Bridging the Gap Between Detection and Tracking: A Unified Approach

Lianghua Huang[1,2] Xin Zhao[1,2*] Kaiqi Huang[1,2,3]
[1]CRISE, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
huanglianghua2017@ia.ac.cn, {xzhao, kqhuang}@nlpr.ia.ac.cn

Abstract

Object detection models have been a source of inspiration for many tracking-by-detection algorithms over the past decade. Recent deep trackers borrow designs or modules from the latest object detection methods, such as bounding box regression, RPN and ROI pooling, and can deliver impressive performance. In this paper, instead of redesigning a new tracking-by-detection algorithm, we aim to explore a general framework for building trackers directly upon almost any advanced object detector. To achieve this, three key gaps must be bridged: (1) Object detectors are class-specific, while trackers are class-agnostic. (2) Object detectors do not differentiate intra-class instances, while this is a critical capability of a tracker. (3) Temporal cues are important for stable long-term tracking while they are not considered in still-image detectors. To address the above issues, we first present a simple target-guidance module for guiding the detector to locate target-relevant objects. Then a meta-learner is adopted for the detector to fast learn and adapt a target-distractor classifier online. We further introduce an anchored updating strategy to alleviate the problem of overfitting. The framework is instantiated on SSD [40] and FasterRCNN [15], the typical one- and two-stage detectors, respectively. Experiments on OTB, UAV123 and NfS have verified our framework and show that our trackers can benefit from deeper backbone networks, as opposed to many recent trackers.

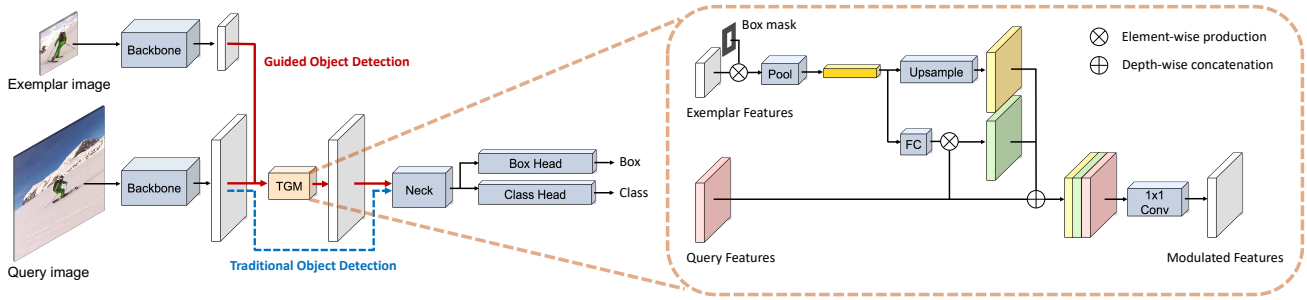
1. Introduction

Visual object tracking refers to the task of sequentially locating a specified moving object in a video, given only its initial state. It is currently an active research area of computer vision and significant progresses have been made in recent years [57, 31, 30, 12]. However, the task is still challenging due to several factors such as object interactions, cluttered background, occlusions and target deformation [43, 24].

Accurate localization of specified objects in complex scenes, as well as distinguishing different objects are crucial for tracking, which, at the same time, are extensively studied in the area of object detection [15, 37, 39]. In effect, object detection algorithms have been the inspiration of many popular trackers. For example, the correlation filters are previously applied in object detection (e.g., UMACE [41] and ASEF [6]) and later improved and adapted for tracking [5, 22, 23, 9, 8]. Similar examples include the bounding box regression [13, 49] used in MDNet [45, 26] and GOTURN [21], structured output SVM [4] applied in Struck [17], region proposal networks [15] used in SiamRPN [34, 59, 32] and the precise ROI pooling [25] used in the ATOM tracker [7], to name a few. Detection modules may either improve the trackers in terms of higher localization precision [13, 25] and/or better discriminability against occlusions and background clutters [4, 6, 15].

The goal of this work is not to redesign a new detection-based tracker. Instead, we would like to explore a simple and universal framework for building trackers upon advanced deep detectors. Our motivations for such framework are three fold: 1) Detection algorithms are specialized in precisely locating and differentiating objects in complex scenes, which may lead to more accuracy and robust trackers. 2) Reusing detection models reduces duplicated works in tracking, thus we can focus more on tracking-specific issues, such as target domain adaptation and temporal dependencies. 3) Such framework potentially enables multi-task models, i.e. joint detection and tracking of visual objects, which is close to the industry's needs. However, developing such a framework is not straightforward and several issues must be addressed. First, object detection only works on specific classes while a tracker is supposed to track arbitrary moving objects [30, 24]. Second, a detector cannot differentiate intra-class instances, which is, however, a crucial ability of a robust tracker. Finally, temporal cues in videos are important for stable tracking while they are not

*Corresponding author



(a) (b)

Figure 1: (a) The overall architecture of our tracking-by-detection framework. The architecture consists of two branches, one for generating target features as guidance while the other is an ordinary object detector. The two branches are bridged through a Target-Guidance Module (TGM). The blue dotted line represents traditional object detection process, while the red arrow denotes the procedure of the proposed guided object detection. (b) The outline of TGM. The input to the module is the exemplar and search image features and it output a modulated feature map with target information incorporated. The follow-up detection process remains intact. Note the detection model in (a) can be replaced by almost any modern object detectors.

considered in object detection approaches.

To address the above issues, in this work, we consider tracking as a joint task of *one-shot object detection* and *few-shot instance classification*. The former is a class-level subtask that finds all target-like candidates, while the latter is an instance-level subtask that tells apart the target from distractors. We propose a target-guidance module for *one-shot object detection*, which is built upon a base detector that consists of a backbone network and several top layers (i.e., detection heads and optional proposal and ROI pooling layers). The module encodes the target and search region features as well as their interactions in the backbone network as a guide to focus the base detector on a small set of target-like objects.

In the second subtask, an instance classifier will be learned to differentiate target from detected distractors. However, direct training of the classifier on such a small sample set will lead to significant overfitting. In this paper, we introduce the *Model-Agnostic Meta-Learning (MAML)* [14, 1] algorithm for solving the problem of few-shot learning. MAML learns sensitive initial parameters that can quickly adapt to new tasks with only a few samples and in a few training iterations. In brief, it *learns to finetune*. We found the detector's classification head to be a good initialization for the instance classifier, thus we simply replace the head by a *meta layer* with identical structure, and learns to quickly finetune it on different targets from large training data. Note in this manner, it only takes one pass of guided-detection to filter out both inter- and intra-class distractors.

The overall framework is outlined in Figure 1, where the base detector can be replaced by almost any modern object detection algorithm. In this paper, we instantiate the framework based on SSD [40] and FasterRCNN [15], the typical one- and two-stage detectors, respectively. The corresponding tracking models are shown in Figure 3 and Figure 4. For SSD, since the detection is performed on multiple layers with different resolutions, we also insert the target-guidance modules several times with different target resolutions. Main contributions of this paper are summarized as follows.

- To the best of our knowledge, we propose the first universal framework for building generic object trackers upon deep learning based object detectors.
- We propose to consider tracking as a joint task of *one-shot object detection* and *few-shot instance classification*, and we present an efficient target-guidance module and a meta-learner to handle respective subtasks.
- We develop a novel anchored updating strategy to avoid model drifts during online learning.
- We test our models on OTB [57], UAV123 [42] and NfS [28] benchmarks and report state-of-the-art tracking performance. Our ablation studies also show that our approach can benefit from deeper backbone networks, as opposed to many recent deep trackers [3, 34].

2. Related Works

Object Detection. Object detection is the task of locating and classifying objects in an image to a set of predefined classes [11, 13, 40, 15, 39]. State-of-the-art methods are based on deep neural networks, where usually a convolutional neural network (CNN) pretrained on ImageNet [10] is employed as the backbone architecture. For one-stage detectors [47, 40, 37], additional convolutional layers are appended to the backbone that perform dense prediction of object classes and locations. For two-stage detectors [15, 19],

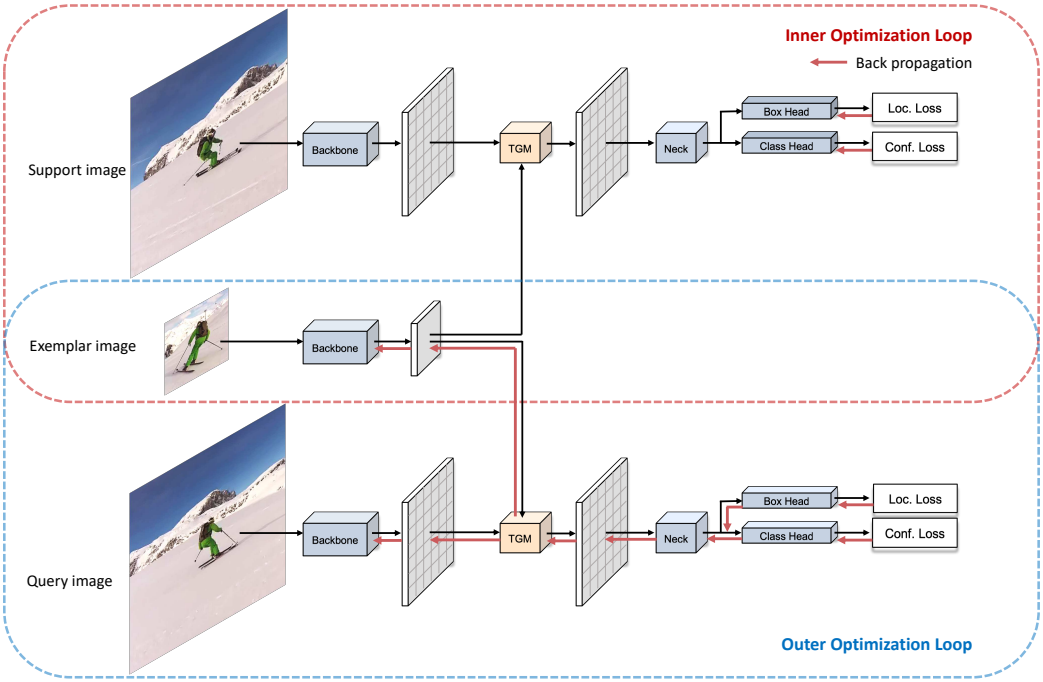


Figure 2: Overview of our training procedure. In the training stage, we sample triplets of *exemplar, support* and *query images* from video frames. Each triplet is chronologically sampled from a same video. We take the *exemplar image* as the guidance and perform detection on the *support* and *query images*. The losses calculated on the *support image* are used to finetune the *meta-layers* (i.e., the detector's heads) of our model, and we expect the updated model to generalize and perform well on the query image, which is realized by backpropagating all parameters of our model based on the losses on query image. The red arrows represent the backpropagation path during optimization. The *inner optimization loop* only updates head layer parameters, while the *outer optimization loop* updates all parameters in the architecture.

a region proposal network (RPN) is first applied to generate class-agnostic object candidates, then each candidate is independently cropped from feature maps and fed into head layers to estimate its class and location. Since shallow and deep CNN layers bear complementary representation, i.e. location-aware and semantic features, respectively, they are usually jointly used for detection in recent approaches [40, 37, 36]. Object detection models are trained on large image datasets, and once trained the models can only perform detection on specific classes. *Our method, on the other hand, generalizes the detection algorithms to any new objects with a single-shot of annotated sample*.

Tracing-by-Detection. Due to the high correlation between the two tasks, many tracking approaches have been motivated by object detection models over the past few decades. They include Tracking-Learning-Detection (TLD) [27], correlation filters based trackers [5, 22, 23, 9, 8], structured output tracking [17], siamese regression network [21], siamese region proposal networks [34, 32] and tracking by overlap minimization [7], to name a few. Modules and designs in object detection algorithms are adopted to either improve accuracy (e.g., bounding box regression used in [21, 45, 34]) and/or enhance stability (e.g., re-detection mechanism used in [27]) of tracking. In this work, rather than redesigning a new tracking-by-detection algorithm, we aim to explore the possibility of a general framework for building trackers upon deep detectors, thereby promoting module reusability and focusing research on tracking-specific issues.

Few-shot Learning. Few-shot learning aims to learn generic representation that can be transferred to novel tasks with only a handful of samples. There are several paradigms on few-shot learning, including metric learning [29, 51, 52], network parameter prediction [55, 2], recurrent neural work that learns gradient updates [44, 48] and learning to finetune [14, 35, 1, 46]. Our work is most related to *learning to finetune* based methods, i.e. the *Model-Agnostic Meta Learning (MAML)* [14, 1] algorithm. MAML learns sensitive initial network parameters that can be quickly finetuned on new tasks with a few samples and in a few iterations. We choose MAML for its simplicity, and that it can work on different models without changing their architectures.

3. Proposed Method

3.1. Overview

In this work, we propose a universal framework for building trackers on detectors consisting of two components: 1) A target-guidance module that leads the base detector to find target-relevant objects; and 2) A few-shot instance classifier that distinguishes target from surrounding distractors. Specifically, the proposed target-guidance module encodes target and search region features as well as their interactions in the backbone, while keeping the remaining detection process intact. On the other hand, the few-shot classifier is developed based on the *Model-Agnostic Meta Learning (MAML)* algorithm [14, 1]. MAML learns sensitive initial parameters for fast finetuning from only a few samples. In our case, we use the detector's classification head as the initial instance classifier, and learns to finetune it on specific targets from large training data. In this manner, the few-shot learning module adds no new parameters to the detector, and during test, one pass of guided-detection is necessary to filter out both inter-class and intra-class distractors. The overall architecture of our framework is shown in Figure 1, while its training procedure is outlined in Figure 2.

In the training phase, we sample triplets from video frames for model optimization. Each triplet consists of three cropped images sampled chronologically from a video, namely the *exemplar, support* and *query images*. The *exemplar image* represents the target guidance, telling the detector "where to look". The *support image* is used to finetune the detector's classification head, as mentioned in the above paragraph, and we expect the updated model to perform well on the *query image*. Finetuning on the *support image* is called *inner optimization loop*, while optimizing on the *query image* is termed *outer optimization loop*. Since the inner and outer optimizations are conducted on different frames, the generalization capability is thereby enhanced during the offline training procedure. The training process of our model is shown in Figure 2, where the complete framework can be trained from end to end.

The rest of this section is organized as follows. Section 3.2 and Section 3.3 describe our target-guidance module and few-shot learner, respectively. Section 3.4 details the online tracking procedure, and we introduce two instantiation examples of our framework based on SSD [40] and FasterRCNN [15] in Section 3.5.

3.2. Target-Guided Object Detection

We introduce a target-guidance module for *one-shot object detection* that is pluggable, in the sense that it does not alter the overall architecture of its base detector. The module takes as input the target and search region features, and outputs a feature map of the same size as the detector's backbone. The follow-up detection procedure remains unchanged. Specifically, the module first performs ROI pooling on target features, followed by a convolutional layer to convert the output to a modulator of size $C \times 1 \times 1$, where C is the number of feature channels; then the modulator is used to re-weight the feature channels of the search image. Afterwards, the original and modulated search region features, as well as the upsampled target global representation, are concatenated and then fed into a 1×1 convolutional layer to merge the features. In this manner, the backbone network encodes both the target and search region features as well as their interactions, providing sufficient information for subsequent detection. The structure of the module is shown in Figure 1 (b).

We adopt the same loss functions as the base detector, i.e., cross entropy loss for classification and smooth l_1 loss for bounding box regression [40, 15]. The losses are computed on the outputs of the RPN (for one- and two- stage detectors) and optional ROI layers (for two-stage detectors). We use Online Hard Negatives Mining (OHNM) [40, 15] to accelerate training and improve performance. Note for two-stage detectors, since we emphasize more on recall in RPN stage and delay the discrimination to ROI layers, we use $3:1$ as the positive-negative weights when calculating the classification loss of RPN.

3.3. Few-Shot Learning for Domain Adaptation

While the target-guidance module proposed in Section 3.2 can focus the detector on target-relevant objects, we found in our early experiments that it remains difficult for the module to distinguish between these detected objects. We hypothesis the main reason to be that the surrounding negatives are not considered in the guidance, thus weakening the discriminability of the detector. To compensate for this, we propose to explicitly learn a classifier on the small set of samples. However, directly training on such small data from scratch is time-consuming and can lead to severe overfitting. Instead, in this work, we solve the problem using few-shot learning [14, 35, 29], which aims to learn transferrable knowledge from large training data that can be generalized to new tasks with only a handful of samples.

Specifically, we adopt the *Model-Agnostic Meta Learning (MAML)* algorithm [14] to train the target-distractor classifier. MAML learns network initialization that can be quickly finetuned on new, unseen tasks with a few training samples and in a few iterations. In brief, it *learns to finetune*. Specific to our model, we found the detector's classification head to be a good initialization for the target-distractor classifier, thus we replace the head layer with an identical *meta layer*, and learns to finetune it on specific targets from large training data. Formally, in the training stage, we sample triplets $\{z_b, s_b, q_b\}_{b=1}^{B}$ from video frames, where B is the batch size, z_b, s_b, q_b represent three cropped

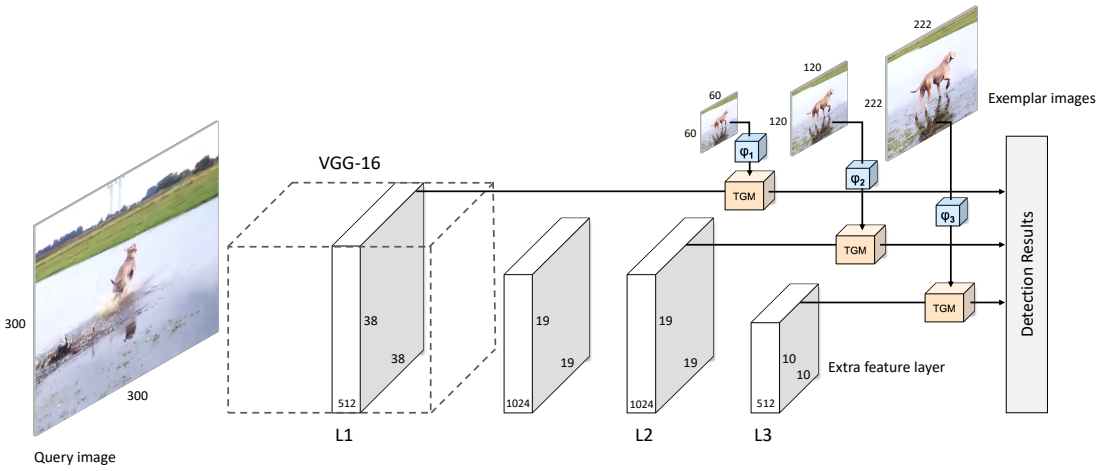


Figure 3: Instantiation of our framework on SSD [40]. We employ SSD with VGG-16 [50] as the backbone. The original SSD performs object detection on 6 different convolutional layers with increased receptive fields, with each being responsible for specific sized objects. In our work, we only use its first 3 backbone layers, denoting as L1, L2 and L3 in the figure. The target-guidance modules are appended to each layer, with increased guidance image resolutions that are consistent with the receptive fields. Operators φ_1, φ_2 and φ_3 represent extracting features at L1, L2 and L3 layers.

images sampled chronologically from a video, namely the *exemplar, support* and *query images*. For the guided detector h_θ with meta-parameter θ, we expect to learn an initial $\theta = \theta_0$ such that, after N steps of gradient update on *support set* (z_b, s_b) to obtain θ_N, the detector performs well on *query set* (z_b, q_b). The ith gradient update step on (z_b, s_b) can be expressed as:

$$\theta_i^b = \theta_{i-1}^b - \alpha \nabla_\theta \mathcal{L}_{(z_b,s_b)}(h_{\theta_{i-1}^b}), \tag{1}$$

where α denotes the learning rate and $\mathcal{L}_{(z_b,s_b)}(h_{\theta_{i-1}^b})$ is the loss computed on support set (z_b, s_b) after $(i-1)$ steps of gradient update. In our case, $\mathcal{L}_{(z_b,s_b)}$ is the classification loss. We can then define the *meta-loss* as:

$$\mathcal{L}_{meta}(\theta_0) = \sum_{i=1}^{B} \mathcal{L}_{(z_b,q_b)}(h_{\theta_N^b(\theta_0)}), \tag{2}$$

where we have explicitly denoted the dependence of θ_N^b on θ_0. $\mathcal{L}_{meta}(\theta_0)$ measures the quality of θ_0 in terms of the total loss of using the initialization across all triplets in the batch. The resulting update for the meta-parameter θ_0 can be expressed as:

$$\theta_0 = \theta_0 - \beta \nabla_\theta \sum_{b=1}^{B} \mathcal{L}_{(z_b,q_b)}(h_{\theta_N^b(\theta_0)}), \tag{3}$$

where β denotes the learning rate for θ_0 and $\mathcal{L}_{(z_b,q_b)}$ is the loss on query set (z_b, q_b). We refer to Eq. (1) as the *inner optimization loop* while Eq. (3) as the *outer optimization loop*. Since finetuning and loss evaluation are conducted on different samples (support and query sets, respectively), the generalization ability of the finetuning is thereby ensured.

An overview of the training procedure is shown in Figure 2. We found in our experiments that, finetuning both classification and regression heads during tracking leads to better performance than adjusting the classification head along, while only slighly increasing the computational costs. Section 4.2 offers detailed quantitative analysis. In this case, the θ_0 in the above formulation represents the parameters of the detector's head layers. We directly employ the detector's loss for optimizing θ_0. Since these settings are directly compatible with most base detectors, the complete framework can be trained from end to end, where only θ_0 is updated during both *inner* and *outer optimization loops* while other parameters of the detector are only updated during *outer optimization loop*.

3.4. Online Tracking Approach

Online Learning. During tracking, we adopt the few-shot learning algorithm described in Section 3.3 to adapt the guided detection model online. Specifically, the head parameters of the detector is updated using Eq. (1) at the first frame as well as for every T frames with online collected samples. Since the few-shot learner has learned fast convergence from a little data, the finetuning is efficient. At the first frame, we use random horizontal flipping and random cropping to generate 16 training samples, and finetune the detector's heads with $N_1 = 5$ training iterations. For other frames, the training samples are collected one per frame, and during updating, we use $N_r = 1$ training iterations to adjust the model.

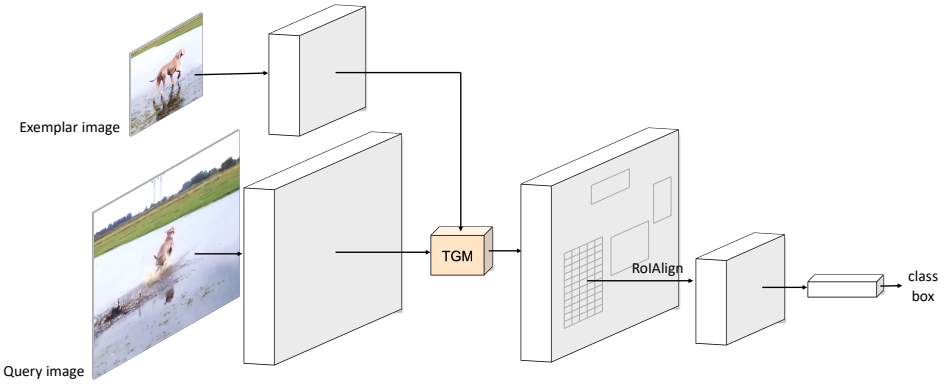


Figure 4: Instantiation of our framework on FasterRCNN [15]. The exemplar and query images are fed into the backbone and bridged using the target-guidance module, while the subsequent region proposal and RoI classification and regression procedure is kept unchanged. We evaluate the model with either VGG [50] or ResNet [20] as the backbone.

Anchored Updating. Although a smaller N_r is set for online updating, there is still a risk of overfitting since the generalization capability of *continuous learning* is never ensured during the offline training stage of the meta-learner. We thereby introduce an anchored updating strategy to alleviate the overfitting, motivated by the anchor loss [33]. Specifically, the parameters learned at the first frame, i.e., $\theta_1 = f(\theta_0; z_1, s_1)$ where $f(\theta; z, s) = \theta - \alpha \nabla_\theta \mathcal{L}_{(z,s)}(h_\theta)$, are stored throughout the tracking procedure. When fine-tuning at step t, the updated parameters θ_t is defined as a composition of updates from the last checkponit θ_{t-1} as well as from the initial parameters θ_0:

$$\theta_t = \lambda f(\theta_0; z_t, s_t) + (1 - \lambda) f(\theta_{t-1}; z_t, s_t). \tag{4}$$

We call θ_0 the anchor parameters. Through Eq. 4, the anchor parameters have fixed weight during online optimization, thus the overfitting problem can be alleviated.

Target Localization. We search the target location using the detection outputs. Specifically, with the detection results before non-maximum suppression (NMS), we cast penalization on position changes with a cosine window, and on scale and aspect ratio changes with a unnormalized Laplace function [34]:

$$p_{sr} = e^{-\frac{|d_r + d_s - 1|}{\sigma}}, \tag{5}$$

where d_r and d_s represent aspect ratio and scale changes, while σ is a scaling factor setting to 0.55 in our experiments. We rank the detections based on their re-weighted scores, and find the best detection result B^*. The target center is updated as the center of B^*, while the target size is smoothly updated to B^* with a learning rate of 0.275.

3.5. Instatiation on SSD and FasterRCNN

While our modules are universal and applicable to different object detectors, in this paper, we instantiate our framework on typical object detection models: SSD [40] and FasterRCNN [15]. SSD is a one-stage object detector that utilizes a single fully convolutional network to predict objects' classes and bounding boxes at dense spatial locations. By contrast, FasterRCNN is a two-stage object detector. It first generates class-agnostic object proposals using a region proposal network (RPN), then these proposals are accurately cropped from feature maps and fed to another network to predict their classes and refined locations.

Our tracker based on SSD is outlined in Figure 3. We use VGG-16 [50] as the backbone. The original SSD detects objects at 6 different backbone layers with increased receptive fields, with each being responsible for detecting objects of a specific range of sizes. In this work, we only use the first 3 backbone layers, with the base object sizes ranging from 30, 60 to 111 pixels. We crop the exemplar image as a square with twice size of the target, in order to introduce context information. Then we rescale it to three different resolutions, i.e., $60 \times 60, 120 \times 120$ and 222×222 according to the base sizes of SSD. The resized exemplar images are served as the detector's guidance at different backbone layers. The search region is set to 5 times the target size and is resized to 300×300. We use three target-guidance modules to bridge the exemplars and the detector at multiple layers. During the inner optimization loop (See Section 3.3), only the detector's head layer parameters are updated.

The FasterRCNN based tracker is shown in Figure 4. Unless specified in the ablation study, we use ResNet-50 [20] as its backbone. The search region is 5 times the target size and is rescaled to 480×480. The exemplar image is of twice the target size, and is rescaled to 192×192. The target-guidance module takes the exemplar and search region features as input and output a feature map with the same size as FasterRCNN's backbone network. The rest detection procedure remains unchanged. During inner optimization loop, only the parameters of RoI head layers are

Table 1: Impact of each component in our method on the tracking performance on OTB-2013 dataset. We compare few-shot learning with brute-force gradient descent, and assess the effectiveness of the anchored updating scheme, finetuning bounding box regression head and multi-resolution guidance images.

	Baseline (SSD)	GD	No Anch. Updating	No Reg. Finetuning	No Multi-Res.
AUC	**0.637**	0.551	0.612	0.609	0.629
OP	**0.813**	0.698	0.780	0.773	0.807

Table 2: Impact of backbone networks on tracking performance, evaluated on OTB-2013. The results show that our tracking performance improves as the network gets deeper from VGG-16 to ResNet-50, while further increasing network depth by using ResNet-101 does not bring improvements.

	VGG-16	ResNet-34	ResNet-50	ResNet-101
AUC	0.639	0.647	**0.656**	0.642
OP	0.793	0.810	**0.829**	0.825

finetuned.

4. Experiments

We conduct a comprehensive evaluation of the proposed tracking framework on four challenging datasets: Object Tracking Benchmark (OTB) of version 2013 [56] and 2015 [57], UAV123 [42] and Need for Speed (NfS) [28]. We also carry out experiments to analyze the effectiveness of our components, as well as the impact of network depth on tracking performance.

4.1. Implementation Details

Training. The base detection models are initialized with the weights pretrained on COCO dataset [38]. We then train our guided models using the GOT-10k [24] dataset, which is a recently proposed tracking dataset that consists of around 10,000 videos belonging to over 560 object classes.

Optimization. We employ stochastic gradient descent (SGD) with 32 triplets in a batch to train our models. The whole architecture is trained from end-to-end for 50,000 iterations with learning rate exponentially decayed from 0.01 to 0.0005. The learning rate of the *inner optimization loop* in Eq. (1) is fixed to $\alpha = 0.05$. We use 5×10^{-4} for weight decay and 0.9 for momentum. Our tracker is implemented in Python, using PyTorch. Our SSD based tracker runs at over 10 fps on an NVIDIA GTX-1080 GPU, while the FasterRCNN based tracker with ResNet-50 backbone runs at 3 fps.

Table 3: State-of-the-art comparison on OTB-2013 and OTB-2015 datasets in terms of the area-under-the-curve (AUC) metric.

	OTB-2013	OTB-2015
SiamFC [3]	0.607	0.582
CFNet [53]	0.611	0.568
DSiam [16]	0.656	-
RASNet [54]	0.670	0.642
SiamRPN [34]	-	0.637
DaSiamRPN [59]	-	**0.658**
SA-Siam [18]	**0.677**	0.657
ECO [8]	**0.709**	**0.694**
MemTrack [58]	0.642	0.626
Ours (SSD)	**0.637**	**0.620**
Ours (FRCNN)	**0.656**	**0.647**

Table 4: State-of-the-art comparison on UAV123 and NfS datasets in terms of the area-under-the-curve (AUC) metric.

	UAV123	NfS
SiamFC [3]	0.523	-
SiamRPN [34]	0.571	-
DaSiamRPN [59]	**0.584**	0.395
ECO [8]	0.537	0.470
Ours (SSD)	**0.531**	**0.491**
Ours (FRCNN)	**0.586**	**0.515**

4.2. Ablation Study

We study the impact of various design choices presented in Section 3. The analysis is performed on OTB-2013 [56] dataset, which consists of 51 tracking videos. We evaluate the trackers based on overlap precision (OP) and area-under-the-curve (AUC) metrics. OP_τ is defined as the percentage of successfully tracked frames with bounding box overlap larger than a threshold τ, while AUC is defined as $\text{AUC} = \int_0^1 \text{OP}_u du$. We use $\tau = 0.5$ for OP in the following evaluation.

Few-shot Learning. We compare our few-shot learning algorithm, introduced in Section 3.3, with a brute force gradient descent (GD) method for model finetuning. For GD, we test with different iterations and learning rates on a validation set of 10 videos sampled from OTB-2015, which are not overlapped with OTB-2013, and find the best setting with the highest AUC for evaluation. Table 1 lists the results. The tracker employing GD decreases the AUC score by 8.6% compared to our baseline. This demonstrates the effectiveness of our learned finetuning, which generalizes better than the brute-force GD. We can also conclude from Table 1 that updating both classification and bounding box regression layers lead to better performance, compared to finetuning the classification head along.

Anchored Updating. We compare the tracker with and without anchored updating. Table 1 shows that introducing anchored updating significantly improves the AUC of our tracker by around 2.5% and OP by 3.3%, which verifies the effectiveness of our anchored updating mechanism.

Guidance Images. When object detection is performed on multiple backbone layers with different resolutions, as in SSD[40], we append our target-guidance module to each of the layer. We compare the performance of using multiple sized guidance images, that are consistent with the layers' receptive fields, with that using same sized guidance image across all layers. The performance comparison is shown in Table 1. By comparing the baseline results with *No. Multi-Res.*, we found that using different exemplar sizes improves performance in terms of both AUC and OP.

Backbone Network Depth. We test our FasterRCNN based tracker with different backbone networks that are increasingly deeper. Four backbones are evaluated, including VGG-16 [50], ResNet-34, ResNet-50 and ResNet-101 [20]. We re-train the trackers with different backbones, and evaluate the performance on OTB-2013. Results are listed in Table 2. Our tracker equipped with ResNet-50 backbone performs significantly better than the VGG-16 and ResNet-34 counterparts, although going deeper to ResNet-101 does not bring further improvement. This suggests that our model can benefit from deeper network, which is opposed to many recent deep trackers [8, 3, 34] where introducing deeper network leads to similar or even worse performance.

4.3. State-of-the-art Comparison

We summarize the comparison of our trackers, equipped with SSD and FasterRCNN (denote as FRCNN in Table 3 and Table 4) detectors, with the state-of-the-art approaches on four challenging tracking datasets.

OTB2013 [56]: The dataset consists of 51 videos and the performance is evaluated using AUC (introduced in Section 4.2). Table 3 shows the results. SiamFC employs a siamese fully convolutional network to perform dense comparison between target and candidates, while SiamRPN extends it with dense bounding box regression. SA-Siam is a two branch SiamFC that introduces the semantic feature branch. DSiam and RASNet extend SiamFC with dynamic updating scheme and attention modules, while CFNet combines the siamese structure with correlation filters. ECO is a correlation filters based tracker that utilizes multi-level features and compact filters, while MemTrack performs online learning based on long-short term memory networks. Our FasterRCNN based tracker achieves an AUC of 65.6%, which is on par with RASNet (67%) and DSiam (65.6%), while outperforms MemTrack and SiamFC.

OTB2015 [57]: OTB2015 is an extension of OTB2013 and it contains 100 tracking videos. The comparison of state-of-the-art trackers on OTB2015 is listed in Table 3. Our FasterRCNN based tracker achieves an AUC of 64.7%, which outperforms SiamRPN by around 1% and MemTrack by around 2.1%. The SSD based method also achieves a comparable performance that is close to the MemTrack. The competitive performance of our trackers on OTB2013 and OTB2015 benchmarks have verified the effectiveness of our framework.

UAV123 [42]: The dataset consists of 123 aerial videos captured from the UAV platform. Table 4 shows the results. The DaSiamRPN tracker achieves a competitive AUC score of 58.4% that improves SiamRPN by around 1.3%. Our FasterRCNN based tracker achieves an AUC of 58.6%, which outperforms DaSiamRPN. Our SSD tracker can also obtain an AUC score of 53.1% that is close to the performance of the more complex ECO tracker.

Need for Speed [28]: The dataset consists of 100 videos captured with high frame-rate cameras. We evaluate the performance on its 30 FPS version. The results are shown in Table 4. ECO and DaSiamRPN achieve AUC of 39.5% and 47%, respectively, while our trackers significantly outperforms them. Our SSD based tracker obtains an absolution gain on AUC by 2.1%, while the FasterRCNN based tracker improves ECO by 4.5%.

5. Conclusion

We propose the first universal framework for constructing generic object trackers upon deep learning based object detectors. The tracking problem is decomposed into a joint task of *one-shot object detection* and *few-shot instance classification*. We introduce a light-weighted target-guidance module for *one-shot object detection*, which encodes target features in the detector's backbone while keeping the follow-up detection stages intact. For the second subtask, we employ a meta-learning algorithm to learn fast convergence of the classifier on a little data. The framework aims to promote module reusability and focusing the research on tracking-specific issues. Our models instantiated on SSD and FasterRCNN show the state-of-the-art performance on four challenging benchmarks. We further show in our ablation study that our FasterRCNN based tracker can benefit from deeper backbones.

Acknowledgement

This work is supported in part by the National Natural Science Foundation of China (Grant No. 61602485 and No. 61673375), the National Key Research and Development Program of China (Grant No. 2016YFB1001005), and the Projects of Chinese Academy of Science (Grant No. QYZDB-SSW-JSC006).

References

[1] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

[2] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, pages 523–531, 2016.

[3] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[4] Matthew B Blaschko and Christoph H Lampert. Learning to localize objects with structured output regression. In *European conference on computer vision*, pages 2–15. Springer, 2008.

[5] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550. IEEE, 2010.

[6] David S Bolme, Bruce A Draper, and J Ross Beveridge. Average of synthetic exact filters. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2105–2112. IEEE, 2009.

[7] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. *arXiv preprint arXiv:1811.07628*, 2018.

[8] 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 6638–6646, 2017.

[9] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision*, pages 472–488. Springer, 2016.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[11] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[12] 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. *arXiv preprint arXiv:1809.07845*, 2018.

[13] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[16] Qing Guo, Feng Wei, Ce Zhou, Huang Rui, and Wang Song. Learning dynamic siamese network for visual object tracking. In *IEEE International Conference on Computer Vision*, 2017.

[17] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2096–2109, 2016.

[18] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4834–4843, 2018.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[20] 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.

[21] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[22] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European conference on computer vision*, pages 702–715. Springer, 2012.

[23] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596, 2015.

[24] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *arXiv preprint arXiv:1810.11981*, 2018.

[25] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018.

[26] Ilchae Jung, Jeany Son, Mooyeol Baek, and Bohyung Han. Real-time mdnet. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 83–98, 2018.

[27] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012.

[28] 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 1125–1134, 2017.

[29] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[30] Matej Kristan and et. al. Leonardis, Ales. The sixth visual object tracking vot2018 challenge results. In *European Conference on Computer Vision*, pages 3–53. Springer, 2018.

[31] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 1–23, 2015.

[32] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. *arXiv preprint arXiv:1812.11703*, 2018.

[33] Bi Li, Wenxuan Xie, Wenjun Zeng, and Wenyu Liu. Learning to update for object tracking with recurrent meta-learner. *IEEE Transactions on Image Processing*, 2019.

[34] 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.

[35] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[36] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[37] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[38] 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 *European conference on computer vision*, pages 740–755. Springer, 2014.

[39] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*, 2018.

[40] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[41] Abhijit Mahalanobis, BVK Vijaya Kumar, Sewoong Song, SRF Sims, and JF Epperson. Unconstrained correlation filters. *Applied Optics*, 33(17):3751–3759, 1994.

[42] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *European conference on computer vision*, pages 445–461. Springer, 2016.

[43] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 300–317, 2018.

[44] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2554–2563. JMLR. org, 2017.

[45] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2016.

[46] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[47] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[48] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[49] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[50] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[51] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[52] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[53] Jack Valmadre, Luca Bertinetto, João Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2805–2813, 2017.

[54] Qiang Wang, Zhu Teng, Junliang Xing, Jin Gao, Weiming Hu, and Stephen Maybank. Learning attentions: residual attentional siamese network for high performance online visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4854–4863, 2018.

[55] Yu-Xiong Wang and Martial Hebert. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision*, pages 616–634. Springer, 2016.

[56] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.

[57] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[58] Tianyu Yang and Antoni B. Chan. Learning dynamic memory networks for object tracking. 2018.

[59] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 101–117, 2018.