SegCompass: Exploring Interpretable Alignment with Sparse Autoencoders for Enhanced Reasoning Segmentation

Supplementary Material

A. Implementation Details

A.1. Pipeline Implementation

Parallelism and Optimization. Our implementation leverages the VERL [43] framework, adapted for multi-modal GRPO [42] training. To manage memory efficiency, we utilize Fully Sharded Data Parallel (FSDP) [61] to partition the MLLM policy parameters across devices. The lightweight segmentation modules, comprising the query head, Q–V attention [47], and mask decoder, are kept unsharded to avoid unnecessary communication overhead. For inference acceleration, the vLLM [20] backend employs tensor parallelism across attention heads. Additionally, image features are precomputed offline, removing the frozen vision backbone from the active training graph.

Distributed Worker Architecture. To reduce computational overhead, we precompute image features offline, thereby excluding the frozen vision backbone from the training loop. Our framework orchestrates three distinct types of FSDP workers: (i) The **actor** contains all trainable modules (including the MLLM and segmentation components) and is responsible for gradient computation and parameter updates. (ii) The **rollout worker** executes the MLLM in inference mode, processing image and text inputs to generate responses trajectories via next-token prediction. (iii) The frozen **reference worker** maintains a copy of the reference policy model to compute the KL divergence term for $\mathcal{L}_{\text{GRPO}}$. Additionally, it utilizes the segmentation modules to decode masks required for calculating mask-based reward scores and group advantages.

Training Workflow. For each annotated sample, the rollout worker first generates a group of G responses using the current policy model via next-token predict, caching both the output tokens and their log probabilities. Subsequently, the frozen reference worker performs a forward pass (without gradients) on the same inputs. This step computes the reference log probabilities for the sampled responses and decode masks for the mask-based reward. For each response and its mask signal, we compute a scalar reward and convert rewards to group advantages. Finally, the actor worker performs a forward pass to obtain the current policy log probabilities for the sampled responses and the predicted segmentation mask. The total loss is composed of the GRPO objective (derived from actor log probabilities, cached rollout log probabilities, reference log probabilities, and advantages) and the segmentation objective (derived from the predicted and ground truth mask. The two objectives are summed and jointly optimized in a single backward pass to update all trainable parameters.

Sparse Autoencoder (SAE). We adopt the architecture of SAE from the methodology proposed in SAE-V [34], which comprises an encoder and a decoder, as shown in Figure 7. The input feature of SAE is the hidden layer state of MLLM. We maintain consistent experimental configurations across LLaVA-1.5-7B [27], LLaVA-1.5-13B and Qwen2.5-VL-7B [2], with the exception of the specific Transformer decoder layer selected for feature extraction. Specifically, we extract features from the 16-th layer for both LLaVA-1.5-7B (32 layers total) and LLaVA-1.5-13B (40 layers total), and the 13-th layer for Qwen2.5-VL-7B (28 layers total). Following the standard practice in SAE, we target these intermediate layers to strike a balance between modality fusion and semantic abstraction. Middle layers are empirically found to encode rich mixed vision-language information, whereas deeper layers tend to be dominated by text-generation patterns (i.e., next-token prediction dynamics). SAE training is conducted before the reasoning segmentation task, during which its parameters are frozen.

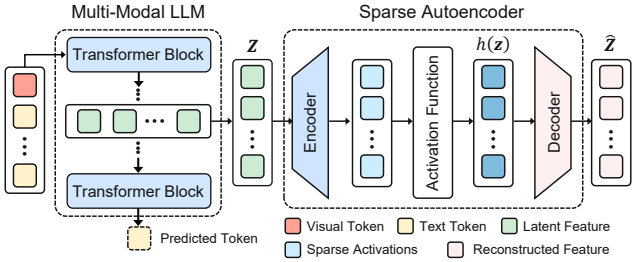

Figure 7. The architecture of Sparse Autoencoder (SAE).

A.2. Training Configuration

Data Preprocessing. We train on the RefCOCO [17, 35] series and gRefCOCO [25] datasets. The maximum prompt length and response length are set to 1,400 tokens and 2,000 tokens, respectively. For Qwen2.5-VL-7B, the input image pixel count range from 3,136 to 705,600. Image exceeding this range are resized accordingly. We initialize the vision branch using SAM ViT-H.

Model Optimization. We optimize the policy model parameter using AdamW with a weight decay of 0.01 and $(\beta_1, \beta_2) = (0.9, 0.999)$. We employ differential learning rates across model components: the base learning rate for the MLLM backbone is set to 2×10^{-6}, with multipliers of $25\times$ for the query codebook, $80\times$ for the slot mapper,

and $10\times$ for the mask decoder. Gradient clipping is applied with a maximum norm of 1.0. The schedule is one cycle with a final division factor of 6.7 and no warmup. Training spans 24,252 steps with gradient checkpointing enabled. The global batch size is 128, comprising 16 image-text pairs with a group size of 8 for GRPO. We utilize bfloat16 precision for model parameters and fp32 for reductions and buffers. For SAE training, we set the learning rate, batch size, and total training epochs to 5×10^{-5}, 64, 10, respectively. Additionally, We employ a warmup phrase during the initial stage of training.

GRPO Settings. We configure GRPO with a group size of 8, a clip ratio of 0.2, and a fixed KL penalty coefficient 0.2. For the policy model, the per-device micro-batch is set to 2 for updates and 8 for experience collection. The policy model is trained using Fully Shared Data Parallel (FSDP) with optimizer state offloading. For the rollout service, we employ a vLLM backend with a tensor parallelism size of 4 for sampling. Inference is executed in bfloat16, supporting a maximum of 64 concurrent sequences and a batch size of 17,408 batch tokens. Chunked prefill is enabled and each sample consists of a single image.

B. Group Relative Policy Optimization

Recently, Reinforcement Learning (RL) has been applied to enhance the reasoning capability of LLMs. By formulating reasoning token generation as a Markov Decision Process, models are trained to maximize the expected return of reasoning paths, guiding optimization toward more structured and coherent reasoning trajectories. Proximal Policy Optimization (PPO) [41] and its variants, Group Relative Policy Optimization (GRPO) [12, 42], are among the most widely adopted algorithms for this purpose.

B.1. PPO Preliminaries

PPO is an actor-critic RL algorithm that optimizes LLMs by maximizing the following surrogate objective:

$$\mathcal{L}_{\text{PPO}} = \mathbb{E}\left[q \sim P(Q), o \sim \pi_{\boldsymbol{\theta}_{\text{old}}}(O|q)\right] \frac{1}{|o|}\sum_{t=1}^{|o|}\min\left[\frac{\pi_{\boldsymbol{\theta}}(o_t|q,\boldsymbol{o}_{<t})}{\pi_{\boldsymbol{\theta}_{\text{old}}}(o_t|q,\boldsymbol{o}_{<t})}A_t, \text{clip}\left(\frac{\pi_{\boldsymbol{\theta}}(o_t|q,\boldsymbol{o}_{<t})}{\pi_{\boldsymbol{\theta}_{\text{old}}}(o_t|q,\boldsymbol{o}_{<t})}, 1-\varepsilon, 1+\varepsilon\right)A_t\right] \tag{8}$$

where $\pi_{\boldsymbol{\theta}}$ and $\pi_{\boldsymbol{\theta}_{old}}$ denote the current and old policy models, respectively. Here, q and o represents questions and outputs sampled from the dataset and the old policy $\pi_{\boldsymbol{\theta}_{old}}$. The ε is a clipping-related hyper-parameter introduced in PPO for stabilizing training. The advantage A_t is computed using Generalized Advantage Estimation (GAE), based on the reward $\{r_{\geq t}\}$ and a learned value function V_ψ. The value function V_ψ is typically approximated by a learnable critic model of the same scale as the policy model. Furthermore, a per-token KL penalty from a reference model is added to the reward at each step to mitigate over-optimization of the reward model, which denoted as:

$$r_t = r_\varphi(q, \boldsymbol{o}_{\leq t}) - \beta \log \frac{\pi_{\boldsymbol{\theta}}(o_t|q,\boldsymbol{o}_{<t})}{\pi_{\text{ref}}(o_t|q,\boldsymbol{o}_{<t})} \tag{9}$$

where r_φ represents the reward model, π_{ref} denotes the reference model (typically the initial policy model), and β is the coefficient of the KL penalty. In PPO, both the policy and critic models must be trained simultaneously, which imposes significant computational demands when model parameters or token counts are large. GRPO has been proposed to reduce the resource consumption associated with PPO while maintaining training stability.

B.2. GRPO Objective

GRPO eliminates the need for a separate value function approximator as in PPO. Instead, it estimates the advantage using the average reward of multiple outputs sampled for the same question. Specifically, for each question q, GRPO samples a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\boldsymbol{\theta}_{\text{old}}}$ and optimizes the policy model by maximizing the following objective:

$$\mathcal{L}_{\text{GRPO}} = \mathbb{E}\left[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\boldsymbol{\theta}_{\text{old}}}(O|q)\right] \frac{1}{G}\sum_{i=1}^{G}\min\left[\frac{\pi_{\boldsymbol{\theta}}(o_i|q)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(o_i|q)}\hat{A}_i, \text{clip}\left(\frac{\pi_{\boldsymbol{\theta}}(o_i|q)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_i\right] - \beta\mathbb{D}_{\text{KL}}\left[\pi_{\boldsymbol{\theta}}\|\pi_{\text{ref}}\right] \tag{10}$$

where ε and β are hyper-parameters, and $\hat{A}_i$ represents the advantage computed using the relative rewards of outputs within each group. For each question q, a group of outputs $\{o_1, o_2, \cdots, o_G\}$ is sampled from the old policy model $\pi_{\boldsymbol{\theta}_{old}}$. A reward model assigns scores to these outputs, yielding G corresponding rewards $\{r_1, r_2, \cdots, r_G\}$. The advantages $\hat{A}_i$ of the tokens in the output is defined as the normalized reward: $\hat{A}_i = \tilde{r}_i = \frac{r_i - \text{mean}(\boldsymbol{r})}{\text{std}(\boldsymbol{r})}$. Additionally, GRPO incorporates the KL divergence between the trained policy and the reference policy directly to the loss function, avoiding complicating the calculation of $\hat{A}_i$. The KL divergence is estimated by the following unbiased estimator:

$$\mathbb{D}_{\text{KL}}\left[,\pi_{\boldsymbol{\theta}} \,\|\, \pi_{\text{ref}}\right] = \frac{\pi_{\text{ref}}(o_i|q)}{\pi_{\boldsymbol{\theta}}(o_i|q)} - \log\frac{\pi_{\text{ref}}(o_i|q)}{\pi_{\boldsymbol{\theta}}(o_i|q)} - 1 \tag{11}$$

B.3. Reward Function Design

Format Score. We score outputs in $\{0.0, 0.9, 1.0\}$. A response is valid only if it contains exactly one `<think> ... </think>` block with non-empty content and exactly one special token for reference position; the special token must appear after `</think>`. Any violation yields 0.0. For valid outputs, the base is 1.0 and is downgraded to 0.9 if the `<think>` content is overly long (more than 2048 characters), or if there is any non-whitespace text before `<think>` or after the special token. Thus the five canonical cases are: invalid (0.0); valid and clean (1.0); valid but

long <think> (0.9); valid but extra text before <think> (0.9); valid but extra text after the special token (0.9).

Single-object. We use a scalar reward that combines a format score with a mask score. For the single-object setting, no assignment is required: the mask score is computed directly from the agreement between the predicted probability mask and the ground-truth mask using a soft IoU overlap in $[0, 1]$. This measures how well the prediction localizes the referent without introducing any auxiliary terms or matching.

Multi-object. For multiple instances, the model outputs K slot masks with confidences. We retain predictions above a fixed confidence threshold and compare them to the G ground truth masks via pairwise soft overlaps. A Hungarian assignment is then solved to maximize total overlap, and the mask score is the mean overlap over the matched pairs. If one side is empty and the other is not, the score is set to 0; if both are empty, it is set to 1.0. The final reward still combines the format score and this mask score, encouraging valid reasoning traces and precise one-to-one coverage in the multi-object case.

C. Additional Analyses

C.1. Correlation Analyses

To verify the effectiveness of the designed concept representation and concentration token in guiding the segmentation process, we examine two key correlations: (1) the correlation between the SAE feature and segmentation during training, and (2) the correlation between the heatmap and segmentation results during inference. In Figure 8 (left),

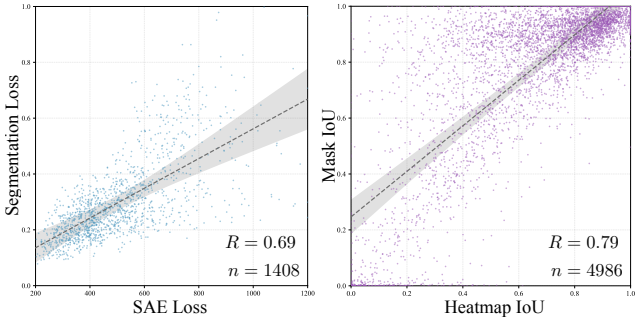

Figure 8. **Correlation analysis.** Left: SAE features vs. segmentation during training. Right: heatmap vs. segmentation during inference. Results for Qwen2.5-VL-7B on gRefCOCO. Ordinary least squares (OLS) regression lines and mean confidence bands are overlaid.

each data point denotes a training batch. The x-axis displays the SAE reconstruction error (MSE) while the y-axis represents the training Dice loss of the predicted masks. Note that the SAE is pretrained and frozen within the segmentation pipeline. Consequently, MSE measures SAE reconstruction quality, whereas Dice loss reflects mask accuracy. The scatter plot reveals a distinct trend, indicating a strong association between SAE reconstruction quality and segmentation performance. In Figure 8 (right), each point corresponds to a test instance. The x-axis represents the cIoU between the heatmap $(\mathcal{H}_k)_{k=1}^{K_s}$ and the ground truth $\boldsymbol{M}_{\text{gt}}$, while the y-axis shows the cIoU between the predicted mask $\hat{\boldsymbol{M}}$ and $\boldsymbol{M}_{\text{gt}}$. We observe a significant positive relationship. These finds demonstrate that both SAE and heatmap quality are positively associated with final mask performance, providing evidence that the interpretable alignment contributes predictive signal.

C.2. Additional Visualizations

As shown in Figure 9, we visualize cases from the ReasonSeg test split. Tokens in regions that match the instruction's semantics show higher responses in the heatmap, and the map peaks on the referred instance, providing a gook prior that helps the decoder recover clean boundaries. In addition, SAE activations over image tokens are tightly linked to the question: high-activation tokens concentrate on foreground areas and correlate with the heatmap intensity. This behavior is consistent with the quantitative trend in Figure 8, where SAE activations for question and instance tokens exceed those for background and instruct the heatmap signal.

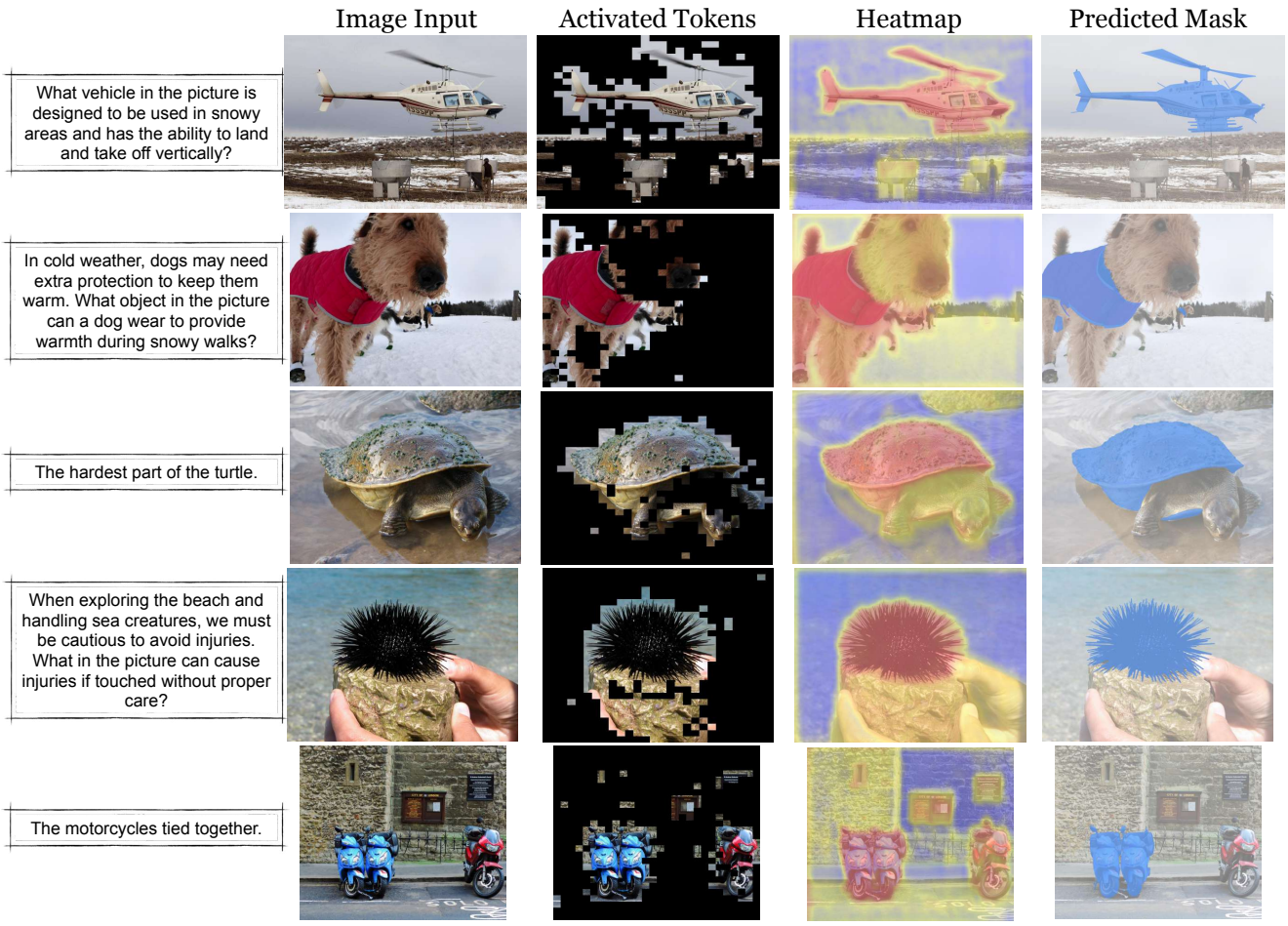

Figure 9. **Additional Visualizations.** From left to right: instruction, image input, activated tokens, heatmap, and predicted mask.