A Supplementary Material

The appendix presents additional details of our tracker in terms of design and experiments, as follows.

- **A.1 Volume of Language-Annotated Training Data**
  We analyze how the volume of language-annotated training data affects tracking performance.
- **A.2 Different Language Models**
  We analyze and compare different language embedding models (*i.e.*, BERT [3] and GPT-2 [7]) in our method.
- **A.3 Details of The Proposed Asymmetrical Searching Strategy (ASS)**
  We present more details about the pipeline of our proposed ASS.
- **A.4 Comparison of ASS and LightTrack [9]**
  We show efficiency comparison of our ASS and another NAS-based tracker LightTrack [9].
- **A.5 Visualization of Tracking Result and Failure Case**
  We visualize the tracking results and analyze a failure case of our tracker.
- **A.6 Activation Analysis of Different Language Descriptions**
  We study the impact of different language descriptions on tracking performance by visualizing their activation maps.
- **A.7 Attribute-based Performance Analysis**
  We conduct attribute-based performance analysis on LaSOT [4], and the results demonstrate the robustness of our tracker in various complex scenarios.

A.1 Volume of Language-Annotated Training Data

Language-annotated data is crucial for our proposed tracker in learning the robust vision-language representations. We analyze the influence of training with different volumes of language-annotated data and the results are presented in Tab. 8a. The default setting in the manuscript is noted as "100%". For settings of "50%" and "75%", the reduced part is filled with the data without language annotation, which keeps the whole training data volume. It shows that as the language-annotated training pairs reduced, the performance on LaSOT [4] gradually decreases ($63.9\% \rightarrow 61.8\% \rightarrow 58.9\%$ in SUC), demonstrating that more language-annotated data helps improve model capacity.

A.2 Different Language Models

As described in Sec. 3.1 of the manuscript, the language model of BERT [3] is adopted to abstract the semantics of the sentence, which directly relates to the learning of vision-language representation. To show the influence of different language models, we compare the results of using BERT [3] and GPT-2 [7], as shown in Tab. 8b. An interesting finding is that GPT-2 [7] even decreases the performances, which is discrepant with recent studies in natural language processing. One possible reason is that the bi-directional learning strategy in BERT [3] can better capture the context information of a sentence than the self-regression in GPT-2 [7].

Table 8: Evaluation for different settings on LaSOT: (a) training with different volumes of language-annotated data, (b) the influence of different language models.

(a)

Settings	SUC (%)	P (%)
50%	58.9	61.4
75%	61.8	64.9
100%	63.9	67.9

(b)

Settings	SUC (%)	P (%)
GPT-2 [7]	59.3	62.3
BERT [3]	63.9	67.9

A.3 Details of The Proposed Asymmetrical Searching Strategy (ASS)

As mentioned in Sec. 3.2 of the manuscript, ASS is designed to adapt the mixed modalities for different branches by simultaneously searching the asymmetrical network $\mathcal{N} = \{\varphi_t, \varphi_s, \varphi_m\}$. The pipeline of ASS consists of two stages. The first stage is pretraining to search architecture and the second one is to retrain it for our VL tracking, as summarized in Alg. 1.

Algorithm 1 Algorithm for Asymmetrical Searching Strategy.

1: **/* Search */**
2: **Input**: *Network* $\mathcal{N}$, *search space* $\mathcal{A}$, *max iteration* $\mathcal{T}$, *random sampling* Γ,
Train dataset: $\mathcal{D}_{train} = \{\mathcal{X}_n, y_n\}_{n=1}^{N}$, $\mathcal{X}_n = \{x_n^v, x_n^l(optional)\}$,
Val dataset: $\mathcal{D}_{val} = \{\mathcal{X}_m, y_m\}_{m=1}^{M}$,
For videos without language annotation: $x^l =$ "***0-tensor***" *or* "***template-embedding***".
3: **Initialize**: *Initialize the network parameters* $\theta_{\mathcal{N}}$.
4: **for** $i = 1 : \mathcal{T}$ **do**
5: **for** $n = 1 : N$ **do**
6: **if** language annotation exists **then**
7: $f_n^l = BERT(x_n^l)$;
8: **else if** $x_n^l =$"***0-tensor***" **then**
9: $f_n^l = zeros\_like[BERT(x_n^l)]$; // Default setting without language annotation
10: **else if** $x_n^l =$"***template-embedding***" **then**
11: $f_n^l = ROI(x_n^v)$; // Robust setting without language annotation
12: **end if**
13: $a = \Gamma(\mathcal{A})$, $p_n = \mathcal{N}(x_n^v, f_n^l; a)$;
14: **end for**
15: Update the network parameters $\theta_{\mathcal{N}}$ with gradient descent:
16: $\theta_{\mathcal{N}}^a \leftarrow \theta_{\mathcal{N}}^a - \alpha\partial\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(p_n, y_n; \theta_{\mathcal{N}}^a)/\partial\theta_{\mathcal{N}}^a$;
17: **end for**
18: $a_{best} = \mathrm{EvolutionaryArchitectureSearch}(\mathcal{A}, \mathcal{D}_{val}; \theta_{\mathcal{N}})$; [6]
19: **Initialize**: *Initialize the network parameters* $\theta_{\mathcal{N}}$.
20: **while** *not converged* **do**
21: **for** $n = 1 : N$ **do**
22: **line 6 - 12**
23: $p_n = \mathcal{N}(x_n^v, f_n^l; a_{best})$;
24: **end for**
25: Update the network parameters $\theta_{\mathcal{N}}$ with gradient descent:
$\theta_{\mathcal{N}}^{a_{best}} \leftarrow \theta_{\mathcal{N}}^{a_{best}} - \alpha\partial\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(p_n, y_n; \theta_{\mathcal{N}}^{a_{best}})/\partial\theta_{\mathcal{N}}^{a_{best}}$;
26: **end while**
27: **Output**: *network parameters* $\theta_{\mathcal{N}}$, a_{best}.
28: **/* Retrain */**
29: Train the searched networks $\theta_{\mathcal{N}}$, a_{best};
30: **/* Inference */**
31: Track the target by $y = \mathrm{argmax}\, p$.

The pretraining stage (line 3-18 in Alg. 1) contains four steps: **1)** ASS first initializes the network parameter $\theta_{\mathcal{N}}$ of $\mathcal{N}$. Concretely, φ_t and φ_s reuse the pretrained supernet of SPOS [6], while φ_m copies the weight of the last layer in φ_t and φ_s. This reduces the tedious training on ImageNet [2] and enables quick reproducibility of our work; **2)** The language model [3] processes the annotated sentence x^l to get corresponding representation f^l. If the language annotations are not provided, two different strategies are designed to handle these cases (*i.e.*, "**0-tensor**" or "**template-embedding**", illustrated in Sec. 4.4 and Tab. 5c of the manuscript); **3)** Then, $\theta_{\mathcal{N}}$ is trained for $\mathcal{T}$ iterations. For each iteration, a subnet a is randomly sampled from search space $\mathcal{A}$ by the function Γ and outputs the predictions p. The corresponding parameters of a would be updated by gradient descent; **4)** After pretraining, Evolutionary Architecture Search [6] is performed to find the optimal subnet a_{best}. The rewarding for evolutionary search is the SUC (success score) on validation data $\mathcal{D}_{val}$. The retraining

stage (line 19-27 in Alg. 1) is to optimize the searched subnet a_{best} following the training pipeline of baseline trackers [5, 1].

Tab. 9 displays the detailed configurations of the searched asymmetrical architecture, providing a complement to Tab. 1 in the manuscript.

Table 9: Configurations of the asymmetrical architecture learned by ASS.

	Stem	Stage1		Stage2		Stage3		Stage4		Output
Layer Name	Convolution Block	$\text{Block}_{\text{ASS}}$ $\times 3$	ModaMixer	$\text{Block}_{\text{ASS}}$ $\times 3$	ModaMixer	$\text{Block}_{\text{ASS}}$ $\times 7$	ModaMixer	$\text{Block}_{\text{ASS}}$ $\times 3$	ModaMixer	Convolution Block
Parameter	$P_{in}=2$ $C_{in}=16$	$P_1=2$ $C_1=64$	$P=1$ $C=64$	$P_2=2$ $C_2=160$	$P=1$ $C=160$	$P_3=1$ $C_3=320$	$P=1$ $C=320$	$P_4=1$ $C_4=640$	$P=1$ $C=640$	$P_{out}=1$ $C_{out}=256$
Output Size	$\frac{H}{2} \times \frac{W}{2}$	$\frac{H}{4} \times \frac{W}{4}$		$\frac{H}{8} \times \frac{W}{8}$		$\frac{H}{8} \times \frac{W}{8}$		$\frac{H}{8} \times \frac{W}{8}$		$\frac{H}{8} \times \frac{W}{8}$

A.4 Comparison of ASS and LightTrack [9]

Despite greatly boosting the tracking performance, Neural Architecture Search (NAS) brings complicated training processes and large computation costs. Considering the complexity, we ease unnecessary steps of ASS to achieve a better trade-off between training time and performance. Taking another NAS-based tracker (*i.e.*, LightTrack [9]) as the comparison, we demonstrate the efficiency of our proposed ASS.

As illustrated in Tab. 10, NAS-based trackers usually need to first pretrain the supernet on ImageNet [2] to initialize the parameters, which results in high time complexity in training. LightTrack even trains the backbone network on ImageNet [2] twice (*i.e.*, the 1st and 4th steps), which heavily increases the time complexity. By contrast, our ASS avoids this cost by reusing the pre-trained supernet from SPOS, which is much more efficient.

Table 10: Pipeline comparison of ASS and LightTrack in term of time complexity.

Steps	**LightTrack [9]**	**ASS in VLT** (Ours)
1st step	Pretraining backbone supernet on ImageNet [2]	Reusing trained backbone supernet of SPOS [6]
2nd step	Training tracking supernet on tracking datasets	Training tracking supernet on tracking datasets
3rd step	Searching with evolutionary algorithm on tracking supernet	Searching with evolutionary algorithm on tracking supernet
4th step	Retraining searched backbone subset on ImageNet [2]	Reusing trained backbone supernet of SPOS [6]
5th step	Finetuning searched tracking subset on tracking datasets	Finetuning searched tracking subset on tracking datasets
Network searching cost	~40 Tesla-V100 GPU days	~3 RTX-2080Ti GPU days

A.5 Visualization of Tracking Result and Failure Case.

As shown in Fig. 5, the proposed $\textbf{VLT}_{\text{SCAR}}$ delivers more robust tracking under deformation, occlusion (the first row) and interference with similar objects (the second row). It demonstrates the effectiveness of learned multimodal representation, especially in complex environments. The third row shows the failure case of our tracker. In this case, the target is fully occluded for about 100 frames and distracted by similar objects, leading to ineffectiveness of our tracker in learning helpful information. A possible solution to deal with this is to apply a global searching strategy, and we leave this to future work. Fig. 6 shows that our $\textbf{VLT}_{\text{TT}}$ achieves the best performance compared to other SOTAs. It demonstrates the resilience of our tracker and effectiveness of proposed multimodal VL tracking in complex environments.

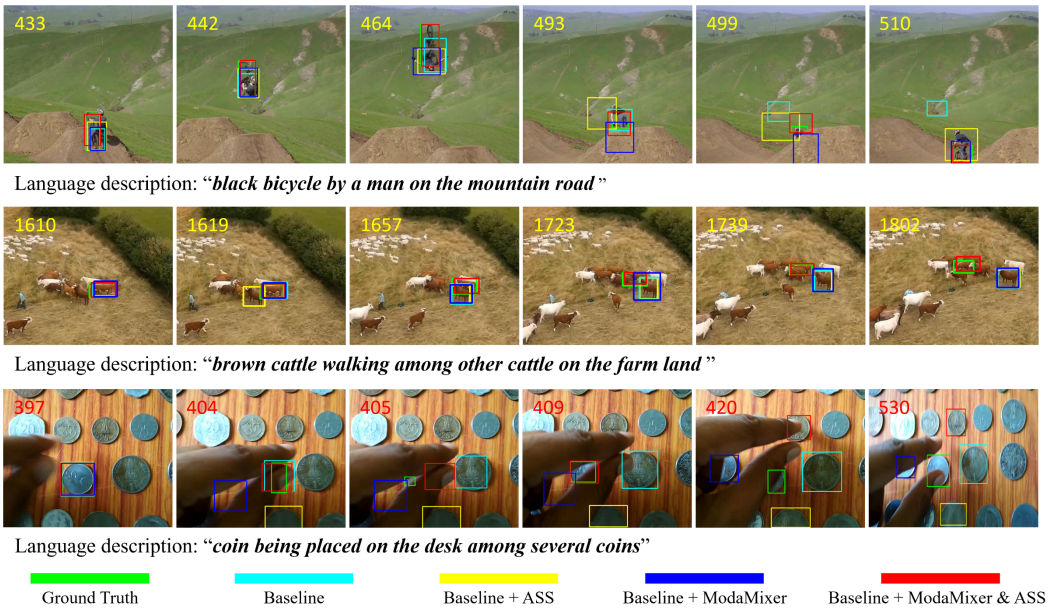


Figure 5: The first two rows show the success of our tracker in locating target object in complex scenarios, while the third row exhibits a failure case of our method when the target is occluded for a long period (with around 100 frames).

Figure 6: Results visualization of different trackers. The comparison shows that our **VLT**$_{\mathrm{TT}}$ could perform robust tracking under complex scenarios (*e.g.,* Deformation, Disappearance, Occlusion and Similar Interferences).

A.6 Activation Analysis of Different Language Descriptions

Language description provides high-level semantics to enhance the target-specific channels while suppressing the target-irrelevant ones. As presented in Fig. 7, we show the effect of different words to evidence that language helps to identify targets. The first row shows that the VLT$_{\mathrm{SCAR}}$ without language description focuses on two birds (red areas), interfered by the same object class. When introducing the word "bird", the response of the similar object is obviously suppressed. With a more detailed "black bird", the responses of distractors almost disappear, which reveals that more specific annotation can help the tracker better locate the target area. Furthermore, we also try to track the target with only environmental description, *i.e.*, "standing on the ground". The result in column 5 shows that the background is enhanced while the target area is suppressed. The comparison evidences

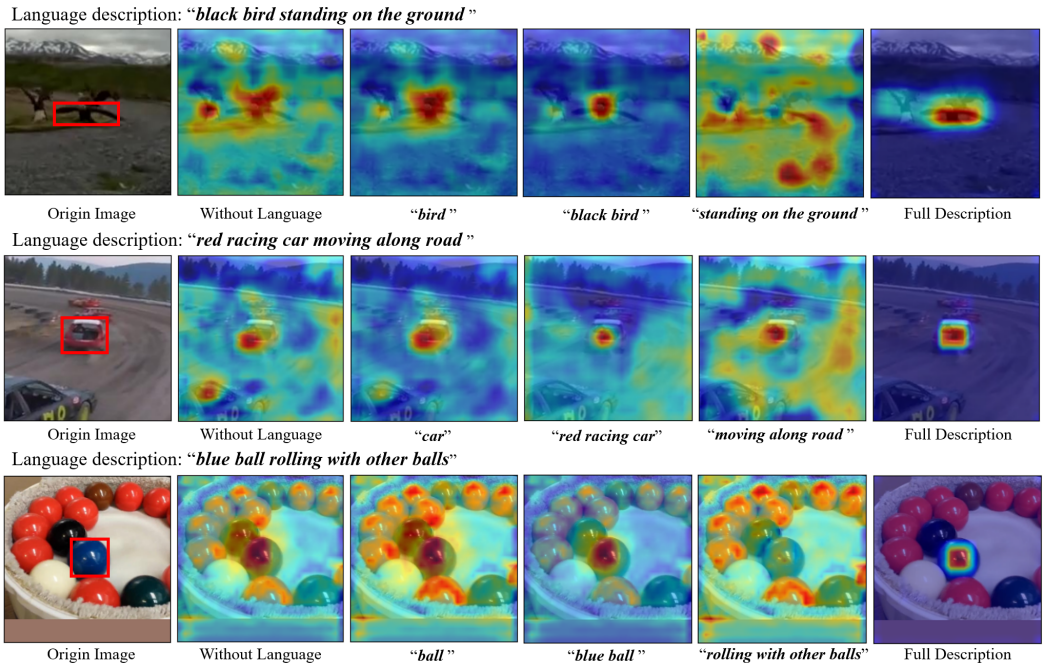
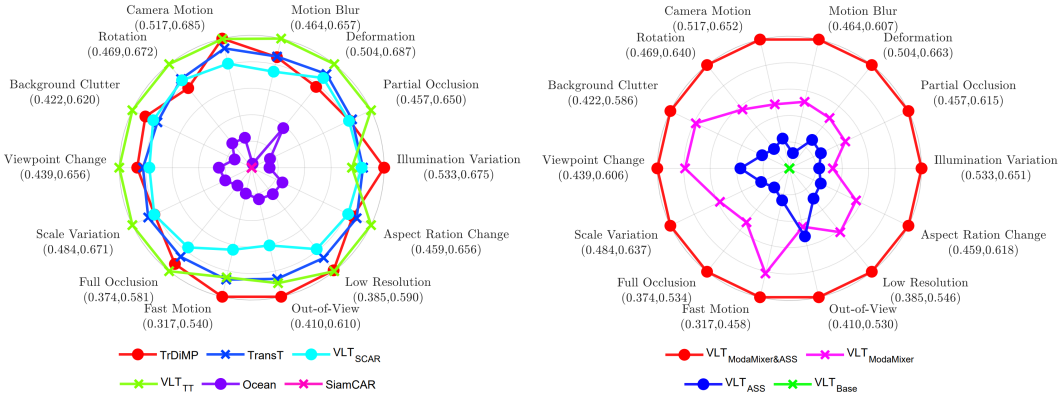


Figure 7: Activation visualization of $\mathbf{VLT}_{\mathrm{SCAR}}$ with different language descriptions using Grad-CAM [8]. The general language description endows our VL tracker the distinguishability between the target and interferences.

that the language description of object class is crucial for the tracker to distinguish the target from the background clutter, while the mere description of the environment (the fourth column) may introduce interference instead. The last column shows the activation maps with full description, where the tracker can precisely locate the target, demonstrating the effectiveness of the learned unified-adaptive vision-language representation.

A.7 Attribute-based Performance Analysis

Fig. 8 presents the attribute-based evaluation on LaSOT [4]. We compare $\mathrm{VLT}_{\mathrm{SCAR}}$ and $\mathrm{VLT}_{\mathrm{TT}}$ with representative state-of-the-art algorithms, as shown in Fig. 8a. It shows that our methods are more effective than other competing trackers on most attributes. Fig. 8b shows the ablation on different components of $\mathrm{VLT}_{\mathrm{SCAR}}$, which evidences that the integration of ModaMixer and ASS is necessary for a powerful VL tracker.



(a) Comparison with different trackers.

(b) Ablation on components of $\mathrm{VLT}_{\mathrm{SCAR}}$.

Figure 8: AUC scores of different attributes on the LaSOT.

References

[1] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 3

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009. 2, 3

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1, 2

[4] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. LaSOT: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 1, 5

[5] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. SiamCAR: Siamese fully convolutional classification and regression for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 3

[6] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, 2020. 2, 3

[7] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019. 1

[8] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 2017. 5

[9] Bin Yan, Houwen Peng, Kan Wu, Dong Wang, Jianlong Fu, and Huchuan Lu. Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 1, 3