Representation Learning for Visual Object Tracking by Masked Appearance Transfer

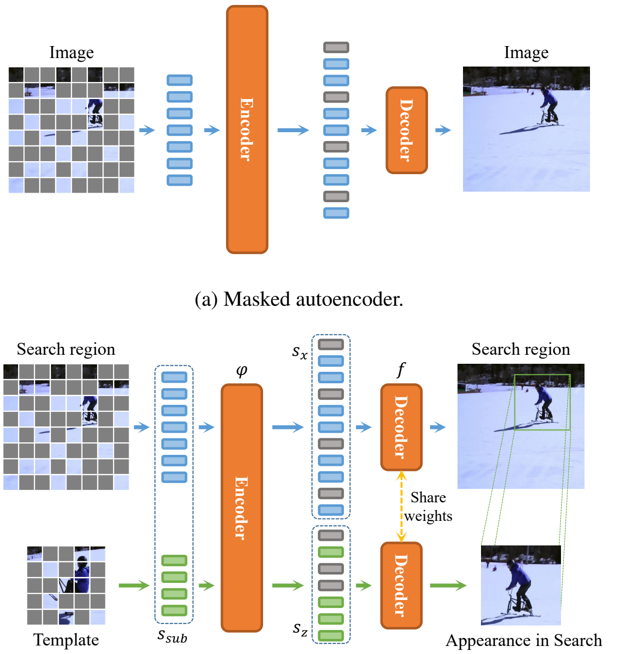
Haojie Zhao[1] Dong Wang[1] Huchuan Lu[1,2*]
[1]Dalian University of Technology, China [2]Peng Cheng Laboratory, China
haojie_zhao@mail.dlut.edu.cn {wdice,lhchuan}@dlut.edu.cn

Abstract

Visual representation plays an important role in visual object tracking. However, few works study the tracking-specified representation learning method. Most trackers directly use ImageNet pre-trained representations. In this paper, we propose masked appearance transfer, a simple but effective representation learning method for tracking, based on an encoder-decoder architecture. First, we encode the visual appearances of the template and search region jointly, and then we decode them separately. During decoding, the original search region image is reconstructed. However, for the template, we make the decoder reconstruct the target appearance within the search region. By this target appearance transfer, the tracking-specified representations are learned. We randomly mask out the inputs, thereby making the learned representations more discriminative. For sufficient evaluation, we design a simple and lightweight tracker that can evaluate the representation for both target localization and box regression. Extensive experiments show that the proposed method is effective, and the learned representations can enable the simple tracker to obtain state-of-the-art performance on six datasets. https://github.com/difhnp/MAT

1. Introduction

Visual object tracking is a computer vision task that highly depends on the quality of visual representation [38]. On this basis, deep representations are adopted for tracking and successfully boost the development of tracking algorithms in previous years. Unlike some primitive trackers (e.g., [10, 28, 37]) that use ready-made deep features, SiamFC [1] integrates a convolutional neural network (CNN) into the tracking model and learns task-specified representations by end-to-end tuning. From here on, end-to-end model training becomes a common practice in siamese-based tracking methods. The assumption of the siamese tracker is that different visual appearances of the same target can be embedded to have similar representations, and tracking can be achieved by similarity matching. Based on this assumption, the learning objective aims to push the representations of the same target to be close to each other.



(a) Masked autoencoder.

(b) Masked appearance transfer.

Figure 1. Comparison between the masked autoencoder [17] and our masked appearance transfer that uses a nontrivial learning objective.

Although we have a clear learning objective, research on the effective representation learning methods for siamese tracking is still lacking. The common practice is to simply fine-tune the ImageNet [33] pre-trained representations. However, they are learned for classification rather than tracking. High ImageNet accuracy does not indicate good performance in visual object tracking [43]. Without good rep-

[*]Corresponding author: Huchuan Lu, lhchuan@dlut.edu.cn

resentations, simple similarity matching cannot bring good tracking performance. Thus, most works focus on the design of tracker neck [7, 36, 47, 50] and tracker head [25, 40, 46]. Within these works, we can notice that the entire model is always trained end-to-end, and representation learning is coupled to the box regression task and target localization task. It means that the representation learning is totally driven by other tracker components. To improve tracking performance, we have to make the neck or head increasingly complex.

On the aforementioned observation, we attempt to improve the tracking performance by learning good representations and propose a simple but effective representation learning method in this paper. This method aims to decouple the representation learning from tracker training, thereby making it a tracker-free method. It employs a transformer-based autoencoder to learn more discriminative visual features for object tracking. This is achieved by setting a nontrivial learning objective for the autoencoder, as illustrated by Figure 1b. The embedded target appearance of the template can be pushed to be close to that in the search region. We further make the learned representations more discriminative by masking out the input images.

To evaluate the learned representations for target localization and box regression, we design a very simple tracker with a matching operator and a lightweight head. We evaluate the proposed method with different model architectures and initial weights. We also compare our simple tracker with many state-of-the-art trackers on the popular LaSOT [14], TrackingNet [31], GOT10k [20], and many other tracking datasets [15, 30, 41]. The experiments demonstrate the effectiveness and generalization ability of our proposed representation learning method. Comparison results show that our simple tracker can obtain state-of-the-art performance with the learned representations.

To summarize, our contributions are presented as follows:

- We propose masked appearance transfer (MAT), a novel representation learning method for visual object tracking that jointly encodes the template and search region images, and learns tracking-specified representation with a simple encoder-decoder pipeline. A nontrivial training objective is proposed to make this method effective.

- We design a simple and lightweight tracker for tracking-specified evaluation that has few parameters and no hyper-parameters. It can evaluate the representations not only for target localization but also for box regression.

- Extensive experiments demonstrate the effectiveness and generalization ability of the proposed method. Extensive comparisons show that the proposed simple tracker can obtain state-of-the-art performance by using the learned tracking-specified representations.

2. Related Work

2.1. Visual Representation Learning

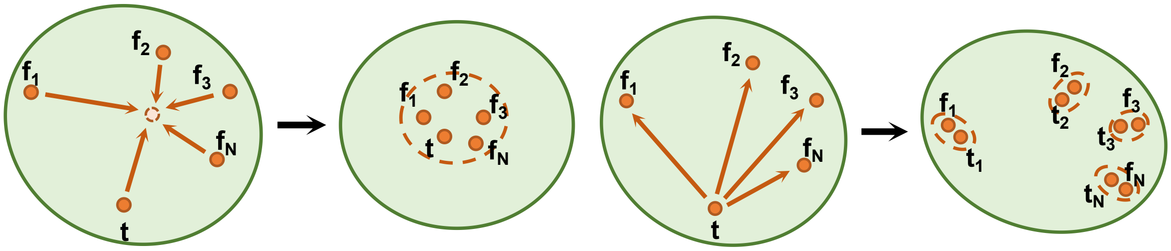
LeCun et al. [23] first proposed the use of the convolutional neural network (CNN) to extract visual representation for document recognition. Krizhevsky et al. [21] designed a deep CNN and train it for image classification successfully, thereby showing the great advantage of deep visual features. After that, many deeper convolutional neural networks, such as VGG [34] and ResNet [19], are proposed to extract better visual representations. Recently, the self-attention-based transformer architecture [35] has shown great potential in natural language processing (NLP). Inspired by the transformer successes in NLP, Dosovitskiy et al. [13] used a pure transformer model to encode visual representation, thereby obtaining excellent image recognition results. Many other vision tasks, such as detection [3, 9, 51], segmentation [16, 42], and tracking [7, 8, 47], also used the transformer model successfully.

To learn visual representation, the vision models are commonly trained with ImageNet [33] in the supervised learning fashion. However, self-supervised methods can also bring good visual representations. Some works, such as SimSiam [5] and MoCov3 [6], used contrastive methods to learn representation. He et al. [17] used an autoencoding method for representation learning. In our work, we also used an autoencoder, but we assign it a nontrivial learning objective to learn the tracking-specified representations.

2.2. Visual Object Tracking.

The siamese-based tracking methods have achieved great success in recent years. SiamFC [1] first formulated object tracking as a similarity matching problem and trains a siamese convolutional network to encode the template feature and search region feature separately. By using the cross-correlation operator, it can generate a score map and locate the target at the peak value. It still uses a coarse multi-scale method to estimate the target size. To estimate a more accurate bounding box, Li et al. [25] introduced the region proposal network (RPN) [32] to the siamese tracker that could regress many box candidates. The best box is selected by finding the peak of the score map. Zhang et al. [49] and Li et al. [24] used deeper networks to improve the tracking performance. Many works, such as TransT [7], STARK [47], and TrDiMP [39], used the transformer for more effective feature fusion.

Recently, some works attempted to design one-stream tracking frameworks that encode the template and search region jointly. Xie et al. [45] performed multi-layer matching within a transformer backbone. Cui et al. [8] proposed mixed attention for simultaneous representation embedding. [48] and [4] used a single ViT model for tracking. Instead of designing a one-stream tracking framework, we



(a) Make all embedded $\mathbf{t}$ and $\mathbf{f}$ be close to each other.

(b) Push the embedded $\mathbf{t}$ close to the embedded $\mathbf{f}$.

Figure 2. Different assumptions on the embedding function, where $\mathbf{t}$ denotes the embedded template appearance, and $\mathbf{f}_i$ denotes the embedded visual appearance w.r.t. frame i.

propose a representation learning method with an autoencoder. Based on the learned representations, we can achieve high-performance tracking.

3. The Proposed Method

Our MAT is a novel tracking-specified representation learning method. This method employs a masked autoencoder, which consisted of an encoder that embedded the search image patches and template patches jointly, and a decoder that reconstructs the original search image and the transferred template separately. With this learning objective, the target representation of the template is trained to be close to that of the search region (Figure 2b), and the learned representations are naturally suitable for visual tracking.

3.1. Reformulation

The siamese-based tracking methods [1, 7, 25, 47] formulate visual tracking as a similarity matching problem. These trackers learn a deep network as the embedding function φ, which can embed the template image $\mathbf{Z} \in \mathbb{R}^{H_z \times W_z \times 3}$ and search region image $\mathbf{X} \in \mathbb{R}^{H_x \times W_x \times 3}$ to the same feature space:

$$\mathbf{t} = \varphi(\mathbf{Z}), \quad \mathbf{f} = \varphi(\mathbf{X}).$$

A matching function is used to locate the target by fusing the embedded $\mathbf{t}$ and $\mathbf{f}$ and generating a score map. The siamese tracker assumes that the same target with different visual appearances can be embedded to be close in the feature space (Figure 2a). On this basis, we can perform similarity matching for tracking. However, in practice, the embedding function is difficult to learn, and many high-performance trackers have to implement complex models to fuse the template feature and search region feature [7, 8, 47, 50].

In this work, we still formulate visual tracking as a similarity-matching problem and still learn an embedding function. What makes the difference is that, in our assumption, the embedding function does not embed all appearances to be close to each other. It is expected to push the embedded target representation of the template close to that of the search region (Figure 2b). With this assumption, we obtained a clear supervising target that can be achieved by using a simple autoencoder, see below.

3.2. Jointly Masked Encoding

An autoencoder can be used for visual representation learning. It first uses an encoder to embed the input image into the visual feature. Then, it uses a decoder to reconstruct the input image with the embedded feature. In contrast to the common autoencoders, our encoder takes two images, namely, the template and search region images, as the inputs, which encode their visual representations jointly.

Specifically, we use a vision transformer model as the encoder. The template image $\mathbf{Z}$ and the search region image $\mathbf{X}$ are cut into non-overlap patches in size of $p \times p$. Then, we use a linear layer to project these patches into vision tokens and reform them into an input sequence $\mathbf{s} \in \mathbb{R}^{(N_z+N_x)\times C}$, where C is embedding dimensions; and $N_z = H_z W_z / p^2$ and $N_x = H_x W_x / p^2$ denote the number of patches w.r.t. $\mathbf{Z}$ and $\mathbf{X}$, respectively. Besides, we add sine-cosine position embeddings to the projected patches.

Input masking has shown its effectiveness in representation learning [12, 17]. It can reduce redundancy and make a difficult task for the autoencoder, thereby learning more discriminative features. We randomly mask out the input tokens according to a pre-defined masking ratio k. The remained sequence $\mathbf{s}_{sub} \in \mathbb{R}^{N_{sub} \times C}$, where $N_{sub} = \lfloor (1-k) \times (N_z + N_x) \rfloor$, is encoded by vision transformer. After encoding, we use a learnable [*mask*] token to fill the masked positions for recovering the sequence length. Specifically, in our method (Figure 1b), we use a encoder φ to jointly encode $\mathbf{s}_{sub}$ and output the recovered $\mathbf{s}_x$ and $\mathbf{s}_z$, : $\varphi(\mathbf{s}_{sub}) \rightarrow \mathbf{s}_x, \mathbf{s}_z$.

Based on the self-attention mechanism, the transformer encoder can have the capability of long-range modeling. We expect it to capture the target-specified correspondence between the template and the search region, generate high-quality representations for similarity matching, and bring satisfactory tracking performance. To this end, we design a nontrivial learning objective for our autoencoder, see below.

3.3. Masked Appearance Transfer

The decoder commonly reconstructs the original input. However, we set a nontrivial learning objective for our autoencoder, thereby enabling the encoder to capture the target-specified correspondence. We use the same decoder to reconstruct the template and search region separately (Figure 1b).

The input search region image $\mathbf{X}$ remains unchanged and was used as the learning objective for the search region branch. The decoder f takes the encoded search region tokens $\mathbf{s}_x$ to reconstruct the original image: $f(\mathbf{s}_x) \to \mathbf{X}$. For the template branch, we crop a new template $\mathbf{T}$ from the search region image as the learning objective: $f(\mathbf{s}_z) \to \mathbf{T}$. By reconstructing this new template, we transfer the target appearance from the template view to the search region view.

Note that this new template can also be reconstructed by $\mathbf{s}_x^{target}$, which is a subset of $\mathbf{s}_x$ and belongs to the target area. By using the proposed nontrivial objective, $\mathbf{s}_z$ can be pushed to be close to $\mathbf{s}_x^{target}$. Thus, the learned representations are suitable for tracking by similarity matching. Some visualization results are shown in Figure 3.

3.4. Evaluation Tracker

To evaluate the representation quality, a common practice is to train a lightweight evaluator on the learned representations. Self-supervised learning (SSL) works (e.g., [5, 17, 18]) usually train a supervised linear classifier. For visual tracking, recent work [43] makes a lightweight tracker by using *cross-correlation* or *discriminative correlation filter* on frozen features. However, this tracker does not use the feature for box regression, which is an essential part of high-performance modern trackers (e.g., [8, 11, 25]). In this work, we use a similar evaluation method and construct a simple tracker on the learned representations.

We design our tracker to be lightweight and hyper-parameter-free. In contrast to the heavy-head design, a lightweight head can make the tracker more sensitive to the quality of representations, and few learnable parameters can make the tracking performance less related to the training setting. We use the depth-wise correlation operator [24] in our tracker to perform similarity matching. This operator has no learnable parameter, and the matching result totally depends on the representations. To make the tracker hyper-parameter-free, we use a CornerNet-based [22] head to regress the bounding box. This head predicts the final box by regressing the left-top corner point and the bottom-right corner point. This head does not have any hyper-parameters, unlike the anchor-based methods [7, 24, 50], because target localization and scale estimation can be achieved simultaneously. It is also lightweight, consisting of only several 3×3 convolution layers. Specifically, we reshape the encoded $\mathbf{s}_z$ and $\mathbf{s}_x$ and use the matching operator to fuse them to generate a feature map. Then, we feed this feature map to the anchor-free head, regressing two corner probability distribution maps. On these maps, we can use *soft-argmax* to gather the corner coordinates, which are formatted to the final bounding box.

With our design, the proposed tracker can demonstrate the quality of representations for localization and regression. With our learned representations, this simple tracker can obtain competitive tracking performance.

4. Experiments

4.1. Datasets and Metrics

LaSOT [14] refers to a large-scale long-term tracking benchmark, which has 280 long-term test videos with an average length of 2,448 frames. It uses Success (AUC) and normalized precision (NP) as the evaluation metrics.

TrackingNet [31] refers to a large-scale visual tracking benchmark that contains 511 short-term test videos. It also uses AUC and NP to reflect the tracking performance.

GOT10k [20] contains more than 10,000 videos, split into train, val, and test sets. It uses average overlap (AO) as the primary metric and also uses success rates (SR) at different thresholds for evaluation.

UAV123 [30] contains 123 test sequences from an aerial viewpoint, which is more challenging for most generic trackers and also uses AUC metric.

TNL2K [41] is a newly proposed dataset, containing 700 test videos with many thermal, adversarial, and virtual scenarios. It uses AUC as the primary metric.

NFS [15] also contains 100 test videos but has many scenarios with fast motion and distractors, and AUC is adopted for evaluation.

4.2. Evaluation Methods

We build a tracker on the learned representation. By referring to [17], we adopt the *freezing* protocol and the *fine-tuning* protocol for tracker evaluation. The tracking performance will reflect the effectiveness of our representation learning method.

Freezing. We freeze the feature encoder and only train the tracker head. This protocol can demonstrate whether the learned representation is tracking-specified.

Fine-tuning. We fine-tune the learned representation. Both the feature encoder and the tracker head are trained. This protocol can demonstrate the advantage of our learned representation in tracking.

4.3. Implementation Details

Autoencoder. The default encoder is ViT-B/16 [13]. The decoder is consistent with MAE [17] and remains unchanged in our experiments. The encoder and decoder are initialized with MAE pre-trained weights. We use two RTX3090 GPUs for all experiments, and all ViT models are from PyTorch Image Models [44].

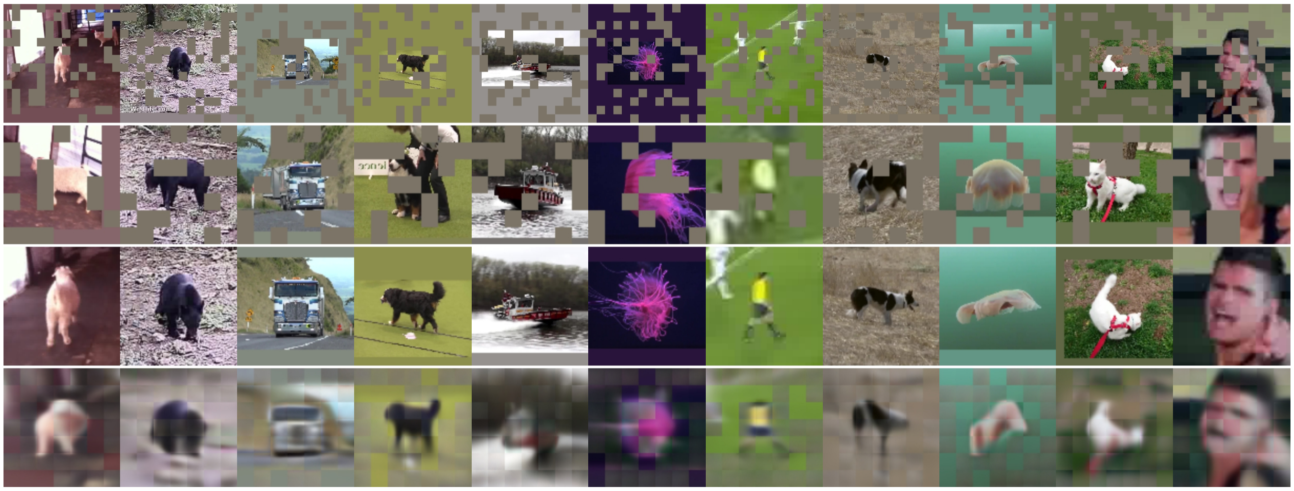


Figure 3. Visualizations of the masked search regions (1st row), masked templates (2nd row), new templates from the search region (3rd row), and reconstructed new templates (4th row). The masking ratio is 25%.

Table 1. Comparison with state-of-the-art trackers on the LaSOT test set in terms of AUC, NP. The best top-3 is marked red, blue, green.

	SiamFC++	DiMP	PrDiMP	TrDiMP	SiamRCNN	TransT	STARK-ST101	ToMP-101	MixFormer-1k	**Ours_freeze**	**Ours_fine-tune**
AUC(%)	54.4	56.9	59.8	63.9	64.8	64.9	67.1	**68.5**	**67.9**	65.2	**67.8**
NP(%)	62.3	65.0	68.8	73.0	72.2	73.8	**77.0**	**79.2**	**77.3**	74.8	**77.3**

During training, the default masking ratio k is set to 25%. We refer to other tracking literature (e.g., [7, 24, 50]) for data preparation. The training pairs are randomly sampled from the training set of VID [33], TrackingNet [31], LaSOT [14], and GOT10k [20]. Color jitter, horizontal flip, scale jitter, and position jitter are used for augmentation. This processing method is also used for tracker training. The template size and search region size are set to 112×112 and 224×224, namely 2 and 4 times the size of the target, for both training and inference. We use the AdamW optimizer [27] to train the model end-to-end. The total training epochs are set to 500 with 64,000 search-template pairs per epoch. We set the weight decay to $5e^{-2}$, and the initial learning rate is set to $1e^{-4}$ and is dropped by a factor of 10 after 200 epochs. Each GPU holds 32 sample pairs and accumulates gradients for two steps. The reconstruction loss is the commonly used mean squared error (MSE).

Tracker. We construct a tracker based on the pre-trained encoder and make it as simple as possible: (1) We borrow a lightweight anchor-free head from [8]. This head predicts only one box and does not require any complex post-processes (e.g., cosine window, size penalty), having no hyper-parameters. (2) We use the simple depth-wise correlation operator [24] as the matching function, which does not have any learnable parameters.

As our clean design, this tracker does not have any hyper-parameters that can avoid performance gains from hyper-parameter search and make our evaluation results more reliable. Besides, it only contains about 2M learnable parameters (v.s. 86M in the ViT-B/16 encoder), running at approximately 120 FPS on RTX3090.

The training set includes the aforementioned datasets and COCO [26]. Training pairs are processed in the aforementioned way. We use the AdamW optimizer [27] to train our tracker for 300 epochs with $64,000$ search-template pairs per epoch. We set the weight decay to $1e^{-4}$, and the initial learning rate is set to $1e^{-4}$ and is dropped by a factor of 10 after 240 epochs. For fine-tuning experiments, the learning rate of the encoder blocks is set to $0.1\times$ base learning rate. Each GPU holds 32 sample pairs and accumulates gradients for two steps. We use a combination of generalized intersection over union (GIoU) loss and L_1 loss to supervise the tracker training. The final loss function is defined as

$$L = \lambda_1 L_{GIoU}(\mathbf{b}, \hat{\mathbf{b}}) + \lambda_2 L_1(\mathbf{b}, \hat{\mathbf{b}}),$$

where $\mathbf{b}$ is the predicted box, and $\hat{\mathbf{b}}$ is the ground truth. We set the loss weight λ_1 to 2 and set λ_2 to 5.

During tracking, we simply take an encoding forward pass on the search region and template. Then, we reshape the encoded tokens to feature maps and fuse them with the matching operator. Finally, the fused feature is fed into the anchor-free head to regress a target box. Note that random masking does not be used for inference.

Table 2. Comparison with state-of-the-art trackers on the TrackingNet test set.

	SiamFC++	DiMP	PrDiMP	TrDiMP	SiamRCNN	TransT	STARK-ST101	ToMP-101	MixFormer-1k	**Ours_freeze**	**Ours_fine-tune**
AUC(%)	75.4	74.0	75.8	78.4	81.2	81.4	**82.0**	81.5	**82.6**	79.8	**81.9**
NP(%)	80.1	80.0	81.6	83.3	85.4	86.7	**86.9**	86.4	**87.7**	85.2	**86.8**

Table 3. Comparison with state-of-the-art trackers on the GOT10k test set. "†" indicates that we use the default train set for MAT pre-training but use the GOT10k train split for tracker training. "*" indicates that we use the GOT10k train split for both MAT pre-training and tracker training.

	Ocean	DiMP	PrDiMP	TrDiMP	SiamRCNN	TransT	STARK-ST101	MixFormer-1k	**Ours_freeze***	**Ours_fine-tune***	**Ours_freeze†**
AO(%)	61.1	61.1	63.4	67.1	64.9	67.1	**68.8**	**71.2**	63.2	64.4	**67.7**
$SR_{0.5}$(%)	72.1	71.7	73.8	77.7	72.8	76.8	**78.1**	**79.9**	72.1	72.8	**78.4**

Table 4. Comparison with state-of-the-art trackers on the UAV123, TNL2K and NFS datasets in terms of AUC (%).

	Ocean	DiMP	PrDiMP	TrDiMP	SiamRCNN	TransT	STARK-ST101	ToMP-101	MixFormer-1k	**Ours_fine-tune**
UAV123	57.4	65.3	68.0	67.5	64.9	**69.1**	**68.2**	66.9	**68.7**	68.0
TNL2K	38.4	44.7	47.0	-	**52.3**	51.0	**52.5**	-	**53.3**	51.3
NFS	49.4	62.0	63.5	**66.2**	63.9	**65.7**	**66.2**	**66.7**	64.2	65.3

4.4. State-of-the-art Comparison

In this paper, we propose a representation learning method for tracking. To evaluate the learned representation, we compare our tracker with some existing state-of-the-art (SOTA) trackers, including SiamFC++ [46], DiMP [2], PrDiMP [11], TrDiMP [39], SiamRCNN [36], Ocean [50], TransT [7], STARK-ST101 [47], ToMP-101 [29], and MixFormer-1k [8]. Both the *freezing* protocol and the *fine-tuning* protocol are used.

Without bells and whistles, our simple tracker achieves satisfactory tracking performance on six large-scale tracking datasets with the learned representation. Table 1 and 2 show that our tracker can obtain the comparable 65.2% AUC on LaSOT with TransT [7], even though the feature encoder is totally frozen. By fine-tuning encoder blocks, our tracker obtains 67.8% AUC and 77.3% NP on the LaSOT test set and obtains 81.9% AUC and 86.8% NP on the TrackingNet test set. Our simple tracker not only has competitive tracking performance, which is very close to recent SOTA trackers ToMP-101 [29] and MixFormer-1k [8] but also has a fast tracking speed.

Table 3 shows the results on the GOT10k test set. By using only the GOT10k train split, our tracker achieves a sub-optimal 63.2% AO. Fine-tuning the feature encoder can bring a 64.4% AO, but the improvement is slight. However, if we use all the aforementioned datasets for MAT pre-training, our tracker can achieve a competitive 67.7% AO. It suggests that our method needs to take sufficient data to learn good representation, and the proposed tracker can demonstrate the quality of different representations for tracking.

Besides, as reported in Tabel 4, our tracker achieves 68.0% AUC on UAV123, which outperforms ToMP-101 [29] by 1.1% and is very close to MixFormer-1k [8]. On TNL2K, our tracker achieves a competitive 51.3% AUC, which is slightly higher than TransT [7] and slightly lower than STARK-ST101 [47]. The multi-modal scenarios in TNL2K result in more challenges for our learned RGB representations. On the NFS dataset, our tracker still achieves results that are comparable with TransT [7] on this dataset and outperforms MixFormer-1k [8] by 1.1%.

It's worth noting that the best several trackers, including STARK-ST101 [47], ToMP-101 [29], and MixFormer-1k [8], have carefully designed online update schemes to achieve the top performance. Our tracker can obtain competitive performance with only a very simple design, highly depending on the offline learned representations. All these results prove the effectiveness of our representation learning method, and the learned representation is exactly suitable for tracking. It also proves the importance of good representations for tracking again.

4.5. Ablation Study

We use the freezing protocol for ablation studies, where the feature encoder is frozen, and only the anchor-free tracker head is trained. Three main-stream datasets, i.e., LaSOT, TrackingNet, GOT10k, are used for evaluation.

Masking ratio. We pre-train the encoder with different masking ratios. Tracker training is initialized by these pre-trained weights. Table 5 shows the influence of the masking ratio on the tracking performance. The masking ratio of 25% works well for tracking, and we use it as the default setting in our experiments. The ratios of 50% and 0% also bring similar good tracking performance. It suggests that

Table 5. A medium-low masking ratio can bring good representation for tracking. The default ratio is marked in gray .

Masking ratio	LaSOT		TrackingNet		GOT10k	
	AUC	NP	AUC	NP	AO	$SR_{0.5}$
75%	63.7	73.4	76.9	82.6	64.3	75.6
50%	**65.6**	**75.1**	78.9	84.4	67.5	78.4
25%	65.2	74.8	**79.8**	**85.2**	**67.6**	**78.5**
00%	63.5	72.5	79.4	84.4	67.5	77.6

Table 6. Component-wise analysis (top block) and studies on some training settings (bottom block).

	LaSOT		TrackingNet		GOT10k	
	AUC	NP	AUC	NP	AO	$SR_{0.5}$
baseline	55.1	62.8	70.9	76.0	56.1	65.2
+JE	44.2	47.9	62.9	66.8	43.4	49.1
+TD	50.8	52.2	56.5	58.9	36.3	40.5
+TD +JE	61.9	71.7	76.7	82.2	61.9	73.1
+TD +JE +MAT	65.2	74.8	**79.8**	**85.2**	67.6	78.5
+LP	**65.3**	**75.0**	79.6	84.9	**69.4**	**80.5**
-DE	**65.3**	74.6	**79.8**	84.9	68.0	78.2
-SJ	64.5	67.4	78.8	84.3	66.4	75.7
-PJ	57.7	59.6	74.3	79.2	58.8	67.7

our method prefers a medium-low masking ratio and is tolerant of the masking ratio. The 75% masking ratio reduces the performance by about 2% over all three benchmarks. Considering that the training samples are not object-centric images, a high masking ratio can mask out too much target information, affecting representation learning.

Masking strategy. By default, we use the *random masking* strategy, which has shown good performance in both vision [17] and language [12] tasks. The *grid-wise* strategy performs similarly to the random strategy in the original MAE. It regularly keeps one of every four patches, and this is equivalent to a 75% masking ratio. We make a comparison on the masking strategies. Table 7 shows the comparison results. We can see that the *grid-wise* strategy also performs similarly to the random strategy in our MAT method. However, its performance is still lower than that of *random masking* on all three datasets.

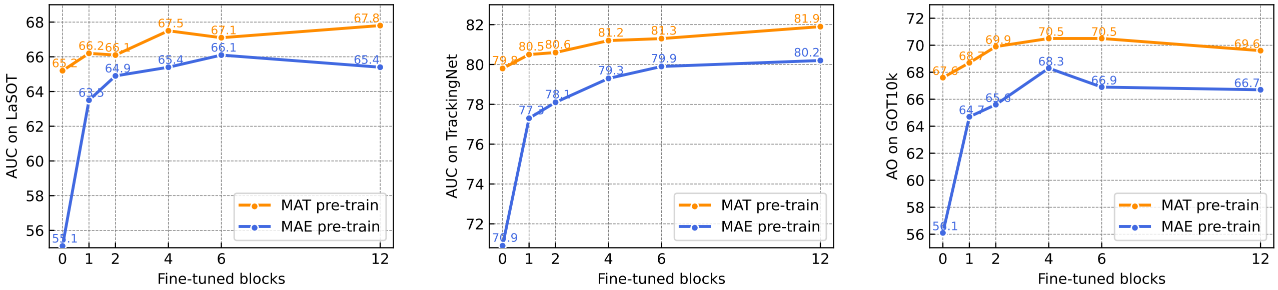
Component-wise Analysis. We conduct a component-wise analysis with the ViT-B/16 model and our proposed tracking method (Table 6). "**baseline**" indicates that we use the MAE pre-trained ViT encoder as the baseline that encodes the template and search region images separately, i.e. a common two-stream architecture. "**JE (joint encoding)**" indicates that we use the one-stream architecture and encode the template and search region jointly. "**TD (tracking data)**" indicates that we use all the aforementioned tracking datasets (i.e. VID, TrackingNet, LaSOT, GOT10k) for model training. "**MAT**" indicates that we use the proposed masked appearance transfer method for representation learning.

"**baseline**" shows that the ImageNet pre-trained MAE representation is not suitable for tracking, having poor tracking performance. It suggests a gap between the classification task and tracking task on representation learning. By directly loading the MAE weights to the one-stream architecture for joint encoding, the variant "**+JE**" obtains 44.2%/47.9%, 62.9%/66.8%, and 43.4%/49.1% performance scores. Compared with the baseline, an obvious performance decline can be observed. It suggests that simply using a one-stream architecture cannot extract better representations for tracking. "**+TD**" indicates we fine-tune the MAE weights with the tracking data by using the original MAE [17] method. The performance decline is because the object diversity of tracking datasets is less than that of ImageNet. Simply tuning with the tracking data can damage the ImageNet pre-trained representation. "**+TD +JE**" indicates that we use the one-stream architecture for joint encoding and use tracking data to fine-tune this one-stream model. Here, we do not perform *masked appearance transfer*, and the decoder still reconstructs the original input. By capturing the target-specified correspondence between the template and search region, joint encoding can improve the tracking performance effectively. "**+TD +JE +MAE**" provides better results. It suggests that the proposed *masked appearance transfer* learning objective makes the encoder learn more discriminative representations, which are more suitable for tracking.

We also analyze some training settings. We replace the default sine-cosine position embeddings with learnable embeddings "**+LP**", and slight gains on GOT10k can be observed. "**-DE**" denotes we do not tune the decoder, the lightweight decoder does not affect the performance significantly. Scale jitter and position jitter are two import data augmentation methods for tracker training, and we also use them in the pre-training phase. Here, we study how these augmentation methods affect representation learning. "**-SJ**" and "**-PJ**" denote the removal of scale jitter and position jitter from the pre-training phase, respectively. It is obvious that position jitter plays an important role in our pre-train method. The randomly positional shift makes appearance transfer a hard task, which forces the encoder to capture more discriminative correspondence.

Comparison with different representations. We compare our learned representations with different representations that were encoded by the original MAE [17] pre-trained ViT-B/16 and MoCov3 [6] pre-trained ViT-S/16. As the popular backbone of most visual trackers, ImageNet-supervised ResNet-50[1] [19] is also selected as a baseline. For a fair comparison, we remove the fourth block of ResNet to make the down-sample stride the same as the ViT model. Thus, nearly all settings are aligned, except for the type of rep-

[1] We use the PyTorch pre-trained weights: https://pytorch.org/vision/stable/models.html



(a) Our tracker can achieve 67.8 AUC by fine-tuning all blocks.

(b) Tracking performance is always improved by tuning more blocks.

(c) Tracker is more likely to overfit on the MAE representations.

Figure 4. Our learned representations perform better than the original MAE representations under the *partial fine-tuning* protocol.

Table 7. Comparison of different masking strategies.

	LaSOT		TrackingNet		GOT10k	
	AUC	NP	AUC	NP	AO	$SR_{0.5}$
random	63.7	73.4	76.9	82.6	64.3	75.6
grid-wise	62.4	72.2	75.6	81.1	62.4	73.1

Table 8. Comparison of different representations.

	LaSOT		TrackingNet		GOT10k		ImageNet1K	params
	AUC	NP	AUC	NP	AO	$SR_{0.5}$	Top-1 Acc.	(M)
ViT-B/16 +MAT	**65.2**	**74.8**	**79.8**	**85.2**	**67.6**	**78.5**	-	-
ViT-B/16	55.1	62.8	70.9	76.0	56.1	65.2	83.6	86
ViT-S/16 +MAT	59.0	68.0	73.1	78.8	59.8	69.9	-	-
ViT-S/16	46.5	52.1	63.5	68.4	47.4	55.8	81.4	22
ResNet-50	54.9	63.2	68.9	74.2	55.1	65.1	76.1	25

resentations. Because we only train the same tracker head, the tracking performance totally depends on the quality of representations.

Table 8 reports the tracking performance on different representations, where "**+MAT**" indicates that we use the proposed MAT learning method to tune the pre-trained weights. Using the same tracker design, the ResNet50 baseline has weak tracking performance. Compared with ResNet50, the "**ViT-S/16 +MAT**" representations improve tracking performance by 4.1%/4.8%, 4.2%/4.6%, and 4.7%/4.8%. The "**ViT-B/16 +MAT**" representations can improve tracking performance by 10.3%/11.6%, 10.9%/11.0%, and 12.5%/13.4%. The proposed method improves the MAE representations successfully.

By changing the model architecture and initial weights to ViT-S/16-MoCov3, our method is still effective. It demonstrates the good generalization ability of our method. These results also suggest that our tracker design (i.e., simple matching operator and simple head) is suitable for demonstrating the representation quality for visual object tracking.

Besides, the tracking performance and ImageNet accuracy or model capability have no clear correlation. High accuracy or big model capability does not denote good tracking performance. It proves the worth of our work that proposes a tracking-specified representation learning method.

4.6. Partial Fine-tuning

To evaluate the representation, we use the *partial fine-tuning* protocol. We still train our tracker using the aforementioned way. However, we fine-tune the last several encoder blocks while freezing the others. The learning rate for these blocks is set to 0.1× base learning rate.

Figure 4 shows that the fine-tuning encoder blocks can always bring performance gains regardless of the representation we use. By fine-tuning, our simple tracker can achieve 67.8% AUC, 81.9% AUC, and 70.5% AO on LaSOT, TrackingNet, and GOT10k, respectively. The fine-tuning results outperform the freezing results by 2.2%, 2.1%, and 2.9%, respectively. Although the fine-tuned encoder blocks can largely improve the original MAE representations for tracking, it is always lower than our method by at least 1.0% when we fine-tune the same number of blocks. Our learned representations perform better over three datasets. The best results with our learned representations outperform the best results with the original MAE representations by 1.7%, 1.7%, and 2.2%, respectively. Besides, the frozen MAT representations can bring comparable performance with the fine-tuned representations, suggesting that the MAT-learned representation is tracking-specified. The partial fine-tuning experiments show the effectiveness of our method again.

5. Conclusion

In this paper, we present a tracking-specified representation learning method. We design a special learning objective that can make the transformer encoder model target-specified correspondence and learn the tracking-specified representations by using an autoencoder. We propose a simple and lightweight tracker that can be used for representation evaluation. We conduct extensive experiments on six tracking datasets. The experimental results show that our representation learning method is effective, and the proposed tracker can obtain competitive tracking performance by using the learned representations.

Acknowledgements The paper is supported in part by National Natural Science Foundation of China (Nos.62293542, U1903215, 62022021), National Key R&D Program of China (No.2018AAA0102001), and Fundamental Research Funds for the Central Universities (No.DUT22ZD210).

References

[1] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *Proceedings of the European Conference on Computer Vision Workshops*, pages 850–865, 2016. 1, 2, 3

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6181–6190, 2019. 6

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision*, pages 213–229, 2020. 2

[4] 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. pages 375–392, 2022. 2

[5] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 2, 4

[6] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9620–9629, 2021. 2, 7

[7] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8126–8135, 2021. 2, 3, 4, 5, 6

[8] Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Mixformer: End-to-end tracking with iterative mixed attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–12, 2022. 2, 3, 4, 5, 6

[9] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1601–1610, 2021. 2

[10] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient convolution operators for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6931–6939, 2017. 1

[11] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7181–7190, 2020. 4, 6

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. pages 4171–4186, 2019. 3, 7

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations*, pages 1–12, 2021. 2, 4

[14] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. LaSOT: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5374–5383, 2019. 2, 4, 5

[15] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1134–1143, 2017. 2, 4

[16] Ruohao Guo, Dantong Niu, Liao Qu, and Zhenbo Li. SOTR: Segmenting objects with transformers. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7157–7166, 2021. 2

[17] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2022. 1, 2, 3, 4, 7

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9726–9735, 2020. 4

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 2, 7

[20] Lianghua Huang, Xin Zhao, and Kaiqi Huang. 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. 2, 4, 5

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012. 2

[22] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Proceedings of the European Conference on Computer Vision*, pages 765–781, 2018. 4

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998. 2

[24] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019. 2, 4, 5

[25] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018. 2, 3, 4

[26] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Proceedings of the European Conference on Computer Vision*, pages 740–755, 2014. 5

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations*, pages 1–11, 2019. 5

[28] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3074–3082, 2015. 1

[29] Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. Transforming model prediction for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–18, 2022. 6

[30] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for UAV tracking. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Proceedings of the European Conference on Computer Vision*, pages 445–461, 2016. 2, 4

[31] Matthias Müller, Adel Bibi, Silvio Giancola, Salman Al-Subaihi, and Bernard Ghanem. TrackingNet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European Conference on Computer Vision*, pages 310–327, 2018. 2, 4, 5

[32] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. 2

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. ImageNet: Large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1, 2, 5

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *Proceedings of the International Conference on Learning Representations*, pages 1–14, 2015. 2

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 2

[36] Paul Voigtlaender, Jonathon Luiten, Philip H. S. Torr, and Bastian Leibe. Siam R-CNN: visual tracking by re-detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6577–6587, 2020. 2, 6

[37] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3119–3127, 2015. 1

[38] Naiyan Wang, Jianping Shi, Dit-Yan Yeung, and Jiaya Jia. Understanding and diagnosing visual tracking systems. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3101–3109, 2015. 1

[39] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1571–1580, 2021. 2, 6

[40] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1328–1338, 2019. 2

[41] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 13763–13773, 2021. 2, 4

[42] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8741–8750, 2021. 2

[43] Zhongdao Wang, Hengshuang Zhao, Ya-Li Li, Shengjin Wang, Philip Torr, and Luca Bertinetto. Do different tracking tasks require different appearance models? In *Advances in Neural Information Processing Systems*, pages 1–15, 2021. 1, 4

[44] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 4

[45] Fei Xie, Chunyu Wang, Guangting Wang, Yue Cao, Wankou Yang, and Wenjun Zeng. Correlation-aware deep tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–15, 2022. 2

[46] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12549–12556, 2020. 2, 6

[47] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10448–10457, 2021. 2, 3, 6

[48] Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint feature learning and relation modeling for tracking: A one-stream framework. In *Proceedings of the European Conference on Computer Vision*, pages 341–357, 2022. 2

[49] Zhipeng Zhang and Houwen Peng. Deeper and wider siamese networks for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4591–4600, 2019. 2

[50] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *Proceedings of the European Conference on Computer Vision*, pages 771–787, 2020. 2, 3, 4, 5, 6

[51] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *Proceedings of the International Conference on Learning Representations*, pages 1–11, 2021. 2