Latent Implicit Visual Reasoning

Supplementary Material

Here, we provide additional details on additional experiments (Section 1), datasets (Section 2), training setup (Section 3), and visualizations (Section 4).

1. Additional Experiments

1.1. Position of latents

We compare placing latents before versus after the prompt. Placing latents after the prompt (our default) as opposed to before the prompt yields higher accuracy across all three tasks. Specifically, for the Localization, Semantic Correspondence, and Functional Correspondence tasks, we get scores of (83.61 vs. 80.33), (64.75 vs. 61.87), and (67.81 vs. 63.70), respectively, for placing the latents after vs. before the prompts. We hypothesize that when latents appear before the prompt, they cannot condition on the question and are farther from answer tokens, making them harder for the model to exploit effectively.

2. Datasets

We evaluate LIVR on **nine** complementary, perception-heavy tasks that together span low-level (visual correspondence, relative reflectance), mid-level (jigsaw, art style classification), and higher-level (localization, counting, visual similarity, semantic correspondence, functional correspondence) visual reasoning. Building on the BLINK benchmark and PixMo-Count, we derive evaluation splits that represent a broad and diverse testbed that stresses the model across heterogeneous task types and difficulty levels.

We provide detailed descriptions of the datasets and task setups used in our experiments. For each of the nine perception-heavy tasks - counting, jigsaw, object localization, visual correspondence, art style, semantic correspondence, functional correspondence, relative reflectance, and visual similarity - we describe: (i) the data sources and train/validation/test splits, and (ii) the VQA-style prompt templates used during training and evaluation.

2.1. Counting

2.1.1. Data Sources and Splits

We use the PixMo-Count dataset [3], available as `allenai/pixmo-count` on HuggingFace, which provides an image URL, an object label (e.g., "people", "cars"), and an integer count for each example, with official `train`, `validation`, and `test` splits.

For training, we construct a 1,000-example subset of the PixMo-Count `train` split by first discarding any examples whose remote image URLs no longer resolve and then restricting to images whose ground-truth counts lie in the range $c \in \{2, 3, \ldots, 10\}$ and sampling examples so that these counts are approximately uniformly represented, matching the range we evaluate on. The PixMo authors note that the official splits may contain overlapping images, so we perform an additional visual de-duplication step between our `train` + `validation` images and the official PixMo-Count `test` images: using CLIP [11] embeddings together with perceptual hashing and SSIM-based image similarity [12], we flag near-duplicate pairs and remove any training/validation example whose image is a near-duplicate of a test image. From the remaining pool, we obtain 1,000 training instances.

For validation and test, we use the official PixMo-Count `validation` and `test` splits, discarding any examples whose remote image URLs no longer resolve; these contain 534 and 528 examples, respectively.

2.1.2. Prompt Template

We phrase Counting as an open-ended task, using the following prompt template:

2.2. Jigsaw

2.2.1. Data Sources and Splits

We construct BLINK-style Jigsaw training and validation sets from COCO [8]. Starting from COCO2017

train2017 and val2017, we sample 1,000 and 250 images for our train and val splits, respectively. For each image, we first sample a horizontal canvas of fixed width 400 px and random height in $[170, 230]$ px, then crop a $400 \times h$ region from the original image. We partition this canvas into four equal quadrants and treat the bottom-right quadrant as the ground-truth patch. The input image is obtained by blacking out this quadrant, while the correct option is the original bottom-right patch. We then sample a distractor patch of the same size from the same COCO image, enforcing that it (i) intersects the canvas, (ii) does not overlap the ground-truth patch, and (iii) has its center at least a fixed fraction of the canvas size away from the ground-truth center. Finally, we randomly assign the ground-truth patch to option A or B, and use the other as the distractor. We ensure that each COCO image (identified by its file stem) appears in at most one of our Jigsaw train/val splits, so there is no cross-split image overlap within our generated data.

For our test split, we use the official BLINK Jigsaw val split (150 examples), which is constructed from the TARA dataset [5] rather than COCO. Because our COCO-based Jigsaw training/validation sets and the BLINK Jigsaw benchmark come from disjoint source datasets, we do not perform any additional train–test de-duplication across them.

2.2.2. Prompt Template

We phrase Jigsaw as a two-way multiple-choice question over candidate patches, using the following prompt template:

Prompt: Given the first image with the lower right corner missing, can you tell which one of the second image or the third image is the missing part? Imagine which image would be more appropriate to place in the missing spot. You can also carefully observe and compare the edges of the images. Select from the following choices.
(A) the second image
(B) the third image

Gold: A

2.3. Object Localization

2.3.1. Data Sources and Splits

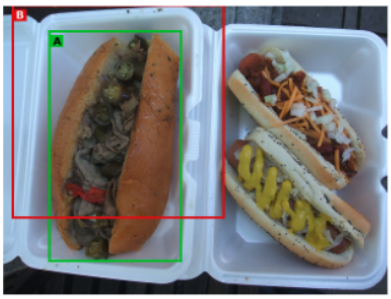
We construct a 1,000-example training set and 250-example validation set for object localization from the COCO 2017 detection splits [8]. Starting from the official train2017 and val2017 annotations, we first filter for non-crowd instances whose bounding boxes cover between 15% and 50% of the image area and whose segmentation masks fill at least 60% of the box area. For each selected instance, we treat its COCO bounding box as the "gold" localization and generate a distractor box by jittering the four box corners until the resulting box attains an IoU in $[0.2, 0.5]$ with the gold box. We then render both boxes onto the image, label them as A and B, and phrase the task as a two-way multiple-choice question asking which box more accurately localizes the object. Because both our COCO-based data and the BLINK Object Localization benchmark are derived from the same underlying COCO imagery and overlay annotated bounding boxes, we perform an explicit visual de-duplication step between our COCO candidates (train + validation) and the BLINK Object Localization val images: using CLIP embeddings together with perceptual hashing and SSIM-based image similarity (on both raw and blurred grayscale images), we flag near-duplicate pairs and remove any COCO example whose image is a near-duplicate of a BLINK image. We also enforce that each COCO image (identified by its image ID) appears in at most one of our object-localization train/val splits, so there is no cross-split image overlap within our generated data. From the remaining pool of candidates, we obtain 1,000 training and 250 validation instances.

For our test split, we use the BLINK Object Localization val split (122 examples).

2.3.2. Prompt Template

We phrase Object Localization as a two-way multiple-choice question over candidate bounding boxes, using the following prompt template:

Object Localization



Prompt: A bounding box is an annotated rectangle surrounding an object. The edges of bounding boxes should touch the outermost pixels of the object that is being labeled. Given the two bounding boxes on the image, labeled by A and B, which bounding box more accurately localizes and encloses the {sandwich}? Select from the following options.
(A) Box A
(B) Box B

Gold: A

2.4. Visual Correspondence

2.4.1. Data Sources and Splits

We construct BLINK-style visual correspondence data from the HPatches sequences dataset [1]. We download the official HPatches sequence release (including homographies) and treat each sequence of six aligned images as a unit. Viewpoint and Illumination sequences are shuffled separately and then split 80/10/10 into `train`, `val`, and `test` subsets, after which we merge the viewpoint and illumination partitions for each split. This ensures that each HPatches sequence, and hence each source/target image pair, appears in exactly one split.

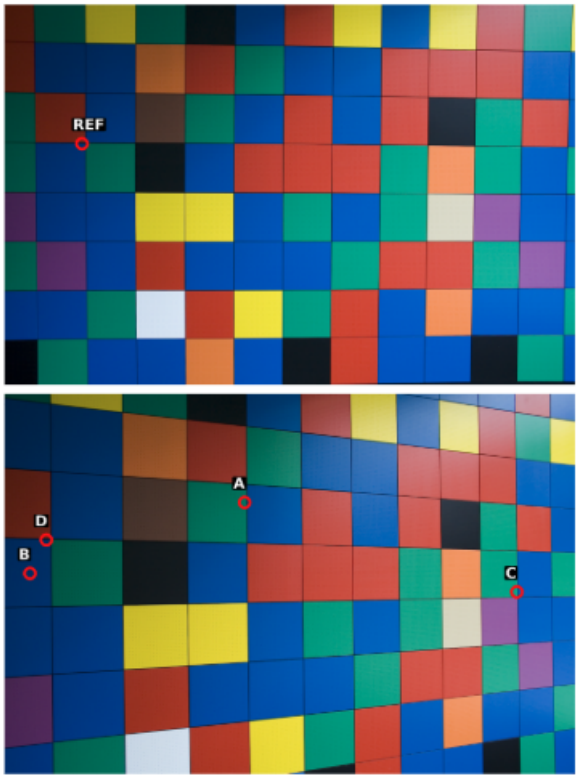
Within each sequence, we consider all forward image pairs (i, j) with $1 \leq i < j \leq 6$ and $j - i \geq 2$, following the six-image HPatches protocol. For a given pair, we use the provided homographies to map points from the source image i to the target image j. On the source image, we repeatedly sample reference points away from the image boundary and from one another, warp them to the target using the $i \rightarrow j$ homography, and retain only those whose projections lie safely inside the target image. For each valid reference point, we create a single multiple-choice example: the source image shows the reference point annotated as "REF", and the target image shows four candidate locations (labeled A–D), one of which is the true correspondence and three of which are distractors sampled to be far from the true location and from each other. We randomly assign the correct correspondence to one of the four labels, yielding BLINK-style MCQs. Across the HPatches sequences, we generate 1,000 training, 500 validation, and 700 test examples.

Because both the BLINK Visual Correspondence benchmark and our training data are derived from HPatches, using the official BLINK split together with strict de-duplication would leave too little training and validation data. Instead, we adopt the BLINK task format but evaluate on the HPatches-based `test` split constructed above.

2.4.2. Prompt Template

We phrase Visual Correspondence as a four-way multiple-choice question over candidate correspondence points, using the following prompt template:

Visual Correspondence



Prompt: A point is circled on the first image, labeled with REF. We change the camera position or lighting and shoot the second image. You are given multiple red-circled points on the second image, choices of "A, B, C, D" are drawn beside each circle. Which point on the second image corresponds to the point in the first image? Select from the following options.
(A) Point A
(B) Point B
(C) Point C
(D) Point D

Gold: D

2.5. Art Style

2.5.1. Data Sources and Splits

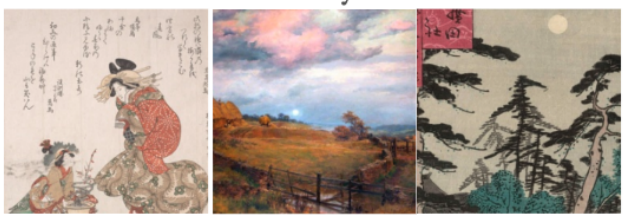
We construct a 1,000-example training set and 250-example validation set for binary art-style classification from the ArtBench-10 dataset [7]. Starting from the official 256×256 ImageFolder variant with `train`/`test` splits, we first build a style-balanced pool of paintings from the ArtBench `train` split. Each example is then converted into a binary multiple-choice question: given a reference painting and two candidate paintings (A and B), the model must decide which candidate shares the same style as the reference. For each reference, we sample a positive candidate from the same ArtBench style (but a different image) and a negative candidate from a different style, and we randomize whether the positive candidate appears as option A or B. We ensure that each underlying ArtBench image appears in at most one of our `train`/`val` splits, so there is no cross-split image overlap within our generated data. Because ArtBench-10 also draws from large online art repositories (including WikiArt), the ArtBench and BLINK corpora can share overlapping images. To prevent train–test leakage, we perform an explicit cross-dataset de-duplication between our ArtBench-based training/validation pool (considering all three images per example: reference and both candidates) and the BLINK Art Style `val` images. Using CLIP image embeddings to retrieve high-similarity pairs, followed by perceptual hashing and SSIM-based image similarity checks, we flag near-duplicate pairs and remove any training/validation example whose images are near-duplicates of BLINK Art Style examples. From the remaining pool of candidates, we obtain 1,000 training and 250 validation examples.

For our `test` split, we use the BLINK Art Style `val` split (117 examples).

2.5.2. Prompt Template

We phrase Art Style as a two-way multiple-choice question over candidate images, using the following prompt template:

Art Style



Prompt: Some most common art painting styles include Realism, Impressionism, Expressionism, Pop Art, and Cubism.
Given the following images of art paintings, use the first image as the reference image, and determine which one of the second or the third image shares the same style as the reference image?
Select from the following choices.
(A) the second image
(B) the third image

Gold: B

2.6. Semantic Correspondence

2.6.1. Data Sources and Splits

For semantic correspondence, we construct a 1,000-example training set and 250-example validation set from SPair-71k [9], which provides object-category image pairs with dense keypoint annotations. We use the official training split (`trn`) and filter to pairs with at least four valid keypoint correspondences (finite, non-negative coordinates in both source and target). From this pool, we construct BLINK-style four-way MCQs: for each selected pair, we mark a single reference point on the source image at one annotated keypoint (labeled "REF"), and on the target image we place four candidate circles (A–D), consisting of the true corresponding keypoint and three distractor keypoints sampled from the remaining annotations. All annotations are drawn directly on the original-resolution images, and we randomly assign the correct correspondence to one of the four labels. From the resulting candidates, we obtain 1,000 training and 250 validation examples. Because both our custom subset and BLINK's Semantic Correspondence task are derived from SPair-71k, we explicitly de-duplicate our SPair-based train/validation pool against the BLINK Semantic Correspondence `val` split. We perform a pair-aware similarity check: for each BLINK pair, we retrieve a small

set of nearest custom pairs under CLIP-based image similarity, consider both aligned and swapped orientations, and treat a pair as a duplicate only if both images pass strict perceptual-hash and SSIM criteria. Any flagged custom examples are removed. We obtain a total of 1,000 training and 250 validation instances.

For our `test` split, we use the BLINK Semantic Correspondence `val` split (139 examples) as the held-out test set, without further resampling or modification.

2.6.2. Prompt Template

We phrase Semantic Correspondence as a four-way multiple-choice question over candidate correspondence points, using the following prompt template:

Semantic Correspondence

Prompt: Humans can find corresponding points for different objects in the same category. For instance, if there are images of two different cats, then the left ear tip of one cat corresponds to the left ear tip of the other cat, and the right front paw of one cat corresponds to the right front paw of the other cat.
Given the following two images, a reference point is annotated on the first image, labeled with REF. You are given multiple red-circled points on the second image, choices of "A, B, C, D" are drawn beside each circle. Select between the choices on the second image and find the corresponding point for the reference point. Which point is corresponding to the reference point?
Select from the following choices.
(A) Point A
(B) Point B
(C) Point C
(D) Point D

Gold: D

2.7. Functional Correspondence

2.7.1. Data Sources and Splits

For functional correspondence, we construct BLINK-style multiple-choice questions from the FunKPoint dataset [6], which provides per-image action labels (e.g., "pour", "scoop") together with five normalized functional keypoints. We first form a global, disjoint 80/10/10 split of FunKPoint images into `train`, `val`, and `test`, using an action-aware balancing scheme so that the per-action image counts are approximately preserved across splits. Each underlying FunKPoint image appears in exactly one split.

Within each split and action, we then pair images that share the same action using a one-use-per-image policy: images are randomly shuffled and grouped into disjoint pairs, ensuring that no image is reused within that split and action. For a given pair, we treat one image as the left (source) and the other as the right (target). On the left image, we sample a reference keypoint index $k \in \{1, \ldots, 5\}$ and mark the corresponding functional keypoint with a red circle labeled "REF". On the right image, we annotate four candidate keypoints (A–D): the keypoint with the same index k (the true correspondence) and three distractor keypoints sampled from the remaining indices $\{1, \ldots, 5\} \setminus \{k\}$. We randomly assign the correct correspondence to one of the four labels and phrase the task as a BLINK-style MCQ: given the action and the left "REF" point, select which candidate (A–D) on the right matches it. Across all actions, this procedure yields 1,000 training, 144 validation, and 146 test examples.

Because both our functional-correspondence data and the BLINK Functional Correspondence benchmark are derived from FunKPoint, using the official BLINK split together with strict de-duplication would leave too little training and validation data. As in our visual-correspondence setup, we therefore adopt the BLINK task format but evaluate on the FunKPoint-based `test` split constructed above, rather than on BLINK's Functional Correspondence split.

2.7.2. Prompt Template

We phrase Functional Correspondence as a four-way multiple-choice question over candidate correspondence points, using the following prompt template:

Functional Correspondence

Prompt: Humans can find corresponding points for the same action between different objects. For instance, if a person uses a pot versus a hammer to "Mash Pound", then the handle of the pot will be the corresponding point to the handle of the hammer because they serve the same function for the action -- to hold; and the bottom of the pot will be the corresponding point to the face of the hammer because they both mash the other object.
Given the following two images, a reference point is annotated on the first image, labeled with REF.
You are given multiple red-circled points on the right image, choices of "A, B, C, D" are drawn beside each circle.
Select from the choices on the second image and find the corresponding point for the reference point, if we use both items for the action: {"Brush/Dust"}. Which point is corresponding to the reference point?
Select from the following choices.
(A) Point A
(B) Point B
(C) Point C
(D) Point D

Gold: D

2.8. Relative Reflectance

2.8.1. Data Sources and Splits

For relative reflectance, we construct three-way multiple-choice questions from the Multi-Illumination Dataset (MID) [10], which provides images of indoor scenes under multiple point-light directions together with per-pixel diffuse albedo. Starting from the official MID training split, we select a subset of scenes and 25 illumination directions per scene, download sRGB images at a fixed mip level, and attach the corresponding albedo images from the released archives. For each RGB–albedo pair, we generate a single example by sampling two spatially separated points on the image, converting the albedo to linear RGB, and computing the local disk-averaged luminance at each point. Let Y_A and Y_B be the luminance values at the two locations; we define the relative difference

$$\text{rel} = \frac{|Y_A - Y_B|}{\max(Y_A, Y_B, 10^{-8})}.$$

If $\text{rel} \leq 0.10$, we assign label (C) "About the same"; otherwise we assign (A) "A is darker" or (B) "B is darker" according to which point has lower luminance. We control the sampling schedule so that the "About the same" class constitutes roughly one quarter of the data. From this generated pool, we randomly subsample 1,000 training and 250 validation examples.

For our `test` split, we use the BLINK Relative Reflectance `val` split (134 examples). Because our training and validation data are derived from MID while BLINK Relative Reflectance is built on IIW [2], no additional cross-dataset de-duplication is required.

2.8.2. Prompt Template

We phrase Relative Reflectance as a three-way multiple-choice question over relative surface brightness, using the following prompt template:

Relative Reflectance

Prompt: Two points are annotated on the image, labeled by A and B. Consider the surface color of the points (the albedo of the surface, without the effect of shading). Which point has darker surface color, or the colors is about the same?
Select from the following choices.
(A) A is darker
(B) B is darker
(C) About the same

Gold: B

2.9. Visual Similarity

2.9.1. Data Sources and Splits

For visual similarity, we use the NIGHTS dataset introduced in the DreamSim work [4] as our training and validation source, NIGHTS provides human-tested triplets consisting of a reference image and two candidates, together

with votes indicating which candidate is perceptually closer to the reference. Each triplet is converted into a three-image example (`image_1`, `image_2`, `image_3`), where `image_1` is the reference, `image_2` and `image_3` are the two candidates in randomized order, and the label is "A" or "B" depending on whether the second or third image is judged more similar to the reference. Because both our NIGHTS triads and BLINK Visual Similarity are derived from the same underlying source, we enforce strict data deduplication. We deduplicate against BLINK by dropping any triad whose reference or candidate image is a near-duplicate of a BLINK Visual Similarity image, using a CLIP-based pre-filter followed by perceptual hashing and SSIM to confirm matches. After de-duplication, we retain 1,000 training and 250 validation triads from NIGHTS.

For our `test` split, we use the BLINK Visual Similarity `val` split (135 examples).

2.9.2. Prompt Template

We phrase Visual Similarity as a two-way multiple-choice question over candidate images, using the following prompt template:

Visual Similarity

Prompt: Given three similar but different images, take the first image as reference. Can you tell which one of the latter two images is most similar to the first one?
Select from the following choices.
(A) the second image
(B) the third image

Gold: B

3. Training Setup

Single-task experiments. Single-task experiments use the following protocol. For each of the nine perception-heavy tasks, we construct an independent training set of 1,000 examples and fine-tune a separate model per task. For a given task, the Direct SFT baseline is trained for 10 epochs on the 1,000 training examples. LIVR uses a two-stage schedule with 4 epochs of Stage 1 (masked bottleneck training) followed by 6 epochs of Stage 2 (unmasked fine-tuning) on the same per-task training set, with $K = 16$ latent tokens. We select the checkpoint with the highest validation accuracy for reporting.

Multi-task experiments. For the multi-task setting with Qwen3-VL-4B-Instruct, we train on a combined dataset of six tasks (counting, localization, visual correspondence, semantic correspondence, functional correspondence, relative reflectance), using 1,000 training examples per task (6,000 total). Direct SFT is trained for 5 epochs, while LIVR is trained for 2 epochs of Stage 1 and 3 epochs of Stage 2, with $K = 16$ latent tokens. We report performance using the final checkpoint.

Optimization details. We adopt LoRA for both attention and MLP modules of the language backbone, with rank $r = 16$, $\alpha = 32$, and dropout 0.05. We keep the vision encoder and projector frozen. For Direct SFT, only the LoRA parameters are trainable; for LIVR we additionally unfreeze the rows of the embedding table corresponding to the K latent tokens, while all other embeddings remain frozen. We use AdamW with learning rate 1×10^{-4}, weight decay 0.01, betas $(0.9, 0.999)$, and $\epsilon = 10^{-8}$. Training uses a per-device batch size of 1 with 8 gradient-accumulation steps, giving an effective batch size of 8. The learning-rate schedule uses a 5% linear warmup followed by cosine decay; LIVR applies a separate schedule to each stage with the same warmup and cosine decay.

Mirage comparison. For the head-to-head comparison with Mirage, we align the number of stages and epochs but keep each method's own two-stage objective. For LIVR, we set $K = 4$ latent tokens and train for 10 epochs in Stage 1 (LIVR's masked bottleneck) and 10 epochs in Stage 2 (LIVR's unmasked fine-tuning). For Mirage, we likewise use $K = 4$ latent tokens, training for 10 epochs with Mirage's Stage 1 objective and 10 epochs with Mirage's Stage 2 objective. In this comparison we use full fine-tuning of the language backbone (while keeping the vision encoder and projector frozen), with AdamW (learning rate 1×10^{-5}, weight decay 0.01, betas $(0.9, 0.999)$, $\epsilon = 10^{-8}$) and the same effective batch size of 8 (per-device batch size 1 with 8 gradient-accumulation steps). The learning-rate schedule again uses a 5% linear warmup followed by cosine decay for each stage.

Compute resources. All experiments are run with a fixed random seed of 42 on a single node equipped with 8 NVIDIA RTX 6000 Ada GPUs, using single-GPU training for each run.

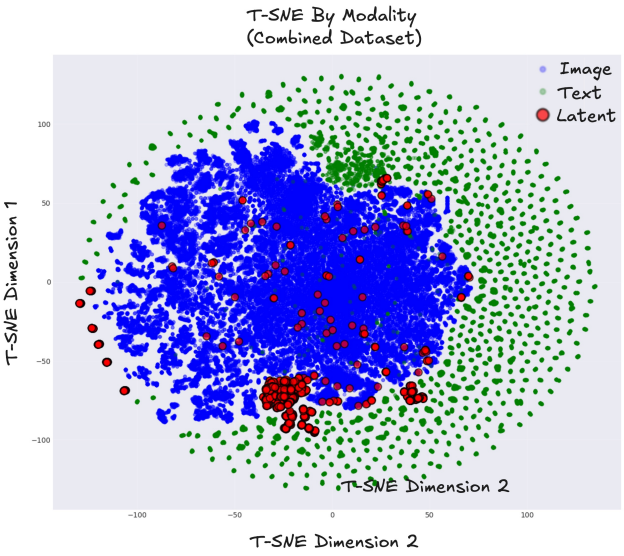


Figure 1. t-SNE Visualization of Different Tokens

4. Additional Visualizations

Latent Token t-SNE. To probe what our latent tokens represent, we visualize token hidden states using t-SNE. We first train Qwen3-VL-4B-Instruct in the multi-task setting on the six-task mixture used in Sec. **??**. We then extract the final-layer hidden states for all token positions (latent, image, and text) from the first 50 evaluation examples of each of the six tasks, and embed them with t-SNE (Fig. 1; red: latent, blue: image, green: text). Latent tokens largely occupy the same region as image tokens in the t-SNE projection, suggesting that many latent representations align with the model's visual feature space. At the same time, a compact cluster of latent tokens forms a distinct region, suggesting that some latents learn specialized representations not fully captured by the image-token manifold.

4.1. Additional Latent to Image Attentions

We show in Figure 2 some additional examples of latent-to-image attentions for Qwen-3-VL-4B. For the Visual Similarity and Art Style tasks, we display the raw attention maps instead. For question types where images serve as the choices (Jigsaw, Visual Similarity, and Art Style), we display a check mark symbol next to the correct answer. We can see that in the Visual Correspondence example, the latent tokens focus correctly on the parking line that is indicated by the REF point. In the Semantic Correspondence task, the model focuses correctly on the toothbrush handle. In the Jigsaw task, the latent tokens actually focus on diagonal ledge of the table in the masked out reference image, and finds this feature in Choice B, which is the correct answer. In Relative Reflectance, the latent tokens focus on the 2 points that it needs to compare. Finally, in the Visual Similarity and the Art Style tasks, the attention maps for the latent tokens of the correct answers are much more similar to attention maps of the reference images. For these tasks, it may be hard for humans to define explicit representations, but our approach allows latent tokens to learn these useful representations implicitly.

References

[1] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[2] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Trans. Graph.*, 33(4), 2014. 6

[3] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 91–104, 2025. 1

[4] Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. In *Advances in Neural Information Processing Systems*, pages 50742–50768, 2023. 6

[5] Xingyu Fu, Ben Zhou, Ishaan Preetam Chandratreya, Carl Vondrick, and Dan Roth. There is a time and place for reasoning beyond the image, 2022. 2

[6] Zihang Lai, Senthil Purushwalkam, and Abhinav Gupta. The functional correspondence problem. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15772–15781, 2021. 5

[7] Peiyuan Liao, Xiuyu Li, Xihui Liu, and Kurt Keutzer. The artbench dataset: Benchmarking generative models with artworks, 2022. 4

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. 1, 2

[9] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Spair-71k: A large-scale benchmark for semantic correspondence, 2019. 4

[10] Lukas Murmann, Michael Gharbi, Miika Aittala, and Fredo Durand. A dataset of multi-illumination images in the wild.

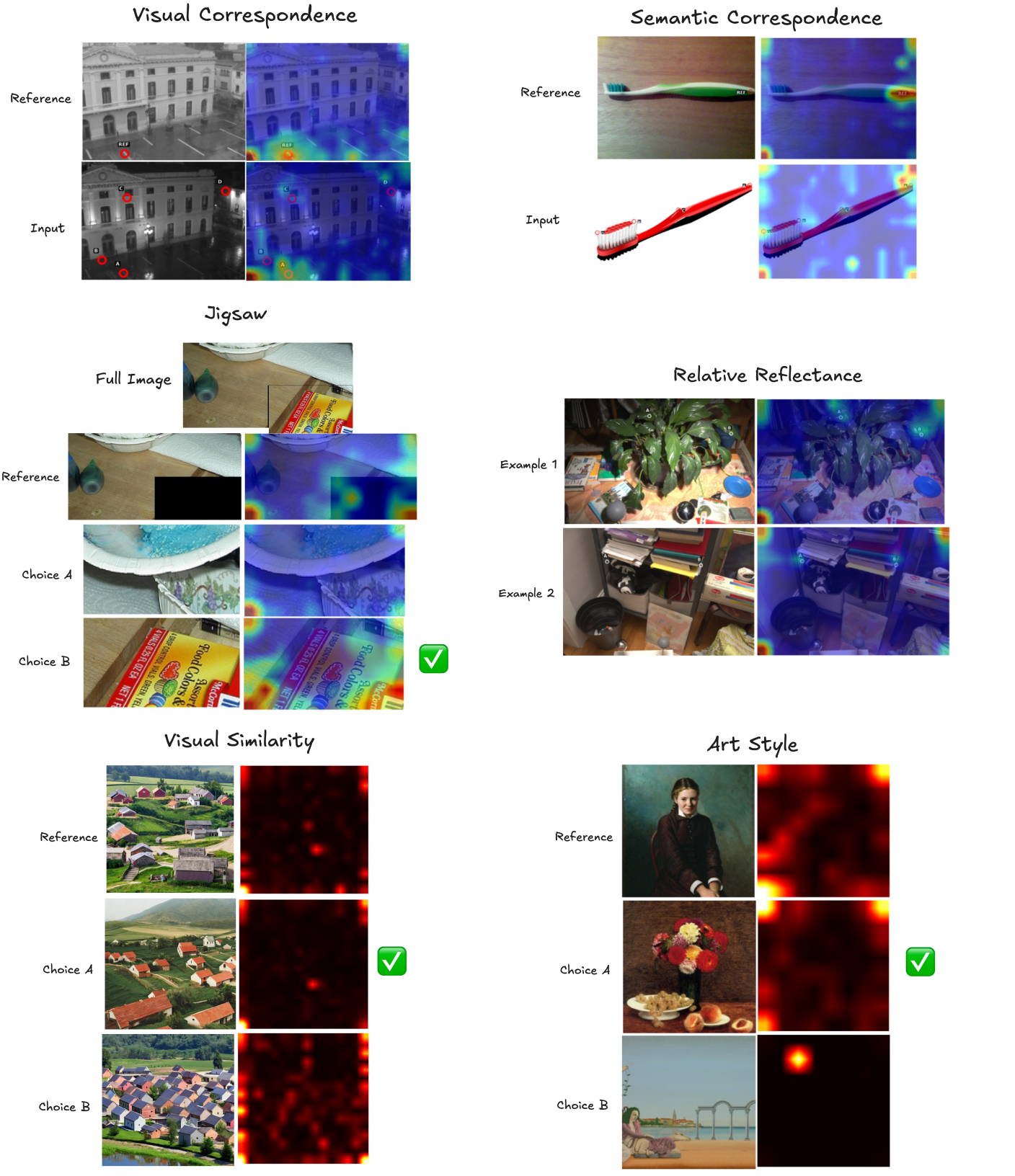


Figure 2. Additional visualizations of latent token to image attention.

In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 6

[11] 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. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 1

[12] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 1