Exploring Efficient and Effective Sequence Learning for Visual Object Tracking

Dongdong Li, **Zhinan Gao**, **Yangliu Kuai***, **Rui Chen**
National University of Defense Technology
{lidongdong12, gaozhinan22, kuaiyangliu09, chenrui23}@nudt.edu.cn

Abstract

Sequence learning based tracking frameworks are popular in the tracking community. In practice, its auto-regressive sequence generation manner leads to inferior performance and high latency compared with latest advanced trackers. In this paper, to mitigate this issue, we propose an efficient and effective sequence-to-sequence tracking framework named FastSeqTrack. FastSeqTrack differs from previous sequence learning based trackers in terms of token initialization and sequence generation manner. Four tracking tokens are appended to patch embeddings and generated in the encoder as initial guesses for the bounding box sequence, which improves the tracking accuracy compared with randomly initialized tokens. Tracking tokens are then parallelly fed into the decoder in a one-pass manner and greatly boost the forward inference speed compared with the auto-regressive manner. Inspired by the early-exit mechanism, we inject internal classifiers after each decoder layer to early terminate forward inference when the softmax confidence is sufficiently reliable. In easy tracking frames, early exits avoid network overthinking and unnecessary computation. Extensive experiments on multiple benchmarks demonstrate that FastSeqTrack runs over 100 fps and showcases superior performance against state-of-the-art trackers. Codes and models are available at https://github.com/vision4drones/FastSeqTrack.

1 Introduction

Visual object tracking is a fundamental computer vision task for estimating the continuous positions of an arbitrary target in a video sequence given its initial location in the first frame. Existing advanced trackers commonly employ powerful feature extraction backbones [Ye *et al.*, 2022; Lin *et al.*, 2022], complicated feature fusion modules [Chen *et al.*, 2021b; Cui *et al.*, 2022] and intricate head networks [Li *et al.*, 2018; Yan *et al.*, 2021a] to achieve impressive performance on public benchmarks [Kristan *et al.*, 2020; Huang *et al.*, 2019]. However, those trackers come at high computation burden and memory usage which results in tracking latency and deployment challenges especially on resource-constrained edge devices. Therefore, how to strike a good balance between tracking accuracy and efficiency remains a critical problem for the tracking community.

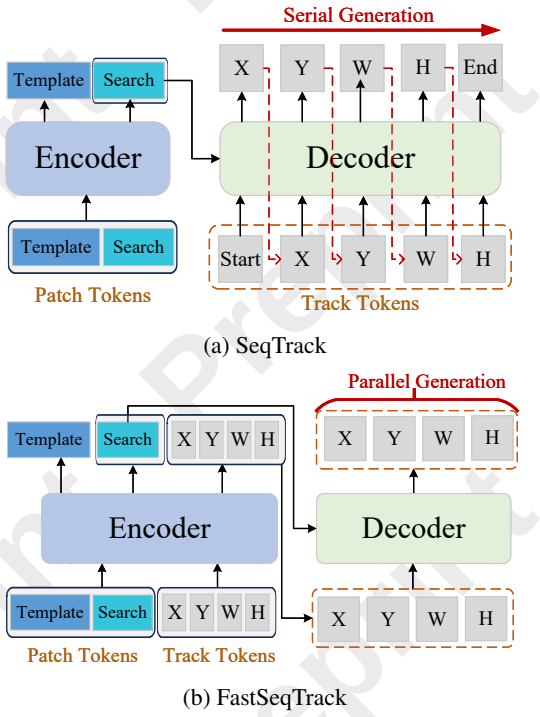

Figure 1: Comparison of tracking frameworks. (a) SeqTrack with a randomly initialized start token inferencing in an auto-regressive sequence generation manner. (b) FastSeqTrack with initially guessed tracking tokens inferencing in a one-pass sequence generation manner.

Recently, sequence learning based trackers model the tracking task as a sequence generation task and enjoy wide popularity due to the simple encoder-decoder architecture and plain cross-entropy loss function. Taking SeqTrack [Chen *et al.*, 2023] as an instance (see Fig.1(a)), it contains: i) a transformer encoder for joint feature extraction and feature fusion from the template and search regions, and ii) a transformer decoder for generating the sequence of bounding box values

from a randomly initialized start token in an auto-regressive manner. Despite success on the visual tracking task [Chen *et al.*, 2023], SeqTrack is inferior to the latest advanced trackers in terms of both tracking accuracy and speed due to the following two reasons. First, four values of the bounding box sequence are generated token-by-token, which means the decoder must run four times per frame during forward inference, inevitably leading to high computation latency. Second, to predict the bounding box sequence, tracking tokens are randomly initialized with word embedding and attend to visual features in the decoder which are restricted to a limited number of layers due to efficiency reasons. Therefore, tracking tokens are unable to fully interact with visual features and accurately predict the bounding box values.

Given the above reasons, two natural intuitions emerge to improve SeqTrack for high-speed and high-accuracy tracking: i) four bounding box values can be parallelly generated from four initial tokens through the decoder in a one-pass manner to enhance the tracking speed and ii) the predicted bounding box could be more accurate given *better* initialized tracking tokens in the decoder.

Guided by these two intuitions, we propose FastSeqTrack as illustrated in Fig.1(b), an efficient and effective sequence learning based tracking framework designed for high-speed and high-accuracy tracking. We divide the transformer architecture of FastSeqTrack into two stages: the encoder as the first stage and the decoder as the second stage. At the **first** stage, four learnable tracking tokens corresponding to four bounding box values are added to the template and search patch embeddings. The encoder takes these tracking tokens and patch embeddings as input, refining the tracking tokens alongside feature extraction. These tracking tokens can be seen as analogous to the concept of anchors in object detection, roughly indicating the initial object locations. Notably, the computational overhead introduced by the tracking tokens is minimal compared to the patch embeddings. Four tracking tokens and patch embedding of the search image are fed into the decoder as queries and visual features respectively. At the **second** stage, four tracking tokens acting as queries, attend to visual features and predict four bounding box values in a one-pass manner. While the original SeqTrack tracker generates the bounding box sequence token-by-token from a randomly initialized start token, FastSeqTrack parallelly generates four bounding box values in a one-pass manner from four initially guessed tracking tokens by the encoder. FastSeqTrack executes forward propagation in the decoder only once, which increases tracking speed by approximately four times during forward decoder inference.

To further improve the tracking efficiency and avoid network overthinking, motivated by the early-exit mechanism [Kaya *et al.*, 2019], we insert an embedding-to-word network after each decoder layer of FastSeqTrack as an internal classifier. The embedding-to-word network samples each bounding box value from the vocabulary based on the maximum likelihood. Forward propagation in the decoder is early terminated when the average maximum likelihood of four tracking tokens exceeds a specific threshold, indicating that the tracking result from the current decoder layers is reliable enough and further forward propagation in subsequent decoder layers is unnecessary. The embedding-to-word networks after each decoder layer are parameter-sharing and thus add no additional parameters compared with SeqTrack.

Our main contributions in this paper are summarized as follows:

1. We propose a novel sequence-to-sequence tracking framework for high-speed and high-accuracy single object tracking, which parallelly generates the bounding box sequence from four better initialized tracking tokens in a one-pass manner.
2. We seamlessly integrate the early-exit mechanism into the transformer decoder without additional parameters, which boosts the tracking speed in easy frames and avoid network overthinking.
3. Our proposal runs at 125 fps and demonstrates state-of-the-art performance across multiple tracking benchmarks.

2 Related Work

Sequence Learning. Sequence-to-sequence learning is initially proposed for natural language modeling [Sutskever *et al.*, 2014; Cho *et al.*, 2014] and then introduced into the object detection field. Pix2Seq [Chen *et al.*, 2021a; Chen *et al.*, 2022b] casts object detection as a language modeling task which takes the observed pixel inputs and generates a sequence of tokens that correspond to multiple bounding boxes and class labels in an auto-regressive manner. Following a similar spirit with Pix2Seq, SeqTrack [Chen *et al.*, 2023] takes a template image and a search image as input and generates a sequence of tokens that correspond to one bounding box. ARTrack [Wei *et al.*, 2023; Bai *et al.*, 2024] further adds the previous tracking results into input to propagate preceding motion dynamics into succeeding frames. It's worth noting that the auto-regressive modeling in SeqTrack and ARTrack is expensive for real-time tracking because the decoder must run four times per frame during forward inference. Different from Pix2Seq where the number of generated tokens is uncertain depending on the number of predicting bounding boxes, SeqTrack and ARTrack generates a fixed number of tokens, i.e., four tokens (each for one bounding box dimension) ignoring the end token. Therefore, in terms of visual tracking, we argue that the auto-regressive sequence generation manner is unnecessary and should be changed into a plain one-pass manner for high efficiency.

Efficient and Effective Tracking. There exist many previous works to achieve a good trade-off between tracking efficiency and accuracy. ECO [Danelljan *et al.*, 2017] is a pioneer correlation filter based tracker which reduces model parameters by factorized convolution operators and achieves a high tracking speed. LightTrack [Yan *et al.*, 2021b] utilizes neural architecture search (NAS) to find lightweight neural networks for deep learning based object tracking. HiT [Kang *et al.*, 2023] introduces the bridge module and achieves both high speed and precision on different devices. These aforementioned trackers achieve high efficiency by reducing flops or network parameters in the tracking model. However, compared with recent top trackers, these trackers has restricted

tracking performance in challenging tracking sequences due to limited parameters in the compact model. Therefore, how to achieve both efficient and effective tracking remains a crucial problem for visual tracking. Some integrated trackers [Fan and Ling, 2017; Li *et al.*, 2020] are proposed observing the phenomenon that a simple network is sufficient for easy frames in a video and more computation should be added to difficult ones. By integrating a simple tracker with a strong one, integrated trackers cope easy and difficult frames with the simple and strong tracker respectively. However, because the simple and strong trackers cooperate on a tracker level and are not integrated in an end-to-end manner, the tracking performance of integrated trackers is inferior to pure CNN or transformer based trackers [Li *et al.*, 2018; Ye *et al.*, 2022].

Early Exits for Tracking. Recently, dynamic routing points out a new direction for solving the speed-precision dilemma with early exits [Kaya *et al.*, 2019]. DyTrack [Zhu *et al.*, 2024] formulates instance-specific tracking as a sequential decision problem in a dynamic transformer network. An IoU token is appended after each encoder layer and predicts an IoU Score with an extra decisioner network (a 3-layer MLP). Forward inference is early terminated when the IoU score is above a threshold. Despite extreme tracking speed, DyTrack achieves only average performance due to two drawbacks. First, compared with features from the end layer, features from intermediate encoder layers (e.g., ViT [Dosovitskiy *et al.*, 2020]) is not discriminative enough for high-accuracy tracking. Second, the IoU score derived from the decisioner network for terminating forward inference is not always consistent with the bounding box predicted by the tracking head. Compared with DyTrack, our proposal maintains the integrity of the encoder and appends the embedding-to-word network after each decoder layer as an internal classifier. The embedding-to-word network both decides the exiting condition and predicts the bounding box. Compared with the extra decisioner network in DyTrack, the embedding-to-word networks in our proposal are parameter-sharing and add no extra parameters to the decoder.

3 Method

This section presents a comprehensive description of FastSeqTrack in detail. First, we provide an overview of FastSeqTrack. Next, we describe the model architecture, including the encoder, the decoder, the early exits and the loss function. Finally, we introduce the training and inference procedures. The overall framework is shown in Fig.2(a).

3.1 Overview

FastSeqTrack mainly consists of an encoder for joint feature extraction and tracking token generation, and a decoder for interacting of visual features and tracking tokens. Due to the one-pass sequence generation manner, the `start` token, `end` token and the casual attention mask for the self-attention modules in the decoder are all removed. During inference, four tracking tokens are generated by the encoder and fed into the decoder as queries. In each frame, four bounding box values are simultaneously predicted in one pass.

3.2 Model Architecture

Encoder. Following SeqTrack, FastSeqTrack employs a standard vision transformer architecture (i.e., ViT) in the encoder. The encoder takes a search image $\boldsymbol{s} \in \mathbb{R}^{3\times H\times W}$ and a template images $\boldsymbol{t} \in \mathbb{R}^{3\times H\times W}$ as input. The search and template images maintain the same size and are partitioned into patches: $\boldsymbol{s}_p \in \mathbb{R}^{N\times(P^2\times 3)}$ and $\boldsymbol{t}_p \in \mathbb{R}^{N\times(P^2\times 3)}$, where (P, P) is the patch size and $N = HW/P^2$ is the patch number. A linear projection $\mathbf{E} \in \mathbb{R}^{(P^2\times 3)\times D}$ is used to map the image patches $\boldsymbol{s}_p$ and $\boldsymbol{t}_p$ to visual embeddings $\boldsymbol{X}_s \in \mathbb{R}^{N\times D}$ and $\boldsymbol{X}_t \in \mathbb{R}^{N\times D}$. Similar to the $[\boldsymbol{cls}]$ tokens in ViT, four randomly initialized tracking tokens $[\boldsymbol{track}] \in \mathbb{R}^{D\times 4}$ are appended to visual embeddings $\boldsymbol{X}_s$ and $\boldsymbol{X}_t$. Learnable position embeddings $\boldsymbol{P} \in \mathbb{R}^{(N+N+4)\times D}$ are added to all the input tokens $[\boldsymbol{X}_s; \boldsymbol{X}_t; \boldsymbol{track}]$. These combined embeddings are fed into the encoder for feature extraction and interaction. During forward propagation, the encoder performs global self-attention of all input tokens, which can be decomposed into three different cross-attention operations, namely $\boldsymbol{X}_s \times \boldsymbol{X}_t$, $[\boldsymbol{track}] \times \boldsymbol{X}_s$ and $[\boldsymbol{track}] \times \boldsymbol{X}_t$. Among them, $\boldsymbol{X}_s \times \boldsymbol{X}_t$ performs feature fusion of the template and search images. $[\boldsymbol{track}] \times \boldsymbol{X}_s$ and $[\boldsymbol{track}] \times \boldsymbol{X}_t$ capture the relationship between tracking tokens and visual features. The tracking tokens $[\boldsymbol{track}]$ and visual features corresponding to the search image $\boldsymbol{X}_s$ output by the last encoder layer are fed into the decoder.

Decoder. The decoder of FastSeqTrack is a casual transformer consisting of N transformer decoder layers. Each decoder layer consists of a mask-free multi-head self attention, a multi-head attention and a feed-forward network (FFN) as shown in Fig.2(b). Different from SeqTrack whose self attention network is masked, the self attention network in the decoder layer of FastSeqTrack is mask-free because four tracking tokens are parallelly fed into the decoder in an one-pass manner. This allows each tracking token to attend to the other three tracking tokens in a non-casual manner. Following the $[\boldsymbol{track}] \times \boldsymbol{X}_s$ operation in the encoder, the multi-head attention network in the decoder allows the tracking tokens $[\boldsymbol{track}]$ to further attend to visual features $\boldsymbol{X}_s$ as queries and learn robust representations for final bounding box prediction. Compared with SeqTrack whose query token (i.e., start token) in the decoder are randomly initialized with word embedding, query tokens (i.e., tracking tokens) of FastSeqTrack are prematurely initialized by the encoder before being fed into the decoder. Therefore, similar to anchors in two-stage object detectors, tracking tokens generated by the encoder contribute to high-accuracy bounding box prediction in the decoder.

Embedding-to-word Network. FastSeqTrack adopts the embedding-to-word network for bounding box regression. Each continuous coordinate of the target bounding box $[x, y, w, h]$ is uniformly discretized into an integer within a vocabulary $\boldsymbol{V} = [1, n_{bins}]$. The embedding-to-word network maps a high-dimensional tracking token to an integer between $[1, n_{bins}]$ corresponding to a word in $\boldsymbol{V}$. The embedding-to-word network is implemented as a multi-layer perceptron (FCN) and a softmax function (SOFTMAX). FCN aligns the dimension of the tracking tokens output by the last decoder layer with the size of the vocabulary n_{bins} while

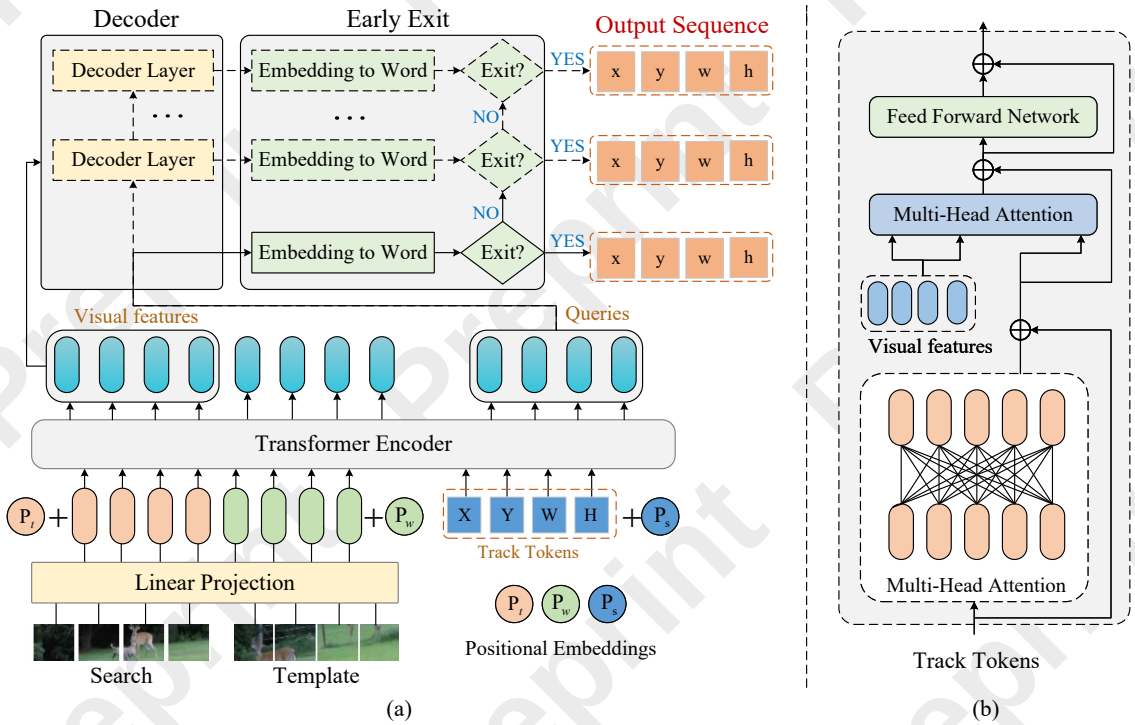

Figure 2: (a) An overview of FastSeqTrack. The key component is an encoder-decoder transformer and an early-exit module. The encoder generates visual features and queries for the decoder. Early exits terminate forward propagation of the decoder layers and output the bounding box when the softmax probability of the embedding-to-word network is above an threshold. Our proposal achieves both high-speed and high-accuracy tracking. (b) Detailed transformer block in the mask-free decoder. The tracking tokens interact with each other through a masked-free multi-head attention mechanism. The visual feature is incorporated into the decoder and interacts with the tracking tokens via a multi-head attention layer.

SOFTMAX samples the predicted bounding box coordinates $\hat{b} = [x,y,w,h]$ from the vocabulary based on the softmax probability.

Early Exits. Different from previous sequence learning based trackers (i.e., SeqTrack) which performs forward propagation through all decoder layers, FastSeqTrack achieves instance-specific tracking by early terminating forward propagation in easy frames to avoid network overthinking. The encoder of FastSeqTrack remain unchanged compared with SeqTrack. In SeqTrack, the embedding-to-word network is appended to the last decoder layer for bounding box prediction. On contrast, in FastSeqTrack, the embedding-to-word network is appended to each decoder layer as an internal classifier as shown in Fig.2(a). The embedding-to-word networks after each decoder layer are parameter-sharing and thus avoid adding extra parameters to the tracking model. Four tracking tokens $[\boldsymbol{track}] \in \mathbb{R}^{D\times 4}$ output by each decoder layer are fed into the embedding-to-word network and mapped to a softmax probability $P \in \mathbb{R}^{n_{bins}\times 4}$. We compute the average softmax score over four generated bounding box values. If the averaged score exceeds a specific threshold τ which means the predicted bounding box is reliable enough, FastSeqTrack terminates forward propagation at a certain intermediate decoder layer and outputs the tracking result. The forward propagation process of the decoder with early exits is summarized in Algorithm 1. In fact, we also insert the embedding-to-word network before the first decoder layer to directly predict the bounding box from the tracking tokens generated from the encoder. In this case, the whole decoder is skipped.

3.3 Training and Inference

FastSeqTrack is trained in an end-to-end fashion with the combination of the cross-entropy loss and the generalized IoU loss [Rezatofighi *et al.*, 2019]. Different from SeqTrack which only predicts with the last decoder layer, all the intermediate decoder layers of FastSeqTrack are responsible for predicting the bounding box. The loss function can be written as

$$L = \lambda_{ce}\sum_{l=1}^{N} L_{ce}(b,\hat{b}_l) + \lambda_{iou}\sum_{l=1}^{N} L_{iou}(b,\hat{b}_l). \qquad (1)$$

where b represents the groundtruth bounding box and $\hat{b}_l$ represents the bounding box predicted by the l_{th} decoder layer. $\lambda_{ce}, \lambda_{iou} \in \mathbb{R}$ are regularization parameters.

During inference, the encoder perceives one search image, an initial template image, a dynamic template image and four tracking tokens. The dynamic template image is online updated. The decoder parallelly predicts the target sequence from four tracking tokens. FastSeqTrack will early terminate

Algorithm 1 Decoder With Early Exits

Input:
Visual features $\boldsymbol{X}_s$.
Tracking tokens $[\boldsymbol{track}]$.
Parameters:
Decoder layers $Block_l, l = 1 : N$.
Embedding-to-word network.
Output:
Predicted bounding box $\hat{b}$.

Let $z_0 = [\boldsymbol{X}_s; \boldsymbol{track}]$ and Earlyexit = $False$.
for layer l = 1 to N **do**
  $z_l = Block_l(z_{l-1})$
  $\hat{b}, score$ =Embedding-to-word(z_l)
  if $score > \tau$ **then**
    Earlyexit = $True$
    return $\hat{b}$
  end if
end for
if Earlyexit = False **then**
  return $\hat{b}$
end if

forward inference of the decoder and boost the tracking speed in easy tracking frames. Forward inference will go deeper in the decoder until the last decoder layer in difficult frames. The last decoder layer will output its predicted bounding box as the final tracking result when outputs from the previous decoder layers are all unreliable.

4 Experiments

4.1 Implementation Details

Model. We adopt ViT-B [Dosovitskiy *et al.*, 2020] as the encoder architecture for FastSeqTrack. The input resolution of the template image and search image are 256×256. The patch size is set to 16×16. Accordingly, the dimension of four tracking tokens in FastSeqTrack is 768, which equals $16 \times 16 \times 3$. The decoder consists of 2 transformer blocks. Accordingly, three embedding-to-word networks are inserted in the decoder, one before the first decoder layer and two after each decoder layer. For fair comparison, we follow all the default parameter setting of SeqTrack for the rest hyperparameters. Our implementation is based on SeqTrack and is far from optimal. We believe there is a big room for future improvement and generalization.

In addition, we present model parameters, trraining flops, and inference speed in Tab. 1. The speed is measured on Intel Xeon Gold 6354 CPU @ 3.00GHz with 64 GB RAM and a single 4090 GPU with 24GB memory. All the models are implemented with Python 3.8 and PyTorch 1.11.0. As reported in Tab. 1, FastSeqTrack and SeqTrack adopt the same backbone and thus maintain the same number of params in the encoder. Compared with SeqTrack, the four tracking tokens slightly increase the encoder flops in FastSeqTrack by 1.5%. There are slightly fewer decoder params in FastSeqTrack than SeqTrack which indicates that the tracking tokens and early exits add almost no extra parameters to the model. During training, SeqTrack has less decoder flops than FastSeqTrack because its self attention is masked. It's worth noting that the encoder flops and decoder flops are measured during offline training and do not directly measure the actual online inference speed which is displayed in the last column in Tab. 1. It's obvious that FastSeqTrack running at over 100 fps is two times faster than SeqTrack during inference due to the one-pass sequence generation manner.

Training. Our training data includes the training splits of COCO [Lin *et al.*, 2014], LaSOT [Fan *et al.*, 2019], GOT-10k [Huang *et al.*, 2019], TrackingNet [Muller *et al.*, 2018] and VastTrack [Peng *et al.*, 2024]. We follow the default training parameters of SeqTrack. The training of FastSeqTrack is conducted on 2 Intel Xeon Gold 6354 CPU @ 3.00GHz with 64 GB RAM and 8 4090 GPUs with 24GB memory. Each GPU holds 16 image pairs, resulting in a total batch size of 128. The regularization parameters $\lambda_{ce}, \lambda_{iou} \in \mathbb{R}$ in Equ. 1 are set to 1 and 5 respectively. The model is trained for a total of 500 epochs with $60k$ image pairs per epoch. The learning rate decreases by a factor of 10 after 400 epochs.

Inference. The online template update interval is set to 1 by default, while the threshold τ in Algorithm 1 is set to 1.6. The vocabulary size is set to 4000 and the softmax likelihood in the vocabulary is directly multiplied by a 1D Hanning window for window penalty.

4.2 State-of-the-Art Comparisons

As detailed in Tab. 2, we compare FastSeqTrack with state-of-the-art trackers on four tracking benchmarks, i.e., VastTrack [Peng *et al.*, 2024], TNL2K [Wang *et al.*, 2021], TrackingNet [Muller *et al.*, 2018] and GOT-10k [Huang *et al.*, 2019]. FastSeqTrack achieves the best performance in most benchmarks and is the only tracker which runs over 100 fps among all trackers. We display the performance and tracking speed of SeqTrack, FastSeqTrack and state-of-the-art trackers on four benchmarks in the following. It's worth noting that we follow the reported fps in the literature for each tracker in Tab. 2.

VastTrack. VastTrack [Peng *et al.*, 2024] is a recently introduced general visual tracking benchmark with a vast object category. The test set comprises 3500 videos from 2115 classes. As reported in Tab. 2, FastSeqTrack achieves the top position with AUC scores of 39.24%, surpassing the second best tracker (i.e., MixformerV2) by 4.04%. The top position in AUC, normalized precision (P_{Norm}) and Precision (P) of FastSeqTrack demontrates the effectiveness of the initially guessed tracking tokens. Further, FastSeqTrack performs better than SeqTrack, getting 4.4% AUC score improvement. Fig. 3 shows the results of attribute-base evaluation, illustrating that FastSeqTrack performs better than other competing trackers on almost all attributes.

TNL2K. TNL2K [Wang *et al.*, 2021] is a recently released large-scale dataset with 700 challenging video sequences. On the large-scale TNL2K benchmark, FastSeqTrack obtains the third best performance with 56.86% AUC score as reported in Tab. 2. Under aligned settings (the same ViT-B encoder architecture and input resolution), FastSeqTrack achieves 0.73% and 0.96% higher AUC score than SeqTrack and OSTrack, respectively.

Model	Encoder Params (M)	Encoder FLOPs (G)	Decoder Params (M)	Decoder FLOPs (M)	Overall Params (M)	Overall FLOPs (G)	Speed (*fps*)
SeqTrack	85.647	65.742	3.465	120.440	89.111	65.862	58
FastSeqTrack	85.647	66.082	3.464	130.232	89.110	66.212	125

Table 1: Details of SeqTrack and FastSeqTrack.

Method	VastTrack [Peng *et al.*, 2024]			TNL2K [Wang *et al.*, 2021]			TrackingNet [Muller *et al.*, 2018]			GOT-10k [Huang *et al.*, 2019]			Speed
	AUC	P_{Norm}	P	AUC	P_{Norm}	P	AUC	P_{Norm}	P	AO	$SR_{0.5}$	$SR_{0.75}$	fps
SeqTrack	34.84	37.70	33.89	56.13	74.01	58.09	83.3	88.3	82.2	74.7	84.7	71.8	58
FastSeqTrack	39.24	42.7	39.79	56.86	74.22	60.35	85.54	90.31	85.53	77.8	88.2	76.8	125
ODTrack [Zheng *et al.*, 2024]	-	-	-	61.7	-	-	85.1	90.1	86.1	77.0	87.9	75.1	32
ARTrackV2 [Bai *et al.*, 2024]	-	-	-	59.2	-	-	84.9	89.3	84.5	75.9	85.4	72.7	72
SimTrack [Chen *et al.*, 2022a]	34.4	34.2	30.3	55.6	-	-	83.4	87.4	-	69.8	78.8	66.0	40
MixformerV2 [Cui *et al.*, 2023]	35.2	36.5	33.0	-	-	-	83.1	88.1	81.6	70.7	80.0	67.8	25
AiATrack [Gao *et al.*, 2022]	-	-	-	-	-	-	82.7	87.8	80.4	69.6	63.2	80.0	38
CSWinTT [Song *et al.*, 2022]	-	-	-	-	-	-	81.9	86.7	79.5	69.4	78.9	65.4	12
STARK [Yan *et al.*, 2021a]	33.4	34.3	30.8	-	-	-	82.0	86.9	-	68.8	78.1	64.1	40
OSTrack [Ye *et al.*, 2022]	33.6	34.5	31.5	55.9	-	-	83.1	87.8	82.0	71.0	80.4	68.2	58
SwinTrack [Lin *et al.*, 2022]	33.0	34.2	30.3	-	-	-	84.0	-	82.8	72.4	-	67.8	98
RTS [Paul *et al.*, 2022]	35.5	36.4	33.1	-	-	-	81.6	86.0	79.4	-	-	-	30
ToMP [Mayer *et al.*, 2022]	34.9	36.4	32.1	45.9	-	-	81.5	86.4	78.9	-	-	-	19
AutoMatch [Zhang *et al.*, 2021]	28.8	31.5	26.6	-	-	-	76.0	-	72.6	65.2	76.6	54.3	50
TransT [Chen *et al.*, 2021b]	29.9	31.4	25.4	50.7	-	-	81.4	86.7	80.3	67.1	76.8	60.9	50
SiamBAN [Chen *et al.*, 2020]	16.0	15.5	9.6	41.0	49	41.7	-	-	-	-	-	-	35
Ocean [Zhang *et al.*, 2020]	27.5	30.2	24.6	38.4	45	37.7	-	-	-	61.1	72.1	47.3	58
DiMP [Bhat *et al.*, 2019]	29.9	31.7	25.7	44.7	-	-	74.0	80.1	68.7	61.1	71.7	49.2	40
SiamPRN++ [Li *et al.*, 2019]	28.1	29.7	24.9	41.3	48	41.2	73.3	80.0	69.4	51.7	61.6	32.5	35
ATOM [Danelljan *et al.*, 2019]	17.2	14.1	10.2	40.1	47	39	70.3	77.1	64.8	55.6	63.4	40.2	30
SiamFC [Bertinetto *et al.*, 2016]	8.5	2.9	4.3	29.5	35.4	28.6	73.3	80.0	69.4	51.7	61.6	32.5	58

Table 2: State-of-the-art comparisons on four large-scale benchmarks. The top three results are highlight with **red**, blue and green fonts, respectively.

TrackingNet. TrackingNet [Muller *et al.*, 2018] is a relatively smaller dataset covering diverse object categories and scenes with 511 videos in the test set. As reported in Tab. 2, FastSeqTrack achieves the best performance in both AUC score and P_{Norm}, surpassing the second best tracker ODTracker by 0.44% in AUC and 0.2% in P_{Norm}. It's worth noting that ODTrack adopts dense temporal tokens for performance improvement and only runs at 32 fps.

GOT-10k. GOT-10k [Huang *et al.*, 2019] test set contains 180 videos covering various common tracking challenges. As reported in Tab. 2, FastSeqTrack achieve the top position in AO, P_{Norm} and P, outperforming SeqTrack by a large margin. Specifically, FastSeqTrack achieves 3.1% gains in AO score over SeqTrack.

4.3 Ablation and Analysis.

We use SeqTrack as the baseline model in our ablation study to investigate the impact of different modules in FastSeqTrack. The result of the baseline is reported in Tab. 3 (#0).

Loss Functions. We compare different loss functions for offline training. The baseline SeqTrack tracker only adopts a simple cross-entropy loss for end-to-end training. The overall cross-entropy loss of the predicted bounding box is calculated by summing the cross-entropy losses of four bounding box values. Therefore, the cross-entropy loss in SeqTrack is easy to be dominated by a certain value with big prediction error. On contrast, as described in Equ. 1, FastSeqTrack adopts both the cross-entropy loss and generalized IoU loss for bounding box regression. Here, as reported in Tab. 3 (#1), SeqTrack with the joint loss function performs much better than SeqTrack with the cross-entropy loss, which indicates that the generalized IoU loss contributes to better bounding box regression.

Sequence Generation Manner. SeqTrack generates the sequence in an auto-regressive manner, which predicts the four bounding box values one by one. As shown in Tab. 3 (#2), we change the auto-regressive manner into the one-pass manner that predictes four bounding box values parallelly. The original input of the decoder in SeqTrack is only a start token which tells the model to begin the sequence generation. While, the input of SeqTrack in the one-pass manner is changed into four randomly generated tracking tokens. Accordingly, the casual attention mask in SeqTrack is removed, allowing four tracking tokens to attend to each other. As reported in Tab. 3 (#2), SeqTrack in the one-pass sequence generation manner is superior to SeqTrack in the auto-gressive manner by 1.6% in the AO score.

Decoder Input. We compare different input tokens to the decoder. Different from four randomly generated tracking tokens in Tab. 3 (#2), the input tracking tokens of the decoder in Tab. 3 (#3) is generated in the encoder as an initial guess. While the input tokens of the decoder in SeqTrack only interact with visual features in the decoder, the input tracking tokens in Tab. 3 (#3) interact with visual features in both the encoder and decoder. From Tab. 3 (#3), we can observe that SeqTrack with initially guessed tracking tokens outperform SeqTrack with randomly initialized tracking tokens by 2.4% in the AO score. The underlying reason might be that initially guessed tracking tokens guarantee saturated fusion of the tracking tokens and visual features.

Early Exits. As shown in Tab. 3 (#4), we insert parameter-sharing early exits into the decoder of SeqTrack. Com-

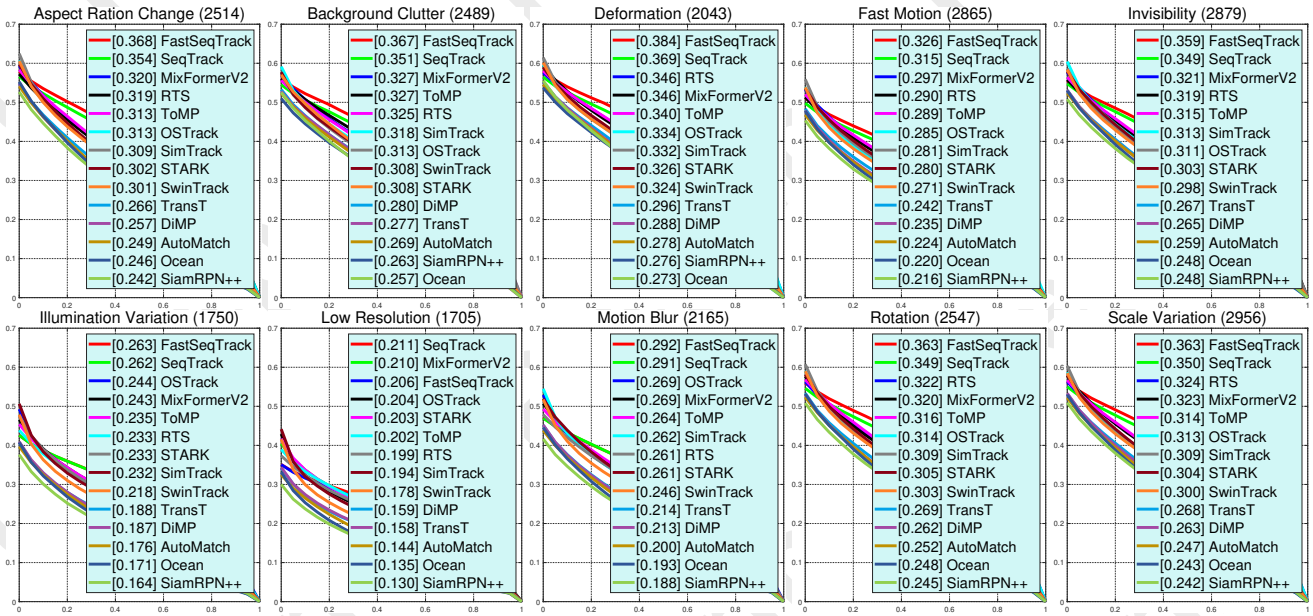

Figure 3: *Success ratio* plots on 10 attributes of the VastTrack dataset. Trackers are ranked by their AUC scores. Ours method has achieved consistently the superior performance on 9 of 10 attributes, which demonstrates the robustness of our approach in challenging tracking scenarios. The horizontal and vertical axes denote the overlap threshold and success rate respectively.

#	Method	AO	$SR_{0.5}$	SR
0	Baseline	-	-	-
1	+GIoU Loss	+2.4	+3.0	+2.3
2	Auto-regressive→One-pass	+1.6	+1.8	+2.8
3	Random Initialization→Initial Guess	+2.4	+2.6	+3.9
4	+Parameter-sharing Early Exits	+0.7	+1.2	+0.1
5	+Independent Early Exits	+0.7	+1.3	+0.0

Table 3: Ablation Study on GOT-10k. + denotes the performance gain compared with the baseline #0.

pared with SeqTrack which performs layer-by-layer inference in the decoder, SeqTrack with parameter-sharing early exits achieves better performance. The underlying reason might be that early exits avoid network overthinking which is destructive when a correct prediction changes into a false prediction by late layers. As shown in Tab. 3 (#5), we also implement SeqTrack with independent early exits which are not parameter-sharing. Overall, independent early exits don't show improved performance. Therefore, we adopt parameter-sharing early exits by default in our implementation of FastSeqTrack.

Visualization of Cross Attention Map. To better understand the effectiveness of parallel sequence generation in FastSeqTrack, we visualize the cross attention map of the last decoder block. Fig. 4 shows cross attention maps as the model generates four tracking tokens. The search images are displayed in the first column. Overall, the cross attention maps in the second and third columns concentrate on the top left corner of the target region which corresponds to the x and y values. On contrast, the cross attentions in the fourth and fifth columns focus on the whole target region to accurately regress the w and h values.

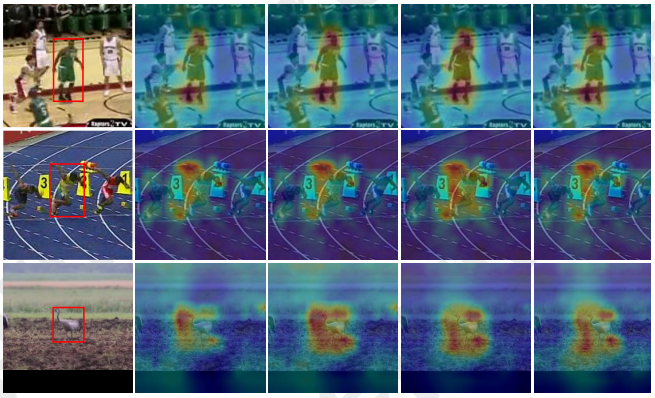

Figure 4: Decoder's cross attention to visual features when generating four tracking tokens. The first column is the search region image, and the second to last columns are the cross attention maps corresponding to x, y, w, h tokens, respectively.

5 Conclusion

In this work, we propose an efficient and effective sequence learning based tracking framework named FastSeqTrack, which sets a new direction for high-accuracy and high-speed visual tracking. Compared with previous sequence learning based trackers, FastSeqTrack improves the tracking token initialization and sequence generation manner. Early exits are integrated into the decoder for further efficiency and avoid network overthinking. Extensive experiments demonstrate FastSeqTrack achieves state-of-the-art performance across multiple datasets. The core ideas in FastSeqTrack are generic and able to be incorporated into any similar sequence learning based trackers.

References

[Bai *et al.*, 2024] Yifan Bai, Zeyang Zhao, Yihong Gong, and Xing Wei. Artrackv2: Prompting autoregressive tracker where to look and how to describe. In *CVPR*, pages 19048–19057, 2024.

[Bertinetto *et al.*, 2016] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip H S Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, pages 850–865, 2016.

[Bhat *et al.*, 2019] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, pages 6182–6191, 2019.

[Chen *et al.*, 2020] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *CVPR*, pages 6668–6677, 2020.

[Chen *et al.*, 2021a] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. In *ICLR*, 2021.

[Chen *et al.*, 2021b] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, pages 8126–8135, 2021.

[Chen *et al.*, 2022a] Boyu Chen, Peixia Li, Lei Bai, Lei Qiao, Qiuhong Shen, Bo Li, Weihao Gan, Wei Wu, and Wanli Ouyang. Backbone is all your need: A simplified architecture for visual object tracking. In *ECCV*, pages 375–392, 2022.

[Chen *et al.*, 2022b] Ting Chen, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David J Fleet, and Geoffrey Hinton. A unified sequence interface for vision tasks. *arXiv preprint arXiv:2206.07669*, 2022.

[Chen *et al.*, 2023] Xin Chen, Houwen Peng, Dong Wang, Huchuan Lu, and Han Hu. Seqtrack: Sequence to sequence learning for visual object tracking. In *CVPR*, pages 14572–14581, 2023.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[Cui *et al.*, 2022] Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Mixformer: End-to-end tracking with iterative mixed attention. In *CVPR*, pages 13608–13618, 2022.

[Cui *et al.*, 2023] Yutao Cui, Tianhui Song, Gangshan Wu, and Limin Wang. Mixformerv2: Efficient fully transformer tracking. In *NIPS*, 2023.

[Danelljan *et al.*, 2017] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient convolution operators for tracking. In *CVPR*, pages 6638–6646, 2017.

[Danelljan *et al.*, 2019] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate tracking by overlap maximization. In *CVPR*, pages 4660–4669, 2019.

[Dosovitskiy *et al.*, 2020] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

[Fan and Ling, 2017] Heng Fan and Haibin Ling. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. In *ICCV*, pages 5487–5495, 2017.

[Fan *et al.*, 2019] 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 *CVPR*, pages 5374–5383, 2019.

[Gao *et al.*, 2022] Shenyuan Gao, Chunluan Zhou, Chao Ma, Xinggang Wang, and Junsong Yuan. AiATrack: Attention in attention for transformer visual tracking. In *ECCV*, pages 146–164, 2022.

[Huang *et al.*, 2019] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE TPAMI*, pages 1562–1577, 2019.

[Kang *et al.*, 2023] Ben Kang, Xin Chen, Dong Wang, Houwen Peng, and Huchuan Lu. Exploring lightweight hierarchical vision transformers for efficient visual tracking. In *ICCV*, pages 9578–9587, 2023.

[Kaya *et al.*, 2019] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *ICML*, pages 3301–3310, 2019.

[Kristan *et al.*, 2020] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, et al. The eighth visual object tracking VOT2020 challenge results. In *ECCV*, pages 547–601, 2020.

[Li *et al.*, 2018] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, pages 8971–8980, 2018.

[Li *et al.*, 2019] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, pages 4282–4291, 2019.

[Li *et al.*, 2020] Dongdong Li, Fatih Porikli, Gongjian Wen, and Yangliu Kuai. When correlation filters meet siamese networks for real-time complementary tracking. *IEEE Trans. Circuits Syst. Video Technol.*, 30(2):509–519, 2020.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755, 2014.

[Lin *et al.*, 2022] Liting Lin, Heng Fan, Yong Xu, and Haibin Ling. Swintrack: A simple and strong baseline for transformer tracking. In *NeurIPS*, 2022.

[Mayer *et al.*, 2022] Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. Transforming model prediction for tracking. In *CVPR*, pages 8731–8740, 2022.

[Muller *et al.*, 2018] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. TrackingNet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, pages 300–317, 2018.

[Paul *et al.*, 2022] Matthieu Paul, Martin Danelljan, Christoph Mayer, and Luc Van Gool. Robust visual tracking by segmentation. In *ECCV*, pages 571–588, 2022.

[Peng *et al.*, 2024] Liang Peng, Junyuan Gao, Xinran Liu, Weihong Li, Shaohua Dong, Zhipeng Zhang, Heng Fan, and Libo Zhang. Vasttrack: Vast category visual object tracking. *arXiv preprint arXiv:2403.03493*, 2024.

[Rezatofighi *et al.*, 2019] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, pages 658–666, 2019.

[Song *et al.*, 2022] Zikai Song, Junqing Yu, Yi-Ping Phoebe Chen, and Wei Yang. Transformer tracking with cyclic shifting window attention. In *CVPR*, pages 8791–8800, 2022.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NeurIPS*, pages 3104–3112, 2014.

[Wang *et al.*, 2021] Xiao Wang, Xiujun Shu, Zhipeng Zhang, Bo Jiang, Yaowei Wang, Yonghong Tian, and Feng Wu. Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark. In *CVPR*, pages 13763–13773, 2021.

[Wei *et al.*, 2023] Xing Wei, Yifan Bai, Yongchao Zheng, Dahu Shi, and Yihong Gong. Autoregressive visual tracking. In *CVPR*, pages 9697–9706, 2023.

[Yan *et al.*, 2021a] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *ICCV*, pages 10448–10457, 2021.

[Yan *et al.*, 2021b] 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 *CVPR*, pages 15180–15189, 2021.

[Ye *et al.*, 2022] Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint feature learning and relation modeling for tracking: A one-stream framework. In *ECCV*, pages 341–357, 2022.

[Zhang *et al.*, 2020] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *ECCV*, pages 771–787, 2020.

[Zhang *et al.*, 2021] Zhipeng Zhang, Yihao Liu, Xiao Wang, Bing Li, and Weiming Hu. Learn to match: Automatic matching network design for visual tracking. In *ICCV*, pages 13339–13348, 2021.

[Zheng *et al.*, 2024] Yaozong Zheng, Bineng Zhong, Qihua Liang, Zhiyi Mo, Shengping Zhang, and Xianxian Li. Odtrack: Online dense temporal token learning for visual tracking. In *AAAI*, pages 7588–7596, 2024.

[Zhu *et al.*, 2024] Jiawen Zhu, Xin Chen, Haiwen Diao, Shuai Li, Jun-Yan He, Chenyang Li, Bin Luo, Dong Wang, and Huchuan Lu. Exploring dynamic transformer for efficient object tracking. *CoRR*, 2024.