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

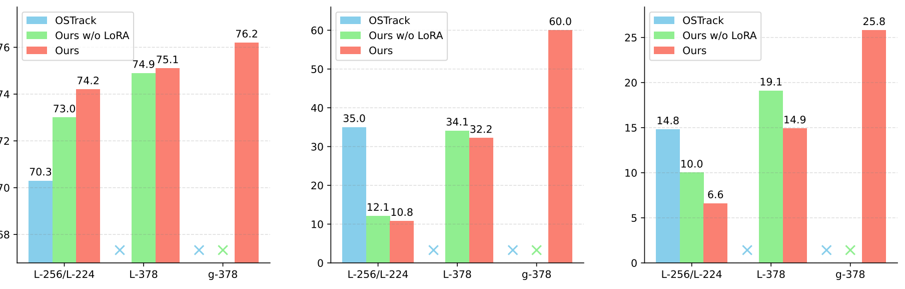
Liting Lin[1], Heng Fan[2], Zhipeng Zhang[3], Yaowei Wang[1,†], Yong Xu[4,1], and Haibin Ling[5,†]

[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

Abstract. Motivated by the Parameter-Efficient Fine-Tuning (PEFT) in large language models, we propose **LoRAT**, a method that unveils the power of larger Vision Transformers (ViT) for tracking within laboratory-level resources. The essence of our work lies in adapting LoRA, a technique that fine-tunes a small subset of model parameters without adding inference latency, to the domain of visual tracking. However, unique challenges and potential domain gaps make this transfer not as easy as the first intuition. Firstly, a transformer-based tracker constructs unshared position embedding for template and search image. This poses a challenge for the transfer of LoRA, usually requiring consistency in the design when applied to the pre-trained backbone, to downstream tasks. Secondly, the inductive bias inherent in convolutional heads diminishes the effectiveness of parameter-efficient fine-tuning in tracking models. To overcome these limitations, we first decouple the position embeddings in transformer-based trackers into shared spatial ones and independent type ones. The shared embeddings, which describe the absolute coordinates of multi-resolution images (namely, the template and search images), are inherited from the pre-trained backbones. In contrast, the independent embeddings indicate the sources of each token and are learned from scratch. Furthermore, we design an anchor-free head solely based on a multilayer perceptron (MLP) to adapt PETR, enabling better performance with less computational overhead. With our design, 1) it becomes practical to train trackers with the ViT-g backbone on GPUs with only memory of 25.8GB (batch size of 16); 2) we reduce the training time of the L-224 variant from 35.0 to 10.8 GPU hours; 3) we improve the LaSOT SUC score from 0.703 to 0.742 with the L-224 variant; 4) we fast the inference speed of the L-224 variant from 52 to 119 FPS. Code and models are available at `https://github.com/LitingLin/LoRAT`.

Keywords: visual object tracking, LoRA, parameter-efficient fine-tuning

† corresponding author



(a) SUC on LaSOT **(b)** Training Time (h) **(c)** Training Memory (GB)

Fig. 1: Comparison of tracking models on performance and training efficiency. "×" indicates failure to train due to insufficient memory. Best viewed in color for all figures.

1 Introduction

Visual tracking, aiming to continuously locate the target object from a sequence given its initial state, is one of the most fundamental problems in computer vision and has been extensively explored in the past decades [1, 2, 10, 18, 28, 34, 66]. In recent years, the tracking community has witnessed considerable advancements [7, 38, 59, 60] by leveraging the Transformer architecture [15, 53], primarily owing to its exceptional ability in capturing long-range dependencies and flexibility in accommodating a wide variety of pre-training models [15, 26, 27, 43]. While Transformer contributes to significant improvements in tracking performance, their usage is *resource-intensive*. In particular, most performance-oriented Transformer trackers require extensive computational resources, typically involving multiple high-end data-center GPUs and prolonged training periods. Figure 1 illustrates the escalating resource requirements for training large-scale trackers, which are becoming increasingly unaffordable for most researchers. The largest model in visual tracking to date, SeqTrack-L384 [6], employs a ViT-L backbone, yet lags behind the scale of available pre-trained ViT models. In this work, we aim to develop methods for training large-scale trackers with manageable resource requirements, thereby making more advanced large-scale models accessible to a broader research community and accelerating the evolution of visual tracking.

To achieve the goal, we draw inspiration from the advancements in Parameter-Efficient Fine-Tuning (PEFT) for large language models (LLMs), which are the first to encounter the prohibitive costs associated with full fine-tuning of large-scale models [14, 37]. PEFT methods fine-tune a relatively small number of parameters while keeping the rest frozen, significantly reducing computational and storage costs. While these methods have shown effectiveness in language models [63] and vision-language models [45, 64], their application to visual tracking, which will present unique challenges and potential domain gaps in our study, has been under-explored. This work is the first to investigate the application of PEFT in visual tracking, assessing its viability, and addressing domain-specific challenges. Among various PEFT research streams, we select Low-Rank Adaptation (LoRA) [30]. LoRA achieves parameter-efficient fine-tuning by adding trainable rank decomposition matrices to certain dense layers of the frozen model, providing competitive performance compared to other PEFT approaches such as adapters [29,40,48] or prompt tuning [33,35,42], without additional inference latency.

However, unlike language models, the downstream task adaptation for vision models usually requires careful design upon the pre-trained backbone network [3, 24]. Hence, we need a basic tracking framework as the starting point. We employ the one-stream framework [60] as our baseline, because it brings minimal changes to the pre-trained ViT model, requiring only an image-pair capable positional encoding module and a head network. This architecture minimizes the extra parameters needed for training, aligning with the purpose of parameter-efficient fine-tuning. Additionally, the encoder-only Transformer architecture has been demonstrated to be more efficient in training trackers [38].

Yet, it is *not* as easy as our first intuition to apply LoRA to the one-stream tracking architecture, mainly due to two key issues below: ① *Ineffective design of positional encoding module.* Most Transformer-based trackers adopt separate positional encodings for template and search region tokens. Albeit effective in full fine-tuning [60], such design is not well aligned with LoRA-style PEFT and thus results in suboptimal performance. We argue that this ineffectiveness comes from the disruption to the original structure of the pre-trained model, which is critical to preserve in PEFT such as LoRA [30]. ② *Inductive biases.* We empirically identify the convolutional head in the one-stream tracker [60] as a bottleneck in achieving convergence with LoRA. We argue that this is caused by strong inductive biases in convolutional network [15], making it hard to adapt the pre-trained model by fine-tuning a small subset of parameters using LoRA.

Our solution. To overcome the above issues in employing LoRA to one-stream tracker [60], we present two embarrassingly *simple but effective* designs. Specifically, in addressing issue ①, we draw inspiration from the well-known BERT [13] and introduce the token type embedding in input embedding. Concretely, we assign specific types to tokens of template and search region. By doing so, we can directly leverage existing positional encoding modules of pre-trained ViT models without disrupting their original structure. Considering different resolutions of template and search regions, we further devise a multi-resolution absolute positional embedding adaptation strategy, enabling the support of multi-resolution inputs to pre-trained ViT models for better performance. To deal with issue ②, we propose a multilayer perceptron (MLP)-based anchor-free head network, instead of convolutional head network as in [60], for target classification and bounding box regression. Such a design effectively avoids the inductive biases when applying LoRA to our baseline one-stream tracker, enabling better performance with less computational overhead as shown in our experiments.

It is worth noting that, despite the simplicity in implementation, our solution is specifically devised to better employ LoRA for improving visual tracking. Particularly, with our two designs, we propose LoRAT by applying LoRA to the one-stream OSTrack using various pre-trained ViTs, and extensively assess its performance on five large-scale benchmarks, including LaSOT [17], LaSOT$_{\text{ext}}$ [16], TrackingNet [46], GOT-10k [31], and TNL2K [55]. Our best variant LoRAT-g-378 sets a new record on LaSOT with 0.762 SUC score when equipped with ViT-g [47], while the lightest variant LoRAT-B-224 still achieves a comparable SUC score of 0.717 on LaSOT yet running at 209 FPS. In addition, the required training resources and time are manageable. The training time of different variants, from the light LoRAT-B-224 to strong LoRAT-g-378, ranges from 5.9 to 60 hours using 8 Nvidia V100 GPUs. Notably, our base LoRAT-B-224 can even be trained on a single customer-grade Nvidia RTX 4090 GPU within 11 hours with 605 inference FPS.

In summary, we make the following **contributions**: ♠ We, for the first time, propose leveraging LoRA to develop efficient and resource-friendly generic object tracking; ♥ We propose two simple yet effective designs to enable better adaption of LoRA for the tracking task; ♣ Our tracker LoRAT, by marrying LoRA to a one-stream tracker, achieves new state-of-the-art performance on multiple benchmarks with reasonable resource requirements.

2 Related Work

2.1 Paradigms of Transformer Tracker

Visual tracking has witnessed significant progress in past decades [2,6–8,11,34, 38,56,59,60,66], particularly with the integration of Transformer architectures. Early exploration focuses on utilizing Transformer to replace or enhance components in traditional tracking frameworks. The work of [54] proposes the first Transformer tracker by using Transformer for feature fusion in Siamese [1] and DCF [2,28] pipelines, respectively. TransT [7] leverages self-attention for feature enhancement and cross-attention for feature fusion. Subsequent research exploits the Transformer's modeling flexibility for improved template-search region relationship representation. The approaches of [23,58] explore feature interaction within the backbone network. Stark [59] and SwinTrack [38] explore joint feature enhancement and feature interaction via Transformer but still separate backbone networks for feature extraction on template and search region.

More recently, the one-stream tracking framework emerges as a significant evolution, with Mixformer [8], SimTrack [5], and OSTrack [60] serving as the vanguard. This paradigm allows for earlier interaction between the template and search region, facilitating joint feature extraction and fusion. The self-attention mechanism of the Transformer architecture plays a crucial role in this context, enhancing the model's ability to capture complex dependencies. Most subsequent Transformer trackers follow this paradigm. GRM [21] introduces adaptive token division to enable more flexible relation modeling. DropTrack [57] adaptively performs spatial-attention dropout in the frame reconstruction to facilitate temporal correspondence learning in videos. SeqTrack [6] and ARTrack [56] add a Transformer decoder on the top a one-stream encoder to predict object bounding boxes in an autoregressive fashion.

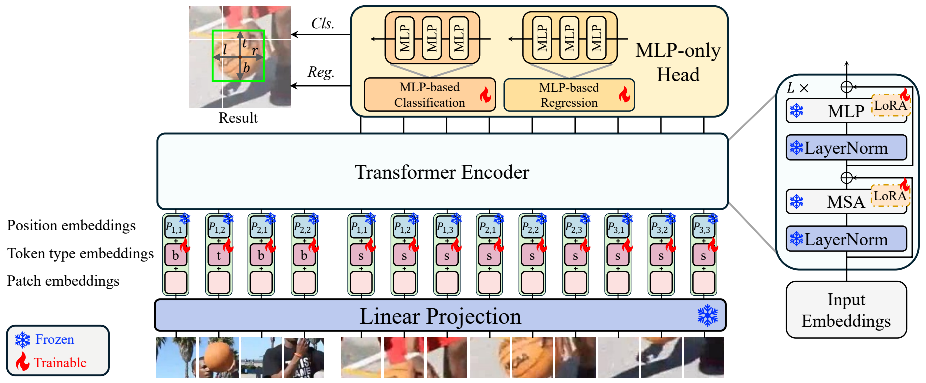


Fig. 2: Architecture of LoRAT. The template and search region are first split and then projected as patch embeddings. Patch embeddings are added with shared position embeddings and token type embeddings as the input embeddings, which are then fed into Transformer encoder for joint feature extraction and fusion. The resulting representations are fed to the MLP-only head network for target classification and anchor-free based bounding box regression. Most network components from the pre-trained ViT model are frozen during training, except for LoRA modules applied on the linear layers in the Transformer encoder, the token type embeddings, and the head network.

2.2 Parameter Efficient Fine Tuning

Parameter Efficient Fine Tuning (PEFT) is a solution to the challenge of fine-tuning large models, which becomes impractical due to high costs. PEFT enables efficient fine-tuning by adjusting a significantly smaller subset of parameters instead of the full model. PEFT methods primarily fall into two categories: prompt tuning-based [33, 35, 41, 42] and adapter-based approaches [29, 40, 48]. Prompt tuning-based approaches utilize continuous vectors as a part of input prompts, which are optimized through gradient descent during fine-tuning. Adapter-based approaches, on the other hand, introduce additional learnable modules either within or parallel to specific network components. One notable family of adapter-based methods is LoRA [30] and its variants [12, 25, 62]. These methods apply low-rank matrices to approximate weight changes during fine-tuning and can be merged with pre-trained weights prior to inference, making them particularly efficient as they do not add any extra inference burden.

3 Adapting ViT for LoRA-friendly Tracker

This section meticulously outlines our approach, beginning with an elucidation of our baseline, the one-stream tracking framework. We then introduce the concept of Low-Rank Adaptation (LoRA), a pivotal technique for the efficient tuning of large-scale models for specialized tasks, emphasizing its role in enhancing model adaptability with minimal computational overhead. Following this, we explore the architectural adjustments necessary to render our model compatible with

LoRA, including modifications to the model's input embedding and an MLP-only head network. An overview of our tracker is shown in Fig. 2.

3.1 Preliminaries

One-Stream Tracker. Our approach builds upon the one-stream tracking framework [60], which utilizes the Transformer's multi-head attention mechanism [53] to attend to different representation subspaces at various positions simultaneously, enabling joint feature extraction and fusion.

One-stream trackers follow the Siamese tracker framework [1], requiring a template image $T \in \mathbb{R}^{H_T \times W_T \times 3}$ and a search region image $S \in \mathbb{R}^{H_S \times W_S \times 3}$ as inputs. The template image T and the search region image S are first projected by a learnable linear projection layer to obtain the template token embeddings and the search region token embeddings. Assuming the image patch resolution is (P, P), the template image is divided into $N_T = N_T^H \times N_T^W$ image patches, and the search region image is divided into $N_S = N_S^H \times N_S^W$ image patches, where $N_T^H, N_T^W, N_S^H, N_S^W = H_T, W_T, H_S, W_S / P$. The resulting patch embeddings are the template token embeddings $\mathbf{T} \in \mathbb{R}^{N_T \times d}$ and the search region embeddings $\mathbf{S} \in \mathbb{R}^{N_S \times d}$, where d represents the hidden dimension of the network.

Subsequently, two positional embeddings, $\mathbf{E}_T$ and $\mathbf{E}_S$, are added to these token embeddings respectively, obtaining the template input embeddings $\mathbf{t}_0 \in \mathbb{R}^{N_T \times d}$ and the search region input embeddings $\mathbf{s}_0 \in \mathbb{R}^{N_S \times d}$:

$$\mathbf{t}_0 = \mathbf{T} + \mathbf{E}_T, \quad \mathbf{s}_0 = \mathbf{S} + \mathbf{E}_S. \tag{1}$$

Then, the template input embeddings $\mathbf{t}_0$ and the search region input embeddings $\mathbf{s}_0$ are concatenated as $\mathbf{z}_0$ and being fed into an L-layer Transformer encoder. The encoder comprises multi-head self-attention mechanisms and feed-forward neural networks. The output of the encoder, $\mathbf{z}_L \in \mathbb{R}^{(N_T + N_S) \times d}$, represents the joint feature representation of the template image and the search region image. Finally, a de-concatenation operation is performed to obtain the feature representations of the template image and the search region, denoted as $\mathcal{T}$ and $\mathcal{S}$, respectively. The search region feature representation $\mathcal{S}$ is then fed into the head network for further processing. The whole process is encapsulated by the following equations:

$$\begin{aligned}
\mathbf{z}_0 &= \mathrm{Concat}(\mathbf{t}_0, \mathbf{s}_0), && && (2)\\
\mathbf{z}'_\ell &= \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, && \ell = 1 \ldots L, && (3)\\
\mathbf{z}_\ell &= \mathrm{MLP}(\mathrm{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, && \ell = 1 \ldots L, && (4)\\
\mathcal{T}, \mathcal{S} &= \mathrm{DeConcat}(\mathbf{z}_L). && && (5)
\end{aligned}$$

Low-Rank Adaptation. Low-Rank Adaptation (LoRA) [30] is a technique for the efficient fine-tuning of large-scale pre-trained models. Typically, these models are initially trained on extensive datasets, which provides a broad foundational knowledge. To adapt the models to specific downstream tasks, models are fine-tuned with domain-specific data. Traditional fine-tuning is computationally intensive, as it involves a full forward and backward pass through the network, requiring significant computational resources and memory.

LoRA addresses these issues by proposing a low-rank matrix decomposition to represent weight updates, which can hugely reduce the number of trainable weights and memory requirements. The principle behind LoRA is that while neural networks typically have full-rank weight matrices during pre-training, the updates required for fine-tuning to a specific task have a low "intrinsic dimension". Remarkably, in many applications, LoRA shows a comparable performance to full fine-tuning of the network.

In the context of fine-tuning weight updates ($\varDelta\varTheta$), LoRA employs considerably smaller matrices ($\varDelta\varPhi$), thus dramatically reducing the parameter space that needs to be learned. For a weight matrix update in the network $\varDelta\theta_i$, LoRA approximates it using:

$$\varDelta\theta_i \approx \varDelta\phi_i = BA$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, with the rank $r \ll \min(d, k)$. This significantly reduces the learning complexity from learning $d \times k$ parameters to learning only $(d + k) \times r$ parameters.

Importantly, adopting LoRA does not increase the inference latency of the model. After fine-tuning, the model's weights are updated by adding the low-rank approximations to the original weights. This approach simplifies the deployment of multiple task-specific models on top of a single large model because the updates ($\varDelta\varPhi$) are much smaller than the original weights ($\varDelta\varTheta$).

3.2 LoRA-friendly Model Design

Decoupled Input Embedding. ViT models require positional encodings to inject positional information. Existing one-stream trackers use separate positional embeddings for template and search region tokens, which disrupt the original structure of the pre-trained ViT model, leading to ineffectiveness in PEFT. We explore multi-resolution positional embedding and token type embedding to address this issue.

Token type embedding. We incorporate token type embedding (also called segment embedding), which is originally proposed in the BERT [13] language model, into our tracker, assigning unique embeddings for template and search region tokens. This approach decouples token type identification from positional embeddings, improving parameter efficiency and flexibility.

We also explore more usage of token type embedding. The cropping strategy [1] commonly used by Siamese trackers to generate tracking templates has several issues that may confuse the tracker: the aspect ratio of the target objects in the templates is variable, some target objects do not have clear semantic meanings, and some target objects in the templates can not be not clearly distinguishable from the background. Token type embedding can help alleviate these issues by explicitly annotating foreground and background parts.

Positional embedding. Most ViT models adapt the 1D absolute positional embedding [15], which only works for fixed resolutions. Meanwhile, the one-stream trackers usually utilize images of two different sizes as inputs. According to the PEFT-friendly model design principle, we have to share the positional embedding across two images. We explore two strategies to reuse existing absolute positional embeddings for different input sizes. Both strategies regard the original 1D absolute embedding in a 2D view. Suppose the original 1D absolute positional embedding is P, the 2D view of P is $P_{2d} = [p_{1,1}, ..., p_{1,w}; p_{2,1}, ..., p_{2,w}; p_{h,1}, ..., p_{h,w}]$. For convenience, we assume that the resolution of search region image is the native resolution of ViT model, thus P_{2d} can be directly applied to search region tokens, *i.e.* the search region positional embedding $P_x = P_{2d}$.

The first approach regards absolute positional embedding as continuous spatial position embedding. In this way, we can direct interpolate the P_x to the expected size of P_z. In this way, $P_z = \text{interpolate}(P_x)$. We call this approach as *interpolation-based* adaptation.

The second approach regards absolute positional embedding as discrete patch index embedding. Thus, the smaller positional embedding for template tokens P_z can be directly taken as a sub-matrix of the P_x, which is formed by taking a block of the entries of the size of template patches ($N_T^H \times N_T^W$) from the top left corner of the P_x. In this way, $P_z = [p_{1,1}, ..., p_{1,N_T^W}; p_{2,1}, ..., p_{2,N_T^W}; p_{N_T^H,1}, ..., p_{N_T^H,N_T^W}]$. We call this approach as *slicing-based* adaptation.

For other commonly used positional encoding schemes, like sinusoidal positional encoding [15], relative positional embedding [50] or Rotary Positional Embedding [51], they do not target on specific resolution, so apply them on template tokens and search region tokens individually.

The final input embeddings are constructed by adding the positional embedding and token type embedding to the patch embeddings:

$$\mathbf{E}_T^{(i,j)} = \begin{cases} \mathbf{E}_{pos}^{(i,j)} + \mathbf{E}_{type}^{T_o}, & \text{if } \mathbf{T}^{(i,j)} \text{ belongs target,} \\ \mathbf{E}_{pos}^{(i,j)} + \mathbf{E}_{type}^{T_b}, & \text{otherwise} \end{cases} \tag{6}$$

$$\mathbf{E}_S^{(i,j)} = \mathbf{E}_{pos}^{(i,j)} + \mathbf{E}_{type}^{S}, \tag{7}$$

where $\mathbf{E}_{type}^{T_o} \in \mathbb{R}^d$, $\mathbf{E}_{type}^{T_b} \in \mathbb{R}^d$, and $\mathbf{E}_{type}^{S} \in \mathbb{R}^d$ are the token type embedding of the template foreground (target object) tokens, the template background tokens, and search region tokens respectively. **Notice**, the positional encoding $\mathbf{E}_{pos}^{(i,j)}$ is **shared** between template and search region.

MLP-only Head Network. To alleviate potential *inductive biases* brought by the convolutional head, such as locality assumptions on data structure [15], which may hinder convergence with LoRA-based fine-tuning, we employ a multi-layer perceptron (MLP) only head.

The head network consists of two branches for target classification and bounding box regression, each implemented as a three-layer perceptron. Both branches take search area tokens $\mathcal{S}$ from the Transformer encoder output as input. The head network applies a center-based anchor-free [52] style. The target classification branch predicts a classification confidence map $\mathbf{R} \in (0,1)^{N_S^H \times N_S^W}$, estimating the IoU between the predicted bounding box and the ground-truth (the estimation value is targeted to 0 during training if the object does not fall in the current anchor). The bounding box regression branch predicts a bounding box map $\mathbf{B} \in (0,1)^{N_S^H \times N_S^W \times 4}$. The anchor with the maximum value on the classification confidence map is considered the responsible anchor. For each anchor center point (x_i, y_j), the bounding box regression branch predicts offsets $(l_{i,j}, t_{i,j}, r_{i,j}, b_{i,j})$. The final bounding box is obtained by $(x_i - l_{i,j}, y_j - t_{i,j}, x_i + r_{i,j}, y_j + b_{i,j})$.

4 Experiments

4.1 Experimental Setup

LoRAT is trained and evaluated on 8 $\times$ NVIDIA V100 GPUs. This section reports the main experimental setup, with more details in **supplementary material**.

Model. We present six variants of LoRAT, namely B-224, B-378, L-224, L-378, g-224, and g-378. These variants are based on three sizes of the Vision Transformer (ViT): ViT-B [15], ViT-L [15], and ViT-g [47], serving as the backbone networks. Each size is configured with two distinct input resolutions. Specifically, for the -224 variants, the template size is set to $[112 \times 112]$, and the search region size is set to $[224 \times 224]$; For the -378 variants, the template size is set to $[196 \times 196]$, the search region size is set to $[378 \times 378]$. By default, all variants employ the DINO v2 [47] pre-trained weights.

Training. The training splits of LaSOT [17], TrackingNet [46], GOT-10k [31] (1k sequences removed for fair comparison following [59, 60], as they overlap with videos used in [32]), and COCO 2017 [39] are used to train the trackers. Additionally, the evaluation results on the GOT-10k test split for trackers are trained only with the GOT-10k training split, following the protocol described in [31]. We train the trackers with 170 epochs, each epoch containing 131,072 image pairs. For the GOT-10k benchmark, we reduce the training epochs to 100 to prevent over-fitting. Each GPU holds 16 samples for an iteration, resulting in a batch size of 128. Following the suggestion in [12], LoRA is applied on all linear layers of ViT backbone, including four projection matrices in the self-attention module and two projection matrices in the MLP module. The rank r of LoRA is set to 64 for all variants. All LoRA linear layers are initialized as the same way as BERT [13] linear layer, truncated normal distribution with std 0.02.

Inference. LoRAT follows conventional steps of Siamese tracking [1] during the inference process: the template is cropped from the first frame of the video sequence, with target object as center; the search area is cropped from the current tracking frame, with the image center as the predicted target center position from the previous frame. To exploit the location prior during tracking, a Hanning window penalty to the classification response map $\mathbf{R}$ output by the head network.

Table 1: Benchmarking our tracker on five large-scale challenging datasets. For GOT-10k evaluation, all the methods follow the one-shot protocol, training only on the train split of GOT-10k. Bold indicates the best results.

Tracker	LaSOT [17]			LaSOT$_{\text{ext}}$ [16]			TrackingNet [46]			GOT-10k [31]			TNL2K [55]	
	SUC	P$_{\text{Norm}}$	P	SUC	P$_{\text{Norm}}$	P	SUC	P$_{\text{Norm}}$	P	AO	SR$_{0.5}$	SR$_{0.75}$	SUC	P
TransT [7]	64.9	73.8	69.0	-	-	-	81.4	86.7	80.3	67.1	76.8	60.9	50.7	51.7
AutoMatch [65]	58.3	-	59.9	37.6	-	43.0	76.0	-	72.6	65.2	76.6	54.3	47.2	43.5
STARK [59]	67.1	77.0	-	-	-	-	82.0	86.9	-	68.8	78.1	64.1	-	-
KeepTrack [44]	67.1	77.2	70.2	48.2	-	-	-	-	-	-	-	-	-	-
MixFormer [8]	70.1	79.9	76.3	-	-	-	83.9	88.9	83.1	-	-	-	-	-
SBT [58]	66.7	-	71.1	-	-	-	-	-	-	70.4	80.8	64.7	-	-
AiATrack [20]	69.0	79.4	73.8	47.7	55.6	55.4	-	-	-	69.6	80.0	63.2	-	-
SimTrack [5]	70.5	79.7	-	-	-	-	-	-	-	69.8	78.8	66.0	55.6	55.7
OSTrack [60]	71.1	81.1	77.6	50.5	61.3	57.6	83.9	88.5	83.2	73.7	83.2	70.8	55.9	56.7
SwinTrack [38]	71.3	-	76.5	49.1	-	55.6	84.0	-	82.8	72.4	80.5	67.8	55.9	57.1
DropTrack [57]	71.8	81.8	78.1	52.7	63.9	60.2	-	-	-	75.9	86.8	72.0	56.9	57.9
SeqTrack [6]	72.5	81.5	79.3	50.7	61.6	57.5	85.5	89.8	85.8	74.8	81.9	72.2	57.8	-
ARTrack [56]	73.1	82.2	80.3	52.8	62.9	59.7	85.6	89.6	86.0	78.5	87.4	77.8	60.3	-
CiteTracker [36]	69.7	78.6	75.7	-	-	-	84.5	89.0	84.2	74.7	84.3	73.0	57.7	59.6
ROMTrack [4]	71.4	81.4	78.2	51.3	62.4	58.6	84.1	89.0	83.7	74.2	84.3	72.4	-	-
MixViT [9]	72.4	82.2	80.1	-	-	-	85.4	**90.2**	85.7	75.7	85.3	75.1	-	-
LoRAT-B-224	71.7	80.9	77.3	50.3	61.6	57.1	83.5	87.9	82.1	72.1	81.8	70.7	58.8	61.3
LoRAT-B-378	72.9	81.9	79.1	53.1	64.8	60.6	84.2	88.4	83.0	73.7	82.6	72.9	59.9	63.7
LoRAT-L-224	74.2	83.6	80.9	52.8	64.7	60.0	85.0	89.5	84.4	75.7	84.9	75.0	61.1	65.1
LoRAT-L-378	75.1	84.1	82.0	**56.6**	**69.0**	**65.1**	85.6	89.7	85.4	77.5	86.2	78.1	62.3	67.0
LoRAT-g-224	74.9	84.5	82.3	53.3	65.4	61.1	85.2	89.8	85.1	77.7	87.7	77.7	61.8	66.6
LoRAT-g-378	**76.2**	**85.3**	**83.5**	56.5	**69.0**	64.9	**86.0**	**90.2**	**86.1**	**78.9**	**87.8**	**80.7**	**62.7**	**67.8**

4.2 Quantitative Comparison with the State-of-the-Art Models

In this section, we present a comprehensive evaluation of our proposed LoRAT, benchmarking it against the latest Transformer-based tracking models across five datasets. We follow the official evaluation protocols to ensure fair and accurate comparison. The results are detailed in Table 1. Additionally, we present efficiency comparison with several SOTA Transformer trackers in Table 2.

LaSOT [17] is a large scale benchmark containing 280 test videos. From Table 1, our smallest variant LoRAT-B-224 achieves 0.717 SUC score, already providing a competitive performance. LoRAT-L-224 provides a SUC score of 0.742, beats all other SOTA models. With the current largest backbone network applied in visual tracking, LoRAT-g-384 achieves a breakthrough SUC score of 0.762, significantly outperforming the previous state-of-the-art model ARTrack [56] by 3.1%.

LaSOT$_{\text{ext}}$ [16] is an extension of LaSOT with 150 additional highly challenging sequences. The results are presented in Table 1. LoRAT-B-378 obtains 0.531 SUC score, outperforms the prior best, ARTrack [56], by 0.4% with a relatively smaller backbone network. LoRAT-L-378 incorporating the ViT-L [47] backbone, attains a breakthrough SUC score of 0.566. Our largest variant, LoRAT-g-378, achieves a comparable SUC score of 0.565.

Tracker	Speed (*fps*)	MACs (G)	#Params (M)
SwinTrack-B-384 [38]	45	69.7	91
OSTrack-256 [60]	130	21.5	-
OSTrack-384 [60]	68	48.3	-
SeqTrack-B256 [6]	38	66	89
SeqTrack-L384 [6]	6	524	309
LoRAT-B-224	209	30	99 (11, 2)
LoRAT-B-378	151	97	99 (11, 2)
LoRAT-L-224	119	103	336 (28, 4)
LoRAT-L-378	63	325	336 (28, 4)
LoRAT-g-224	50	378	1216 (71, 9)
LoRAT-g-378	20	1161	1216 (71, 9)

Table 2: Comparison on efficiency with state-of-the-art Transformer trackers. The values in parentheses for the # params of our trackers represent LoRA and extra components (token type embedding and head), respectively. The speed of all trackers was re-evaluated on our machine.

TrackingNet [46] is a large-scale object tracking benchmark. The test split of TrackingNet includes 511 video sequences with first frame annotated. The results in Table 1 demonstrate that all variants of our tracker obtain competitive SUC scores from 0.835 to 0.860. The largest variant LoRAT-g-378 gets the best 0.860 SUC score, outperforming the previous best model by 0.4%.
GOT-10k [31] provides 180 videos for testing. Following the official protocol, we report the results on the models trained only on GOT-10k train split. As reported in Table 1, on the GOT-10k benchmark, the model size greatly impact the final results, more than the input resolution. our LoRAT-g-378 gets a new state-of-the-art AO score of 78.9%.
TNL2k [55] is a recent tracking dataset with 700 test videos. As shown in Table 1, LoRAT-L-378 achieves 2.0% gain over ARTrack [56] at 0.623 SUC score. LoRAT-g-378 achieves the best performance with a SUC score of 0.627.
Efficiency Comparison. We compare LoRAT with other Transformer trackers on efficiency in Table 2, including running speed, floating point operations and number of parameters. Thanks to the highly efficient one-stream architecture, zero inference cost of LoRA, our tracker has a quite impressive running speed to MACs ratio. Our lightest variant LoRAT-B-224, while delivering SOTA performance, operates at an impressive 209 frames per second (fps). The LoRAT-L-378 variant, defeat other trackers on most datasets, but still have a 63 fps. Remarkably, the LoRAT-g-378 variant, leveraging the largest ViT backbone to date, maintains a practical speed of 20 fps. Besides, our tracker also exhibits efficiency in storage requirements. For instance, the visual tracking adaptation module in the LoRAT-g-378 variant, which includes LoRA, token type embedding, and the head, requires only 80 million parameters. This is substantially less than the 1.1 billion parameters of the ViT-g model.

4.3 Ablation Experiments

To gain deep insights into LoRAT, we perform detailed ablation studies on three datasets, including LaSOT [17], LaSOT$_{\text{ext}}$ [16], and TNL2k [55].
Performance improvement of LoRA. We compare the performance of LoRAT with full-finetune variants on 4 relatively small variants, as shown in Ta-

Table 3: Ablation on the performance improvement of LoRA on our tracker. All variants are trained with full parameter fine-tuning. The results are compared with our default settings, *i.e.* the variants trained with LoRA.

Variant	LaSOT [17]		LaSOT$_{\text{ext}}$ [16]		TNL2K [55]	
	SUC	P	SUC	P	SUC	P
B-224	70.9 (↓**0.8**)	76.2 (↓**1.1**)	50.0 (↓**0.3**)	57.1 (↓**0.0**)	58.1 (↓**0.7**)	60.1 (↓**1.2**)
B-378	73.2 (↑**0.3**)	78.8 (↓**0.3**)	52.9 (↓**0.2**)	60.5 (↓**0.1**)	60.0 (↑**0.1**)	63.3 (↓**0.4**)
L-224	73.0 (↓**1.2**)	79.3 (↓**1.6**)	52.6 (↓**0.2**)	59.8 (↓**0.2**)	60.8 (↓**0.3**)	64.3 (↓**0.8**)
L-378	74.9 (↓**0.2**)	81.8 (↓**0.2**)	55.2 (↓**1.4**)	63.1 (↓**2.0**)	61.7 (↓**0.6**)	65.7 (↓**1.3**)

ble 3. LoRA improves the performance of the tracker on all variants, indicating that our model design correctly unleashes the power of parameter-efficient fine-tuning, promoting the performance of visual tracking through mitigating catastrophic forgetting [22] during fine-tuning [14, 61].

Input embedding. This ablation verifies the effectiveness of the proposed *decoupled* input embedding scheme. The experiments are conducted on the ViT-L [15] backbone with two input resolution settings. From the results in Table 4, we observe that: 1) Freezing positional embedding shows slightly better performance than the unfreezing one (⑤ *vs.*①). 2) Adopting "shared positional embedding", "token type embedding" and "foreground indicating embedding" (our model) respectively outperforms the "separated positional embedding", "non-token embedding" and "non-foreground indicating embedding" version (⑤-⑧). 3) Unfreezing the position embedding will degrade the effectiveness of our proposed foreground indicating embedding (④ *vs.*③, ⑧ *vs.*⑦) 4) With a higher resolution of input images, foreground indicating embedding significantly boost the performance of our tracker (⑩ *vs.*⑨).

MLP-only head network. We evaluate the performance of MLP-only and convolutional head networks under different fine-tuning settings. As shown in Table 5, the OSTrack [60] convolutional head yields similar competitive results to the MLP-only head in the full fine-tuning setting but fails to converge in the LoRA-based fine-tuning setting. We attribute this is due to the inductive biases in the convolution layer, which may not satisfy the low "intrinsic dimension" assumption required by low-rank adaptation. Additionally, an MLP-only head offers computational efficiency.

Aligned comparison with OSTrack. We provide aligned performance comparison with OSTrack [60] by replacing only the backbone. For fair comparison, OSTrack is trained without the early candidate elimination module. Input resolution of variants is adjusted according to the patch size of ViT. As shown in Table 6, all variants of OSTrack are inferior to our base model. Besides, the performance of the DINOv2 backbone on OSTrack lags behind the default setting (MAE backbone). These results demonstrate the effectiveness of our model design.

Table 4: Ablation on the input embedding configurations. All experiments are conducted on ViT-L [15] backbone with two input resolution settings and with LoRA-based fine-tuning. We examine five factors: 1) Freezing positional embeddings during training; 2) Sharing positional embeddings between template and search region images; 3) Using token type embeddings 4) Incorporating foreground object indicator embeddings; 5) Input resolution (224 vs 378).

	Frozen P. Emb.	Shared P. Emb.	Type Emb.	Foreg. Indic.	Res.	LaSOT [17] SUC	LaSOT [17] P	LaSOT$_{\text{ext}}$ [16] SUC	LaSOT$_{\text{ext}}$ [16] P	TNL2K [55] SUC	TNL2K [55] P
①					224	73.9	80.4	51.9	59.0	60.6	64.3
②		✓			224	73.9	80.6	52.7	59.8	60.8	64.8
③		✓	✓		224	74.2	81.1	52.9	60.2	60.8	64.7
④		✓	✓	✓	224	74.0	80.8	52.8	60.1	61.2	65.5
⑤	✓				224	73.5	80.2	51.8	59.1	60.7	64.4
⑥	✓	✓			224	73.8	80.6	53.7	61.4	60.6	64.5
⑦	✓	✓	✓		224	74.0	80.7	52.4	59.6	60.7	64.7
⑧	✓	✓	✓	✓	224	74.2	80.9	52.8	60.0	61.1	65.1
⑨	✓	✓	✓		378	74.4	82.6	55.2	63.2	62.3	67.1
⑩	✓	✓	✓	✓	378	75.1	82.0	56.6	65.1	62.3	67.0

Table 5: Ablation on head modules. Conv head: Replace the 3-layers MLP-base head in our tracker with 5 stacked Conv-BN-ReLU layers; OSTrack [60] head: Replace the head in our tracker with OSTrack [60] convolution-based head.

Variant	LaSOT [17] SUC	LaSOT [17] P_{Norm}	LaSOT [17] P	LaSOT$_{\text{ext}}$ [16] SUC	LaSOT$_{\text{ext}}$ [16] P_{Norm}	LaSOT$_{\text{ext}}$ [16] P	TNL2k [55] SUC	TNL2k [55] P	Speed *fps*	#Params M
LoRA:										
L-224	74.2	83.6	80.9	52.8	64.7	60.0	61.1	65.1	119	336
+ Conv head	4.7	6.4	1.6	4.5	9.4	0.8	0.0	0.1	119	336
+ OSTrack [60] head	5.4	5.1	1.3	4.9	9.0	1.2	0.1	0.1	119	340
Full Fine-tune:										
L-224	73.0	81.8	79.3	52.6	64.1	59.8	60.8	64.3	119	307
+ Conv head	71.7	80.5	78.0	51.7	63.1	59.0	58.8	62.0	109	309
+ OSTrack [60] head	72.4	80.9	78.5	51.9	63.2	59.3	59.3	62.6	104	311

Different pre-training methods. We conduct experiments on four ViT variants with different pre-training methods [19,26,47,49]. From Table 8, we observe that the self-supervised DINOv2 outperforms all other ViT variants on most datasets. Compared with the results in Table 6, our model design effectively unleash the power of the large-scale pre-training.

Training efficiency of LoRA. We verify the training efficiency of LoRA on two machines: one with 8 NVIDIA V100 GPUs (Table 7), one with a single NVIDIA 4090 GPU (Table 9). The results demonstrate that fine-tuning with

LoRA significantly reduces training time and GPU memory consumption, especially for large models.

Due to space limitation, please refer to **supplementary material** for more experimental results and analysis.

Table 6: OSTrack variants implemented using official codes.

Variant	Pre-training	LaSOT SUC
B-256	MAE	68.7
L-256	MAE	70.3
B-224	DINOv2	66.2
L-224	DINOv2	67.1

Table 7: Training time and max GPU memory usage of our tracker.

	Variant	Time(h)	Memory(GB)
LoRA	B-224	5.9	2.4
	B-378	12.2	5.7
	L-224	10.8	6.6
	L-378	32.2	14.9
	g-224	22.3	14.0
	g-378	60.0	25.8
Full Fine-tuning	B-224	5.9	3.2
	B-378	12.5	6.5
	L-224	12.1	10.0
	L-378	34.1	19.1
	g-224	29.3	27.1
	g-378	Out of Memory	

Table 8: Ablation on the impact of ViT backbone with various pre-training methods. The base is B-224 trained with *full fine-tuning*.

Pre-training	LaSOT [17]		$\text{LaSOT}_{\text{ext}}$ [16]		TNL2K [55]	
	SUC	P	SUC	P	SUC	P
MAE [26]	69.9	74.8	47.9	54.7	55.2	56.0
CLIP [49]	70.1	75.0	48.7	55.0	56.8	58.3
EVA-02 [19]	70.3	75.6	49.9	56.5	57.2	59.2
DINOv2 [47]	70.9	76.2	50.0	57.1	58.1	60.1

Table 9: Training time, max training GPU memory usage, max training batch size and inference FPS of our tracker on a single customer-grade NVIDIA RTX 4090 GPU.

	Variant	Time(h)	Memory(GB)	Batch	Speed (*fps*)
LoRA	B-224	11.0	14.1	128	605
	B-378	33.8	20.3	64	285
	L-224	34.0	19.5	64	289
	L-378	110.3	14.7	16	150
Full Fine-tuning	B-224	11.7	16.1	128	605
	B-378	36.5	12	32	285
	L-224	38.5	13.7	32	289
	L-378	117.8	18	16	150

5 Conclusion

In this work, we introduce a novel approach to visual tracking by applying Low Rank Adaptation (LoRA) within a one-stream tracking framework. Our contributions include two innovative designs that enable effective adaptation of pre-trained models for tracking, significantly reducing the resource requirements. The proposed tracker, LoRAT, not only achieves state-of-the-art performance on multiple benchmarks but also demonstrates the feasibility of training advanced tracking models with manageable resources. This study bridges the gap between advanced tracking technologies and resource accessibility, marking a pivotal step towards sustainable advancements in the field of visual tracking.

Acknowledgments. This work is supported by Pengcheng Laboratory Research Project No. PCL2023A08 and No. U20B2052. Heng Fan, Zhipeng Zhang and Haibin Ling receive no financial support for the research, authorship, and/or publication of this article.

References

1. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: Fully-convolutional siamese networks for object tracking. In: ECCV Workshops (2016)
2. Bhat, G., Danelljan, M., Gool, L.V., Timofte, R.: Learning discriminative model prediction for tracking. In: ICCV (2019)
3. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. In: NeurIPS (2020)
4. Cai, Y., Liu, J., Tang, J., Wu, G.: Robust object modeling for visual tracking. In: ICCV (2023)
5. Chen, B., Li, P., Bai, L., Qiao, L., Shen, Q., Li, B., Gan, W., Wu, W., Ouyang, W.: Backbone is all your need: A simplified architecture for visual object tracking. In: ECCV (2022)
6. Chen, X., Peng, H., Wang, D., Lu, H., Hu, H.: SeqTrack: Sequence to sequence learning for visual object tracking. In: CVPR (2023)
7. Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., Lu, H.: Transformer tracking. In: CVPR (2021)
8. Cui, Y., Jiang, C., Wang, L., Wu, G.: MixFormer: End-to-end tracking with iterative mixed attention. In: CVPR (2022)
9. 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)
10. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: Eco: Efficient convolution operators for tracking. In: CVPR (2017)
11. Danelljan, M., Gool, L.V., Timofte, R.: Probabilistic regression for visual tracking. In: CVPR (2020)
12. Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: QLoRA: Efficient finetuning of quantized llms. arXiv preprint arXiv:2305.14314 (2023)
13. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019)
14. Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.M., Chen, W., et al.: Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. arXiv preprint arXiv:2203.06904 (2022)
15. 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)
16. 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)
17. 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)
18. Fan, H., Ling, H.: Siamese cascaded region proposal networks for real-time visual tracking. In: CVPR (2019)
19. 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)
20. Gao, S., Zhou, C., Ma, C., Wang, X., Yuan, J.: AiATrack: Attention in attention for transformer visual tracking. In: ECCV (2022)

21. Gao, S., Zhou, C., Zhang, J.: Generalized relation modeling for transformer tracking. In: CVPR (2023)
22. Goodfellow, I.J., Mirza, M., Xiao, D., Courville, A., Bengio, Y.: An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv preprint arXiv:1312.6211 (2013)
23. Guo, M., Zhang, Z., Fan, H., Jing, L., Lyu, Y., Li, B., Hu, W.: Learning target-aware representation for visual tracking via informative interactions. In: IJCAI (2022)
24. Hao, Y., Song, H., Dong, L., Huang, S., Chi, Z., Wang, W., Ma, S., Wei, F.: Language models are general-purpose interfaces. arXiv preprint arXiv:2206.06336 (2022)
25. Hayou, S., Ghosh, N., Yu, B.: Lora+: Efficient low rank adaptation of large models. arXiv preprint arXiv:2402.12354 (2024)
26. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: CVPR (2022)
27. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
28. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(3), 583–596 (2014)
29. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for nlp. In: ICML (2019)
30. 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)
31. Huang, L., Zhao, X., Huang, K.: Got-10k: A large high-diversity benchmark for generic object tracking in the wild. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(5), 1562–1577 (2021)
32. Kristan, M., Matas, J., Leonardis, A., Vojíř, T., Pflugfelder, R., Fernández, G., Nebehay, G., Porikli, F., Čehovin, L.: A novel performance evaluation methodology for single-target trackers. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(11), 2137–2155 (2016)
33. Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (2021)
34. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: Siamrpn++: Evolution of siamese visual tracking with very deep networks. In: CVPR (2019)
35. Li, X.L., Liang, P.: Prefix-tuning: Optimizing continuous prompts for generation. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (2021)
36. Li, X., Huang, Y., He, Z., Wang, Y., Lu, H., Yang, M.H.: CiteTracker: Correlating image and text for visual tracking. In: ICCV (2023)
37. Lialin, V., Deshpande, V., Rumshisky, A.: Scaling down to scale up: A guide to parameter-efficient fine-tuning. arXiv preprint arXiv:2303.15647 (2023)
38. Lin, L., Fan, H., Zhang, Z., Xu, Y., Ling, H.: SwinTrack: A simple and strong baseline for transformer tracking. In: NeurIPS (2022)
39. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV (2014)

40. Lin, Z., Madotto, A., Fung, P.: Exploring versatile generative language model via parameter-efficient transfer learning. In: Findings of the Association for Computational Linguistics: EMNLP 2020 (2020)
41. Liu, X., Ji, K., Fu, Y., Tam, W., Du, Z., Yang, Z., Tang, J.: P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (2022)
42. Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., Tang, J.: Gpt understands, too. arXiv:2103.10385 (2021)
43. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021)
44. Mayer, C., Danelljan, M., Paudel, D.P., Van Gool, L.: Learning target candidate association to keep track of what not to track. In: ICCV (2021)
45. Mou, C., Wang, X., Xie, L., Wu, Y., Zhang, J., Qi, Z., Shan, Y., Qie, X.: T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. arXiv preprint arXiv:2302.08453 (2023)
46. Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., Ghanem, B.: TrackingNet: A large-scale dataset and benchmark for object tracking in the wild. In: ECCV (2018)
47. 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)
48. 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)
49. 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)
50. Shaw, P., Uszkoreit, J., Vaswani, A.: Self-attention with relative position representations. In: NAACL (2018)
51. Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., Liu, Y.: Roformer: Enhanced transformer with rotary position embedding. arXiv preprint arXiv:2104.09864 (2021)
52. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: Fully convolutional one-stage object detection. In: ICCV (2019)
53. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NIPS (2017)
54. Wang, N., Zhou, W., Wang, J., Li, H.: Transformer meets tracker: Exploiting temporal context for robust visual tracking. In: CVPR (2021)
55. 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)
56. Wei, X., Bai, Y., Zheng, Y., Shi, D., Gong, Y.: Autoregressive visual tracking. In: CVPR (2023)
57. Wu, Q., Yang, T., Liu, Z., Wu, B., Shan, Y., Chan, A.B.: DropMAE: Masked autoencoders with spatial-attention dropout for tracking tasks. In: CVPR (2023)
58. Xie, F., Wang, C., Wang, G., Cao, Y., Yang, W., Zeng, W.: Correlation-aware deep tracking. In: CVPR (2022)
59. Yan, B., Peng, H., Fu, J., Wang, D., Lu, H.: Learning spatio-temporal transformer for visual tracking. In: ICCV (2021)
60. 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)

61. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: ICCV (2023)
62. Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., Zhao, T.: Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. In: ICLR (2023)
63. Zhang, R., Han, J., Liu, C., Zhou, A., Lu, P., Li, H., Gao, P., Qiao, Y.: LLaMA-adapter: Efficient fine-tuning of large language models with zero-initialized attention. In: ICLR (2024)
64. Zhang, R., Zhang, W., Fang, R., Gao, P., Li, K., Dai, J., Qiao, Y., Li, H.: Tip-adapter: Training-free adaption of clip for few-shot classification. In: ECCV (2022)
65. Zhang, Z., Liu, Y., Wang, X., Li, B., Hu, W.: Learn to match: Automatic matching network design for visual tracking. In: ICCV (2021)
66. Zhang, Z., Peng, H., Fu, J., Li, B., Hu, W.: Ocean: Object-aware anchor-free tracking. In: ECCV (2020)