

Search-TTA: A Multimodal Test-Time Adaptation Framework for Visual Search in the Wild - Supplementary Material

Anonymous Author(s)

Affiliation

Address

email

1 A Additional Dataset Details



Figure A.1: Examples of satellite images in the full 437k dataset (each with different taxonomies), used for CLIP fine-tuning.



Figure A.2: Examples of satellite images in the 80k training and 4k validation dataset (each with same taxonomies), used for AVS validation and to generate score maps for VLM fine-tuning.

2 In this section, we provide more details about our Autonomous Visual Search (AVS) dataset com-
3 position and generation process, in addition to the information provided in Sec 4. Our taxonomic
4 location dataset contains **437k** non-overlapping satellite images, each with multiple coordinates of
5 different taxonomic targets (Fig. A.1). More specifically, this dataset contains **2.7M** satellite-image
6 to taxonomy pairs [1], which can be used directly to fine-tune our satellite image encoder. After fur-
7 ther processing, we eventually obtain **4k** validation images that each contain between 3-20 counts of
8 the same taxonomy per image. We convert the remaining images with 3-20 counts of taxonomic tar-
9 gets into **80k** score maps, which are used to fine-tune our VLM baselines and to train our RL policy.
10 Examples of these same-taxonomy satellite images can be seen in Fig. A.2. Each of these Sentinel-2
11 level 2A satellite images [2] has a resolution of 10m and covers approximately $2.56\text{km} \times 2.56\text{km}$ of
12 land mass.

13 A.1 Dataset Composition

14 We visualize the spatial distribution of our dataset in Fig. A.3, where the color intensity reflects the
 15 taxonomy counts in each cell (1° latitude \times 1° longitude). Despite filtering our dataset to cater to
 16 our AVS task, our **80k** and **4k** dataset subsets still appear visually representative of the original **473k**
 17 dataset distribution.

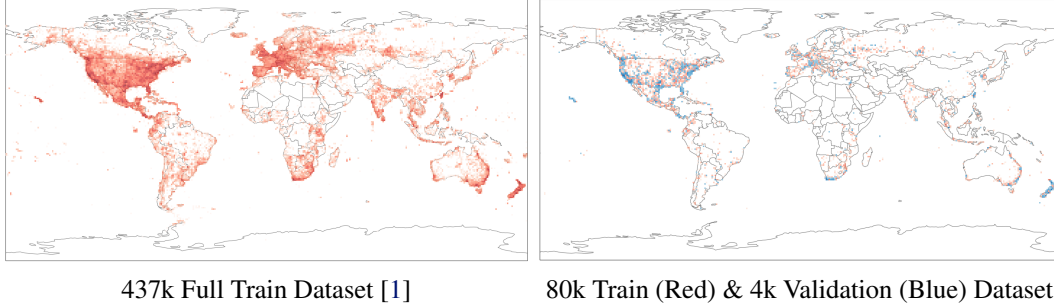


Figure A.3: Geographic coverage of datasets used in training and validation.

18 A.2 Taxonomic Statistics

19 We visualize the distribution of the taxonomy counts for the training and validation datasets in
 20 Fig. A.4. We note a natural decreasing trend in the number of same-taxonomy targets within the
 21 $2.56\text{km} \times 2.56\text{km}$ of land mass covered by each image. Overall, the average number of taxonomic
 22 counts per image for our **80k** training and **4k** validation datasets are 4.6 ± 2.6 and 3.1 ± 1.1 respec-
 23 tively. This results in a very sparse target distribution per image, which makes our AVS task more
 24 challenging given a budget constraint.

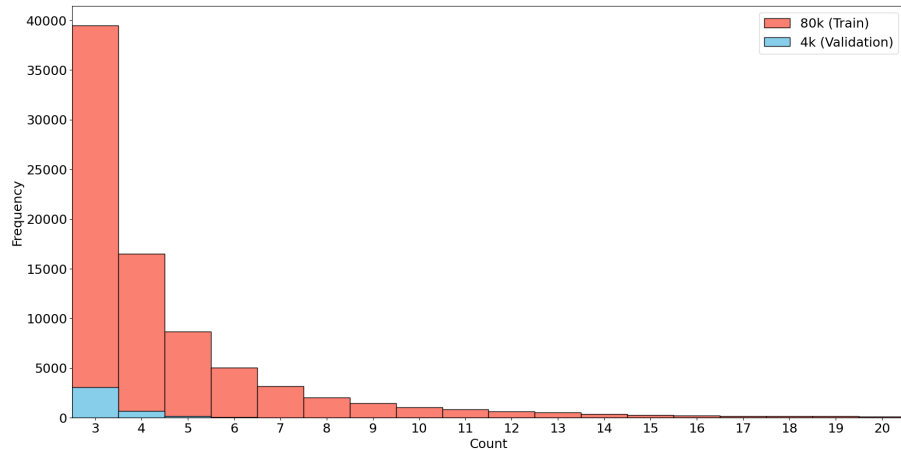


Figure A.4: Histogram counts for 80k Train (Red) & 4k Validation (Blue) Dataset

25 In addition, we visualized the break down of the taxonomy categories of both the **80k** training
 26 dataset (Table A.1) and the **4k** validation dataset (Table A.2). This distribution is similar to the
 27 original distribution from the iNaturalist 2021 challenge where this dataset originates from [3].

28 A.3 Score Map Generation

29 We generate our **80k** training score masks using a custom process because our AVS dataset only
 30 includes point locations, and conversion to segmentation masks with likelihood scores is non-trivial.
 31 This is done in two stages, using the same-taxonomy satellite images that are not used in the **4k**
 32 validation set. First, we use GSNet [4], an open-vocabulary semantic segmentation model, to obtain

Table A.1: Taxonomy Distribution (80k)

| Category | Count | Relative Percentage (%) |
|-------------------|--------|-------------------------|
| Plants | 166913 | 45.4 |
| Insects | 109247 | 29.7 |
| Birds | 42756 | 11.6 |
| Fungi | 9830 | 2.7 |
| Reptiles | 12828 | 3.5 |
| Mammals | 4109 | 1.1 |
| Ray-finned Fishes | 6094 | 1.7 |
| Amphibians | 8764 | 2.4 |
| Mollusks | 1674 | 0.5 |
| Arachnids | 3651 | 1.0 |
| Animalia | 1785 | 0.5 |

Table A.2: Taxonomy Distribution (4k)

| Category | Count | Relative Percentage (%) |
|-------------------|-------|-------------------------|
| Plants | 6798 | 51.0 |
| Insects | 2254 | 16.9 |
| Birds | 2338 | 17.5 |
| Fungi | 331 | 2.5 |
| Reptiles | 561 | 4.2 |
| Mammals | 125 | 0.9 |
| Ray-finned Fishes | 239 | 1.8 |
| Amphibians | 246 | 1.8 |
| Mollusks | 75 | 0.6 |
| Arachnids | 149 | 1.1 |
| Animalia | 218 | 1.6 |

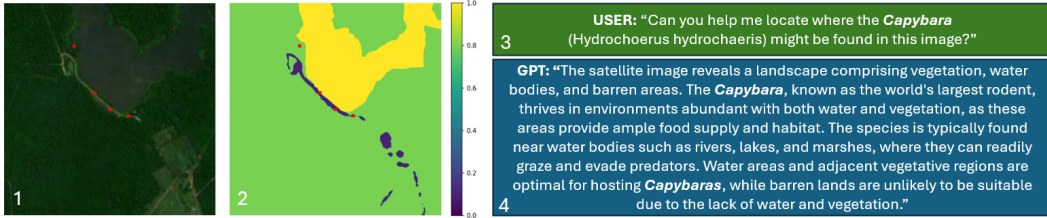


Figure A.5: (1) Satellite map where Capybaras can be found. (2) Score map of where Capybaras are likely to be found, used for fine-tuning VLM baselines and training RL policy. (3-4) Question and answer pair, used to fine-tune VLMs such as LISA [7].

label maps of the low-resolution Sentinel-2 images based on broad landmark names (i.e. *Urban*, *Water*, *Vegetation*, *Barren*). Since GSNet has been pretrained on a diverse set of satellite images with varying spatial resolution, we fine-tune it with the FLAIR semantic segmentation dataset [5] to enhance its segmentation abilities specifically for low-resolution images. However, despite the high-quality segmentation masks, mislabeling commonly occurs. Hence, we use GPT4o [6], augmented with human-labeled examples, to rectify the wrong labels. We then use GPT4o again to score the masks based on the likelihood of finding the specified taxonomies per image. It concurrently generates a conversation that explains the rationale for why certain landmarks are more suitable for the specified taxonomies, and less so for others. After filtering erroneous GPT4o generations, we are left with **80k** score maps for each taxonomic-image pair. A full example can be seen in Fig. A.5. The prompts used for relabeling and scoring by GPT4o is shown in Appendix E.1.

B Additional Search-TTA Details

In this section, we provide more details of the Search-TTA framework, to supplement the information in Section 5. We elaborate on how Search-TTA generates and periodically updates the probability distribution outputs that serve as a strong prior to the RL search planner.

B.1 Qualitative Analysis

We provide snapshots of AVS with the RL planner to provide a better understanding of how it works. In Fig. B.1, the RL planner begins with an initial prior generated by the satellite image CLIP encoder, which represents the probability distribution of where *Marmots* (*Mammalia Rodentia Sciuridae Marmota*) can be found. The Search-TTA significantly improves the probability distribution outputs in the associated region where the first target is found. This steers the RL planner to exploit the high probability region to locate all of the targets within 119 steps. Without TTA, the probability distribution is static and the RL planner takes 230 steps before locating all marmots.

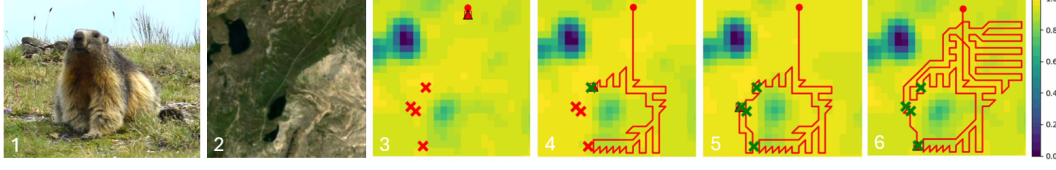


Figure B.1: (1) Ground image of a *Marmot*. (2) Satellite image where marmots can be found. (3) Initial probability output. (4) TTA significantly increases probability values in region where the first Marmot is found. (5) Improved priors leads to efficient search. (6) Inefficient search without TTA.

B.2 Algorithm

We provide a pseudo-code to illustrate the entire pipeline of the Search-TTA framework using the RL planner (algorithm 1).

Algorithm 1: The Search-TTA Framework (with RL Search Policy)

Input: Satellite-image encoder f_θ ; Ground-image encoder h_ψ ; Search policy π_ϕ ,
Satellite image s , Ground image g , Budget \mathcal{B}
Initialize: $\theta \leftarrow \theta_{\text{base}}$, step $t = 0$, measurements $O = [0, \dots, 0]$, grid map $\mathcal{M} = (n \times n)$, $\alpha_{\text{pos}, i} = 4$
// --- Generate Probability Distribution ---
 $z \leftarrow f_\theta(s)$, $y \leftarrow h_\psi(g)$;
 $P \leftarrow \text{cosineSim}(z, y)$;
 $r \leftarrow \text{kmeans}(z)$;
while $t \leq \mathcal{B}$ **do**
 // --- Generate Action ---
 $a_t \sim \pi_\phi(\cdot \mid o_t, P)$;
 $s_{t+1} \sim T(o_t, a_t)$;
 // --- Collect Measurements ---
 $d \leftarrow \text{Observe}(o_{t+1})$;
 $O \leftarrow \text{Update}(O, d)$;
 // --- Perform TTA ---
 for every k **steps do**
 $\theta \leftarrow \theta_{\text{base}}$;
 $\gamma \leftarrow \gamma_{\min} + (t/n^2) \cdot (\gamma_{\max} - \gamma_{\min})$;
 $\alpha_{\text{neg}, i} \leftarrow \min(\beta(O_r/L_r)^\gamma, 1)$;
 $L(\lambda) = \sum_{i=1}^n \alpha_{\text{pos}, i} \cdot \log \lambda(x_i) - \int_{A, i} \alpha_{\text{neg}, i} \cdot \lambda(x) dx$, where $P \approx \lambda$;
 $\theta \leftarrow \theta + \gamma \cdot \nabla_\theta L(\lambda)$;

B.3 Training details for Satellite Image Encoder

We fine-tune our satellite image encoder [8] with the hyperparameters in Table B.1. This was performed using two NVIDIA A6000 GPUs, which took 3 epochs (3.5 days) before convergence (lowest CLIP validation score). During fine-tuning, we keep our BioCLIP [9] model frozen.

B.4 Kmeans Clustering

We rely on Kmeans clustering of the satellite image encoder output to obtain clusters of embeddings that are deemed semantically similar by CLIP [10]. We determine the best k by taking the average of the silhouette score criterion [11] and the elbow criterion [12]. The silhouette score measures clustering quality by contrasting each satellite patch feature’s average distance to its own cluster with the closest alternative cluster. On the other hand, the elbow method charts the within-cluster sum-of-squares across candidate k values and pinpoints where additional clusters yield only marginal variance reduction. Combining both methods balances the silhouette score’s tendency to favor fewer clusters with the elbow method’s subjectivity based on a ‘knee’ point. In practice, we set the max averaged k to be 4 based on the approximate number of possible broad semantic landmarks, should the elbow method overestimates k too significantly.

Table B.1: Training Hyperparameters (Satellite Image Encoder)

| Hyperparameter | Value |
|-------------------------|-----------------------------|
| Batch Size | 32 |
| Learning Rate | 1e-4 |
| Learning Rate Schedule | min 1e-6 (Cosine Annealing) |
| Temperature (τ) | 0.07 |
| Optimizer | AdamW |
| Optimizer β | (0.9, 0.98) |
| Optimizer ϵ | 1e-6 |
| Accumulate Grad batches | 64 |
| Projection Dimension | 512 |
| Ground Image Encoder | ViT-B/16 (BioCLIP [9]) |
| Satellite Image Encoder | ViT-L/14@336px (CLIP [8]) |

Table B.2: Effect of SPPP Weighting Coefficient on Targets Found (%) \uparrow

| Parameters | | $B = 256$ | | | $B = 384$ | | |
|---------------|----------|-------------|-------------|-------------|-------------|-------------|-------------|
| β | γ | All | Bot. 5% | Bot. 1% | All | Bot. 5% | Bot. 1% |
| $\frac{1}{6}$ | 3 | 58.0 | 33.8 | 25.6 | 76.9 | 58.2 | 51.2 |
| $\frac{1}{6}$ | 4 | 58.0 | 32.4 | 24.6 | 76.5 | 59.5 | 54.4 |
| $\frac{1}{6}$ | 1 | 57.6 | 31.8 | 25.4 | 76.0 | 56.0 | 52.7 |
| $\frac{1}{6}$ | 0 | 57.2 | 29.8 | 19.4 | 75.8 | 54.4 | 44.2 |
| 1 | 2 | 57.1 | 30.2 | 24.2 | 75.3 | 51.8 | 46.8 |
| $\frac{1}{6}$ | 2 | 58.1 | 34.1 | 29.1 | 76.1 | 56.0 | 50.8 |

74 B.5 Varying SPPP-based Online Adaptation Hyperparameters

75 We perform grid search to determine the optimal hyperparameters for our negative weighting coef-
76 ficient $\alpha_{\text{neg},i} = \min(\beta(O_r/L_r)^\gamma, 1)$, where O_r is the number of patches observed in region r and
77 L_r is the number of patches in that region. β balances the relative weightage between positive and
78 negative measurements in the loss function, while γ scales the weightage of negative measurements
79 given the same amount of region explored. We summarize our results in Table B.2. When we re-
80 move the negative weighting coefficient ($\gamma = 0$), we observe one of the poorest performance (2nd
81 from the bottom). This is because all negative samples are weighted equally heavily even at the start
82 of AVS, causing premature collapsing of probability distribution modes. Hence, this highlights the
83 importance of our uncertainty weighting scheme. In addition, if we remove the relative weighting
84 factor ($\beta = 0$), we note the worse performance possibly due to over-penalizing negative measure-
85 ments. Lastly, we note that $\gamma = 3$ yields the best performance, which is on-par with LISA when
86 $B = 384$, unlike our reported values with $\gamma = 2$ (second best).

87 In order to achieve stable updates to the output probability distribution, we reset the satellite encoder
88 weights back to the base weights before running TTA updates. During TTA updates, we use the
89 Adam optimizer, and employ a learning rate schedule that increasing our learning rate from 1e-6
90 to 1e-5 depending on how much of the search space has been covered. This learning rate schedule
91 allows the model to learn more effectively when more measurements are collected [13].

92 C Additional Baseline Details

93 In this section, we provide additional information on how we set up our baselines for fair compari-
94 son, on top of the details provided in Section 6.

95 C.1 Planner Baselines

96 We compare Search-TTA with an Attention-based Reinforcement Learning (RL) planner [14] and a
97 greedy Information Surfing (IS) planner [15]. The RL planner is non-myopic in nature as it learns
98 dependencies at multiple spatial scales across the entire search domain. This allows agents to bal-

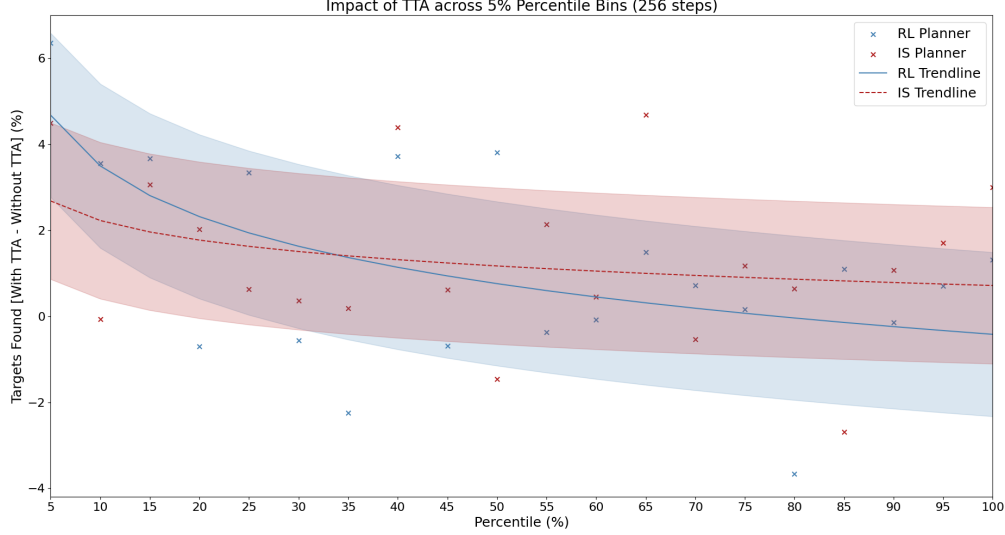


Figure C.1: Performance difference (due to TTA) for RL (blue) and IS (red) planners at 256 steps.

99 ance trade-offs between short-term exploitation and long-term exploration given the probability dis-
 100 tribution map. On the other hand, the IS planner drives agents in the direction of the derivative of
 101 the information map to maximize short-term gains. Such an approach tends to be greedy in nature
 102 and may suffer from overexploitation of local maxima. By design, the RL planner allows movement
 103 to all eight neighboring cells, while the IS planner is limited to the four cardinal directions. Note
 104 that we do not intend to compare the performance between RL and IS, but rather how test-time
 105 adaptation improves each of the planners independently.

106 **RL Policy Training:** We train our RL planner’s attention-based neural network using the soft actor-
 107 critic (SAC) algorithm [16], which learns a policy by maximizing return while maximizing entropy.

$$\pi^* = \operatorname{argmax}_{(\pi, a_t)} \mathbb{E} \left[\sum_{t=0}^T \gamma^t (r_t + \alpha \mathcal{H}(\pi(\cdot | o_t))) \right], \quad (\text{C.1})$$

108 where \mathcal{H} denotes entropy, π^* the optimal policy, γ the discount factor, and α the adaptive temper-
 109 ature term. We utilize a subset of the score maps of varying probability distributions from Sec A.3
 110 to pre-train our RL policy. Similar to [14, 17], we define the viewpoints ψ in the search domain \mathcal{M}
 111 as graph vertices, each connected via edges to its adjacent (≤ 8) neighbors. In addition to positional
 112 information, each of these nodes are augmented with the agent’s visitation history and scores from
 113 Search-TTA’s output probability distribution. This graph can then be used as the RL agent’s observa-
 114 tion to output a stochastic policy $\pi_\theta(a^t | o^t)$. Our reward function penalizes distance travelled while
 115 incentivizes agents to travel to high probability regions. We find that adding rewards for targets
 116 found causes the planner to generate inefficient routes, possibly due to the sparse target distributions
 117 that may provide confusing reward signals. We train our policy using an AMD Ryzen threadripper
 118 3970x and four NVIDIA A5000 GPUs, which took 20k episodes (80 hours) to converge.

119 **Bottom Percentile Comparison:** To determine Search-TTA’s effectiveness given poor CLIP pre-
 120 dictions, we break down the percentage of targets found into the bottom 5% and 1% percentiles.
 121 We measure the quality of CLIP predictions by taking the average scores of the pixels where the
 122 targets are located on the predicted score map, and deem a CLIP prediction to be poor if most tar-
 123 gets are located in the lowest-scoring regions. We then plot the performance gains (due to TTA) for
 124 both RL and IS planners for both 256 (Fig. C.1) and 384 steps (Fig. C.2), given bins of 5-percentile
 125 increments. Note that the logarithmic trendlines fit the datapoints well, indicating that the per-
 126 formance difference is most significant at the bottom percentiles. This illustrates how Search-TTA is
 127 particularly effective in the low percentile range.

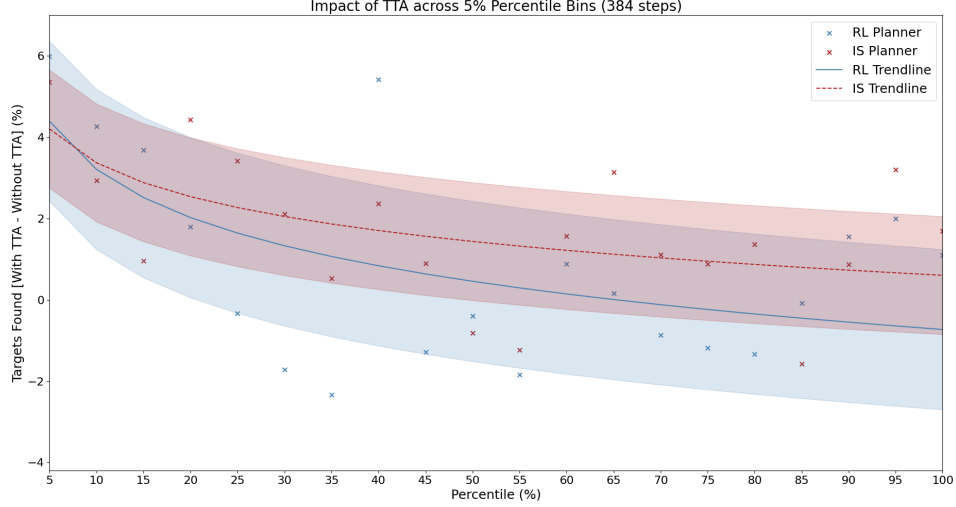


Figure C.2: Performance difference (due to TTA) for RL (blue) and IS (red) planners at 384 steps.

Table C.1: Scaling up Dataset Size for VLM (LISA [7]) Fine-tuning - 384 Steps

| Frameworks | Found (%) \uparrow | Explored (%) \uparrow | S_T (Start) \downarrow | S_T (Mid) \downarrow | S_T (Last) \downarrow |
|---------------------|----------------------|-------------------------|----------------------------|--------------------------|---------------------------|
| LISA (80k) | 76.9 | 66.1 | 105.0 | 123.0 | 263.3 |
| LISA (55k) | 76.8 | 66.1 | 106.2 | 124.1 | 263.5 |
| LISA (24k) | 75.9 | 66.2 | 111.0 | 128.2 | 266.0 |
| LISA (no fine-tune) | 72.6 | 64.8 | 108.8 | 126.1 | 258.5 |

128 C.2 VLM Baselines

129 We evaluate the effectiveness of Search-TTA’s CLIP vision backbone by replacing it with different
 130 state-of-the-art VLMs. Most of these VLMs comprise a reasoning module (e.g. LLaVA [18] or
 131 Qwen2-VL [19]) that processes image and text inputs to output reasoning embeddings. These em-
 132 beddings are then passed into segmentation modules such as SAM [20] to generate the appropriate
 133 masks. LISA [7] (we use LISA-7B) connects and fine-tunes LLaVA and SAM modules in an end-to-
 134 end manner. Unlike the original setup where LISA is trained on binary masks, we directly fine-tune
 135 LISA with the **80k** score maps (with text-based question-answer) from our custom training dataset.
 136 In addition, we remove the final threshold layer in the SAM module to output continuous score dis-
 137 tributions. On the other hand, LLM-Seg [21] (which uses LLaVA-7B) decouples the modules by
 138 initially assigning SAM the task of producing mask proposals, which are then evaluated and chosen
 139 by LLaVA. Unlike LISA, since there are no straightforward methods to fine-tune LLM-Seg with
 140 continuous score maps directly, we apply a binary threshold to our score maps for training (without
 141 text-based question-answer). We also incorporate the scores generated by LLM-Seg into all binary
 142 mask proposals to output a score map. Lastly, we introduce two fully decoupled baselines that output
 143 landmark names and scores from LLaVA-13B and Qwen2-VL-7B. These landmark names are then
 144 fed as input into GroundedSAM [22] to generate segmentation masks, which are then aggregated
 145 with the VLM’s per-region scores to obtain score masks. Note that unlike LISA and LLM-Seg, we
 146 do not fine-tune LLaVA and Qwen2-VL. The prompts used are shown in Appendix E.2.

147 **Scaling Up VLM Training Dataset:** To justify the score map dataset size of **80k**, we experi-
 148 ment with varying the amount of training data used to fine-tune our strongest vision model baseline
 149 (LISA-7B). As seen from Table C.1, scaling up the dataset generally improves search performance.
 150 However, we note that the performance gain becomes marginal when we scale our dataset from 55k
 151 to 80k. This indicates generating additional data may not yield further performance gain. Hence, it
 152 is a reasonable choice to stop at a dataset size of **80k** (also due to the cost of running GPT4o). Note
 153 that the steps taken to find the last target are not indicative of search performance, as we do not take
 154 the average of steps taken for episodes where the last target is not found.

Table C.2: Scaling Up Dataset Size for CLIP Fine-tuning - Targets Found (%) \uparrow

| Dataset | | | $B = 256$ | | | $B = 384$ | | |
|--------------|------|-----|-------------|-------------|-------------|-------------|-------------|-------------|
| Type | Size | TTA | All | Bot. 5% | Bot. 1% | All | Bot. 5% | Bot. 1% |
| Full | 473k | Y | 58.1 | 34.1 | 29.1 | 76.1 | 56.0 | 50.8 |
| Full | 473k | N | 57.0 | 27.8 | 19.4 | 75.3 | 50.1 | 44.0 |
| Half | 200k | Y | 56.5 | 26.0 | 22.1 | 75.0 | 49.6 | 48.0 |
| Half | 200k | N | 55.6 | 21.7 | 13.6 | 74.3 | 42.5 | 26.3 |
| AVS Only | 80k | Y | 53.7 | 35.0 | 30.1 | 73.7 | 60.2 | 61.6 |
| AVS Only | 80k | N | 52.8 | 21.7 | 14.4 | 72.1 | 41.9 | 37.7 |
| No fine-tune | - | Y | 48.8 | 24.4 | 20.9 | 66.2 | 51.3 | 39.7 |
| No fine-tune | - | N | 47.7 | 18.4 | 10.0 | 68.0 | 39.6 | 28.5 |

Scaling Up CLIP Dataset: We measure the performance of our satellite image CLIP encoder fine-tuned with data sets of different sizes in Table C.2, to justify why we choose to use the full data set of **473k** images. In particular, we fine-tune CLIP with images from the full **437k** dataset (that contains out-of-domain taxonomies), from the **80k** AVS dataset (used to generate score maps for VLM fine-tuning), the **200k** dataset downsampled from the full dataset. In addition, we conduct a study in which we do not fine-tune the CLIP model at all. We note an increasing trend in search performance as we scale the dataset. Despite the 473k dataset containing out-of-domain taxonomies, we end up choosing it to allow for fairer comparison with our VLM baselines. This is because the VLMs have the added advantages of using CLIP as a foundation and also being pretrained on much larger datasets. In addition, our results indicate that TTA improves search performance for CLIP models trained on all dataset sizes, highlighting the versatility of TTA across different configurations.

C.3 AVS Framework Baselines

We evaluate the effectiveness of the Search-TTA framework by comparing its performance with existing AVS baselines in the remote sensing domain. Similar to our setup, VAS [23] and PSVAS [24] models an AVS problem where an agent, guided by aerial imagery and operating under a fixed query budget, aims to maximize the number of targets found. VAS utilizes end-to-end reinforcement learning to co-train a feature extraction network and a policy network. The detection results gathered during the search process are then piped back into the feature extraction network for prediction updates. On the other hand, PSVAS decouples its prediction module from its policy network. PSVAS pretrains its prediction module using supervised learning, and jointly optimizes both modules using reinforcement learning. During test-time, it uses detection results from the search process to directly update the weights of their prediction module. Note that their vision backbones (ResNet [25]) are not foundation models and must be trained on specific classes. In addition, both methods do not perform realistic path planning, but instead allow for querying of non-adjacent cells (i.e. teleporting). For fair comparison, we retain their ability to choose where it wants to query, but also perform Dijkstra path planning to consider the cells on-route to their query locations. We weigh our Dijkstra cost function with a combination of factors, which aims to minimize distance, maximize travelling along paths of high probability (output from their vision module), and avoiding visited cells.

D Additional Experiments and Analysis

In this section, we provide information on the additional experiments we conducted, to supplement the information in Section 6. Unless stated otherwise, we conduct all experiments on the **4k** validation dataset, discretize the search space to 24×24 grids, and perform TTA updates every 20 steps. In addition, we randomize the agent’s starting point for each episode but keep it consistent across baselines.

Table D.1: Comparing Against Prompt Learning - Targets Found (%) \uparrow

| Method | LR | Inference Time (s) | $\mathcal{B} = 256$ | | | $\mathcal{B} = 384$ | | |
|----------------|--------------|--------------------|---------------------|-------------|-------------|---------------------|-------------|-------------|
| | | | All | Bot. 5% | Bot. 1% | All | Bot. 5% | Bot. 1% |
| Weights (Ours) | (1e-6, 1e-5) | 0.15 | 58.1 | 34.1 | 29.1 | 76.1 | 56.0 | 50.8 |
| Prompt [26] | (1e-3, 1e-2) | 0.05 | 57.3 | 38.6 | 28.0 | 75.4 | 60.3 | 40.8 |
| No TTA | – | – | 57.0 | 27.8 | 19.4 | 75.3 | 50.1 | 44.0 |

Table D.2: Comparing Against Text-Based TTA (384 Steps) - RMSE (%) \downarrow

| Method | TTA Type | <i>Aves Charadriiformes (Shorebirds)</i> | | | | |
|----------------------------|----------|--|----------------|-------------|----------------|-------------|
| | | First (0%) | Quartile (25%) | Mid (50%) | Quartile (75%) | Last (100%) |
| CLIP | SPPP | 59.2 | 58.4 | 57.0 | 53.3 | 50.4 |
| | - | 59.2 | 59.2 | 59.2 | 59.2 | 59.2 |
| Qwen2+GroundedSAM [19, 22] | Text | 62.4 | 62.6 | 60.2 | 61.6 | 60.9 |
| | - | 62.4 | 62.4 | 62.4 | 62.4 | 62.4 |
| LLaVA+GroundedSAM [18, 22] | Text | 59.8 | 59.8 | 60.2 | 58.6 | 62.0 |
| | - | 59.8 | 59.8 | 59.8 | 59.8 | 59.8 |

D.1 Varying TTA Methodologies

Prompt Learning: We compare our approach to prompt learning [26], where we perform gradient updates on our satellite image prompt instead of our model weights. From Table D.1, we noticed that a different learning rate range works better for prompt learning, likely due to the number of parameters that are updated during backpropagation as observed in [27]. For fair comparison, we use the same hardware (1x NVIDIA A5000 GPU) to log the inference time. From our results, we observe that weights fine-tuning achieves better averaged performance but has a slower inference time. While the number of parameters updated for prompt learning is significantly less, we only observe three times faster in inference speed likely due to overheads with PyTorch’s computational graphs. We leave comparison with other fine-tuning methods such as LoRA [28] to future works.

Text-based TTA: We study the effect of an alternative text-based TTA strategy aside from our SPPP-based strategy. Instead of integrating the VLMs into our Search-TTA framework which may be time-consuming to test, we design a simple experiment using the region-based statistics logged during our Search-TTA’s search process (for *Aves Charadriiformes / Shorebirds*). For each region defined by kmeans clustering, we logged the number of targets found and the ratio of the number of patches explored, at the 25%, 50%, 75%, and 100% search checkpoints. We pass these statistics, along with their initial landmark and score predictions (at 0% checkpoint), into the VLM and prompt them to reconsider their predictions. From Table D.2, we note the inconsistency in RMSE values throughout the different checkpoints, in contrast to the consistent improvements made with our SPPP-based strategy. This highlights the importance of a principled approach to TTA to achieve consistent results. We share the prompt design in Appendix E.2.

D.2 Additional Baseline Comparisons

Search-TTA with Larger Budget: In addition to the experimental results presented in Sec. 6.1 (where $\mathcal{B} = 256$), we present results for $\mathcal{B} = 384$ in Table D.3. Similarly, our results show a general improvement in percentage targets found (especially in the bottom percentile), speed of locating targets, and quality of score maps generated (as measurements are collected during the search process). This highlights Search-TTA’s consistency in improving search performance.

Varying Vision Model: The results in Sec. 6.2 (Table 2) present us with a few more insights. As shown in Table B.2, Search-TTA with $\beta = 1/9$ and $\gamma = 3$ can outperform LISA even when $\mathcal{B} = 384$. Furthermore, we note the significantly longer inference speed for the Qwen2-VL+GroundedSAM and LLaVA+GroundedSAM baselines. This is because, unlike the other VLMs, Qwen2-VL and LLaVA are required to output landmark names and scores in text, which involves significantly more token generation compared to the custom *[SEG]* token used in LISA and LLM-Seg. LLaVA is

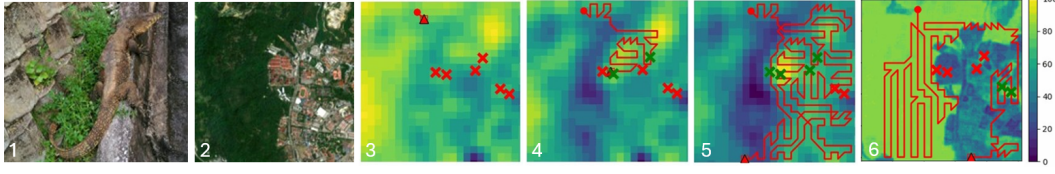


Figure D.1: (1) Ground image of a *Monitor Lizard*. (2) Satellite image where *Monitor Lizards* can be found. (3) Initial CLIP probability output. (4) TTA significantly increases probability values in urban region where the two lizards were found, while decreasing probability values in the forest where negative measurements were taken. (5) Improved priors leads to efficient search. (6) LISA [7] VLM output probability distribution with higher scores for the forest region, but is unable to perform online adaptation when no targets were found for a prolonged period.

Table D.3: Evaluating effects of TTA on different planners (CLIP Vision Backbone), $\mathcal{B} = 384$

| Planner Type | Found (%) \uparrow | | | Steps To Target, $S_T \downarrow$ | | | RMSE (%) \downarrow | | | Explored \uparrow |
|------------------|----------------------|-------------|-------------|-----------------------------------|--------------|-------|-----------------------|------|-------------|---------------------|
| | All | Bot. 5% | Bot. 1% | First | Mid | Last | First | Mid | Last | |
| RL (TTA) [17] | 76.1 | 56.0 | 50.8 | 100.8 | 117.9 | 255.2 | 45.9 | 47.0 | 53.5 | 65.7 |
| RL (no TTA) [17] | 75.3 | 50.1 | 44.0 | 102.2 | 120.0 | 256.9 | 45.9 | 45.9 | 45.9 | 65.7 |
| IS (TTA) [15] | 71.8 | 45.2 | 34.9 | 112.7 | 131.0 | 262.7 | 45.9 | 47.3 | 53.4 | 60.3 |
| IS (no TTA) [15] | 70.2 | 39.9 | 32.7 | 114.3 | 131.4 | 261.3 | 45.9 | 45.9 | 45.9 | 60.3 |
| Lawnmower [29] | 71.5 | – | – | 148.3 | 245.4 | 297.7 | – | – | – | 66.8 |

Table D.4: Comparing AVS frameworks (*Animalia Chordata Reptilia Squamata*)

| Frameworks | $\mathcal{B} = 256$ | | | | | $\mathcal{B} = 384$ | | | | |
|------------------|---------------------|--------------|---------------|-------------|--------------|---------------------|--------------|---------------|--------------|--------------|
| | Found (%) | Explored (%) | S_T (Start) | S_T (Mid) | S_T (Last) | Found (%) | Explored (%) | S_T (Start) | S_T (Mid) | S_T (Last) |
| CLIP+RL (TTA) | 55.7 | 44.4 | 74.8 | 82.5 | 156.8 | 72.7 | 66.0 | 94.8 | 107.3 | 248.8 |
| CLIP+RL (no TTA) | 52.5 | 44.4 | 83.4 | 89.1 | 161.1 | 72.5 | 65.8 | 100.8 | 117.6 | 267.5 |
| PSVAS [24] | 47.3 | 43.1 | 83.7 | 174.3 | 251.0 | 70.0 | 62.1 | 109.0 | 235.0 | 319.9 |
| VAS [23] | 46.8 | 44.1 | 75.5 | 165.7 | 199.8 | 68.5 | 64.6 | 89.3 | 192.6 | 257.7 |
| Lawnmower [29] | 43.5 | 44.6 | 116.0 | 163.8 | 170.6 | 71.4 | 66.8 | 149.1 | 237.1 | 286.8 |

222 slower compared to Qwen2-VL because we use a LLaVA-13B model compared to the Qwen2-VL-
223 7B model. Note that we take all inference speed measurements on a single NVIDIA A5000 GPU.

224 In addition, we provide snapshots of Search-TTA and LISA to compare their performance when
225 searching for *Monitor Lizards* (*reptilia Aquamata Varanidae*). From Figure D.1, we can see that
226 Search-TTA significantly increases probability values in the urban region and decreases proba-
227 bility values in the forest region after collecting positive and negative samples respectively. This
228 results in a more efficient search (4 targets found). In contrast, LISA over-exploits its initial belief
229 where *Monitor Lizards* are more likely to be found in the forest region, and is unable to correct its
230 probability distribution despite many negative measurements (only 2 targets found).

231 **AVS Baseline:** In addition to the experimental results presented in Sec. 6.3 (*Animalia Chordata*
232 *Aves Charadriiformes*), we present results for the same AVS framework baselines when searching
233 for *Animalia Chordata Reptilia Squamata* in Table D.4. Likewise, we notice the same trend where
234 Search-TTA outperforms almost all baselines in terms of percentage targets found and steps taken
235 to locate targets. This highlights Search-TTA’s versatility, given that it is able to outperform AVS
236 baselines (pretrained on specific taxonomies) with just a single model.

237 **Experimental Validation:** In addition to the details for our AVS hardware-in-the-loop experiments
238 described in Sec 6., we also rescale our Yosemite Valley simulated environment from $865\text{m} \times 865\text{m}$
239 to $280\text{m} \times 280\text{m}$. This is due to the crazyflie’s limited flight time of 5 minutes. We position 3D
240 bear models according to the approximate locations of sightings recorded on iNaturalist [3] from
241 2020 to date. The simulated drone flies at an altitude of 30m and is equipped with a 90° FOV
242 camera mounted on a gimbal (tilted at 30° to achieve better bear detection rates). We conduct both
243 experiments, with and without TTA, using $\mathcal{B} = 300\text{s}$ (traveled 2058m and 2062m respectively).

244 E Prompt Engineering

245 E.1 Score Map Generation Prompts

GPT4o Relabelling Prompt

You are an AI visual assistant that can analyze a single satellite image that is very zoomed out (covering around 2 km over the width and 2 km over the height of the image). You are given the bounding box of the segmentations of different regions, represented as (x1, y1, x2, y2) with floating numbers ranging from 0 to 1. These values correspond to the top-left x, top-left y, bottom-right x, and bottom-right y. These values correspond to the x and y coordinates. x-coordinates increase from left to right, and y-coordinates increase from top to bottom. The closer the instance is to the top edge, the smaller the y-coordinate. Assume that point1 [0, 0] is at the upper left, and point2 [1, 1] is at the bottom right.

The image contains different regions defined by bounding boxes. In the input data the regions are numbered from 0 to 3. The region names are: "Urban Area", "Barren", "Water", "Vegetation". Your task is to verify if the region inside the bounding box seems correct as seen in the image. Generate a conversation between yourself (the AI assistant) and a user asking about the photo. Verify the region names for each region given in the input data and tell me if it's name matches visually as seen from the image. When a region is incorrect then also remap the region name to the correct landmark name. Your responses should be in the tone of an AI that is "seeing" the image and answering accordingly.

When using all the provided information, directly generate the conversation. Always answer as if you are directly looking at the image.

Mapping:

```
{
  "Region 0": "Urban Area",
  "Region 1": "Barren",
  "Region 2": "Water",
  "Region 3": "Vegetation"
}
```

You must return your response in the JSON format:

```
{
  "conversation": [
    {
      "from": "human",
      "sat_key": sat_key,
      "taxonomy": taxonomy,
    },
    {
      "from": "gpt",
      "landmarks": {
        "Region i": "Correct/Incorrect",
        "Region j": "Correct/Incorrect",
        "Region k": "Correct/Incorrect",
      },
      "corrected_landmarks": {
        "Region i": {
          "name": "Urban Area/Barren/Water/Vegetation",
        },
        "Region j": {
          "name": "Urban Area/Barren/Water/Vegetation",
        },
        "Region k": {
          "name": "Urban Area/Barren/Water/Vegetation",
        },
      },
    }
  ]
}
```

{Examples}

{Input_Data}

Double check if the region names are correct. In the answer if the region name seems incorrect then remap it to the correct landmark name, else leave it as it is. Once again, please output your response in the JSON format only.

246

GPT4o Scoring Prompt

You are an AI visual assistant that can analyze a single satellite image that is very zoomed out (covering around 2 km over the width and 2 km over the height of the image). A specific animal/plant location within the image is given, along with detailed coordinates. The locations are in the form of coordinates, represented as (x,y) with floating numbers ranging from 0 to 1. These values correspond to the x and y coordinates. x-coordinates increase from left to right, and y-coordinates increase from top to bottom. The closer the instance is to the top edge, the smaller the y-coordinate. Assume that point1 [0, 0] is at the upper left, and point2 [1, 1] is at the bottom right.

You are also given the name of the animal/plant as taxonomy. Along with the exact coordinates in the image, you are also given the bounding box of the area where the animal/plant was found, represented as (x1, y1, x2, y2) with floating numbers ranging from 0 to 1. These values correspond to the top-left x, top-left y, bottom-right x, and bottom-right y. The values follow the same format as the coordinates.

The image contains different landmarks defined by bounding boxes. If a coordinate lies inside a particular bounding box and that bounding box belongs to a specific landmark, that means the animal/plant was found in that landmark. Your task is to generate a conversation between yourself (the AI assistant) and a user asking about the photo. Your responses should be in the tone of an AI that is "seeing" the image and answering accordingly. Using the image, taxonomy, coordinates, and bounding box information, provide a score for each landmark and a detailed explanation of where the animal/plant could be found among those landmarks based on the landmark semantics from the image. Add a simple question in the conversation, asking about where to find the particular animal/plant in the image (use common name as well as taxonomy for question).

Given the explanation, evaluate a score based on the likelihood of finding the queried animal/plant in each landmark. The length of the 'landmarks' list must match the length of the 'score' list. Probability scores must range between 0.0 and 1.0. It is acceptable for multiple consecutive landmarks to have the same scores. For example, a frog may have the same score for both water and land-type landmarks. Scores can be any value between 0.0 and 1.0, such as 0.1, 0.3, 0.5, 0.7, or 0.9. You must be more conservative, where if you are not sure, you should assign a lower score. If the animals cannot exist inside this landmark, please assign a score of 0.0. For example, a land animal or a non-aquatic plant cannot live inside the water body.

The scoring system is defined as follows:

1.0: Almost guaranteed to find the animal in the landmark
0.8: Very likely to find the animal in the landmark
0.6: Likely to find the animal in the landmark
0.4: Unlikely to find the animal in the landmark
0.2: Very unlikely to find the animal in the landmark
0.0: Almost impossible to find the animal in the landmark

You must answer with an explanation, landmarks, and scores for each region. Score the area of a particular region not only based on the number of targets actually present but also using the semantic information of the region. Provide a detailed explanation for your scoring, describing the relationship between the region's semantics and the animal/plant.

When using all the provided information, directly generate the conversation. Always answer as if you are directly looking at the image. Do not mention bounding box or region numbers explicitly, instead use the assigned landmark names. Only output the landmark name, not the landmark coordinates. Your answer may include multiple landmark types. It is acceptable for multiple consecutive landmarks to have the same scores.

You must return your response in the JSON format:

```
{
  "conversation": [
    {
      "from": "human",
      "sat_key": sat_key,
      "taxonomy": taxonomy,
      "common_name": common_name,
      "question": question using taxonomy,
    },
    {
      "from": "gpt",
      "explanation": answer,
      "landmarks": {
        "landmark1": {
          "score": score,
        },
        "landmark2": {
          "score": score,
        },
        "landmark3": {
          "score": score,
        }
      }
    }
  ]
}
```

```

    },
  }
}

{Examples}

{Input_Data}

Once again, please output your response in the JSON format.

```

248

249 E.2 VLM Baselines Prompts

Llava Inference Prompt (LLaVA+GroundedSAM Baseline)

Using the image as a reference, where can {animal} be found? Give me 1-2word high-level landmark names where it can be found in the image. Your response will be a json object where the landmark names are the keys and the probability of it being found in the landmark as values e.g. `{{"barren_land": 0.6}}`. Return just between 3 to 5 landmarks. Your response must be a single json object enclosed in double quotes without additional text.

250

Llava TTA Prompt (LLaVA+GroundedSAM Baseline)

You are provided with a heat map of the satellite image earlier, region statistics showing the percentage of the region in the satellite image that has been explored and number of {animal} found in these regions. Region Statistics:\n{explore_info}\n Use these information to update the probabilities of {animal} being found in the landmarks generated previously: {orig_response}. Do not associate the region Rn with any of the landmarks, they are not related in any way. Return your answer as a JSON object in this format: {new_dict}, where the keys are enclosed with double quotes. Begin your answer with explanation and reasoning steps for calculating the new probability values for {landmark_names}, then return the JSON object. You must enclose the JSON object within ```json tag. e.g. ```json{{"<landmark>": <new value>}}```.

251

Qwen Inference Prompt (Qwen2+GroundedSAM Baseline)

Using the image as a reference, where can {animal} be found? Give me 1-2word high-level landmark names where it can be found in the image. Your response will be a json object where the landmark names are the keys and the probability of it being found in the landmark as values e.g. `{{"barren_land": 0.8}}`. Return just between 3 to 5 landmarks. The key should be landmark names, not any other animals or food. Your response must be a single json object enclosed in double quotes without additional text, and do not return the examples given as a result.

252

Qwen TTA Prompt (Qwen2+GroundedSAM Baseline)

You are provided with a satellite image,a heat map of the satellite image and region statistics and tasked to use these information to come up with pairs of landmark:probabilities, where probabilities denotes the chances of finding {animal} in the landmark. The heat map, together with region statistics shows the percentage of the region in the satellite image that has been explored, as well as the number of {animal} found in it. Region Statistics:\n{explore_info}\nSuppose the original response {orig_response}, you need to use the heat map and region statistics information to come up with the new probabilities associated with the given landmark, using the satellite image as a reference. Do not associate regions Rn with {landmark_names}, they are not related to one another. Return your answer in this format: {new_dict} with new probability values (between 0 and 1) you calculated from the region statistics. Begin your response with explanation and reasoning steps for calculating new values for {landmark_names}, and return a single JSON object in {new_dict} format with the new probability values, enclosed in double quotes for the key and values. Enclose the json object within the ```json tag. e.g. ```json{new_dict}```. The JSON key should be enclosed by double quotes.

253

254 References

- 255 [1] S. Sastry, S. Khanal, A. Dhakal, A. Ahmad, and N. Jacobs. Taxabind: A unified embedding
256 space for ecological applications. In *2025 IEEE/CVF Winter Conference on Applications of*
257 *Computer Vision (WACV)*, pages 1765–1774. IEEE, 2025.

- [2] H. Van der Werff and F. Van der Meer. Sentinel-2a msi and landsat 8 oli provide data continuity for geological remote sensing. *Remote sensing*, 8(11):883, 2016.
- [3] iNaturalist. inaturalist. URL <https://www.inaturalist.org>.
- [4] C. Ye, Y. Zhuge, and P. Zhang. Towards open-vocabulary remote sensing image semantic segmentation.
- [5] A. Garioud, S. Peillet, E. Bookjans, S. Giordano, and B. Wattrelos. Flair 1: semantic segmentation and domain adaptation dataset. 2022. doi:10.13140/RG.2.2.30183.73128/1. URL <https://arxiv.org/pdf/2211.12979.pdf>.
- [6] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- [7] X. Lai, Z. Tian, Y. Chen, Y. Li, Y. Yuan, S. Liu, and J. Jia. Lisa: Reasoning segmentation via large language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9579–9589, 2024.
- [8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.
- [9] S. Stevens, J. Wu, M. J. Thompson, E. G. Campolongo, C. H. Song, D. E. Carlyn, L. Dong, W. M. Dahdul, C. Stewart, T. Berger-Wolf, et al. Bioclip: A vision foundation model for the tree of life. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19412–19424, 2024.
- [10] J. Ma, P.-Y. Huang, S. Xie, S.-W. Li, L. Zettlemoyer, S.-F. Chang, W.-T. Yih, and H. Xu. Mode: Clip data experts via clustering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [11] K. R. Shahapure and C. Nicholas. Cluster quality analysis using silhouette score. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 747–748, 2020.
- [12] D. Marutho, S. Hendra Handaka, E. Wijaya, and Muljono. The determination of cluster number at k-mean using elbow method and purity evaluation on headline news. In *2018 International Seminar on Application for Technology of Information and Communication*, pages 533–538, 2018.
- [13] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018. URL <https://arxiv.org/abs/1706.02677>.
- [14] Y. Cao, T. Hou, Y. Wang, et al. Ariadne: A reinforcement learning approach using attention-based deep networks for exploration. In *2023 IEEE ICRA*, 2023.
- [15] P. Lanillos, S. K. Gan, E. Besada-Portas, G. Pajares, and S. Sukkarieh. Multi-uav target search using decentralized gradient-based negotiation with expected observation. *Information Sciences*, 282:92–110, 2014.
- [16] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications, 2019.
- [17] D. M. S. Tan, Y. Ma, J. Liang, Y. C. Chng, Y. Cao, and G. Sartoretti. Ir2: Implicit rendezvous for robotic exploration teams under sparse intermittent connectivity. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13245–13252. IEEE, 2024.

- [18] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.
- [19] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, et al. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.
- [20] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.
- [21] J. Wang and L. Ke. Llm-seg: Bridging image segmentation and large language model reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1765–1774, 2024.
- [22] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.
- [23] A. Sarkar, M. Lanier, S. Alfeld, J. Feng, R. Garnett, N. Jacobs, and Y. Vorobeychik. A visual active search framework for geospatial exploration. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 8316–8325, 2024.
- [24] A. Sarkar, N. Jacobs, and Y. Vorobeychik. A partially-supervised reinforcement learning framework for visual active search. *Advances in Neural Information Processing Systems*, 36: 12245–12270, 2023.
- [25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [26] M. Shu, W. Nie, D.-A. Huang, Z. Yu, T. Goldstein, A. Anandkumar, and C. Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. *Advances in Neural Information Processing Systems*, 35:14274–14289, 2022.
- [27] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models, 2020. URL <https://arxiv.org/abs/2001.08361>.
- [28] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nZeVKeeFYf9>.
- [29] H. Choset. Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113 – 126, October 2001.