Perception Tokens Enhance Visual Reasoning in Multimodal Language Models

Supplementary Material

6. Ablation study

In this section, we analyze the impact of various design choices and data configurations on the performance of our proposed method. We focus on three aspects: (1) the impact of including or excluding specific steps in the Chain of Thought (CoT) reasoning process for the 3D task of relative depth estimation, (2) the use of standard text tokens versus new perception pixel tokens for the 2D task of object counting, and (3) the effect of incorporating a perception token reconstruction loss during fine-tuning.

6.1. Chain of thought steps

For our 3D task of relative depth estimation, the Chain of Thought (CoT) questions in the fine-tuning data include two steps: (1) identifying the coordinates or locations of the points marked in the image, and (2) generating the depth map and determining which point is closer to the camera based on pixel values in the depth map. This study evaluates the impact of including or excluding these steps in the question prompts during fine-tuning.

We experiment with three variations of fine-tuning data configurations:

1. Direct Labeling Baseline: The model is fine-tuned solely on direct labeling data, where the question prompts directly ask which point is closer to the camera and provide the label as the answer. These prompts do not include either step (1) or step (2), see baselines section.
2. Step (2) Only: This model is fine-tuned with prompts that exclude step (1) (point location identification) but include step (2), asking the model to answer based on the depth map alone.
3. Aurora: Our proposed Aurora model uses prompts that include both steps (1) and (2), explicitly guiding the model through point location identification before generating the depth map.

All models are evaluated on the harder BLINK datasets we introduced. As shown in Tab. 4, the results demonstrate that having both steps in the prompts provides the most significant performance improvement. This suggests that guiding the model through a multi-step reasoning process in the prompts enables it to better capture spatial relationships and achieve more accurate depth estimations.

6.2. Text tokens vs. Perception tokens

In this ablation study, we evaluate the impact of using perception tokens compared to standard text tokens for the object counting subtask. Perception tokens are represented in the format `PIXEL_X`, where `X` is a number between 0 and 336, indicating pixel locations for object bounding boxes. For comparison, we replace these perception tokens with regular text tokens in the fine-tuning data, such that `PIXEL_100` is replaced with `100`, and so on.

As shown in Tab. 5, models utilizing perception tokens achieve higher performance across all three counting benchmarks: BLINK [11], SEED-Bench [23], and CV-Bench [40]. This demonstrates the effectiveness of perception tokens in explicitly encoding spatial information for improved counting accuracy.

6.3. Perception token reconstruction loss

The aim of this ablation study is to assess whether adding the perception token reconstruction loss, despite its increased computational cost, significantly improves model performance. Incorporating this loss requires adding the decoder for the specific task, which increases computation time and resource requirements. Not using it makes the system lighter and faster by just using the token classification loss. Therefore, we evaluate whether the performance gains justify the additional overhead.

To this end, we fine-tune two models based on LLaVA 1.5 13B [27] using a dataset of 20,000 samples only for depth map generation. Each sample includes a prompt such as *"What is the depth map for the image?"* and a response containing sequences of depth tokens. Both models are fine-tuned for 10 epochs: one with the reconstruction loss and one without it (both with cross entropy loss).

The reconstruction loss is computed as the mean squared error (MSE) between the ground truth depth map, which is the output of the VQVAE decoder when provided with the ground truth depth tokens, and the predicted depth map, which is generated by decoding the depth tokens produced by the LLM. A soft merging technique is used in reconstruction, where a "soft token" is created by averaging the embeddings of all potential tokens, weighted by their prediction probabilities from the LLM.

The models are evaluated on two datasets: (1) 124 images from the relative depth subtask of BLINK [11], and (2) 1000 random images from the Visual Genome dataset [21], for which depth maps were generated using Depth Anything [49]. The evaluation metric is the mean squared error (MSE) between the ground truth decoded depth maps and the depth maps reconstructed from the model's output tokens.

As shown in Tab. 6, the results indicate that incorporating the reconstruction loss does not significantly improve model performance. Fig. 5 further illustrates qualitative results, highlighting the visual differences in the predicted

Model	CoT steps: Coordinates	CoT steps: Depth	HardBLINK 3 Points	HardBLINK 4 Points	HardBLINK 5 Points
Direct Labeling Baseline	✗	✗	58.9	52.4	41.1
Step (2) only	✗	✓	56.4	56.4	50
AURORA (Ours)	✓	✓	**66.9**	**60.5**	**54.8**

Table 4. Performance comparison of models trained with different Chain of Thought question prompt variations for relative depth estimation on the harder BLINK datasets. Models with both steps in the prompts (AURORA) achieve the best performance.

Model	Token Type	CV-Bench Counting	SEED-Bench Counting	BLINK Counting
AURORA	Standard	52.2	50.6	38.3
AURORA	Perception	**56.0**	**54.6**	**45.8**

Table 5. Comparison of model performance using perception tokens and standard tokens for the object counting task across three benchmarks: BLINK, SEED-Bench, and CV-Bench. Perception tokens consistently improve accuracy.

depth maps with and without the reconstruction loss. While the reconstruction loss enforces consistency between the generated and ground truth depth tokens, its overall contribution is marginal in this setup. This study suggests that omitting the reconstruction loss may be a more efficient choice, especially when computational cost is a concern. Future work could explore its impact in larger datasets or more complex tasks to better understand its potential benefits.

Model	Recons Loss	Mean Squared Error↓ BLINK	Mean Squared Error↓ Visual Genome
LLaVA 1.5	✓	0.092	**0.074**
LLaVA 1.5	✗	**0.087**	0.076

Table 6. MSE evaluation of models with and without reconstruction loss on subsets BLINK and Visual Genome datasets.

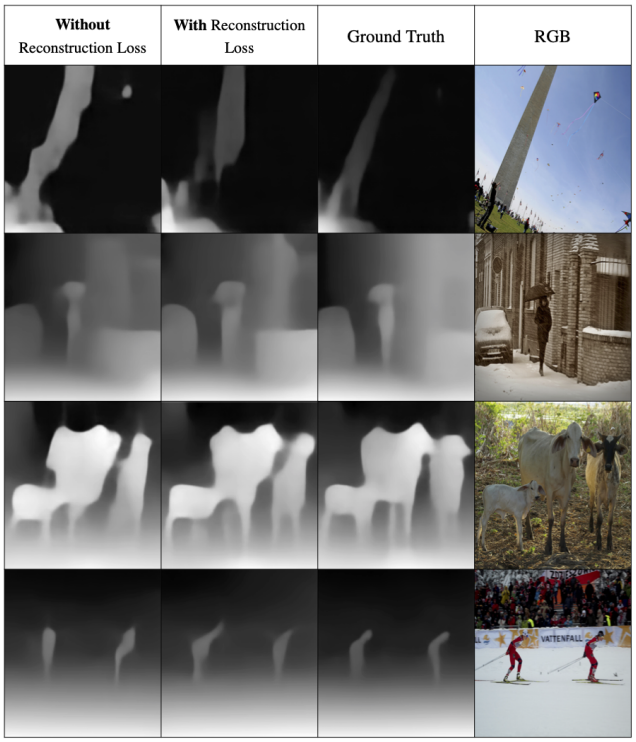

Figure 5. Qualitative comparison of predicted depth maps with and without reconstruction loss.

7. Cross-task generalization

To assess the generalizability of AURORA trained on depth generation and chain-of-thought (CoT) data for the relative depth task, we evaluate it on a different depth-related task. Specifically, we use the Depth subtask from CV-Bench [40], which involves identifying which of two objects, highlighted with red and blue bounding boxes, is closer to the camera. Similar to the BLINK evaluations for relative depth, we remove options from question prompts in these evaluations too.

As shown in Tab. 7, our model outperforms both the base LLaVA 1.5 13B model and the fine-tuning baseline, demonstrating its generalization capabilities across depth-related tasks.

8. Cross-model generalization

We applied AURORA to train LLaVA-OneVision 7B [25], the latest model in the LLaVA family. As shown in Tab. 8,

Model	CV-Bench Depth
LLaVA 1.5 13B	62.2
Fine-tuned LLaVA	60.0
AURORA (Ours)	**64.8**

Table 7. Performance comparison on the CV-Bench Depth subtask, highlighting our model's generalization ability.

the results align with those observed for LLaVA 1.5, highlighting the robustness of our approach across different model variants.

Model	BLINK 2 Pts	BLINK 3 Pts	BLINK 4 Pts	BLINK 5 Pts	Average
OneVision 7B	66.9	43.5	43.5	38.7	48.1
Fine-tuned OneVision 7B	**76.6**	59.6	58.1	54.1	62.1
OneVision 7B + AURORA	75.8	**62.1**	**62.1**	**55.6**	**63.9**

Table 8. Training LLaVA-OneVision with Aurora.

9. Implementation details

Computation resources. We train Aurora models on single-node machines equipped with 8 A40 GPUs. Each training run completes in less than 10 hours.

Model architecture and token expansion. Our approach builds on the LLaVA 1.5 13B model [28], a pre-trained multimodal language model. To support depth-related tasks, we expand the tokenizer by introducing 130 tokens for depth maps and 336 tokens for bounding box coordinates, increasing the vocabulary size beyond the original 32,000 tokens. These additions require modifications to the token embedding layer (embed_tokens) and the language model head (lm_head) to accommodate the new tokens.

Fine-tuning approach. We apply LoRA [16] to the language model for efficient fine-tuning. The vision backbone is kept frozen while the embed_tokens and lm_head layers are fully trained. This strategy enables the model to integrate depth and bounding box information without overwriting its pre-trained knowledge.

We fine-tune the model for 10 epochs, using the same LoRA parameters and learning rates as LLaVA. Finetunning follows a cross-entropy loss for next-token prediction, treating the new tokens identically to the original vocabulary.

Inference and decoding. During inference, we use a temperature of 0 for deterministic generation and employ constrained decoding techniques [3, 7, 12, 43]. For depth map generation, the model outputs exactly 100 depth tokens between `DEPTH_START` and `DEPTH_END`, ensuring consistent and structured results.

Curriculum learning for reasoning. To enhance the model's reasoning capabilities, we employ progressive chain-of-thought (CoT) for curriculum learning. Fine-tuning starts with atomic tasks, such as depth map and bounding box predictions, and gradually incorporates multi-tasking data, including CoT reasoning and direct labeling tasks. For instance, in the depth-related task, we use 20,000 samples for depth generation and 1,000 multi-tasking samples (comprising 500 unique images with sequential CoT and direct labeling questions).

In the first epoch, the model is fine-tuned exclusively on 20,000 depth generation samples. Starting from the second epoch, we introduce multi-tasking data by mixing 18,000 random depth generation samples with 2,000 multi-tasking samples (the 1,000 multi-tasking samples repeated twice). This ratio is progressively adjusted in subsequent epochs, culminating in the 10th epoch, where the model is exposed to 2,000 depth generation samples and 18,000 multi-tasking samples. This staged approach ensures a smooth transition from basic tasks to complex reasoning, effectively reinforcing the model's ability to handle multi-step reasoning challenges.

Fine-tuning data. As discussed in the Methods section, each task is supported by three sub-datasets. For the depth task, these include (1) depth generation data, (2) Chain-of-Thought (CoT) reasoning data, and (3) direct labeling data. Similarly, for the counting task, the sub-datasets consist of (1) bounding box prediction data, (2) CoT reasoning data, and (3) direct labeling data.

Fig. 6 and Fig. 7 present representative samples from each sub-dataset for the depth and counting tasks, respectively.

Depth Generation Data

What is the depth map for the image?

<DEPTH_START><DEPTH_67><DEPTH_67><DEPTH_70>....<DEPTH_3><DEPTH_30><DEPTH_END>

Chain of Thought Data

Multiple points are circled on the image, labeled by letters beside each circle. Which point is the closest to the camera?\nTo answer this question, let's think through it step by step, and we know the image is 336 x 336. First, what are the coordinates of points in the image? Second, what is the depth map for the image? Which point has a higher pixel value on the depth map? Remember, higher values indicate that the point is closer to the camera.

Point A is located at (x = 244 y = 12), Point B is located at (x = 84 y = 183). The depth map for the image is <DEPTH_START><DEPTH_65><DEPTH_20>...<DEPTH_32><DEPTH_END>. Since point B has a higher pixel value on the depth map, the answer is that point B is closer to the camera.

Direct Labeling Data

Multiple points are circled on the image, labeled by letters beside each circle. Which point is the closest to the camera?

(B)

Figure 6. Examples of sub-datasets for the depth task: (1) depth generation, (2) Chain-of-Thought reasoning, and (3) direct labeling.

Bounding Box Data

Please locate all of the boats in this image.

Here are the bounding boxes for the boats:\nBounding box 1: (<PIXEL_0>, <PIXEL_0>, <PIXEL_286>, <PIXEL_242>)\nBounding box 2: (<PIXEL_0>,<PIXEL_41>,<PIXEL_34>,<PIXEL_106>)\nBounding box 3: (<PIXEL_263>,<PIXEL_61>,<PIXEL_335>,<PIXEL_128>)

Chain of Thought Data

How many hats are in the image?.\n To solve this, let's carefully analyze the image. First, identify the bounding boxes associated with the hats. After that, count how many distinct bounding boxes correspond to hats.

Based on the analysis, here are the bounding boxes for the hats:\nBounding box 1: (<PIXEL_193>,<PIXEL_49>,<PIXEL_232>,<PIXEL_80>)\nSince there is 1 distinct bounding box, we conclude there is 1 hat.

Direct Labeling Data

How many hats are in the image?

1

Figure 7. Examples of sub-datasets for the counting task: (1) bounding box prediction, (2) Chain-of-Thought reasoning, and (3) direct labeling.