APPENDIX: A Visual Active Search Framework for Geospatial Exploration

In this appendix, we provide details that could not be included in the main paper owing to space constraints, including: (A) Performance of VAS under uniform query cost; (B) VAS Pseudocode; (C) Policy architecture and training hyperparameter; (D) Search Performance Comparison with Different Feature Extractor Module; (E) More Visual Illustration of VAS and the Most Competitive Greedy Selection baseline Method; (F) Assessment of VAS and other Baseline Methods with a Different Evaluation Metric; (G) Search Performance Comparison with Other Policy Learning Algorithm (PPO); (H) Sensitivity Analysis of *VAS*; (I) Efficacy of TTA on Search Tasks involving Large Number of Grids; (J) Saliency map visualization of *VAS*;

A. Performance of VAS under Uniform Query Cost

In this section, we report the performance of VAS under uniform query cost. The results are presented in the following Table 8 for the *small car* target class and in Table 9 for the *building* class from the xView dataset. We observe significant improvements in performance of the proposed VAS approach compared to all baselines, ranging from $11-25\%$ improvement relative to the most competitive *greedy selection* approach.

Table 8: **ANT** comparisons for the *small car* target class.

Method	$\mathcal{C}$ = 12	$\mathcal{C}$ = 15	$\mathcal{C}$ = 18
random search (N = 30)	4.57	5.66	6.85
greedy classification (N = 30)	5.31	6.24	7.25
greedy selection [30] (N = 30)	5.47	6.45	7.46
active learning [37] (N = 30)	5.28	6.21	7.22
conventional AS [15] (N = 30)	4.86	5.97	6.92
VAS (N = 30)	**6.03**	**7.24**	**8.24**
random search (N = 48)	3.80	4.97	5.98
greedy classification (N = 48)	4.69	5.48	6.79
greedy selection [30] (N = 48)	4.92	5.81	6.98
active learning [37] (N = 48)	4.68	5.46	6.78
conventional AS [15] (N = 48)	3.96	5.45	6.14
VAS (N = 48)	**5.62**	**6.81**	**7.86**
random search (N = 99)	3.12	3.61	4.45
greedy classification (N = 99)	3.68	4.22	4.97
greedy selection [30] (N = 99)	3.81	4.52	5.28
active learning [37] (N = 99)	3.65	4.19	4.93
conventional AS [15] (N = 99)	3.24	3.87	4.61
VAS (N = 99)	**4.61**	**5.64**	**6.55**

We also present the results for *large vehicle* and *ship* target class from DOTA dataset in the following Table 10 and 11 respectively. We see the proposed VAS performs noticeably better than all baselines, ranging from 16–56% relative to the state-of-the-art greedy selection approach. The experimental outcomes in different settings are qualitatively similar to the settings under Manhattan distance-based query cost.

Table 9: **ANT** comparisons for the *building* target class.

Method	$\mathcal{C}$ = 12	$\mathcal{C}$ = 15	$\mathcal{C}$ = 18
random search (N = 30)	5.54	7.18	8.58
greedy classification (N = 30)	5.88	7.72	9.21
greedy selection [30] (N = 30)	6.39	7.95	9.52
active learning [37] (N = 30)	5.86	7.68	9.16
conventional AS [15] (N = 30)	5.76	7.37	8.87
VAS (N = 30)	**7.56**	**9.02**	**10.41**
random search (N = 48)	4.97	6.41	7.66
greedy classification (N = 48)	5.68	6.95	8.40
greedy selection [30] (N = 48)	5.93	7.26	8.71
active learning [37] (N = 48)	5.68	6.93	8.37
conventional AS [15] (N = 48)	5.22	6.67	7.84
VAS (N = 48)	**6.85**	**8.29**	**9.65**
random search (N = 99)	4.35	5.37	6.44
greedy classification (N = 99)	4.92	6.02	7.41
greedy selection [30] (N = 99)	5.38	6.53	7.79
active learning [37] (N = 99)	4.91	6.00	7.40
conventional AS [15] (N = 99)	4.55	5.64	6.75
VAS (N = 99)	**6.75**	**8.27**	**9.46**

Table 10: **ANT** comparisons for the *large vehicle* target class.

Method	$\mathcal{C}$ = 12	$\mathcal{C}$ = 15	$\mathcal{C}$ = 18
random search (N = 36)	3.44	4.08	5.19
greedy classification (N = 36)	3.95	4.62	5.56
greedy selection [30] (N = 36)	4.18	4.86	5.89
active learning [37] (N = 36)	3.92	4.60	5.54
conventional AS [15] (N = 36)	3.71	4.22	5.28
VAS (N = 36)	**5.14**	**6.05**	**7.00**
random search (N = 64)	3.40	4.03	5.14
greedy classification (N = 64)	3.87	4.59	5.55
greedy selection [30] (N = 64)	3.99	4.77	5.67
active learning [37] (N = 64)	3.85	4.54	5.51
conventional AS [15] (N = 64)	3.61	4.12	5.26
VAS (N = 64)	**6.30**	**7.65**	**8.90**

B. VAS Pseudocode

We have included the pseudocode of our proposed Visual Active Search algorithm in table 1.

C. Policy architecture, training hyperparameter, and the details of TTA

In table 12, we detail the *VAS* policy architecture with number of target grids as N. We use a learning rate of 10^{-4},

Table 11: **ANT** comparisons for the *ship* target class.

Method	$\mathcal{C}$ = 12	$\mathcal{C}$ = 15	$\mathcal{C}$ = 18
random search (N = 36)	2.69	3.38	4.46
greedy classification (N = 36)	3.21	3.99	5.11
greedy selection [30] (N = 36)	3.44	4.23	5.32
active learning [37] (N = 36)	3.18	3.95	5.07
conventional AS [15] (N = 36)	2.97	3.56	4.77
VAS (N = 36)	**4.58**	**5.34**	**6.23**
random search (N = 64)	2.54	3.01	4.21
greedy classification (N = 64)	3.34	3.74	4.94
greedy selection [30] (N = 64)	3.62	3.95	5.10
active learning [37] (N = 64)	3.32	3.71	4.93
conventional AS [15] (N = 64)	2.87	3.38	4.53
VAS (N = 64)	**5.04**	**6.50**	**7.38**

Algorithm 1 The VAS algorithm.

Require: A search task instance (x_i, y_i); budget constraint $\mathcal{C}$; search policy $\psi(x_i, o, B)$ with parameters θ;

Initialize $o^0 = [0...0]$; $B^0 = \mathcal{C}$; step $t = 0$
while $B^t > 0$ **do**
  $\tilde{y} = \psi(x_i, o^t, B^t)$
  $j \leftarrow Sample_{j \in \{Unexplored\ Grids\}}[\tilde{y}]$
  Query grid cell with index j and observe true label $y^{(j)}$.
  Obtain reward $R^t = y^{(j)}$.
  Update o^t to o^{t+1} with $o^{(j)} = 2y^{(j)} - 1$.
  Update B^t to B^{t+1} with $B^{t+1} = B^t - c(k, j)$ (assuming we query k'th grid at $(t-1)$).
  Collect transition tuple (τ) at step t, i.e., τ^t = (state = (x_i, o^t, B^t), action = j, reward = R^t, next state = (x_i, o^{t+1}, B^{t+1})).
  $t \leftarrow t + 1$
end while
Update the search policy parameters, i.e., θ using *REINFORCE* objective as in 3 based on the collected transition tuples (τ^t) throughout the episode.
Return updated search policy parameters, i.e., θ.

batch size of 16, number of training epochs 200, and the Adam optimizer to train the policy network in all results. We add a self-supervised head r to the VAS policy architecture for TTT. The architecture of self-supervised head is detailed in table 13. We applied a series of 4 up-convolution layers with intermediate ReLU activations followed by a tanh activation layer on the semantic features extracted using ResNet34. For FixMatch, our VAS architecture remains unchanged, and we apply only spatially invariant augmentations (e.g auto contrast, brightness, color, and contrast) and ignore all translation augmentations (translate X, translate Y, ShearX etc.) to obtain the augmented version of the input image. We update the model parameters after every query step using a cross-entropy loss between a pseudo-target and a predicted vector as described below. We define the pseudo-target vector as follows. Whenever a query j is successful ($y_j = 1$), we construct a label vector as the one-hot vector with a 1 in the jth grid cell. However if $y_j = 0$, we associate each queried grid cell with a 0, and assign a uniform probability distribution over all unqueried grids. Prediction vector is the "logit" representation obtained from the VAS policy. We used the Adam optimizer with a learning rate of 10^{-4} for both TTT and FixMatch.

Table 12: VAS Policy Architecture

Layers	**Configuration**	**o/p Feature Map size**
Input	RGB Image	3 × 2500 × 3000
Feat. Extraction	ResNet-34	512 × 14 × 14
Conv1	c:N k:1 × 1	N × 14 × 14
Tile1	Grid State (o)	N × 14 × 14
Tile2	Query Left (B)	1 × 14 × 14
Channel Concat	Conv1,Tile1,Tile2	$(2N + 1)$ × 14 × 14
Conv2	c:3 k:1 × 1	3 × 14 × 14
Flattened	Conv2	588
FC1+ReLU	$(588- > 2N)$	2N
FC2	$(2N- > N)$	N

Table 13: Self-supervised head Architecture

Layers	**Configuration**
Input: Latent Feature	36 × 14 × 14
1st Up-conv layer	in-channel:36;out-channel:36;k:3 × 3;stride:2;padd:0
Activation Layer	ReLU
2nd Up-conv layer	in-channel:36;out-channel:24;k:3 × 3;stride:2;padd:1
Activation Layer	ReLU
3rd Up-conv layer	in-channel:24;out-channel:12;k:2 × 2;stride:4;padd:1
Activation Layer	ReLU
4th Up-conv layer	in-channel:12; out-channel:3; k:2 × 2; stride:2; padd:0
Normalization layer	tanh

D. Search Performance Comparison with Different Feature Extractor Module

In this section, we compare the performance of VAS with different feature extraction module. We use state-of-the-art feature extraction modules, such as ViT [6] and DINO [4] for comparison. The Vision Transformer (ViT) [6] is a transformer encoder model (BERT-like) pretrained on a large collection of images in a self-supervised fashion, namely ImageNet-21k (a collection of 14 million images), at a resolution of 224 × 224 pixels, with patch resolution of 16 × 16. Note that, we use off the shelf pretrained ViT model provided by huggingface (google/vit-base-patch16-224-in21k). We call the resulting policy *VAS-ViT*. Similar to

ViT, DINO [4] is also based on transformer encoder model. Images are presented to the DINO model as a sequence of fixed-size patches (resolution 8x8), which are linearly embedded. For our experiment, we use DINO pretrained on ImageNet-1k, at a resolution of 224x224 pixels. For our experiments, we use pretrained DINO model provided by huggingface (facebook/dino-vits8). We call the resulting policy as VAS-DINO. In table 14, 15 we report the performance of VAS-ViT and VAS-DINO and compare them with VAS.

Table 14: **ANT** comparisons with different feature extraction module for the *small car* target class on xView.

Method	$\mathcal{C}$ = 25	$\mathcal{C}$ = 50	$\mathcal{C}$ = 75
VAS-DINO (N = 30)	4.56	7.41	9.83
VAS-ViT (N = 30)	**4.64**	7.47	9.86
VAS (N = 30)	4.61	**7.49**	**9.88**
VAS-DINO (N = 48)	4.52	7.41	9.59
VAS-ViT (N = 48)	4.56	7.44	**9.68**
VAS (N = 48)	**4.56**	**7.45**	9.63

Table 15: **ANT** comparisons with different feature extraction module for the *large vehicle* target class on DOTA.

Method	$\mathcal{C}$ = 25	$\mathcal{C}$ = 50	$\mathcal{C}$ = 75
VAS-DINO (N = 36)	4.56	6.75	8.03
VAS-ViT (N = 36)	4.60	**6.82**	**8.09**
VAS (N = 36)	**4.63**	6.79	8.07
VAS-DINO (N = 64)	5.27	8.44	10.45
VAS-ViT (N = 64)	5.31	**8.51**	10.48
VAS (N = 64)	**5.33**	8.47	**10.51**

E. More Visual Illustration of VAS and the Most Competitive Greedy Selection baseline Method

In this section, we provide additional visualization of comparative exploration behaviour of VAS and and the most competitive greedy selection baseline approach. In figure 7, we compare the search strategy with *large vehicle* as a target class. In figure 8, we compare the behaviour with *small car* as a target class. In figure 9, we analyze the exploration behaviour with *ship* as a target class.

These additional visualizations again justify the efficacy of VAS over the strongest baseline method.

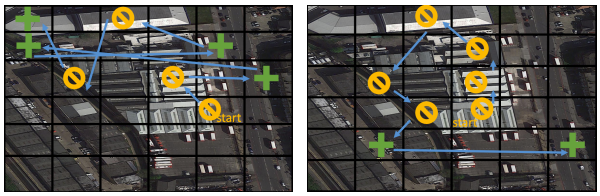


Figure 7: Comparison of policies learned using *VAS* (left) and the *greedy selection* baseline method (right).

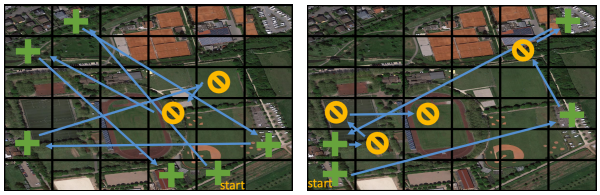


Figure 8: Comparison of policies learned using *VAS* (left) and the *greedy selection* baseline method (right).

Figure 9: Comparison of policies learned using *VAS* (left) and the *greedy selection* baseline method (right).

F. Assessment of VAS and other Competitive Baseline Methods with a Different Evaluation Metric

We additionally compare the search performance of VAS with all the baseline methods using a metric, which we call *Effective Success Rate (ESR)*. A naïve way to evaluate the proposed approaches is to simply use *success rate*, which is is the fraction of total search steps K that identify a target object. However, if K exceeds the total number of target objects in x, normalizing by K is unreasonable, as even a perfect search strategy would appear to work poorly. Consequently, we propose *effective success rate (ESR)* as the efficacy metric, defined as follows:

$$\textbf{ESR} = \frac{\#\,Targets\;Discovered}{\min\{\#\,Targets\,, K\}} \tag{4}$$

Thus, we divide by the number of targets one can possibly discover given a search budget K, rather than simply the search budget.

F.1. Results on the xView Dataset with ESR as Evaluation Metric

We initiate our analysis by assessing the proposed methodologies using the xView dataset, for varying search

budgets $K \in \{12, 15, 18\}$ and number of grid cells $N \in \{30, 48, 99\}$. We also consider two target classes for our search: *small car* and *building*. As the dataset contains variable size images, take random crops of 2500×3000 for $N = 30$, 2400×3200 pixels for $N = 48$, and 2700×3300 for $N = 99$, thereby guarantees uniform grid cell dimensions across the board.

Table 16: **ESR** comparisons for the *small car* target class on the xView dataset.

Method	$K = 12$	$K = 15$	$K = 18$
random search ($N = 30$)	0.598	0.632	0.704
greedy classification ($N = 30$)	0.619	0.675	0.718
greedy selection [30] ($N = 30$)	0.627	0.684	0.729
VAS ($N = 30$)	**0.766**	**0.826**	**0.861**
random search ($N = 48$)	0.489	0.517	0.558
greedy classification ($N = 48$)	0.512	0.551	0.589
greedy selection [30] ($N = 48$)	0.524	0.568	0.596
VAS ($N = 48$)	**0.694**	**0.722**	**0.741**
random search ($N = 99$)	0.336	0.369	0.378
greedy classification ($N = 99$)	0.365	0.384	0.405
greedy selection [30] ($N = 99$)	0.376	0.395	0.418
VAS ($N = 99$)	**0.564**	**0.587**	**0.602**

Table 17: **ESR** comparisons for the *building* target class on the xView dataset.

Method	$K = 12$	$K = 15$	$K = 18$
random search ($N = 30$)	0.663	0.681	0.697
greedy classification ($N = 30$)	0.701	0.734	0.767
greedy selection [30] ($N = 30$)	0.708	0.740	0.786
VAS ($N = 30$)	**0.854**	**0.886**	**0.912**
random search ($N = 48$)	0.526	0.547	0.556
greedy classification ($N = 48$)	0.548	0.569	0.585
greedy selection [30] ($N = 48$)	0.552	0.574	0.604
VAS ($N = 48$)	**0.677**	**0.716**	**0.738**
random search ($N = 99$)	0.443	0.462	0.483
greedy classification ($N = 99$)	0.460	0.482	0.504
greedy selection [30]	0.469	0.488	0.514
VAS ($N = 99$)	**0.654**	**0.676**	**0.690**

The results are presented in Table 16 for the *small car* class and in Table 17 for the *building* class. We see significant improvements in performance of the proposed *VAS* approach compared to all baselines, ranging from ~15–50% improvement relative to the most competitive state-of-the-art method, *greedy selection*.

F.2. Results on the DOTA Dataset with ESR as Evaluation Metric

We also conduct our experiments on the DOTA dataset. We use *large vehicle* and *ship* as our target classes. In both cases, we also report results with non-overlapping pixel grids of size 200×200 and 150×150 ($N = 36$ and $N = 64$, respectively). We again use $K \in \{12, 15, 18\}$.

Table 18: **ESR** comparisons for the *large vehicle* target class on the DOTA dataset.

Method	$K = 12$	$K = 15$	$K = 18$
random search ($N = 36$)	0.460	0.498	0.533
greedy classification ($N = 36$)	0.602	0.624	0.641
greedy selection [30] ($N = 36$)	0.618	0.637	0.647
VAS ($N = 36$)	**0.736**	**0.744**	**0.767**
random search ($N = 64$)	0.389	0.405	0.442
greedy classification ($N = 64$)	0.606	0.612	0.618
greedy selection [30] ($N = 64$)	0.612	0.618	0.626
VAS ($N = 64$)	**0.724**	**0.738**	**0.749**

Table 19: **ESR** comparisons for the *ship* target class on the DOTA dataset.

Method	$K = 12$	$K = 15$	$K = 18$
random search ($N = 36$)	0.491	0.564	0.590
greedy classification($N = 36$)	0.602	0.629	0.657
greedy selection [30] ($N = 36$)	0.609	0.638	0.665
VAS ($N = 36$)	**0.757**	**0.764**	**0.776**
random search ($N = 64$)	0.334	0.379	0.417
greedy classification ($N = 64$)	0.524	0.541	0.559
greedy selection [30] ($N = 64$)	0.531	0.552	0.576
VAS ($N = 64$)	**0.700**	**0.712**	**0.733**

The results are presented in Tables 18 and 19, and are broadly consistent with our observations on the xView dataset, with *VAS* outperforming all baselines by ~16–25%, with the greatest improvement typically coming on more difficult tasks (small K compared to N).

G. Search Performance Comparison with Other Policy Learning Algorithm (PPO)

We conduct experiments with other policy learning algorithm, such as PPO. With PPO [23], the idea is to constrain our policy update with a new objective function called the clipped surrogate objective function that will constrain the policy change in a small range $[1 - \epsilon, 1 + \epsilon]$. Here, ϵ is a hyperparameter that helps us to define this clip range. In all our experiment with PPO, we use clip range $\epsilon = 0.2$ as provided in the main paper [23]. We keep all other hyperparameters including policy architecture fixed. We call

the resulting policy *VAS-PPO*. In table 20, 21 we present the result of VAS-PPO and compare the performance with VAS. our experimental finding suggests that PPO doesn't yield any extra benefits in spite of having added complexity overhead due to the clipped surrogate objective.

Table 20: **ANT** comparisons with different policy learning algorithm for the *small car* target class on xView.

Method	$\mathcal{C}$ = 25	$\mathcal{C}$ = 50	$\mathcal{C}$ = 75
VAS-PPO (N = 30)	4.15	6.82	9.16
VAS (N = 30)	**4.61**	**7.49**	**9.88**
VAS-PPO (N = 48)	4.03	6.87	9.02
VAS (N = 48)	**4.56**	**7.45**	**9.63**

Table 21: **ANT** comparisons with different policy learning algorithm for the *large vehicle* target class on DOTA.

Method	$\mathcal{C}$ = 25	$\mathcal{C}$ = 50	$\mathcal{C}$ = 75
VAS-PPO (N = 36)	4.01	6.24	7.56
VAS (N = 36)	**4.63**	**6.79**	**8.07**
VAS-PPO (N = 64)	4.89	7.93	10.12
VAS (N = 64)	**5.33**	**8.47**	**10.51**

H. Sensitivity Analysis of *VAS*

We further analyze the behavior of *VAS* when we intervene the outcomes of past search queries o in the following ways: (i) Regardless of the "true" outcome, we set the query outcome to be "unsuccessful" at every stage of the search process and observe the change in exploration behavior of *VAS*, as depicted in fig 10, 11, 12. (ii) Following a similar line, we also enforce the query outcome to be "successful" at each stage and observe how it impacts in exploration behavior of *VAS*, as depicted in fig 10, 11, 12.

Early VAS steps are similar between strictly positive and strictly negative feedback scenarios. This is due to the grid prediction network's input similarity in early stages of VAS. The imagery and search budget are constant between the two, and the grid state vector between the two are mostly the same (as they are both initialized to all zeros). Following from step 7 we see VAS diverge. A pattern that emerges is that when VAS receives strictly negative feedback, it begins to randomly explore. After every unsuccessful query, VAS learns that similar areas are unlikely to contain objects of interest and so it rarely visits similar areas. This is most clear in figure 12 where we see at step 11 it explores an area that's completely water. It then visits a distinctive area that's mostly water but with land (and no harbor infrastructure). In strictly positive feedback scenarios we see VAS aggressively exploit areas that are similar to ones its already seen, as those areas have been flagged as having objects of interest. Consider the bottom row for each of figures 10, 11, and 12. In figure 10, after a burn in phase we see VAS looking at roadsides starting in step 9. In figure 11, VAS seeks to capture roads. By step 15, VAS has an elevated probability for nearly the entire circular road in the upper left of the image. In figure 12, VAS seeks out areas that look like harbors. Together these examples demonstrate a key feature of reinforcement learning: the ability to explore and exploit. Additionally, they show that VAS is sensitive to query results and uses the grid state to guide its search. In fig 13, 14, 15, we provide a similar visualization of VAS under Manhattan distance based query cost.

I. Efficacy of TTA on Search Tasks involving Large Number of Grids

We conduct experiments with number of grids N as 900. We train VAS using small car as target while evaluate with building as target class. We report the result in table 22. We observe a significant improvement (up to **4%**) in search performance by leveraging TTA in our proposed VAS framework. Specifically, the performance gap becomes more noticeable as the search budget increases. We observe a similar trend when we train VAS with building as target and evaluate using small car as target as presented in table 23. Such results reinforce the importance of TTA in scenarios (especially when the search budget is large) when the search target differs between training and execution environments.

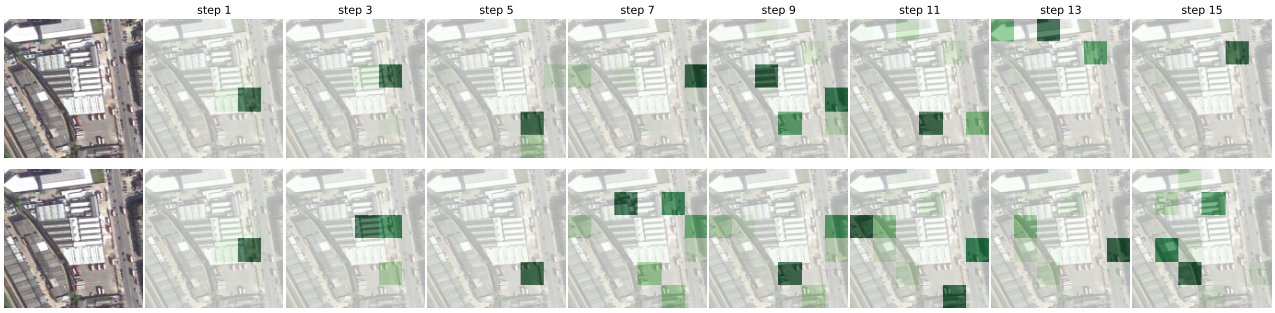
Table 22: Comparative results on xView dataset with *small car* and *Building* as the target class during training and inference respectively under uniform query cost setting.

Method	$\mathcal{C}$ = 18	$\mathcal{C}$ = 24	$\mathcal{C}$ = 30	$\mathcal{C}$ = 60
without TTA (N = 900)	3.32	4.30	5.41	10.39
Stepwise TTA (N = 900)	3.38	4.37	5.54	10.68
Online TTA (N = 900)	**3.41**	**4.42**	**5.60**	**10.81**

Table 23: Comparative results on xView dataset with *building* and *small car* as the target class during training and inference respectively under uniform query cost setting.

Method	$\mathcal{C}$ = 18	$\mathcal{C}$ = 24	$\mathcal{C}$ = 30	$\mathcal{C}$ = 60
without TTA (N = 900)	1.61	2.07	2.60	4.93
Stepwise TTA (N = 900)	1.63	2.10	2.66	5.04
Online TTA (N = 900)	**1.66**	**2.15**	**2.71**	**5.12**

(a) The original image

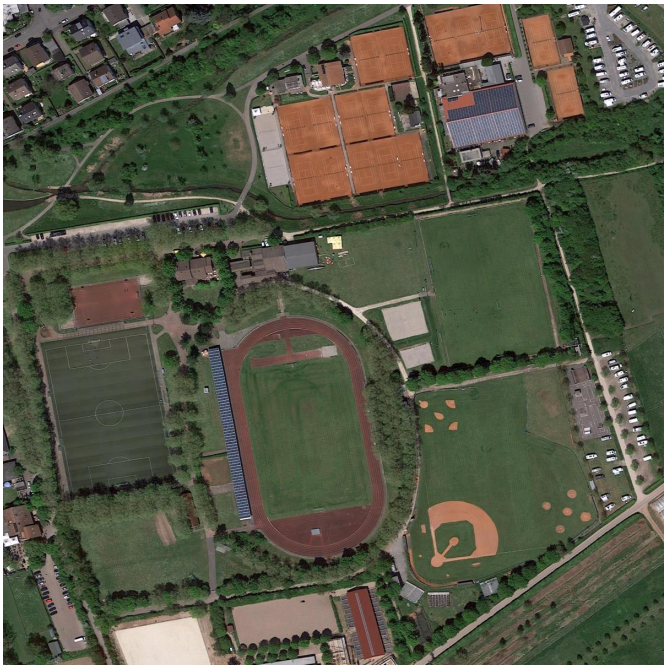


(b) ***(Top row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**unsuccessful**". ***(Bottom row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**successful**".

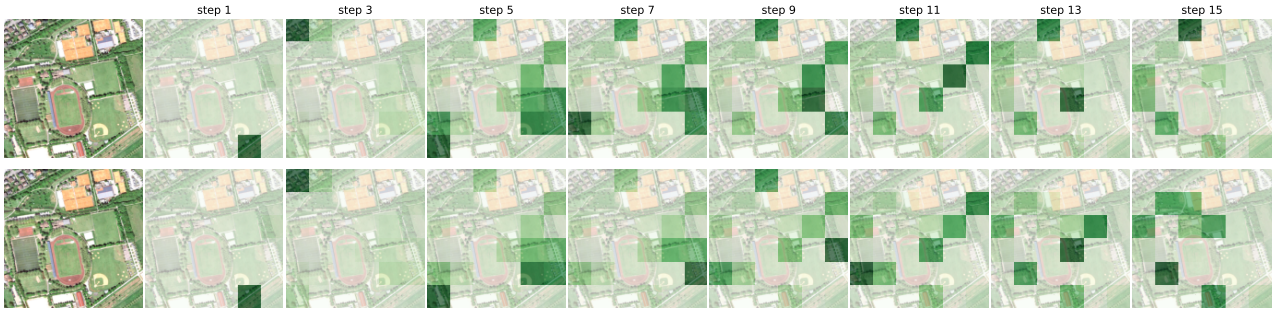
Figure 10: Sensitivity Analysis of *VAS* with a sample test image and *large vehicle* as target class under uniform query cost.

J. Saliency map visualization of *VAS*

In Figure (16,17,18), we show the saliency maps obtained using a pre-trained VAS policy at different stages of the search process. Note that, at every step, we obtain the saliency map by computing the gradient of the output that corresponds to the query index with respect to the input. Figure 16 corresponds to the large vehicle target class while the Figure 17 and Figure 18 correspond to the small vehicle. All saliency maps were obtained using the same search budget (K = 15). These visualizations capture different aspects of the VAS policy. Figure 16 shows its adaptability, as we see how heat transfers from non-target grids to the grids containing targets as search progresses. By comparing saliency maps at different stages of the search process, we see that, VAS explores different regions of the image at different stages of search, illustrating that our approach implicitly trades off exploration and exploitation in different ways as search progresses. Figure 17 shows the effect of supervised training on VAS policy. If we observe the saliency



(a) The original image

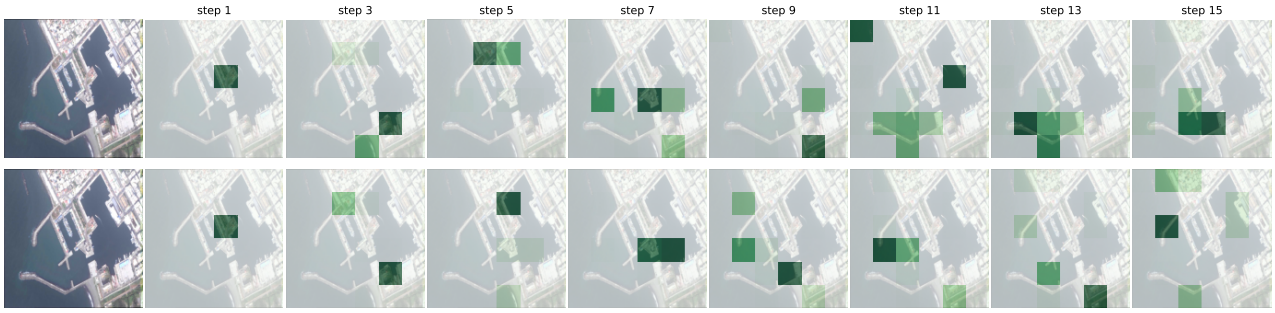


(b) ***(Top row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**unsuccessful**". ***(Bottom row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**successful**".

Figure 11: Sensitivity Analysis of *VAS* with a sample test image and *car* as target class under uniform query cost.

maps across time, we see that VAS never searches for small vehicles in the sea, having learned not to do this from training with similar images. Additionally, we notice that the saliency map's heat expands from left to right as the time step increases, encompassing more target grids, leading to the discovery of more target objects. We observe similar phenomena in figure 18. We can see that while earlier in the search process queries tend to be less successful, as the search evolves, our approach successfully identifies a cluster of grids that contain the desired object, exploiting spatial correlation among them. Additionally, at different stages of the search process, VAS identifies different clusters of grids that include the target object.

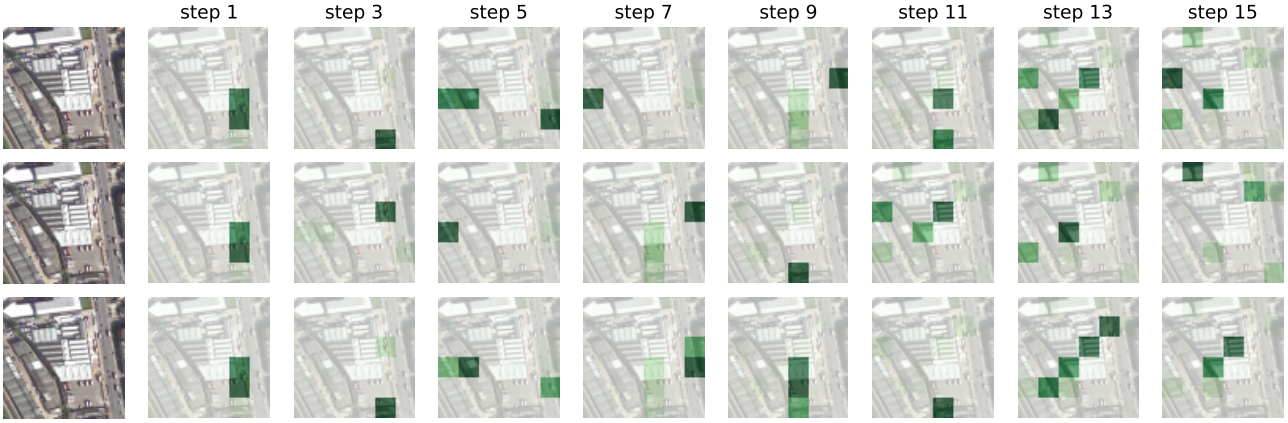
(a) The original image



(b) ***(Top row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**unsuccessful**". ***(Bottom row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**successful**".

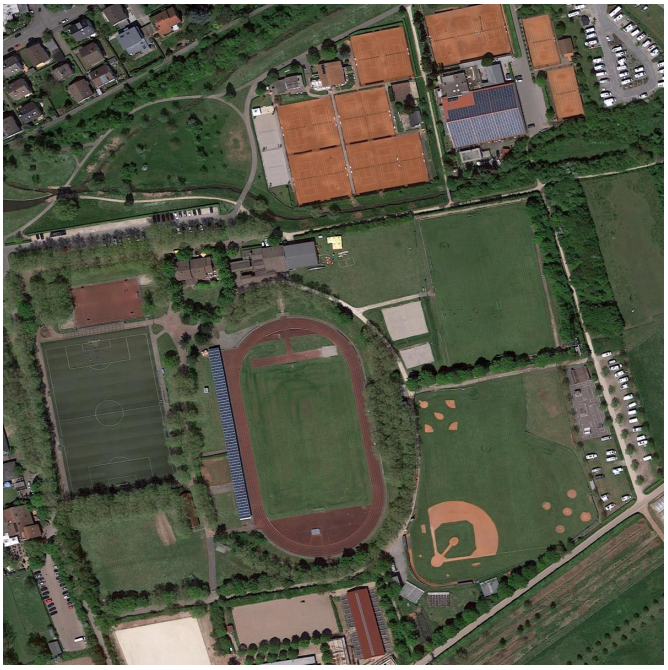
Figure 12: Sensitivity Analysis of *VAS* with a sample test image and *ship* as target class under uniform query cost.

(a) The original image

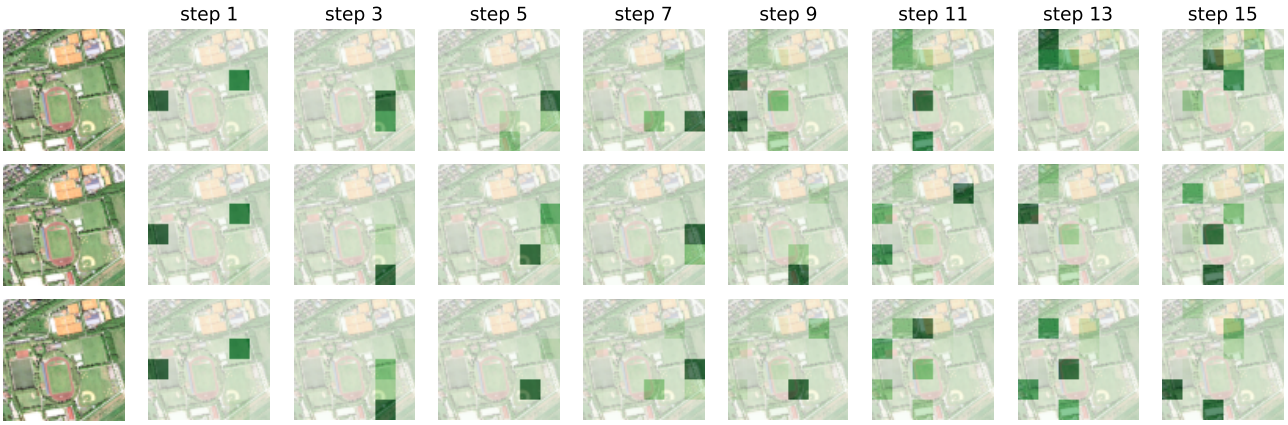


(b) ***(Top row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS. ***(Middle row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**unsuccessful**". ***(Bottom row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**successful**".

Figure 13: Sensitivity Analysis of *VAS* with a sample test image and *large vehicle* as target class under distance based query cost.



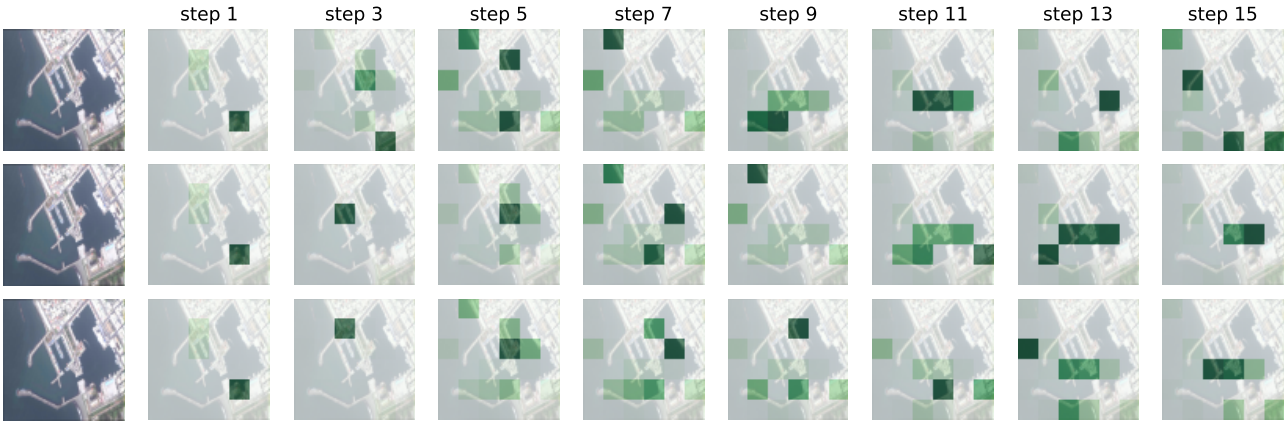
(a) The original image



(b) ***(Top row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS. ***(Middle row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**unsuccessful**". ***(Bottom row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**successful**".

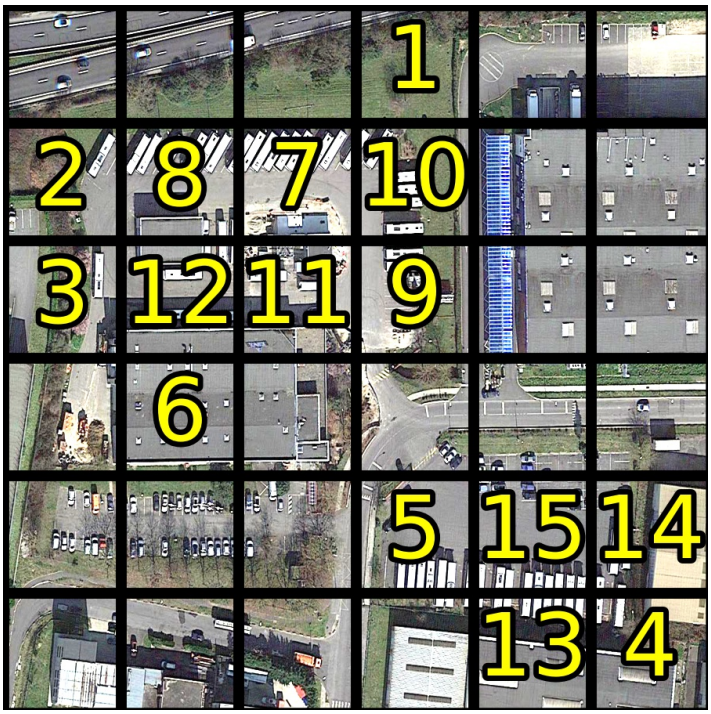
Figure 14: Sensitivity Analysis of *VAS* with a sample test image and *car* as target class under distance based query cost.

(a) The original image

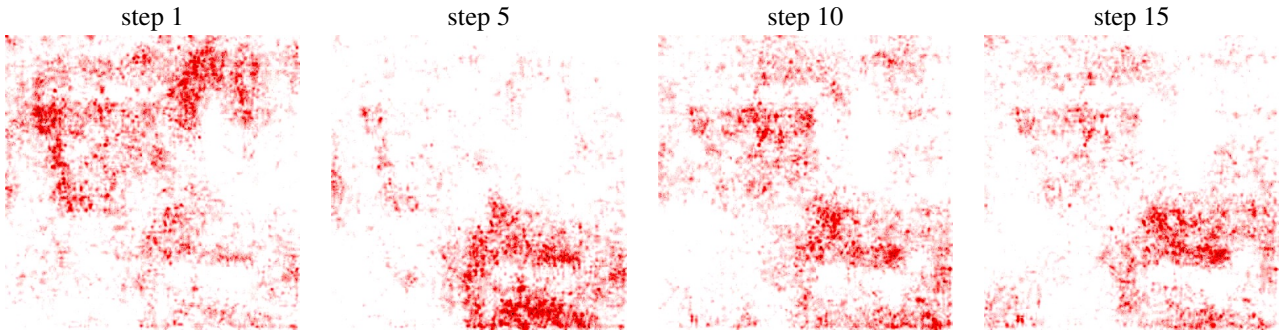


(b) ***(Top row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS. ***(Middle row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**unsuccessful**". ***(Bottom row)*** Query sequences, and corresponding heat maps (darker indicates higher probability), obtained using VAS while enforcing the query outcomes at every stage being "**successful**".

Figure 15: Sensitivity Analysis of *VAS* with a sample test image and *ship* as target class under distance based query cost.



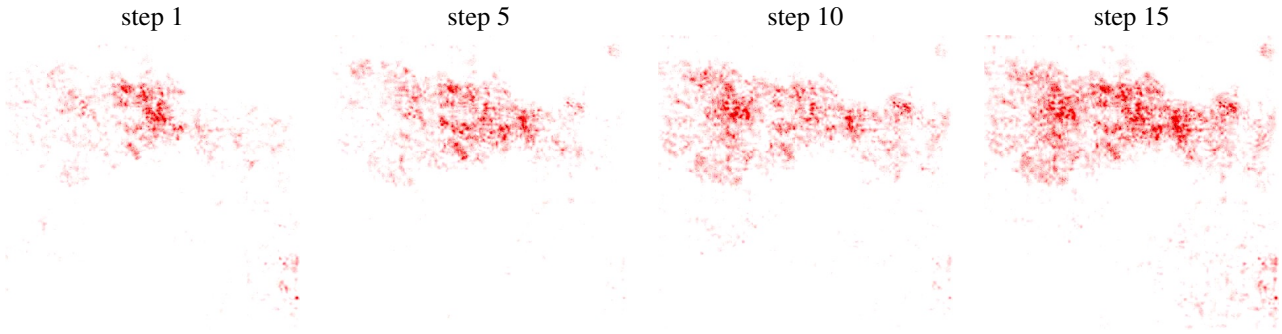
(a) The original image with query sequence.



(b) Saliency maps (red indicates high saliency), obtained using VAS at different stages of search process with *large vehicle* as target.

Figure 16: Saliency map visualization of *VAS* under uniform cost budget.

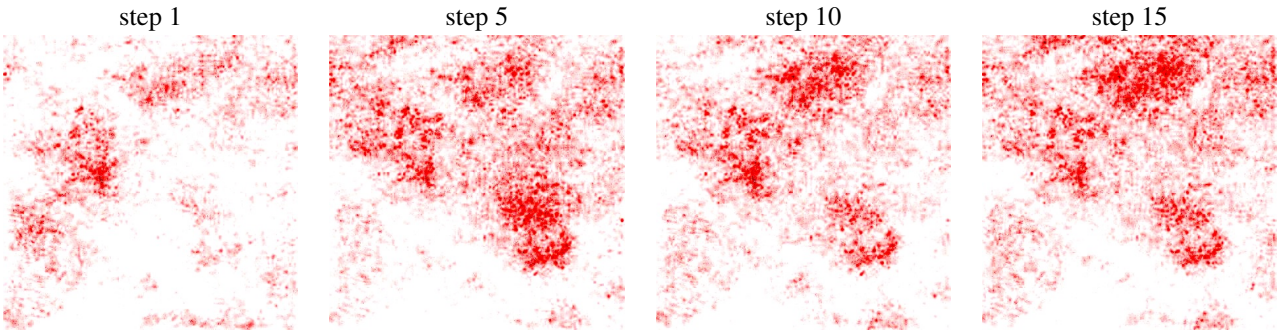
(a) The original image with query sequence.



(b) Saliency maps (red indicates high saliency), obtained using VAS at different stages of search process with *small car* as target.

Figure 17: Saliency map visualization of *VAS* under uniform cost budget.

(a) The original image with query sequence.



(b) Saliency maps (red indicates high saliency), obtained using VAS at different stages of search process with *small car* as target.

Figure 18: Saliency map visualization of *VAS* under uniform cost budget.