Tracking Meets LoRA: Faster Training, Larger Model, Stronger Performance

Liting Lin[1], Heng Fan[2], Zhipeng Zhang[3], Yaowei Wang[1,†], Yong Xu[4,1], and Haibin Ling[5,†]

[1] Pengcheng Laboratory, China
[2] Department of CSE, University of North Texas, USA
[3] KargoBot, China
[4] School of Computer Science & Engineering, South China Univ. of Tech., China
[5] Department of Computer Science, Stony Brook University, USA
lt.lin@qq.com, heng.fan@unt.edu, zhipeng.zhang.cv@outlook.com
wangyw@pcl.ac.cn, yxu@scut.edu.cn, hling@cs.stonybrook.edu

Appendix

A More Implementation Details

This section provides additional implementation details for our proposed LoRAT tracker. The hyper-parameters used during training and testing are summarized in Table 1.

A.1 General Settings

Input Data. Following common practice in Siamese Trackers [2], the template image and search region image are cropped from input images. The template image has a background area factor of 2, while the search region image has a background area factor of 4 for -224 variants and 5 for -378 variants, as in [26].
Training. We apply common data augmentation strategies, including random color jittering and horizontal flipping, to the image pairs. Additionally, we employ the 3-Augment [22] strategy, with a minor modification ensuring that both images in a pair undergo the same transformation. The models are trained using AdamW [15] optimizer, with a learning rate of 1e-4 and a weight decay of 0.1. The learning rate is adjusted with cosine scheduler. The weight decay for all one-dimensional parameters is set to 0 according to [23]. To stabilize training, the model undergoes a 2-epoch linear warmup with an initial learning rate multiplier of 1e-3. Drop path [13] is applied to -L and -g variants with rates of 0.1 and 0.4, respectively. For full fine-tuning settings, we use layer-wise lr decay values [1] of 0.7/0.8/0.9 for -B/-L/-g variants respectively. The models are trained with automatic mixed precision enabled. The optimized CUDA kernels built into PyTorch, including torch.compile and memory-efficient attention [19], are used to reduce GPU memory usage and training time.
Inference. For all performance and efficiency evaluation experiments, automatic mixed precision and memory-efficient attention [19] are enabled.

† corresponding author

Table 1: Hyper-parameters used in our models.

Item	Value
overall:	
template area factor	2
search region area factor	4 (-224) / 5 (-378)
horizontal flip	0.5
color jitter	0.4
batch size	128
epochs	170 / 100 (GOT-10k)
optimizer	AdamW
lr	1e-4
weight decay	0.1
drop path	0.0 (B) / 0.1 (L) / 0.4 (g)
clip max norm	1.0
warmup epochs	2
warmup lr mult	1e-3
BCE loss coef	1.0
GIoU loss coef	1.0
hann window panelty	0.45
LoRA:	
r	64
full fine-tune:	
layer-wise lr decay	0.7 (B) / 0.8 (L) / 0.9 (g)

A.2 Loss Functions

Classification Loss. The classification branch in the head network aims to generate the classification confidence map $\mathbf{R}$. We adopt the binary cross-entropy (BCE) loss for classification, formulated as:

$$\begin{aligned} \mathcal{L}_{cls} &= \frac{1}{N_{\text{pos}}} \sum_{i,j} -(\bar{y}_{i,j} \log(\hat{y}_{i,j}) + (1-\bar{y}_{i,j}) \log(1-\hat{y}_{i,j})), \\ \text{where } \bar{y} &= \begin{cases} \text{IoU}(B_{i,j}^{\text{pred}}, B^{\text{gt}}) & y_{i,j}=1, \\ 0 & y_{i,j}=0, \end{cases} \end{aligned} \tag{1}$$

$y_{i,j}$ represents the ground-truth label of the location (i,j) on the classification confidence map $\mathbf{R}$ (1 for foreground, 0 for background), $\hat{y}_{i,j}$ represents the *IoU-aware classification score* [27], $B_{i,j}^{\text{pred}}$ is the target object bounding box predicted by the bounding box regression branch, and B^{gt} is the ground-truth bounding box, N_{pos} is the number of positive samples.

Table 2: Performance of our tracker on three additional benchmarks. Bold indicates the best results.

	LoRAT -B-224	LoRAT -B-378	LoRAT -L-224	LoRAT -L-378	LoRAT -g-224	LoRAT -g-378
OTB100 [25]	71.4	71.7	72.3	72.0	**72.6**	**72.6**
NFS [12]	64.0	66.6	66.0	66.7	65.0	**68.1**
UAV123 [16]	72.6	71.9	73.3	72.5	**74.2**	73.9

Bounding Box Regression Loss. We adopt the GIoU [21] loss for bounding box regression, formulated as:

$$\mathcal{L}_{box} = \frac{1}{N_{\text{pos}}} \sum_{i,j} \mathbb{1}_{y_{i,j}=1} \text{GIoU}(B^{\text{gt}}, B_{i,j}^{\text{pred}}) \tag{2}$$

where $\mathbb{1}_{y_{i,j}=1}$ is the indicator function, ensuring that only positive samples contribute to the loss.

The overall loss function is the sum of the classification loss and the bounding box regression loss, with equal weights (1.0) assigned to both loss functions.

B Results on Additional Benchmarks

In this section, we report the performance of the tracker on three additional benchmarks, including OTB100 [25], NFS [12] and UAV123 [16] in Table 2.

C More Ablation Studies

C.1 Performance under Different Pre-training Methods with LoRA Enabled

We conduct experiments on four ViT variants with various pre-training techniques, including MAE [9], CLIP [20], EVA-02 [8], and DINOv2 [17], to assess the impact of pre-training methods on performance. All variants are trained with default settings (with LoRA enabled). The input sizes of the template and search region images are adjusted according to the patch size of the ViT model. For models with patch size 14, the template size is set to [112 × 112], the search region size is set to [224 × 224], and the model is named with the postfix -224. For models with patch size 16, the template size is set to [128 × 128], the search region size is set to [256 × 256], and the model is named with the postfix -256.

From Table 3, we observe that the variants with the self-supervised DINOv2 backbone outperform other variants under the LoRA settings as well. The performance of the variants with EVA-02 pre-training is also competitive, indicating the effectiveness of its vision-language pre-training. However, the inference FPS of EVA-02 variants is significantly lower than other variants because of its several custom enhancement structures compared to the original plain ViT architecture.

Table 3: Ablation on the impact of ViT backbone with various pre-training methods.

	Variant	LaSOT [7] SUC	LaSOT [7] P	LaSOT$_{ext}$ [6] SUC	LaSOT$_{ext}$ [6] P	TNL2K [24] SUC	TNL2K [24] P	Speed fps
LoRA	*MAE* [9] *pre-training:*							
	B-256	69.2	73.9	49.6	56.4	55.9	57.6	215
	CLIP [20] *pre-training:*							
	B-256	69.2	73.9	49.3	55.4	56.5	58.0	203
	EVA-02 [8] *pre-training:*							
	B-224	70.5	75.9	47.3	53.6	56.1	57.3	66
	DINOv2 [17] *pre-training:*							
	B-224	71.7	77.3	50.3	57.1	58.8	61.3	209
Full Fine-tuning	*MAE* [9] *pre-training:*							
	B-256	69.9	74.8	47.9	54.7	55.2	56.0	215
	CLIP [20] *pre-training:*							
	B-256	70.1	75.0	48.7	55.0	56.8	58.3	203
	EVA-02 [8] *pre-training:*							
	B-224	70.3	75.6	49.9	56.5	57.2	59.2	66
	DINOv2 [17] *pre-training:*							
	B-224	70.9	76.2	50.0	57.1	58.1	60.1	209

Table 4: Performance, training time, max training GPU memory usage, inference FPS of various OSTrack variants. All OSTrack variants are implemented using official codes.

Variant	Pre-training	LaSOT [7] SUC	LaSOT [7] P	LaSOT$_{ext}$ [6] SUC	LaSOT$_{ext}$ [6] P	TNL2K [24] SUC	TNL2K [24] P	Time h	Memory GB	Speed *fps*
B-256	MAE	68.7	74.6	46.6	52.4	55.7	56.1	12.6	5.3	124
L-256	MAE	70.3	76.1	46.9	52.7	56.4	57.2	35.0	14.8	52
B-224	DINOv2	66.2	70.7	45.7	50.8	53.5	52.2	11.8	4.8	122
L-224	DINOv2	67.1	71.9	43.5	47.5	54.1	52.9	34.1	13.4	53

C.2 Detailed Comparison with OSTrack

We provide detailed performance and efficiency metrics for OSTrack variants in Table 4. Compared to our trackers, OSTrack requires significantly longer training times and higher GPU memory usage during training, while achieving lower performance.

C.3 Performance under Different Initialization Methods of LoRA Weights

Weight initialization is a crucial step for training neural networks. We compare three initialization methods: kaiming uniform (as in the official LoRA implementation), gaussian normal (as in the offical paper [10]), and truncated normal distribution (the initialization method used in BERT [4] and ViT [5]). The experiments are conducted on our LoRAT-L-224 variant. From the results in Table 5,

Table 5: Experiments on LoRA initialization methods. The base model is LoRAT-L-224.

Method	LaSOT [7]			LaSOT$_{\text{ext}}$ [6]			TNL2K [24]	
	SUC	P$_{\text{Norm}}$	P	SUC	P$_{\text{Norm}}$	P	SUC	P
Kaiming uniform	73.7	83.0	80.2	52.9	64.8	60.2	61.0	65.0
Gaussian normal	73.2	82.4	79.7	52.9	64.9	60.3	60.5	64.4
Truncated normal	74.2	83.6	80.9	52.8	64.7	60.0	61.1	65.1

Table 6: Ablation comparison of LoRA *vs*. other PEFT methods. The base model is LoRAT-L-224 (last line). LoRA$_{qv}$ indicates the LoRA module is added to the query and value projection matrix of the ViT backbone. LoRA$_{qkvo}$ indicates the LoRA module is added to the multi-head self-attention modules of the ViT backbone. LoRA$_{mlp}$ indicates the LoRA module is added to the MLP layers of the ViT backbone. LoRA$_{all}$ indicates the LoRA module is added to all linear layers of the ViT backbone.

PEFT	LaSOT [7]		LaSOT$_{\text{ext}}$ [6]		TNL2k [24]		#Params	Trainable	Speed
	SUC	P	SUC	P	SUC	P	M	%	FPS
-	73.0	79.3	52.6	59.8	60.8	64.3	307	100	119
Adapter [18]	55.0	52.9	38.3	39.8	45.4	39.2	314	3.3	78
VPT-Deep ($p = 5$) [11]	58.0	56.8	41.3	44.2	48.0	43.5	308	1.4	83
VPT-Deep ($p = 10$) [11]	59.3	58.9	42.1	45.4	49.4	45.5	308	1.5	83
$(\text{IA})^3$ [14]	59.5	60.0	43.1	47.5	49.4	47.0	308	1.4	119
LoRA$_{qv}$($r = 4$) [10]	66.2	70.0	48.5	54.9	55.7	57.1	308	1.5	119
LoRA$_{all}$($r = 16$) [10]	73.0	79.3	52.9	60.4	60.4	64.3	315	3.6	119
LoRA$_{qv}$($r = 64$) [10]	71.9	77.7	52.9	59.2	59.4	63.0	314	3.3	119
LoRA$_{qkvo}$($r = 64$) [10]	72.9	79.5	52.7	60.3	60.5	64.7	320	5.3	119
LoRA$_{mlp}$($r = 64$) [10]	72.8	78.9	52.2	59.6	60.3	63.9	323	6.2	119
LoRA$_{all}$($r = 64$) [10]	74.2	80.9	52.8	60.0	61.1	65.1	336	9.7	119

we observe that the truncated normal initialization achieves the best overall performance across the three benchmarks.

C.4 Comparison with other PEFT methods

We compare LoRA with other PEFT methods, including a representative of adatper-based PEFT method [18], a representative of prompt tuning-based PEFT method for vision model [11], and a recent zero inference computational cost PEFT method [14]. As shown in Table 6, LoRA outperforms all other PEFT methods on various settings.

Table 7: Ablation on attention mechanism. S-MAM: Replace the self-attention module in ViT backbone with Slimming Mixed Attention Module [3].

Variant	LaSOT [7]			$\text{LaSOT}_{\text{ext}}$ [6]			TNL2k [24]	
	SUC	P_{Norm}	P	SUC	P_{Norm}	P	SUC	P
LoRA:								
L-224	74.2	83.6	80.9	52.8	64.7	60.0	61.1	65.1
+ S-MAM [3]	73.5	82.6	80.1	52.4	64.1	59.8	61.0	64.9
Full Fine-tune:								
L-224	73.0	81.8	79.3	52.6	64.1	59.8	60.8	64.3
+ S-MAM [3]	73.0	82.1	79.6	52.5	64.0	59.9	60.0	63.2

C.5 Full self-attention *vs*. S-MAM

Slimming Mixed Attention Module (S-MAM) is a variant of self-attention proposed in MixViT [3], pruning the template-to-search area cross-attention. We compare the performance of full self-attention (performing self-attention operation on flattened template and search region tokens) and S-MAM in Table 7. Full self-attention outperforms S-MAM in both fine-tuning with LoRA and full fine-tuning settings. The performance of our tracker with S-MAM also gets better result on LoRA-based fine-tuning, verifying that our model design can adapt to other one-stream tracking models.

References

1. Bao, H., Dong, L., Piao, S., Wei, F.: BEit: BERT pre-training of image transformers. In: ICLR (2022) 1
2. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: Fully-convolutional siamese networks for object tracking. In: ECCV Workshops (2016) 1
3. Cui, Y., Jiang, C., Wu, G., Wang, L.: MixFormer: End-to-end tracking with iterative mixed attention. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–18 (2024) 6
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019) 4
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021) 4
6. Fan, H., Bai, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Huang, M., Liu, J., Xu, Y., et al.: LaSOT: A high-quality large-scale single object tracking benchmark. International Journal of Computer Vision **129**, 439–461 (2021) 4, 5, 6
7. Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., Ling, H.: LaSOT: A high-quality benchmark for large-scale single object tracking. In: CVPR (2019) 4, 5, 6

8. Fang, Y., Sun, Q., Wang, X., Huang, T., Wang, X., Cao, Y.: EVA-02: A visual representation for neon genesis. arXiv preprint arXiv:2303.11331 (2023) 3, 4
9. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: CVPR (2022) 3, 4
10. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: ICLR (2022) 4, 5
11. Jia, M., Tang, L., Chen, B.C., Cardie, C., Belongie, S., Hariharan, B., Lim, S.N.: Visual prompt tuning. In: ECCV (2022) 5
12. Kiani Galoogahi, H., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: ICCV (2017) 3
13. Larsson, G., Maire, M., Shakhnarovich, G.: FractalNet: Ultra-deep neural networks without residuals. In: ICLR (2016) 1
14. Liu, H., Tam, D., Mohammed, M., Mohta, J., Huang, T., Bansal, M., Raffel, C.: Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In: NeurIPS (2022) 5
15. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019) 1
16. Mueller, M., Smith, N., Ghanem, B.: A benchmark and simulator for uav tracking. In: ECCV (2016) 3
17. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: DINOv2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023) 3, 4
18. Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., Gurevych, I.: AdapterFusion: Non-destructive task composition for transfer learning. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume (2021) 5
19. Rabe, M.N., Staats, C.: Self-attention does not need $o(n^2)$ memory. arXiv preprint arXiv:2112.05682 (2021) 1
20. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML (2021) 3, 4
21. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: CVPR (2019) 3
22. Touvron, H., Cord, M., Jégou, H.: DeiT III: Revenge of the ViT. In: ECCV. Springer (2022) 1
23. Van Laarhoven, T.: L2 regularization versus batch and weight normalization. arXiv preprint arXiv:1706.05350 (2017) 1
24. Wang, X., Shu, X., Zhang, Z., Jiang, B., Wang, Y., Tian, Y., Wu, F.: Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark. In: CVPR (2021) 4, 5, 6
25. Wu, Y., Lim, J., Yang, M.H.: Object tracking benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(9), 1834–1848 (2015) 3
26. Ye, B., Chang, H., Ma, B., Shan, S., Chen, X.: Joint feature learning and relation modeling for tracking: A one-stream framework. In: ECCV (2022) 1
27. Zhang, H., Wang, Y., Dayoub, F., Sünderhauf, N.: VarifocalNet: An iou-aware dense object detector. In: CVPR (2021) 2