IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 8, 2018, accepted July 23, 2018, date of publication August 8, 2018, date of current version August 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2862878

Domain Adaptation Tracker With Global and Local Searching

FEI ZHAO[1], TING ZHANG[1,2], YI WU[3,4], (Member, IEEE), JINQIAO WANG[1], (Member, IEEE), AND MING TANG[1], (Member, IEEE)

[1]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing 100190, China
[2]Research and Development Center, China National Electronics Import and Export Corporation, Beijing 100036, China
[3]Department of Medicine, Indiana University School of Medicine, Indianapolis, IN 46202, USA
[4]School of Information Engineering, Nanjing Audit University, Nanjing 211815, China

Corresponding author: Jinqiao Wang (jqwang@nlpr.ia.ac.cn)

This work was supported by the Natural Science Foundation of China under Grant 61772527.

ABSTRACT For the convolutional neural network (CNN)-based trackers, most of them locate the target only within a local area, which makes the trackers hard to recapture the target after drifting into the background. Besides, most state-of-the-art trackers spend a large amount of time on training the CNN-based classification networks online to adapt to the current domain. In this paper, to address the two problems, we propose a robust domain adaptation tracker based on the CNNs. The proposed tracker contains three CNNs: a local location network (LL-Net), a global location network (GL-Net), and a domain adaptation classification network (DA-Net). For the former problem, if we come to the conclusion that the tracker drifts into the background based on the output of the LL-Net, we will search for the target in a global area of the current frame based on the GL-Net. For the latter problem, we propose a CNN-based DA-Net with a domain adaptation (DA) layer. By pre-training the DA-Net offline, the DA-Net can adapt to the current domain by only updating the parameters of the DA layer in one training iteration when the online training is triggered, which makes the tracker run five times faster than MDNet with comparable tracking performance. The experimental results show that our tracker performs favorably against the state-of-the-art trackers on three popular benchmarks.

INDEX TERMS Convolutional neural networks, domain adaptation, online training, visual tracking.

I. INTRODUCTION

Visual tracking, as a fundamental and challenging task in computer vision field, has been studied for a long time. The tracking algorithms have been applied in augmented reality, autonomous driving, intelligent surveillance, etc.

Because of scale variation, occlusion, fast motion, and many other challenges, there are lots of difficulties in visual tracking task. To deal with these challenges, the trackers must have two key abilities: one ability is the tracker should adapt to the appearance variations of the target which is being tracked; the other is the tracker should have the ability to distinguish the target from the background. Unfortunately, most of the trackers based on hand-craft features perform poorly with the two abilities [1]–[3].

With the help of deep neural networks, especially the convolutional neural networks (CNNs), huge progress has been made in computer vision field recently, such as image classification and recognition [4], object detection [5]–[7], segmentation [8], optical flow estimation [9], etc. The CNNs show the power both in regression and classification tasks, which are also helpful to the visual tracking task.

Because CNNs can learn powerful features from the large-scale training datasets, the trackers using CNNs can model the target robustly. In recent years, more and more trackers based on CNNs perform better than the ones with hand-crafted features. These CNNs based trackers can be divided into two categories roughly: offline trackers and online trackers. In this paper, the former category refers to the trackers whose models are trained before the tracking phase, and there is no parameter updating during the tracking phase. The latter category refers to the trackers whose models are updated during the tracking phase.

For the offline trackers [10]–[16], they learn to regress the location of the target within a search patch [11]–[16], or distinguish the target from the candidates [10]. For example, GOTURN [11] learns to regress the coordinates of the

target directly. SiamFC [12] learns to regress a response map which reflects the location of the target within a search patch. SINT [10] matches the target from the first frame with the candidates in a new frame, and the tracker returns the most similar pair by the learned matching function.

For the online trackers [17]–[24], they mainly contain the trackers using correlation filters with CNNs features [17], [18], and the trackers using CNNs based classifications to distinguish the target from the candidates [19]–[21], [24]. Besides, CREST [23] reformulates the correlation filters as a CNN, and ADNet [22] locates the target by sequentially actions which are learned by reinforcement learning.

Although these CNNs based trackers perform well on the popular benchmarks, they have some disadvantages as follows. First, most of these trackers locate the target within a local search patch. Once the tracker drifts into the background or moves fast, they can hardly recapture the target in the subsequent frames. Second, the offline trackers are sensitive to the appearance variations of the target. Third, although the online trackers are robust to the appearance variations of the target, the online training phase is time-consuming.

To address these problems, some trackers have been proposed. For example, EBT [25] locates the target based on the region proposals generated within the entire frame, which makes the tracker be not limited to a local search patch. But the region proposals contain the other objects with a high probability. To improve the online updating speed, ECO [17] uses a factorized convolution operator to reduce the number of parameters in the model, but it also a bottleneck.

In this paper, we propose a robust domain adaptation tracker to ameliorate these problems. Our tracker contains three CNNs: a local location network (LL-Net), a global location network (GL-Net), and a domain adaptation classification network (DA-Net). The LL-Net generates target candidates based on multiple target templates and a local search patch. Besides, we use the outputs of the LL-Net to detect whether the tracker drifts into the background. The GL-Net recaptures the target within the entire frame when the drifting is detected. Furthermore, we utilize the DA-Net which is updated online to distinguish the target from the candidates.

The main contributions of ours can be summarized as follows:

- We propose a global location network (GL-Net) to locate the target when the drifting is detected, and the GL-Net makes our tracker be not limited to a local search patch. Further, do not like the EBT [25], the GL-Net can be trained end-to-end on large-scale training datasets to learn a similarity function, which guides the GL-Net pay attention on the object which similar to the target being tracked. The GL-Net makes our tracker robust in tracking as shown in Figure 1.
- We propose a domain adaptation classification network (DA-Net) with a domain adaptation (DA) layer. By pre-training the DA-Net offline, we can only update the parameters of the DA layer in one training iteration during the online training phase, which makes the tracker run faster significantly comparing against MDNet [19] with comparable tracking performance as shown in Figure 1.
- We reduce the risk of over-fitting during the online training phase because we only update the DA layer with a few parameters.
- The experimental results on three popular benchmarks demonstrate the proposed tracker performs favorably against the state-of-the-art trackers.

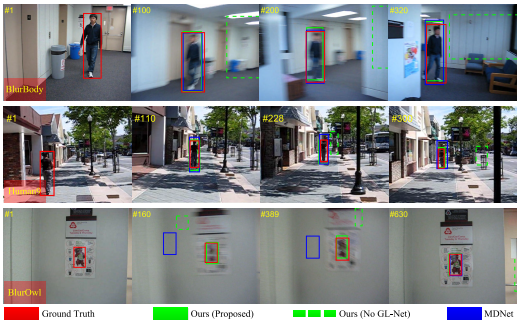

FIGURE 1. Tracking results on OTB. The proposed tracker is trained only one iteration when the online training is triggered, which makes it run faster than MDNet.

The rest of this paper is organized as follows. In Section II, we review some other CNNs based trackers related to ours firstly. Then in Section III, we described the proposed tracker, such as the architectures of the networks, training methods, and the overall tracking algorithm with implementation details. In Section IV, we show and discuss the tracking performance of the proposed tracker on three popular benchmarks. Section V is the conclusion of this paper.

II. RELATED WORK

In this section, we only discuss the CNNs based trackers related to ours. Here, we roughly divide these CNNs based trackers into two categories: offline trackers and online trackers.

A. OFFLINE TRACKERS

For the offline trackers, the CNNs were trained on training datasets before the tracking phase. During the tracking phase, the parameters of the CNNs would not be updated.

Held *et al.* [11] proposed the GOTURN which could run at 100 fps on GPU. The CNN in this tracker contained two parts: the convolutional layers and fully connected layers. The former learned to extract features of the input and the latter learned to regress the coordinates of the target within a search patch. The tracker could learn the relationship between appearance and motion.

Bertinetto *et al.* [12] proposed the SiamFC which could also run in real-time. This tracker learned a similarity function between a target patch and a search patch based on

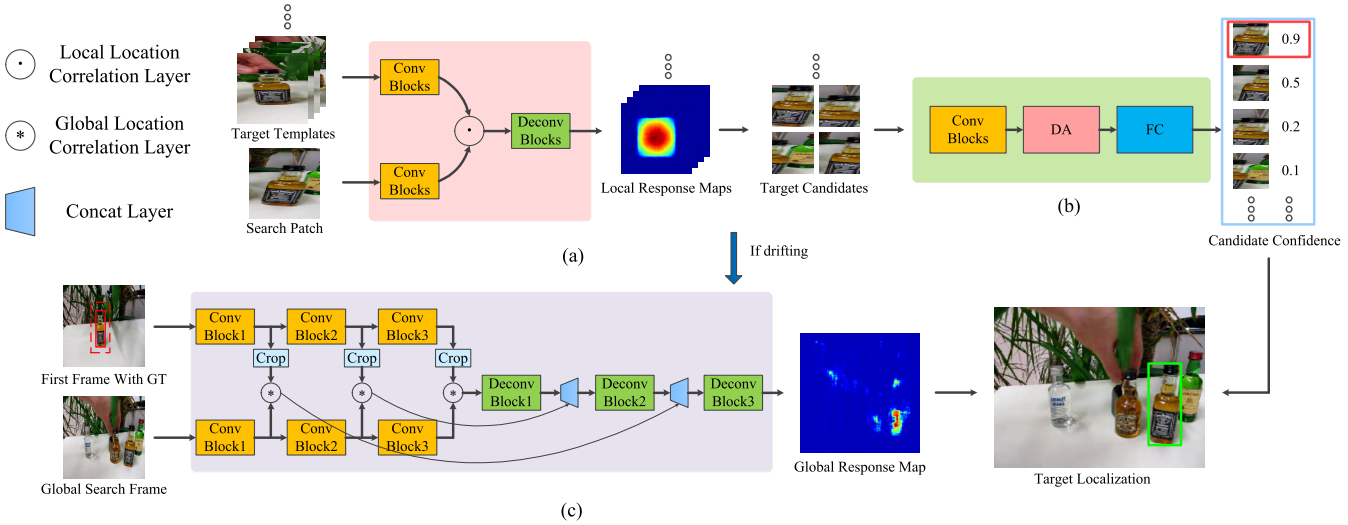

FIGURE 2. Pipeline of the proposed tracking algorithm. (a) Local location network (LL-Net). (b) Domain adaptation classification network (DA-Net). (c) Global location network (GL-Net).

a fully-convolutional network. The correlation layer made the tracker regress the response map efficiently which reflected the location of the target within the search patch.

Based on SiamFC [12], CFNet [13], SA-Siam [16] and EAST [15] were proposed recently. In CFNet, Valmadre *et al.* [13] proposed a differentiable layer which was trained as a correlation filter learner. In SA-Siam, He *et al.* [16] proposed a twofold Siamese network which was composed of a semantic branch and an appearance branch. The two branches were trained and learned the semantic features and appearance features separately. The heterogeneity of the features and the attention mechanism improved the tracking performance. In EAST, Huang *et al.* [15] proposed an adaptive approach for adaptive tracking with deep feature cascades. They trained an agent by reinforcement learning, which could decide whether to use the deeper features to locate the target.

Tao *et al.* [10] proposed the SINT which distinguished the most similar patch matching with the target appearing in the first frame from the candidates. Using a Siamese network, they learned a matching function offline. Because the tracker cropped large numbers of candidates and used optical flow, it could not run in real-time.

All of these trackers could not track the target moving fast because of the limited search area. Besides, due to no online training, they were sensitive to the appearance variations of the targets.

B. ONLINE TRACKERS

For the online trackers, one of the obvious difference to the offline trackers is that during the tracking phase, the parameters of the CNNs will be updated based on the first frame and the tracking results of the video sequences.

Nam and Han [19] proposed the MDNet which was composed of the shared layers and the domain-specific layers. The tracker was trained by multi-domain learning and achieved outstanding performance on the public benchmarks.

Based on the MDNet [19], SANet [26], TCNN [20], and BranchOut [24] were proposed recently. In SANet, Fan and Ling [26], modeled the structure of the target by a recurrent neural network (RNN) [27], and incorporated it into CNN to improve the robustness of the tracker. In TCNN, Nam *et al.* [20] proposed a tracker which managed multiple target appearance models in a tree structure. With the help of the tree structure, TCNN [20] could update the model smoothly along tree paths. In BranchOut, Han *et al.* [24] proposed a regularization technique of CNNs for visual tracking, which could learn the robust appearance of the target during the online training phase.

Yun *et al.* [22] proposed the ADNet which could locate the target by sequentially actions learned by deep reinforcement learning. The tracker reduced the computation complexity in tracking because it did not crop a large number of candidates.

Song *et al.* [23] proposed the CREST which modeled discriminative correlation filters as a CNN. Meanwhile, Song *et al.* [23] utilized the residual learning to take appearance variations of the target into account.

Like the offline trackers mentioned above, these online trackers cropped the candidates within a limited area, which made them hardly recapture the target once drifting into the background. Besides, they spent a significant amount of time in the online training phase.

III. PROPOSED TRACKER

The pipeline of our tracking algorithm is shown in Figure 2. During the tracking phase, we keep N target templates, which can be denoted as $\mathcal{P} = \{P_n\}_{n=1}^{N}$. We crop the search patch two times larger than the size of the target in the current frame, and the search patch is centered on the location of target tracked in the last frame. Within the search patch, the

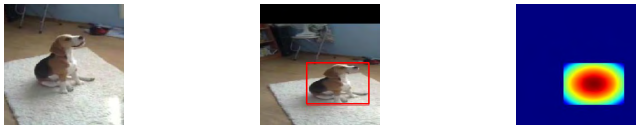

FIGURE 3. Training examples for the LL-Net (target patch, search patch, label). The target is in the red bounding box.

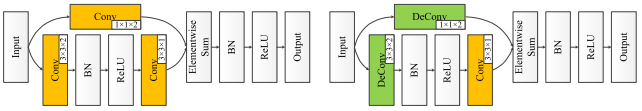

FIGURE 4. Convolutional block (left) and Deconvolutional block (right). "BN" denotes the Batch Normalization. The numbers indicate *kernel_width* × *kernel_height* × *stride*.

FIGURE 5. TRR in two situations. Each set contains the images: target patch, search patch and response map. (a) no drifting (TRR:0.294). (b) drifting (TRR:0.844).

LL-Net generates a response map R_n for each target template P_n first. Based on the response maps (the predicted location of the target) and the search patch, we crop the candidate targets as $\{X_n\}_{n=1}^{N}$. Second, we use the DA-Net to score all of the target candidates. Third, we detect whether the tracker drifts into the background based on the response map R^* corresponding to the candidate with the highest confidence score. If there is no drifting, we locate the target based on R^*. Otherwise, the GL-Net is used to search for the target appearing in the first frame within the global area of the current frame. In the drifting situation, we locate the target based on the global response map generated by the GL-Net. The proposed tracker consists of three CNNs: the LL-Net, the GL-Net, and the DA-Net. We will describe the details next.

A. LOCAL LOCATION NETWORK

We utilize the LL-Net to locate the target within a local area and detect whether the tracker drifts into the background based on the output of the LL-Net.

1) ARCHITECTURE

The LL-Net is a fully-convolutional Siamese network as shown in Figure 2(a). The input of the LL-Net contains two patches: target template patch and a search patch. The output is a response map which reflects the location of the target within the search patch. The sizes of the two patches of the input are $256 \times 256 \times 3$, and the size of the output is $256 \times 256 \times 1$.

Because the sizes of the target template patch and the search patch are the same, and in order to achieve high location precision, we apply the correlation layer which is proposed in [9]. In this paper, we call this layer as the *local location correlation layer* which is designated as $\odot$.

The "Conv Blocks" contains 4 convolutional blocks with the same architecture, and they share the same parameters in the two streams. The numbers of output feature maps are (32, 64, 128, 256), and the sizes are $(128^2, 64^2, 32^2, 16^2)$ respectively. The "Deconv Blocks" contains 4 deconvolutional blocks with the same architecture. The numbers of output feature maps are (128, 64, 32, 1), and the sizes of them are $(32^2, 64^2, 128^2, 256^2)$ respectively. As in ResNet [4], we design the architectures of these blocks as shown in Figure 4. In each block, the numbers of the channels are the same, and the differences are kernel size and stride.

2) OFFLINE TRAINING

We use ImageNet Video (VID) [28], ALOV300++ [29], UAV123 [30], and NUS-PRO [31] to train the proposed trackers from scratch, and we remove all of the videos that overlap with the testing data.

The training set consists of a target patch, a search patch, and a corresponding label, which are created as [32] except the label. In this paper, we define the label as

$$v(x,y) = \begin{cases} \dfrac{1}{2\pi\sqrt{wh}} e^{-(\frac{x^2}{2w^2} + fracy^2 2h^2)}, & \text{if } \|x - x_c\|_2 < 0.5w \\ & \text{and } \|y - y_c\|_2 < 0.5h \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $v(x, y)$ denotes the response value of the point at the coordinates (x, y) within the label. h denotes the height of the target, and w is the width. The center coordinates of the target are (x_c, y_c). $\|\cdot\|_2$ denotes the Euclidean distance. One set of training examples is shown in Figure 3. From left to right: target patch, search patch, and response label. We set the search patch 2 times the side length of the target for the LL-Net. We fill the area beyond the extent of the image with 0, and train the LL-Net with the $L2$ loss.

We utilize Adam [33] optimizer to train the LL-Net with Tensorflow [34]. The networks are trained for $1M$ iterations with learning rate $1e-4$ initially, and the batchsize is 50.

3) DRIFTING DETECTION

As shown in Figure 5, the appearance of the response map generated by the LL-Net changes obviously when the tracker drifts into the background. Based on this, we define the ratio of the area of the predicted target A to the area of the search patch as "Target-to-Response Ratio" (TRR), which is used to detect drifting. A is defined as

$$A = L_v^x \times L_v^y \tag{2}$$

where the L_v^x denotes the farthest distance in the x direction between any two elements with the response values higher than $v\ (= 0.06)$ within the generated response map. Similarly, L_v^y returns the farthest distance in y direction. When the

Algorithm 1 Proposed Tracking Algorithm

Input :
Pretrained local location network (LL-Net);
Pretrained global location network (GL-Net);
Pretrained domain adaptation classification network (DA-Net);
Initial target state Loc_1 (center location, width and height) in the first frame by the ground truth;

Output:
The predicted target state Loc_t^* in the t^{th} frame;

Initialize the target templates $\mathcal{P} = \{P_n\}_{n=1}^N$ by Loc_1;
Creating the training set T_1 from the first frame;
Training the "DA" layer of the DA-Net by training set T_1, $t \leftarrow 1$;
repeat
  For each target template in $\mathcal{P}$, predict a candidate target X_t^i by the LL-Net;
  Predict the confidence score for each candidate target X_t^i by DA-Net;
  Calculate the score s_t^* by Equation 3;
  Get the candidate target X_t^* whose score is s_t^*;
  Predict the response map R_t^* and location Loc_t^* by X_t^*;
  if $TRR(R_t^*) > 0.6$ **then**
    Locate the target by the GL-Net;
    Update the Loc_t^* by the result of GL-Net;
  end
  if $t\%30 == 0$ **then**
    Create training set T_t from the last 10 frames;
    Update the "DA" layer by T_t;
  end
  if $t\%14 == 0$ **then**
    Update the $\mathcal{P}$;
  end
  $t \leftarrow t + 1$;
until *the end of the sequence*;

TRR is higher than 0.6, the tracker will use the GL-Net to search for the target appearing in the first frame within the global area. In the ablation study, we compare against the "PSR" (Peak to Sidelobe Ratio) [35] in tracking.

During the tracking phase, we set the edge length of the search patch in the current frame two times as large as that of the target in the last frame. Meanwhile, during the offline training phase, the LL-Net learns to adapt to the translation and scale changes of the target. Based on the above observations, the TRR value is around 0.25 when no drifting is detected. If there is no target in the search patch, the maximum response would appear randomly in the search patch. This observation is shown in Figure 5. In a word, the high value of the TRR indicates that the drifting is detected. However, if the threshold of the TRR is set too high, some drifting situations will be missed. The experimental results show that 0.6 is a suitable value.

B. GLOBAL LOCATION NETWORK

We utilize the GL-Net to locate the target within the global area of the current frame when the drifting is detected.

1) ARCHITECTURE

The GL-Net is a fully-convolutional Siamese network as shown in Figure 2(c). The input of the GL-Net contains two patches: the entire first frame and the entire current frame. The output is a response map which reflects the location of the target within the current frame. The sizes of the two patches of input are $512 \times 512 \times 3$, and the size of the output is $512 \times 512 \times 1$.

Because the sizes of the two inputs of each $\circledast$ layer are different, and in order to achieve higher matching speed, we apply the correlation layer proposed in [12]. In this paper, we call this layer as the *global location correlation layer* which is designated as $\circledast$.

There are 3 convolutional blocks with the same architecture in each stream of the GL-Net. For these convolutional blocks, the numbers of output feature maps are (32, 64, 128), and the sizes are $(256^2, 128^2, 64^2)$ respectively. For the stream whose input is the first frame, we get the feature maps corresponding to the target by cropping them from each output of the convolutional block based on the ground truth location of the target. Here, the width and heights of the cropped feature maps are 1.5 times the ground truth. We resize the cropped feature maps as $(16^2, 8^2, 4^2)$, and we convolve them with the feature maps generated by the other stream by $\circledast$ layer. The outputs of the $\circledast$ layers are resized as $(256^2, 128^2, 64^2)$. There are 3 deconvolutional blocks with the same architecture. The numbers of output feature maps are (128, 64, 1), and the sizes of them are $(128^2, 256^2, 512^2)$ respectively. The architectures of these blocks are the same as in the LL-Net. In each block, the numbers of the channels are the same, and the differences are kernel size and stride.

The feature maps in the deep convolutional layers contain more semantic information, and the feature maps in the early layers contain more location information of the target. To take full advantage of this information, we use hierarchical convolution operation and merge these feature maps for the Deconv blocks as shown in Figure 2(c).

2) OFFLINE TRAINING

The training datasets are similar to the LL-Net. In each training set, we use the two frames extracted from the same video sequence instead of the local patches, and they both contain the same object. The two frames are at most 10 frames apart. The corresponding label is created as Equation 1. We train the GL-Net with the $L2$ loss.

We utilize Adam [33] optimizer to train the GL-Net with Tensorflow [34]. The networks are trained for

$1M$ iterations with learning rate $1e-4$ initially, and the batchsize is 12.

3) TARGET RECAPTURING

If the drifting is detected, we use the GL-Net to locate the target within the global area of the current frame as shown in Figure 2. The center of the target is located at the point with the maximum value within the response map, and the size is the same as in the last frame.

C. DOMAIN ADAPTATION CLASSIFICATION NETWORK

Nam and Han [19] viewed each video sequence as a domain, and trained the CNN based classification network by multi-domain learning. In this way, the network in [19] not only learned whether a certain object was the tracking target in the current video sequence but also learned whether the predicted bounding box was precise.

In this paper, the DA-Net only need to learn whether the predicted bounding box is precise due to the two location networks (LL-Net and GL-Net). The reason is that in [19], any other objects near the tracking target might be cropped as candidates. But in our tracker, the most of the candidates contain the target which is being tracked because of the location networks with similarity learning. Based on these, the reasons why we can train the DA-Net in very few iterations in our tracker online are summarized as follows:

- The DA-Net in our tracker learns a simpler function than [19] with the help of the two location networks.
- We train the DA-Net by single-domain learning offline before the tracking phase on large-scale training datasets.
- We only train the domain adaptation (DA) layer of the DA-Net online. The number of the parameters of the DA layer is fewer than the fully connected layers.

As shown in Figure 2, a set of target candidates $\mathcal{C} = \{X_n\}_{n=1}^{N}$ is generated based on the output of the LL-Net. For the n^{th} target candidate X_n, the DA-Net generates the confidence score s_n. The one among the candidates $\mathcal{C}$ with the maximum value s^* is considered as the final target, which is expressed as:

$$s^* = \max_{n \in \mathcal{C}} s_n \tag{3}$$

1) ARCHITECTURE

The size of the input of the DA-Net is $128 \times 128 \times 3$. The output is the confidence score which reflects how likely of the candidate is the target being tracked.

As shown in Figure 2(b), the DA-Net consists of three parts: ''Conv Blocks'', DA layer, and fully connected layers. The architecture of the ''Conv Blocks'' is the same as the LL-Net. ''DA'' is the ''domain adaptation'' layer. In order to make the features extracted by the ''Conv Blocks'' suitable for classification with a few parameters, we implement a 1×1 convolutional layer with stride 1 and a max pooling layer with stride 2 as the DA layer. The ''FC'' contains two fully connected layers with 512 nodes followed by ReLU and Dropout. The last layer is a binary classification layer, and we only consider the positive score.

FIGURE 6. A batch of training examples for the DA-Net. Each sample pair contains a positive patch and a negative patch.

2) DOMAIN ADAPTATION TRAINING

We call the process of training the domain adaptation classification network (DA-Net) is the domain adaptation training. The domain adaptation training can be divided into two phases: the offline training and the online training.

The offline training can be subdivided into three stages: first, training all of the parameters of the DA-Net; second, training the parameters of the DA layer and the FC layers with the ''Conv Blocks'' fixed; third, training the DA layer with other parameters fixed.

During the online training, only the parameters of the DA layer are updated. The online training is triggered in two situations in a video sequence: the DA layer is initialized for the current video sequence in the first frame with the ground truth, and the DA layer is updated every 30 frames in a periodic online updating.

The training datasets for the offline training are the same as the location networks (LL-Net and GL-Net). Each training batch contains 128 sample pair. Each sample pair contains a positive patch and a negative patch for the same object, and they are cropped in a frame selected randomly. Some examples are shown in Figure 6.

3) IMPLEMENTATION DETAILS

During the offline training phase, we train the DA-Net in three stages: in the first stage, we train the DA-Net for 100K iterations with learning rate $1e-4$ initially; in the second stage, we train the DA-Net for 50K iterations with learning rate $1e-5$ initially; in the third, we train the DA-Net for 10K iterations with learning rate $1e-5$ initially.

We collect the training data from the current video in the last 10 frames when the online training phase is triggered. Based on the IoU with the ground truth (in the first frame) or the estimated bounding box (in the subsequent frames), we create the positive samples more than 0.7 and the negatives less than 0.3. The experimental results shown in Section IV demonstrate that we can achieve a competitive performance by training the DA-Net only one iteration when the online training is triggered. We training the DA-Net with the softmax cross-entropy loss. Besides, in order to improve the tracking performance, we utilize the Hard Negative Mining (HNM) technique. We utilize Adam [33] optimizer to train the DA-Net with Tensorflow [34], and the batchsize is 256.

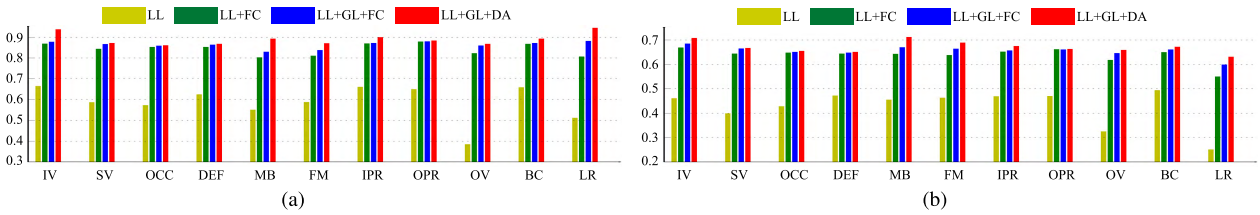

FIGURE 7. Precision and success score of four variants for 11 attributes on OTB-100. (a) Precision Score. (b) Success Score.

TABLE 1. Precision and success score of different variants on OTB.

Variants	Iterations Initial	Iterations Periodic	OTB-2013 Precision	OTB-2013 Success	OTB-50 Precision	OTB-50 Success	OTB-100 Precision	OTB-100 Success	FPS
LL+GL+DA	1	1	0.927	0.705	0.867	0.643	0.895	0.687	3
LL+GL+DA	1	0	0.929	0.709	0.857	0.636	0.884	0.681	3
LL+GL+DA	0	0	0.922	0.703	0.821	0.614	0.867	0.670	3
LL+GL+FC	30	15	0.885	0.686	0.849	0.635	0.885	0.682	2
LL+GL+FC	30	10	0.897	0.689	0.846	0.629	0.883	0.680	2
LL+GL+FC	30	5	0.918	0.708	0.838	0.629	0.878	0.678	2.3
LL+GL+FC	20	5	0.919	0.697	0.845	0.625	0.885	0.675	2.4
LL+GL+FC	10	5	0.877	0.676	0.790	0.590	0.846	0.655	2.5
LL+GL+FC	5	5	0.859	0.665	0.744	0.566	0.825	0.643	2.5
LL+GL+FC	5	0	0.857	0.646	0.715	0.544	0.803	0.621	3
MDNet	30	10	0.938	0.713	0.881	0.646	0.905	0.682	0.6

D. THE ALGORITHM AND IMPLEMENTATION DETAILS

The tracking algorithm proposed in this paper is shown in Algorithm 1. During the offline training phase, we first train the LL-Net from scratch. Then, we use the parameters of the LL-Net to initialize the convolutional blocks of the GL-Net and DA-Net. During the tracking phase, we update the DA layer only in one training iteration when the online training is triggered.

In the proposed tracker, we save 12 target templates when track in a new video sequence. The template set is updated in every 14 frames. When updating the target templates, we delete the earliest saved template and save the latest one. We don't update the template cropped in the first frame whenever. The target template is cropped based on the target location predicted in the current frame. The width and height of the target template are twice the size of the target.

We use the response map generated by the LL-Net or the GL-Net to locate the target. The center of the target is based on the point with the maximum response value within the response map. The width and height of the target are calculated by L_v^x and L_v^y respectively in Equation 2. We use the Hann window on the response maps.

We set the hyper-parameters based on the experimental results considering both the efficiency and performance. For example, if the threshold of the TRR is too high, some drifting situations will be missed. If the network is updated too frequently, the running speed will be too low, otherwise, the network cannot adapt to the appearance changes of the target. If the templates are updated frequently, the risk of decay is high; otherwise, it is hard to describe the appearance variations. Therefore, we set the hyper-parameters as in this section.

IV. EXPERIMENTS

A. BENCHMARK AND EVALUATION METRICS

The experiments in this section are conducted on the object tracking benchmarks (OTB) [36], [37], Temple Color 128 (TC-128) [38] and VOT2017 [39]. The OTB consists of three datasets: OTB-2013, OTB-50 and OTB-100.

We use two evaluation metrics on both OTB and TC-128, which are precision and success rate. For the two evaluation metrics, we show the precision plot and the success plot respectively with the one-pass evaluation (OPE) [36], [37]. We use three measures on VOT2017 [39]: accuracy, robustness and expected average overlap.

The proposed tracker runs at 3 fps using Tensorflow [34] in Python with Intel 2.4 GHz CPU and a single NVIDIA GTX TITAN X GPU.

B. ABLATION STUDY

In this section, all of the ablation studies are conducted on OTB [36], [37]. First, we show that the proposed tracker can achieve high tracking performance with very few training iterations in Table 1. Then, we evaluate the performance of four variants on different attributes in Figure 7, which presents the online training, GL-Net and DA-Net can improve the tracking performance.

Table 1 shows the precision and success score of different variants on OTB [36], [37] with different training iterations during the online training phase. The "LL" denotes the LL-Net, "GL" denotes the GL-Net, "DA" denotes

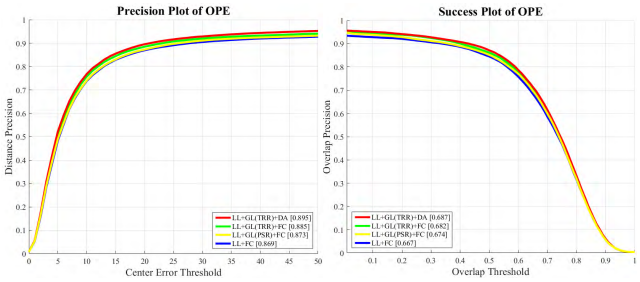

FIGURE 8. Precision and success score of four variants on OTB-100.

TABLE 2. Precision and success score of different variants on OTB-100 and TC-128.

Trackers	OTB-100		TC-128		FPS
	Precision	Success	Precision	Success	
ECO [17]	0.902	0.697	0.799	0.605	3
MDNet [19]	0.905	0.682	0.816	0.606	0.6
C-COT [18]	0.894	0.679	0.783	0.581	0.2
Ours	0.895	0.687	0.794	0.602	3

the DA-Net. "FC" denotes the classification network without the DA layer (the other architectures are the same as the "DA"), and all of the parameters of it has been trained offline. During the online training, we only update the fully connected layers of the "FC" and the DA layer of the "DA" respectively. "Initial" denotes how many iterations do we train the networks online on the first frame in one video sequence, and we call this training phase as initial training. "Periodic" denotes when the online training is triggered periodically, how many iterations do we train the networks, and we call this training phase as periodic training. The experimental results show that the proposed tracker which consisted of the LL-Net, GL-Net, and DA-Net outperforming other variants in all of the three datasets. Meanwhile, the experimental results also show that the proposed tracker can adapt to the current video sequence well enough with one training iteration during the initial and periodic training, which helps it run faster than other variants. The Table 1 also shows that the proposed tracker (LL+GL+DA) runs about 5 times faster than MDNet [19] with comparable performance.

Figure 7 shows the tracking performance of the four variants for 11 attributes on OTB-100 [37]. The "LL", "FC", "GL" and "DA" are the same as in Table 1. The "LL+FC" and "LL+GL+FC" both are trained 30 iterations in the initial training, and 15 iterations in the periodic training. The "LL+GL+DA" is trained 1 iteration in the initial training and 1 iteration in the periodic training. The experimental results in Figure 7 show that the online training and the contributions of ours ("GL" and "DA") are both improve the tracking performance on almost all of the attributes.

Figure 8 shows the tracking performance of the four variants on OTB-100 [37]. The "LL", "FC", "GL" and "DA" are the same as in Figure 7. "TRR" denotes the variant which detects the drifting by Target-to-Response Ratio, and

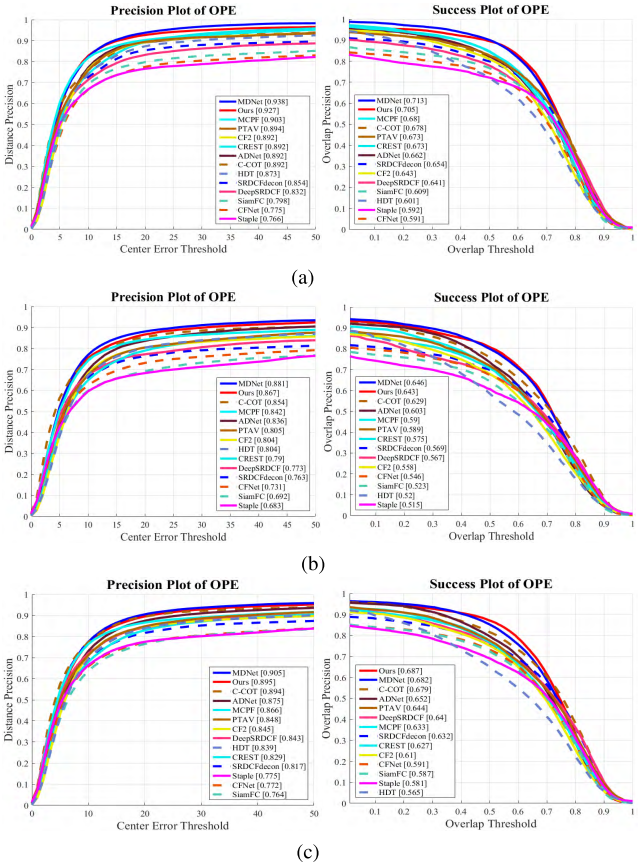

FIGURE 9. Overall Comparison on OTB datasets with other 13 state-of-the-art trackers. (a) OTB-2013. (b) OTB-50. (c) OTB-100.

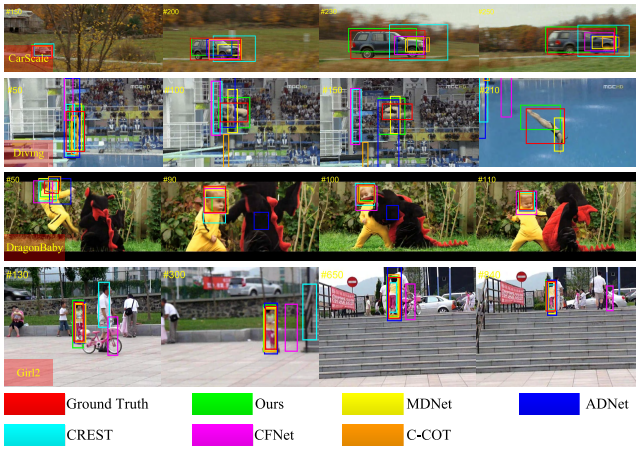

FIGURE 10. Qualitative comparisons with the state-of-the-art trackers on OTB: *CarScale, Diving, DragonBaby* and *Girl2*.

the "PSR" denotes the variant which detects the drifting by Peak to Sidelobe Ratio [35]. The experimental results show that the TRR is better than PSR for the proposed tracker.

In the following experiments, we use the "LL+GL+DA" which trained 1 iteration during the initial and periodic training as our tracker to compare with other state-of-the-art trackers.

TABLE 3. Average scores for different attributes on OTB-100 with the form of "precision score/success score". The best results are shown in red, the second in blue and the third in green.

	IV	SV	OCC	DEF	MB	FM	IPR	OPR	OV	BC	LR
HDT	79.9/52.8	80.7/48.8	75.7/52.3	80.3/53.8	76.0/56.5	79.5/56.2	82.9/55.1	79.2/53.1	66.1/47.7	84.4/58.5	88.1/39.9
Staple	76.0/58.9	72.7/52.7	71.0/54.0	73.1/54.5	66.8/53.0	68.7/53.4	75.1/54.5	72.4/53.1	66.6/48.2	75.0/57.0	69.0/40.3
SiamFC	71.9/56.7	73.5/55.8	71.0/54.3	67.6/50.3	68.3/54.6	72.7/56.6	73.0/55.7	74.7/55.9	67.0/51.4	69.0/53.0	90.0/63.0
CFNet	74.4/57.4	74.3/55.9	70.2/53.6	65.7/49.1	72.6/58.2	74.3/58.3	79.4/59.1	75.2/56.1	64.9/48.5	73.1/55.0	84.4/59.1
CF2	84.0/61.7	80.3/55.7	77.6/57.8	79.1/56.9	80.0/62.1	80.7/61.2	86.5/59.6	81.5/57.8	65.8/49.4	85.2/61.1	88.9/47.3
SRDCFdecon	81.4/64.5	80.5/61.6	75.1/58.8	73.5/55.0	78.4/63.4	75.3/60.3	76.1/57.1	78.3/59.2	64.0/51.9	85.0/65.2	74.2/52.7
DeepSRDCF	76.9/61.7	82.0/61.4	80.8/59.9	76.4/56.0	79.3/63.5	79.2/62.4	80.2/58.6	82.1/60.7	78.3/56.3	84.1/63.7	84.8/57.4
CREST	85.5/64.0	78.5/57.8	76.9/59.0	75.7/56.5	78.3/64.9	76.9/62.3	83.8/61.1	82.8/61.5	73.2/57.3	82.9/62.7	86.1/47.5
PTAV	86.0/65.3	79.3/59.2	83.2/63.2	81.4/60.4	79.6/62.6	77.7/61.5	83.0/61.4	82.5/61.6	73.5/57.8	87.6/65.8	80.6/50.1
MCPF	86.4/62.7	86.1/61.0	84.8/62.0	80.1/56.8	81.6/59.5	82.7/59.5	87.5/62.1	85.6/62.1	76.2/56.0	82.4/61.0	95.8/59.3
ADNet	89.1/66.1	86.6/63.8	82.4/61.6	84.2/62.2	82.8/66.3	82.4/64.1	85.9/62.5	86.8/63.5	79.9/60.7	91.3/66.2	92.0/56.7
C-COT	86.2/68.0	88.1/66.4	88.8/67.4	83.9/61.0	87.0/70.2	86.1/67.5	86.0/62.6	88.6/65.4	89.8/66.1	88.2/66.3	97.6/64.1
MDNet	91.8/69.5	89.2/66.3	85.1/65.0	89.4/65.6	86.8/68.2	87.9/67.5	90.4/66.2	89.9/66.6	79.9/61.7	91.7/67.6	93.7/64.3
Ours	93.9/70.9	87.3/66.8	86.2/65.6	86.9/65.2	89.4/71.3	87.2/69.0	90.1/67.6	88.5/66.4	86.9/66.0	89.4/67.3	94.6/63.2

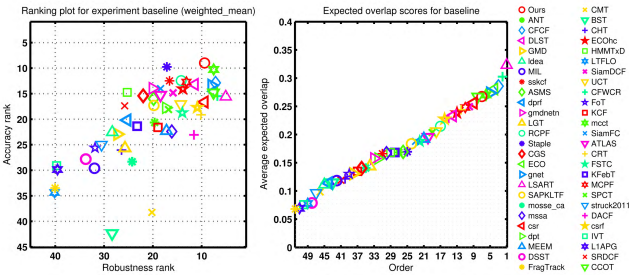

FIGURE 11. Overall Comparison on TC-128 dataset with other 13 state-of-the-art trackers.

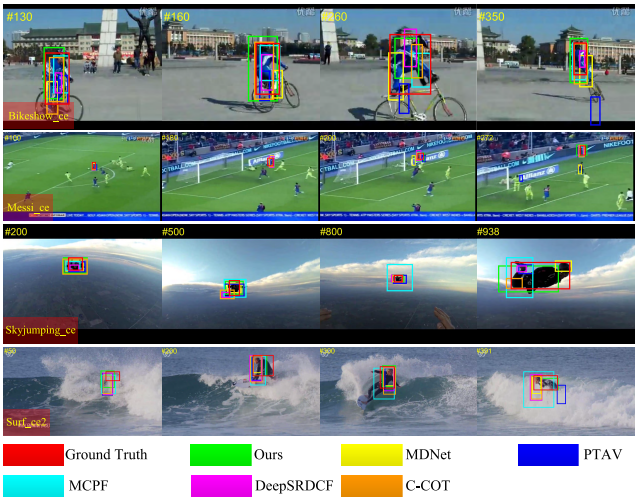

FIGURE 12. Qualitative comparisons with the state-of-the-art trackers on TC-128: *Bikeshow_ce*, *Messi_ce*, *Skyjumping_ce* and *Surf_ce2*.

C. SPEED COMPARISON

The speed comparison against the other top 3 trackers (with CNN features) with Intel 2.4 GHz CPU and a single NVIDIA GTX TITAN X GPU are shown in Table 2. The results in Table 2 show that the proposed tracker achieves the state-of-the-art tracking performance with the comparable tracking speed.

D. EVALUATION ON OTB

We compare the proposed tracker with 13 state-of-the-art trackers: MDNet [19], ADNet [22], MCPF [40], CREST [23], PTAV [41], CFNet [13],C-COT [18], HDT [42], SRDCFdecon [43], Staple [44], SiamFC [12], CF2 [45], and DeepSRDCF [46].

We show the precision plot and success plot on OTB [36], [37] by one-pass evaluation (OPE) in Figure 9. In Figure 9, we conduct the experiments on OTB-2013, OTB-50 and OTB-100. To show the tracking performance on different attributes, we show the precision and success score of these trackers in Table 3. The results show that the proposed tracker achieves the state-of-the-art performance. Besides, our tracker runs faster than the MDNet [19] and C-COT [18] which run slower than 1 fps with our hardware. We also show the qualitative tracking results in Figure 10.

E. EVALUATION ON TC-128

We compare the proposed tracker with 13 state-of-the-art trackers: MDNet [19], C-COT [18], PTAV [41], MCPF [40], DeepSRDCF [46],SRDCFdecon [43], SiamFC [12], CF2 [45], Staple [44], SRDCF [47], HDT [42], MUSTer [48] and DSST [49].

We show the precision plot and success plot on TC-128 [38] by one-pass evaluation (OPE) in Figure 11. To show the tracking performance on different attributes, we show the precision and success score of these trackers in Table 4. The results show that the proposed tracker achieves the state-of-the-art performance. We also show the qualitative tracking results in Figure 12.

F. EVALUATION ON VOT2017

We compare our tracker against all of the trackers which participated in the challenge. The results are shown in Figure 13. Figure 13 shows the accuracy-robustness ranking plot and the expected average overlap (EAO) ranking plot. In both of the ranking plots, the trackers at the top-right corner have better performance than others. The experimental results in Figure 13 show that the proposed tracker outperforms the most of the trackers in the VOT2017 challenge.

TABLE 4. Average scores for different attributes on TC-128 with the form of "precision score/success score". The best results are shown in red, the second in blue and the third in green.

	IV	SV	OCC	DEF	MB	FM	IPR	OPR	OV	BC	LR
DSST	58.8/42.4	53.5/34.5	49.1/34.8	50.8/36.4	45.4/33.5	43.1/35.4	50.5/36.9	51.5/37.8	36.2/29.3	55.2/38.8	40.0/25.5
HDT	68.1/48.7	66.9/45.0	63.2/47.5	74.8/53.2	57.7/43.3	61.3/48.0	63.7/47.5	65.4/48.3	47.6/39.8	77.7/51.9	58.5/32.5
Staple	64.6/51.1	64.2/48.5	58.1/45.9	73.6/54.0	56.6/40.8	59.5/48.7	59.0/46.5	63.4/49.2	52.4/38.8	68.7/49.7	56.0/36.3
SiamFC	62.3/47.4	68.1/49.5	62.9/46.7	67.6/48.6	55.7/40.7	62.5/48.7	64.9/48.7	66.9/50.0	51.7/39.4	67.5/46.5	73.7/46.6
MUSTer	67.6/53.6	58.8/43.8	59.4/45.9	69.5/51.4	47.9/35.7	48.3/41.4	56.1/43.3	57.3/44.6	43.6/34.2	76.1/53.5	57.4/35.0
SRDCF	66.9/51.8	65.3/48.7	63.6/48.2	75.6/53.5	58.0/42.9	56.2/45.3	60.2/45.7	61.0/46.1	50.8/40.9	71.2/50.5	54.7/36.0
CF2	72.2/53.3	68.7/49.0	62.3/47.8	80.5/56.3	63.8/45.2	63.1/50.5	63.5/48.6	67.4/50.8	47.4/38.9	74.5/50.6	58.0/32.6
SRDCFdecon	71.5/56.7	72.8/55.1	69.0/53.1	79.4/55.9	64.0/48.2	62.0/50.9	64.6/49.4	68.1/51.5	56.9/48.5	77.0/54.7	70.5/43.0
DeepSRDCF	67.9/53.6	69.8/52.0	68.8/52.0	77.3/54.2	69.4/50.7	62.7/50.5	64.9/48.4	68.4/51.4	63.4/52.2	77.9/54.7	78.5/50.2
PTAV	75.7/59.4	65.9/51.2	71.8/55.0	78.4/57.0	63.1/45.7	59.9/51.5	64.2/51.0	68.8/53.9	58.7/48.8	83.9/58.2	66.5/37.7
MCPF	78.6/56.8	80.2/54.6	76.6/56.4	84.6/58.4	71.1/49.1	70.6/53.3	75.7/53.6	75.7/55.3	60.7/47.4	83.2/57.1	69.5/41.3
C-COT	77.2/58.7	78.2/58.7	75.4/57.7	86.0/60.7	73.6/53.9	68.5/55.7	72.4/54.9	75.7/56.9	69.6/54.5	80.5/56.7	82.0/52.3
MDNet	82.2/63.3	81.4/60.6	79.7/60.4	85.2/62.3	72.0/51.8	71.8/56.2	75.4/57.7	79.3/60.8	60.5/48.6	86.5/60.7	75.0/47.2
Ours	83.3/62.0	82.0/61.4	78.9/59.8	88.2/63.1	70.3/51.4	74.4/58.5	78.7/60.3	80.6/61.6	61.7/50.1	82.9/59.8	58.9/41.9

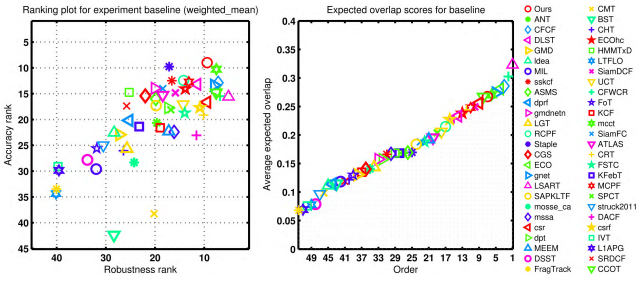

FIGURE 13. Accuracy-robustness ranking plot and expected average overlap on VOT2017.

V. CONCLUSION

In this paper, we propose a domain adaptation tracker containing three neural networks: an LL-Net, a GL-Net and a DA-Net. With the help of the LL-Net and GL-Net, we not only can locate the target templates within the search area precisely but also recapture the target when the drifting is detected. Meanwhile, we proposed a domain adaptation classification network and the corresponding training method. By updating the DA layer of the DA-Net during the online training phase, the tracker can adapt to the current domain (video sequence) quickly. In practice, the proposed tracker achieves the state-of-the-art performance with only one training iteration both in the initial training and the periodic training. All of these make the proposed tracker achieve the ability to adapt to the appearance variations of the target, and to distinguish the target from the background. Our tracker can achieve the comparable performance comparing against the MDNet [19] with five times faster.

REFERENCES

[1] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. CVPR*, Jun. 2012, pp. 1822–1829.

[2] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.

[3] S. Hare *et al.*, "Struck: Structured output tracking with kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2096–2109, Oct. 2016.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[5] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 379–387.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.

[7] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. (2017). "Mask R-CNN." [Online]. Available: https://arxiv.org/abs/1703.06870

[9] A. Dosovitskiy *et al.*, "Flownet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2758–2766.

[10] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1420–1429.

[11] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 749–765.

[12] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 850–865.

[13] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-end representation learning for correlation filter based tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5000–5008.

[14] K. Chen and W. Tao. (2016). "Once for all: A two-flow convolutional neural network for visual tracking." [Online]. Available: https://arxiv.org/abs/1604.07507

[15] C. Huang, S. Lucey, and D. Ramanan, "Learning policies for adaptive tracking with deep feature cascades," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 105–114.

[16] A. He, C. Luo, X. Tian, and W. Zeng. (2018). "A twofold siamese network for real-time object tracking." [Online]. Available: https://arxiv.org/abs/1802.08817

[17] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. (2016). "ECO: Efficient convolution operators for tracking." [Online]. Available: https://arxiv.org/abs/1611.09224

[18] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 472–488.

[19] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4293–4302.

[20] H. Nam, M. Baek, and B. Han. (2016). "Modeling and propagating CNNs in a tree structure for visual tracking." [Online]. Available: https://arxiv.org/abs/1608.07242

[21] H. Fan and H. Ling. (2016). "SANeT: Structure-aware network for visual tracking." [Online]. Available: https://arxiv.org/abs/1611.06878
[22] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1349–1358.
[23] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. H. Lau, and M.-H. Yang, "CREST: Convolutional residual learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2574–2583.
[24] B. Han, J. Sim, and H. Adam, "BranchOut: Regularization for online ensemble tracking with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 521–530.
[25] G. Zhu, F. Porikli, and H. Li, "Beyond local search: Tracking objects everywhere with instance-specific proposals," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 943–951.
[26] H. Fan and H. Ling, "SANet: Structure-aware network for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jul. 2017, pp. 2217–2224.
[27] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.
[28] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[29] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014.
[30] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for UAV tracking," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 445–461.
[31] A. Li, M. Lin, Y. Wu, M.-H. Yang, and S. Yan, "NUS-PRO: A new visual tracking challenge," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 335–349, Feb. 2015.
[32] F. Zhao, M. Tang, Y. Wu, and J. Wang, "DenseTracker: A multi-task dense network for visual tracking," in *Proc. IEEE Conf. Multimedia Expo*, Jul. 2017, pp. 607–612.
[33] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980
[34] M. Abadi *et al.* (2016). "TensorFlow: Large-scale machine learning on heterogeneous distributed systems." [Online]. Available: https://arxiv.org/abs/1603.04467
[35] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2544–2550.
[36] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 2411–2418.
[37] Y. Wu, J. Lim, and M. H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.
[38] P. Liang, E. Blasch, and H. Ling, "Encoding color information for visual tracking: Algorithms and benchmark," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5630–5644, Dec. 2015.
[39] M. Kristan *et al.*, "The visual object tracking VOT2017 challenge results," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, Oct. 2017, pp. 1949–1972.
[40] T. Zhang, C. Xu, and M.-H. Yang, "Multi-task correlation particle filter for robust object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4819–4827.
[41] H. Fan and H. Ling, "Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5487–5495.
[42] Y. Qi *et al.*, "Hedged deep tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4303–4311.
[43] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1430–1438.
[44] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr, "Staple: Complementary learners for real-time tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1401–1409.
[45] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 3074–3082.
[46] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, Dec. 2015, pp. 621–629.
[47] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4310–4318.
[48] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao, "Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 749–758.
[49] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proc. Brit. Mach. Vis. Conf.*, Nottingham, U.K., Sep. 2014, pp. 1–11.

FEI ZHAO received the B.E. degree from the Hebei University of Technology, China, in 2012, and the M.S. degree from the Beijing Institute of Technology, China, in 2015. He is currently pursuing the Ph.D. degree with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, China. His research interests include deep learning and visual tracking.

TING ZHANG received the B.Sc. degree in communication engineering from Beijing Jiaotong University, China, in 2013, and the Ph.D. degree in pattern recognition and intelligent system from the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, in 2018. She is currently an Engineer with the Research and Development Center, China National Electronics Import and Export Corporation. Her research interests include deep learning and face recognition.

YI WU received the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2009. From 2009 to 2016, he was an Assistant Professor with the Nanjing University of Information Science and Technology, Nanjing. From 2010 to 2012, he was a Post-Doctoral Fellow with Temple University, Philadelphia, PA, USA. From 2012 to 2014, he was a Post-Doctoral Fellow with the University of California, Merced, CA, USA. He is currently a Research Assistant Professor with the Indiana University School of Medicine. Before joining Indiana University, he was a Runze Professor with Nanjing Audit University, Nanjing, China. His research interests include computer vision, medical image analysis, and deep learning. His tracking benchmark is one of the most popular tracking evaluation platforms.

JINQIAO WANG received the B.E. degree from the Hebei University of Technology, China, in 2001, the M.S. degree from Tianjin University, China, in 2004, and the Ph.D. degree in pattern recognition and intelligence systems from the National Laboratory of Pattern Recognition, Chinese Academy of Sciences, in 2008. He is currently a Professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His research interests include pattern recognition and machine learning, object detection and tracking, image and video processing, mobile multimedia, and intelligent video surveillance.

MING TANG (M'06) received the B.S. degree in computer science and engineering and the M.S. degree in artificial intelligence from Zhejiang University, Hangzhou, China, in 1984 and 1987, respectively, and the Ph.D. degree in pattern recognition and intelligent system from the Chinese Academy of Sciences, Beijing, China, in 2002. He is currently a Professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His current research interests include computer vision and machine learning.