant trained without this objective. Removing $\mathcal{L}_{\text{NextLat}}$ leads to consistent degradation across most benchmarks, with the most notable drops on V* (80.1 vs. 81.5), V^*_{DA} (83.5 vs. 86.1), MMVP (69.3 vs. 73.5), and IQ (24.0 vs. 28.2). The only benchmark where the ablated variant is competitive is V^*_{RP} (75.0 vs. 74.5), suggesting that the next-latent objective is most beneficial for tasks requiring holistic visual understanding rather than simple relative positioning.

Model	V^*	V^*_{DA}	V^*_{RP}	MMVP	Counting	IQ	Jigsaw	Rel. Ref	Spatial Rel	Avg
Single-type, single tool call										
PEARL	**81.5**	**86.1**	74.5	**73.5**	68.3	**28.2**	53.1	39.6	89.5	**64.1**
PEARL w/o $\mathcal{L}_{\text{NextLat}}$	80.1	83.5	**75.0**	69.3	65.0	24.0	52.7	**42.5**	89.5	62.2
PEARL w/o $\mathcal{L}_{\text{NextLat}}$ (Mean Pooling h_t)	79.6	83.5	73.7	71.0	66.7	27.3	**53.3**	38.1	87.4	62.5
PEARL w/o $\mathcal{L}_{\text{JEPA}}$	79.5	82.6	**75.0**	71.0	68.3	24.0	**53.3**	35.9	**90.9**	62.4
PEARL w/o $\mathcal{L}_{\text{NextLat}}, \mathcal{L}_{\text{JEPA}}$	79.1	82.6	73.7	65.7	67.5	26.7	45.3	33.6	88.8	59.7

Table 3: Ablation of $\mathcal{L}_{\text{NextLat}}$ on the single-type, single tool call setting (LVR data, Qwen2.5-VL-7B-Instruct). Removing the next-latent prediction objective consistently degrades performance, confirming that encouraging hidden states to act as belief states improves the trajectory embeddings learned by $\mathcal{L}_{\text{JEPA}}$. **Bold** denotes best result per column.

These results support the theoretical motivation for $\mathcal{L}_{\text{NextLat}}$: by encouraging hidden states to converge to belief states (sufficient summaries of past context), the objective produces more informative encoding targets for $\mathcal{L}_{\text{JEPA}}$. Without this regularizer, the final hidden state of the decoder is a less reliable sequence representation, which in turn weakens the predictive embedding alignment signal. $\mathcal{L}_{\text{NextLat}}$ therefore acts as a necessary complement to $\mathcal{L}_{\text{JEPA}}$ rather than a redundant auxiliary objective. We also report accuracy of a model trained by replacing $\mathcal{L}_{\text{NextLat}}$ with mean pooling of hidden states to optimize the $\mathcal{L}_{\text{JEPA}} + \mathcal{L}_{\text{VLM}}$ where we see drops on V^*_{DA} (86.1 vs 83.5) and MMVP (73.5 vs 71.0).

A.4 Ablation for LoRA Rank

Table 4 reports a LoRA rank sweep for PEARL on Qwen2.5-VL-7B, trained on LVR data. Two observations stand out. First, PEARL exceeds LVR's full-decoder finetuning (63.12) at *every* rank, including r=16 (63.38), despite adapting only a small fraction of the parameters; this reinforces our earlier finding that the gains come from the predictive embedding objective rather than from finetuning capacity. Second, performance improves gently with rank up to r=128 (64.73) and then plateaus, with r=256 (64.57) offering no further benefit. We therefore use r=64 throughout, as it captures most of the available gain while remaining parameter-efficient.

Method	Avg Accuracy (%)
LVR full decoder finetuning	63.12
PEARL r=16	63.38
PEARL r=32	63.31
PEARL r=64	64.07
PEARL r=128	64.73
PEARL r=256	64.57

Table 4: LoRA Rank Sweep Results on Qwen2.5-VL-7B using LVR Data

A.5 Training and Inference Cost

PEARL requires two forward passes during training (one for the input view, one for the trajectory view), but this overhead is training-only. We measured both training and inference costs on 2x H200 GPUs on the single-step data used by LVR for training.

Metric	PEARL (LoRA)	LVR (full FT)	Ratio
GPU-hours / 1k examples	$\sim$0.55	$\sim$0.37	PEARL 1.5x more
GPU-sec / example	$\sim$1.98	$\sim$1.33	PEARL 1.5x more
Measured TFLOP / example	145,638	353,970	LVR 2.43x more

Table 5: Training cost of PEARL (LoRA) versus LVR (full finetuning).

Training cost. Training costs are reported in Table 5. As can be seen, PEARL uses approximately 1.5x more wall-clock time per example than LVR during training, reflecting the

additional trajectory-view forward pass. However, in terms of measured FLOPs, PEARL is 2.43x more efficient per example than LVR's full-decoder finetuning. The wall-clock gap is narrower than the FLOP gap because LVR benefits from sequence packing which is not enabled for PEARL.

Inference cost. At inference, PEARL achieves 1.7–2.5x lower latency than LVR while using comparable FLOPs per sample, because it avoids autoregressive latent token generation entirely. PEARL also achieves substantially higher hardware utilization ($\sim$127 vs $\sim$51–75 TFLOP/s) since standard VLM decoding is more efficient than interleaved latent-discrete generation (see Table 6 below).

Model	Latency (s)	Samples /s	Gen tok.	FLOP /sample	TFLOP /s
PEARL (direct, 4 tok)	0.76	1.32	3	96.5	$\sim$127
LVR (steps=4)	1.32	0.76	16	98.5	$\sim$75
LVR (steps=8)	1.52	0.66	20	98.6	$\sim$65
LVR (steps=16)	1.93	0.52	28	98.7	$\sim$51

Table 6: Inference cost of PEARL versus LVR across latent reasoning steps. FLOP/sample is in TFLOP.

We further note that LVR's inference overhead is conservative in this comparison because the LVR training data uses cropping operations, which produce smaller image regions and thus fewer latent tokens. For tool operations that output full-resolution images (e.g., depth estimation, segmentation, chart highlighting as in ThinkMorph), the number of latent tokens required for reconstruction would scale with the full image patch count, further widening the latency gap.

PEARL's inference cost is constant regardless of tool type, since it predicts a single trajectory-level embedding rather than reconstructing per-patch latent tokens. The LoRA sweep further shows that PEARL achieves its gains even at $r = 16$, making the method parameter-efficient relative to full-decoder finetuning.

B Examples and Dataset Statistics

B.1 Qualitative Cross-Model Comparison and Failure Analysis

For qualitative comparisons and failure analysis, we conducted a systematic cross-model comparison on BLINK, V*, and MMVP, identifying cases where PEARL, LVR, and PixelReasoner (PR) each uniquely succeed or fail.

Where PEARL wins: direct visual reading without reasoning overhead. PEARL's clearest genuine wins are on MMVP tasks requiring fine-grained scene understanding, where both LVR and PR select the same wrong answer. For example:

- "Do all the bottles in the picture face the same direction?" PEARL correctly detects the misaligned bottle (B=No), while both LVR and PR answer A=Yes.
- "Is the edge of the heart in the picture white or dark-colored?" PEARL correctly reads the dark border (B), while both LVR and PR say white (A).
- "In this image, are the berries placed on a dish or in bowls of yogurt?" PEARL correctly identifies the dish (A), while both LVR and PR infer yogurt bowls (B).

These wins suggest that PEARL's direct decoding, without intermediate latent-token generation or explicit reasoning chains, avoids over-reasoning errors that affect both LVR and PR on fine visual detail tasks. On Counting tasks, PEARL also reads off correct object counts where LVR and PR overshoot or undershoot (e.g., correctly counting 3 plates holding sandwiches where LVR guesses 4 and PR guesses 2).

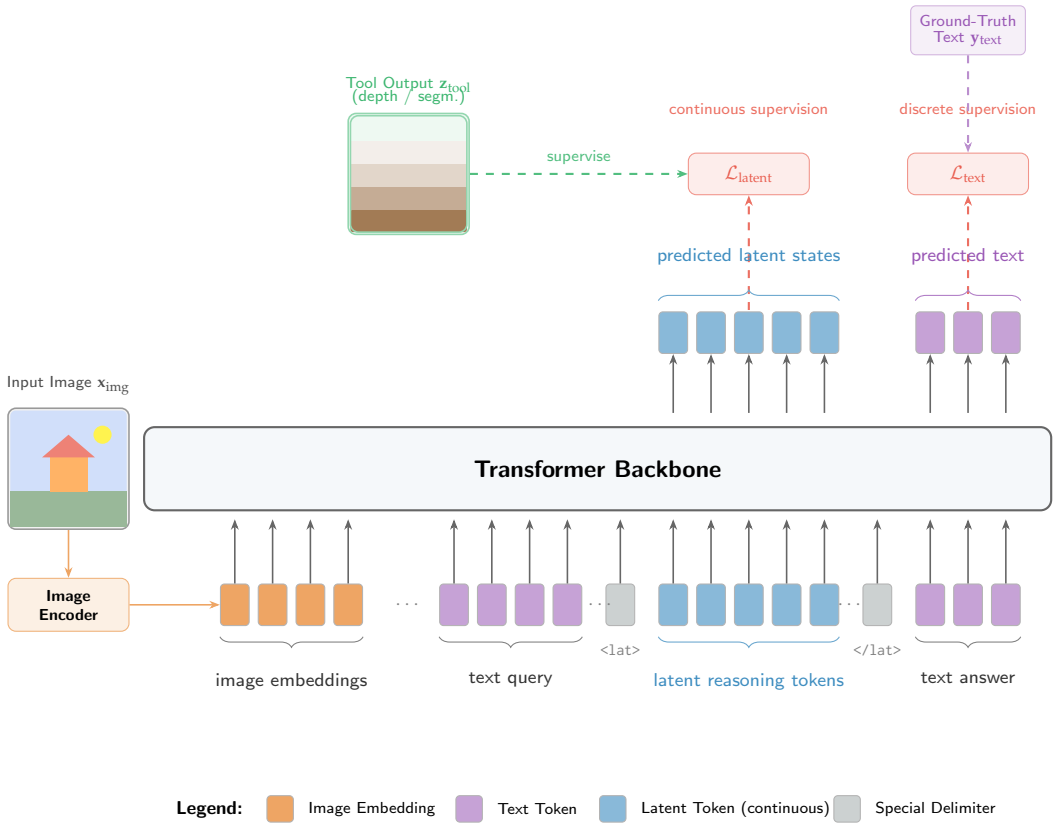


Figure 6: **Training architecture for latent-augmented multimodal reasoning.** The input image $\mathbf{x}_{\text{img}}$ is encoded into image embeddings and concatenated with text query tokens. The model autoregressively generates a sequence of continuous *latent reasoning tokens* (delimited by `<lat>`...`</lat>`), followed by discrete text answer tokens. Latent tokens are supervised with a continuous regression loss $\mathcal{L}_{\text{latent}}$ against a visual tool output $\mathbf{z}_{\text{tool}}$ (e.g. a depth map or segmentation mask), while text tokens are supervised with a standard cross-entropy loss $\mathcal{L}_{\text{text}}$ against ground-truth text $\mathbf{y}_{\text{text}}$.

Where PEARL loses: tasks requiring explicit point localization. PEARL's most systematic failures occur on Relative Reflectance, where the task requires localizing two annotated points to their underlying objects and comparing brightness. In 13 cases, PR succeeds where both PEARL and LVR fail, because PR's explicit step-by-step reasoning identifies what each point sits on (e.g., "Point A on hair (dark), Point B on glasses (reflective, slightly lighter)"), whereas PEARL and LVR default to "about the same." This reveals a clear boundary of learning for internalization: tasks requiring explicit spatial decomposition of marked regions benefit from inference-time reasoning that PEARL's trajectory-level embeddings struggle with.

Similarly, on V* direct_attributes, PEARL occasionally misidentifies fine-grained colors (e.g., predicting "silver" when the correct answer is "white" for a watch). These are however specific attribute-level errors rather than systematic reasoning failures. Our analyses provides insight into the boundary between tasks that benefit from internalization versus those requiring explicit inference-time reasoning.

B.2 Reconstruction-based Multimodal Latent Reasoning

Figure 6 illustrates the general training architecture shared by reconstruction-based latent reasoning methods. The input image is first passed through an image encoder to produce a sequence of image embeddings, which are concatenated with text query tokens and fed into a transformer backbone. The model is then trained to autoregressively generate

a sequence of continuous latent reasoning tokens, delimited by special <lat> ... </lat> markers, before switching to discrete text generation to produce the final answer. The latent tokens are supervised with a continuous regression loss $\mathcal{L}_{\text{latent}}$ against the output of an external visual tool (e.g., a depth map or segmentation mask), while the text answer tokens are supervised with a standard cross-entropy loss $\mathcal{L}_{\text{text}}$ against ground-truth text. A key characteristic of this design is the autoregressive dependency among latent tokens: each generated latent token is fed back as input to predict the next, effectively requiring the model to "imagine" the tool output token-by-token before transitioning back to discrete generation. This training-inference asymmetry — where many latent tokens are used during training but only a small fixed number are decoded at test time — is a central limitation that PEARL is designed to avoid.

B.3 Training Data

The LVR dataset (Li et al., 2025) contains over 450k training samples; however, we find that loss curves plateau well before exhausting the data, and therefore train for at most 2,500 steps, reporting results on the best checkpoint. The ThinkMorph dataset (Gu et al., 2025) comprises 6k samples per subset across four tool types, yielding 24k training samples in total. Its reasoning steps are notably more verbose than those in LVR or PixelReasoner, which contributes to the training–test distribution mismatch observed for smaller models in Section 5. We utilize the SFT dataset from PixelReasoner (Su et al., 2024), which contains $\sim 4k$ samples with sequential image transformations over $\mathcal{I}_0$, comprising between 0 and 3 sequential crops of the original image.

In Section B.5, we provide examples of training samples from Thinkmorph (Gu et al., 2025), Viscot (Shao et al., 2024)–which was used by LVR (Li et al., 2025), and PixelReasoner's (Su et al., 2024) finetuning dataset.

B.4 Evaluation Data

We draw our evaluation tasks from MMVP (Tong et al., 2024), V* (Wu & Xie, 2023), and subsets of the Blink dataset (Fu et al., 2025). Table 7 provides a brief description of each dataset along with the number of evaluation samples.

Dataset	Description	No. Samples
V*	Object attributes	191
MMVP	Perception Robustness	300
Counting	Counting objects	120
IQ	Pattern Matching	150
Jigsaw	Multi-image jigsaw resolution	150
Relative Reflectance	Perception	134
Spatial Relation	Relation between objects	143

Table 7: Overview of datasets and number of evaluation samples.

B.5 Dataset Examples

To illustrate the differences in trajectory format across our three training regimes, we show representative examples from PixelReasoner (Su et al., 2024), Viscot (Shao et al., 2024) (used by LVR), and ThinkMorph (Gu et al., 2025). The examples highlight how the tool type and reasoning style vary across datasets — from concise bounding-box crops to the more verbose, open-ended reasoning of ThinkMorph — which underlies the format-mismatch effects discussed in Section 5.

PixelReasoner Example #13: Visual Question Answering with Tool Use

System

You are a helpful assistant. You may call one or more functions to assist with the user query. Available tools:

- `crop_image_normalized(bbox_2d, target_image)` — zoom into a bounding-box region of an image.
- `select_frames(target_frames)` — select frames from a video.

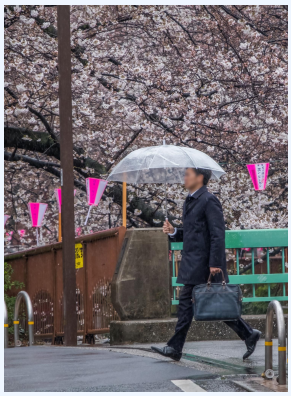
For each function call, return a JSON object inside `<tool_call>...</tool_call>` tags.

User

What does the text on the pink lantern say?

A: Sakura Festival **B:** Cherry Blossom Viewing
C: Hanami Celebration **D:** Spring Lanterns

Guidelines: Understand the visual information and the query. Determine whether to use the provided tools. Reason step by step and put your final answer inside $\{\}$.



Assistant

The image shows a serene outdoor scene with cherry blossoms in full bloom. A person holding an umbrella walks across a bridge. Pink lanterns with text are strung along the area, likely part of a festival.
I will zoom in to read the text on the pink lantern more clearly.

tool_call

```
{"name": "crop_image_normalized",
 "arguments": {"bbox_2d": [0.0, 0.3, 0.4, 0.6], "target_image": 1}}
```

Tool Result

Here is the cropped image:

Assistant

I can see the lanterns more clearly now, but the text needs a closer look.

```
tool_call
{"name": "crop_image_normalized",
 "arguments": {"bbox_2d": [0.1, 0.1, 0.2, 0.2], "target_image": 2}}
```

Tool Result

Here is the cropped image:

Assistant

The close-up clearly shows the text on the pink lantern reads **"Sakura Festival"**.

$$\{A\}$$

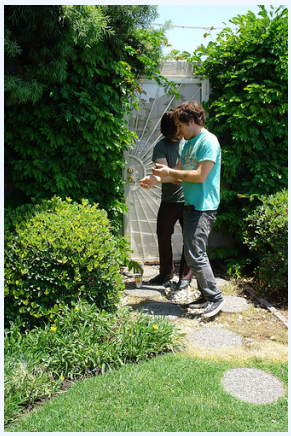
Viscot (LVR) Example

System

You are a helpful assistant.

User

What are the young men doing with their hands? Provide a short and direct response.



Assistant

To determine what the young men are doing with their hands, I will focus on the hand region of the individuals visible in the image. I will draw a red bounding box around that region at coordinates `[0.482, 0.396, 0.548, 0.45]`.

Tool Result

Assistant

They are looking at their hands.

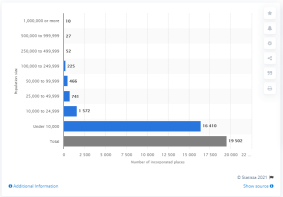
ThinkMorph Example: Chart Refocus

System

You are a helpful assistant.

User

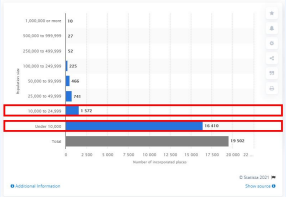
What was the total amount of towns, cities, and villages in the US in 2019 with population less than 25000?



Assistant

To answer the question about the total number of towns, cities, and villages in the US in 2019 with a population less than 25,000, I need to identify the relevant segments in the chart that correspond to this population range. The visualization displays the number of incorporated places grouped by population brackets. Specifically, I should look for the bars labeled 'Under 10,000' and '10,000 to 24,999', since both categories fall under the threshold of 25,000 residents. No other category applies, as the next smallest bracket, '25,000 to 49,999', exceeds the limit set by the query. By highlighting these two horizontal bars, I can clearly indicate the areas that collectively represent all places with populations below 25,000. This focus directly connects to the population constraint in the question and isolates the relevant data needed to compute the total. Therefore, the manipulation involves drawing a red bounding box around both the 'Under 10,000' and the '10,000 to 24,999' rows in the chart.

Tool Result



Assistant

To determine the total number of towns, cities, and villages in the US in 2019 with a population less than 25,000, I first identify the relevant population categories in the chart. The two categories that fit this criterion are 'Under 10,000' and '10,000 to 24,999', which are highlighted with red boxes in the edited visualization. The numbers associated with these categories are 16,410 for 'Under 10,000' and 1,572 for '10,000 to 24,999'. Summing these two values gives the total number of incorporated places with populations below 25,000: 16,410 + 1,572 = 17,982. Therefore, the total amount of towns, cities, and villages in the US in 2019 with population less than 25,000 is 17,982.

ThinkMorph Example: Visual Search

System

You are a helpful assistant.

User

What is the fruit that is on the plate called?

Assistant

To answer the question about the fruit on the plate, begin by visually scanning the image for a plate. The plate is located in the lower right portion of the image, on the tabletop in front of the child. Next, look closely at what is placed on the plate. There is a pale yellow, curved object with a thick peel partially removed, exposing the inner fruit. The shape, color, and partially peeled nature are strong visual cues that this is a fruit. The red bounding box should be drawn around this item on the plate to highlight it as the focal point for identifying the fruit.

Tool Result

Assistant

With the fruit on the plate now highlighted by the red bounding box, it's easy to closely examine its characteristics. The fruit has a distinctive elongated, curved shape and a pale yellow peel that is partially removed, revealing the soft, creamy flesh inside. These visual details—such as the color, the way the peel splits away from the fruit, and the overall shape—are classic identifiers for a banana. By focusing on the region inside the red bounding box, we can confidently determine that the fruit on the plate is a banana.

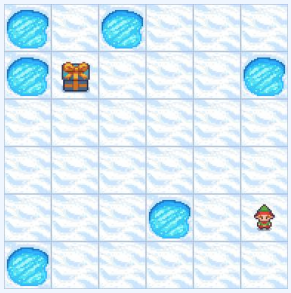
ThinkMorph Example: Spatial Navigation

System

You are a maze solver. Your goal is to guide a player from the start to the goal on a grid map while avoiding holes. The player can move one square at a time in the directions left (L), right (R), up (U), or down (D). The frozen lake is not slippery; the player will always move in the intended direction. Moving off the edge or falling into a hole results in failure. Reaching the goal means success.

User

Provide your solution as a sequence of moves wrapped in `boxed`{}, such as {L,R,U,D}. The moves should be comma-separated.



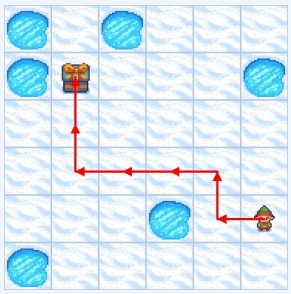
Assistant

To get a clear sense of the challenge, I should first lay out the details of the maze. The grid is 6 rows by 6 columns. The starting point is in the bottom right corner, specifically at row 5, column 5 if counting from the top-left as (0,0). The goal is near the top left, at row 1, column 1.
There are several ice holes scattered around the grid:

-
- The first row has holes at positions (0,0) and (0,2).
- The second row contains two holes: one at (1,0) and another at (1,5).
- The fourth row has a hole at (4,3).
- The last row (row 5) has a hole at (5,0).

All other squares are safe frozen surfaces. Time to move on to plotting the solution and finding the optimal path.

Tool Result



Assistant

Reviewing my solution, the red path begins at the starting square in the bottom right and follows the moves: Left, Up, Left, Left, Left, Up, Up.
Step by step:

- First, move left to avoid the edge.
- Then, up to stay clear of the hole below.
- Continue left three times, traversing safe frozen squares and bypassing the hole above.
- Go up twice to reach the goal square near the top left, successfully avoiding all holes and edges.

Each move lands on a safe frozen surface, and the path finishes directly on the goal. This confirms the route is valid and optimal.
Final solution: {L,U,L,L,L,U,U}

C Limitations and Future Work

While PEARL demonstrates strong performance across three training regimes and multiple model scales, several limitations remain.

Planning over Learned Embeddings. PEARL internalizes the *effects* of tool use in the latent space but does not explicitly plan sequences of actions at inference. The predictive embeddings learned by $\mathcal{L}_{\text{JEPA}}$ encode a holistic representation of the full expert trajectory, which implicitly captures planning structure, but the model does not reason step-by-step over these representations at test time. A natural extension would be to use the learned embeddings as a latent world model for explicit multi-step planning, enabling the model to reason about longer action horizons without invoking external tools.

Interpretability. Because PEARL operates entirely in a continuous embedding space, the learned trajectory representations are not directly interpretable. Unlike reconstruction-based methods, which at least nominally produce latent tokens aligned with intermediate image edits, PEARL makes no claim about what individual dimensions of the embedding encode. While the t-SNE visualizations in Figure 5 confirm that the representations are structured and task-discriminative, understanding *what* the model has internalized about tool use remains an open question. Developing probing methods or disentangled representations that make the learned latent structure more transparent is an important direction for future work.

Training Data Coverage. Our experiments are limited to settings for which open-source expert trajectory data exists. In particular, we do not evaluate the multiple-type, multiple tool call setting due to the absence of suitable training data. As more diverse trajectory

datasets become available, we expect PEARL's trajectory-level embedding objective to generalize naturally to richer tool-use settings, given that it places no constraints on the number or type of tools present in $\mathcal{R}$.

Training Cost. PEARL requires two separate forward passes per training example to encode the input view and the trajectory view independently, which roughly doubles the training-time compute relative to standard SFT. While this overhead does not affect inference, it may be prohibitive at very large scale. Exploring more efficient encoding strategies, such as shared encoders with cross-view masking or cached trajectory embeddings, is a promising avenue for reducing this cost.