Triplet Loss in Siamese Network for Object Tracking

Xingping Dong[1] and Jianbing Shen[1,2](✉)

[1] Beijing Lab of Intelligent Information Technology, School of Computer Science,
Beijing Institute of Technology, Beijing, China
{dongxingping,shenjianbing}@bit.edu.cn
[2] Inception Institute of Artificial Intelligence, Abu Dhabi, UAE
http://github.com/shenjianbing/TripletTracking

Abstract. Object tracking is still a critical and challenging problem with many applications in computer vision. For this challenge, more and more researchers pay attention to applying deep learning to get powerful feature for better tracking accuracy. In this paper, a novel triplet loss is proposed to extract expressive deep feature for object tracking by adding it into Siamese network framework instead of pairwise loss for training. Without adding any inputs, our approach is able to utilize more elements for training to achieve more powerful feature via the combination of original samples. Furthermore, we propose a theoretical analysis by combining comparison of gradients and back-propagation, to prove the effectiveness of our method. In experiments, we apply the proposed triplet loss for three real-time trackers based on Siamese network. And the results on several popular tracking benchmarks show our variants operate at almost the same frame-rate with baseline trackers and achieve superior tracking performance than them, as well as the comparable accuracy with recent state-of-the-art real-time trackers.

Keywords: Siamese network · Triplet loss · Object tracking
Real-time

1 Introduction

Object tracking containing single object tracking [8,9] and multi-object tracking [24,25] remains an important problem with many applications, such as automated surveillance, and vehicle navigation [34]. In single object tracking, powerful feature selecting is one of the key step to improve tracking accuracy. In the recent years, this strategy has been widely used for many correlation filter (CF) based trackers. For example, Henriques *et al.* [12] applied the Histogram of Oriented Gradients (HOG) feature instead of gray feature in [11] to achieve more robust tracking performance. Danelljan *et al.* [5] tried to use the color name to process color sequence. More recently, the pre-trained deep networks are applied to extract feature from raw image for improving accuracy, such as

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11217, pp. 472–488, 2018.
https://doi.org/10.1007/978-3-030-01261-8_28

DeepSRDCF [6], CCOT [7], MCPF [36], and ECO [4]. Besides CF trackers, some deep learning based trackers focus on designing an end-to-end network to achieve more powerful and suitable feature for its tracking system. MDNet [20] used a multi-domain convolutional neural network to extract common feature inside various samples during the off-line training phase. Then, the trained network is refined frame by frame in different sequences through online training. This tracker achieved excellent performance on OTB-2013 [32] and won the main challenge in VOT-2015 [18]. However, its running speed is less than 1 frame-per-second (fps), which is far below the real-time requirement for processing videos (30 fps). The slow speed is caused by the online training. Thus, in order to satisfy the real-time requirement in practical application, recent work like SiamFC [2] still uses the deep network for off-line training to achieve powerful feature while try to avoid online training for acceleration.

Although SiamFC utilizes deep network to extract powerful feature, it does not take full advantage of the relationship among the input samples. SiamFC addresses the tracking task as similarity learning in an embedding space. The similarity function is constructed with a Siamese network trained in off-line phase. The inputs include an exemplar image enclosing the object and a larger search image where the sliding-windows with the same size of exemplar can be viewed as instances, i.e. candidate object bounding boxes. According to the distance between the location of object and an instance, it is labeled as positive when its distance is less than a threshold, otherwise, it is labeled as negative. The logistic loss is applied to maximize the similarity scores on exemplar-positive pairs and minimize them on exemplar-negative pairs. This training method only utilizes the pairwise relationship on samples and ignores the underlying connections inside the triplet: exemplar, positive instance and negative instance.

In this paper, we try to make the best of the triplet inputs to achieve more powerful features by adding a novel triplet loss into the Siamese framework. For each triplet, we define a matching probability to measure the possibility assigning positive instance to exemplar compared with the negative instance. Then, our goal is to maximize the joint probability among all triplets during training. The proposed triplet loss not only can further mine the potential relationship among exemplar, positive instance and negative instance, but also contains more elements for training at most situation. Here, we give an intuitive example. In object tracking, the number of exemplar is 1 since only one object bounding box is given in the first frame. While the numbers of positive and negative instance usually are more than 1. We can set them in a batch as M and N, respectively. In SiamFC, at most $M+N$ pairwise elements (M exemplar-positive pairs + N exemplar-negative pairs) can be applied for training. However, our method can produce MN triplet-wise elements (the combination of M exemplar-positive pairs and N exemplar-negative pairs). If $M > 2$ and $N > 2$, then $MN > M + N$. It indicates our method will get more elements for training to enhance the performance. In the other situation, we can also get approximate number of elements. This example illustrates our loss is able to make better use of the combination of samples to achieve more powerful features. For clearer explanation, the training framework of triplet loss is shown in Fig. 1.

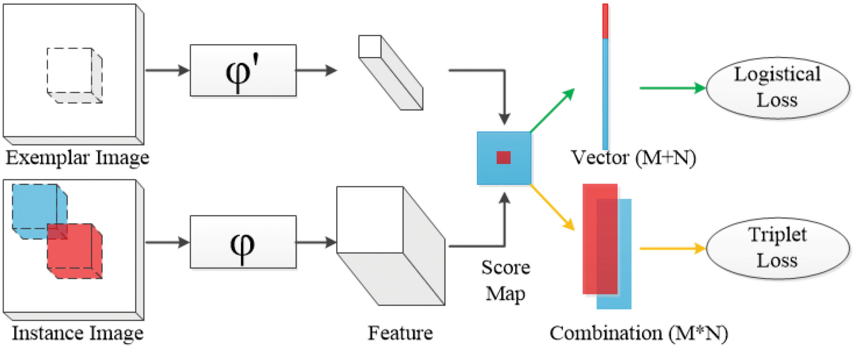


Fig. 1. Training framework of the triplet loss in Siamese network. We also give the original logistic loss for comparison. Given the same feature extraction in baselines [2], [28], we can apply the triplet loss to the score map. In contrast to use the vectorization of score map in logistic loss, we utilize the combination between positive scores (red) and negative scores (blue). The red rectangle means a positive score matrix produced by repeating M positive scores N times and the blue rectangle is a negative score matrix by repeating N negative scores M times. In fact, our loss is suitable for the network with same branches ($\phi' = \phi$ in SiamFC [2]) or different branches ($\phi' \neq \phi$ in CFnet [28]). (Color figure online)

Furthermore, we give the theoretical analysis between the original logistic loss and the proposed triplet loss to prove the effectiveness of our method. Firstly, the logistic loss is reformulated to be comparable with our triplet loss. Then we analyze their difference by comparing their gradients on various inputs. We find the triplet loss can offer larger absolute gradient when the similarity score of exemplar-positive pair vp is not more than the one of exemplar-negative pair vn. It means that the triplet loss will give stronger feedback for back-propagation when the network gives wrong similarities ($vp \leq vn$). This advantage will improve the tracking accuracy since if $vp \leq vn$, the tracking system will produce tracking error by labeling the negative instance as the object.

In fact, our triplet loss is suitable for the Siamese network with different structures. In our experiments, we applied the triplet loss to three existing trackers based on Siamese networks: SiamFC [2], CFnet2 [28], and SiamImp [28]. The experimental results on the famous tracking benchmark OTB-2013 have shown that all variants with our loss outperform original trackers and achieve similar high speed (55 - 86 fps) beyond real-time requirement. In three tracking benchmarks: OTB-2013 [32], OTB-100 [33] and VOT-2017 [15], our trackers achieve comparable results compared with recent state-of-the-art real-time trackers.

2 Related Works

Trackers with Siamese network: With the development of deep learning in recent years, many classical networks are introduced into object tracking, such as Siamese network [27], [2], [28]. Tao *et al.* [27] trained a Siamese network to learn a matching function in the off-line phase. In the online tracking phase, the learned matching function is applied to find the most similar patch in new frame compared with the initial patch of object in the first frame. This Siamese Instance search Tracker (SINT) performs well in OTB-2013 [32] while its speed is only 2 fps. In order to improve running speed, Bertinetto *et al.* [2] omitted the fully connected layers to reduce computation and only apply 5 fully convolutional layers to train an end-to-end Siamese network (SiamFC) for similarity function. Then, the similarity function is directly applied to online track without complex fine-tuning strategies. Therefore, SiamFC achieves high frame-rates beyond real-time, nearly at 86 fps with GPU. Another related tracker CFnet [28] regards the correlation filter as a network layer to compute the similarity between the generating convolutional features of Siamese network. It enables the learning deep features to be tightly coupled to the correlation filter. The experimental results show that 2 convolutional layers with CF layer in Siamese network (CFnet2) will achieve comparable performance and speed (75 fps) compared with SiamFC containing 5 convolutional layers. Otherwise, CFnet proposes an improved Siamese network (SiamImp) by modifying the structure in some convolutional layers of SiamFC [2]. SiamImp outperforms SiamFC in tracking accuracy on OTB-2013 and OTB-100 while it operates at lower speed, nearly 52 fps.

To prove the generality of the proposed triplet loss for network structure, we apply it to three real-time trackers SiamFC, CFnet2 and SiamImp, which own similar Siamese frameworks but different network structures.

Triplet loss in computer vision: Triplet loss has been widely applied for numerous applications in computer vision, such as face recognition [23], image retrieval [14], [37], [26], and person re-identification [3], [30], [13]. Here we illustrate some works for reference. Schroff *et al.* [23] proposed a FaceNet for face recognition and clustering by combining the triplet loss [31] and the deep convolutional network. To ensure fast convergence, an online triplet mining method is proposed by selecting hardest sample pairs (face patch pairs) on each batch. In order to further mine the underlying connection among triplets, Song *et al.* [26] applied a structured loss for training by lifting the vector of pairwise distances within the batch to the matrix of pairwise distances. Hermans *et al.* [13] systematically evaluated several variants of classic triplet loss and proposed a novel batch hard loss with the soft margin for person re-identification. Their method randomly sampled some instances to construct small set as a batch and selected some hardest instances to compute the loss. In contrast to most existing approaches with margin-based triplet loss above, our method uses a probability-based triplet loss to avoid manually selecting the suitable margin.

3 Revisiting the Siamese Network for Tracking

Bertinetto *et al.* [2] proposed a Siamese network with fully convolutional layers for object tracking (SiamFC) by transferring tracking task to exemplar matching in an embedding space. The tacking object patch is usually given in the first frame of a sequence and it can be viewed as an exemplar. The goal is to find a most similar patch (instance) from each frame in the semantic embedding space. How to learn a powerful embedding function is the key step for this matching problem. The authors of SiamFC apply a fully convolution Siamese deep network to represent this embedding function. Two network branches are designed to process the special inputs in the tracking task. One input is the object bounding box in the first frame, which is called as exemplar input. The other instance input is the searching region in each subsequent frame including the candidate patches to be matched. These two network branches can be seen as an identical transformation ϕ for different inputs, since they share the same parameters. Denote the exemplar as z and the instance as x, then the similar function is defined as

$$f(z, x) = g(\phi(z), \phi(x)), \tag{1}$$

where g is a simple similarity metric such as vectorial angle and cross correlation.

In SiamFC, the cross correlation function is applied for g, and the formulation of function f is transferred as follows:

$$f(z, x) = \phi(z) * \phi(x) + b. \tag{2}$$

Then, a logistic loss is applied to define the pairwise loss function for training, which is formulated as follows:

$$L_l(\mathcal{Y}, \mathcal{V}) = \sum_{x_i \in \mathcal{X}} w_i \log(1 + e^{-y_i \cdot v_i}). \tag{3}$$

where $\mathcal{Y}$, $\mathcal{V}$, $\mathcal{X}$ are respectively the sets of ground-truth label, similarity score, instance input. $y_i \in \{+1, -1\}$ is the ground-truth label of a single exemplar-instance pair (z, x_i). v_i is the similarity score of (z, x_i) i.e. $v_i = f(z, x_i)$. w_i is the weight for an instance x_i, and $\sum_{x_i \in \mathcal{X}} w_i = 1, w_i > 0, x_i \in \mathcal{X}$. In SiamFC, the balance weights are used for loss according to the number of positive and negative instances. The formulation of balance weights is defined as follows:

$$w_i = \begin{cases} \frac{1}{2M}, & y_i = 1 \\ \frac{1}{2N}, & y_i = -1 \end{cases} \tag{4}$$

where M, N are the number of positive instance set $\mathcal{X}_p$ and negative instance set $\mathcal{X}_n$ i.e. $M = |\mathcal{X}_p|$, $N = |\mathcal{X}_n|$. (In SiamFC, $M = 13$, $N = 212$.)

4 Siamese Network with Triplet Loss

As mentioned before, we can split the instances set $\mathcal{X}$ in SiamFC [2] to positive instances set $\mathcal{X}_p$ and negative instances set $\mathcal{X}_n$. Considering the other exemplar input, we can construct triplet tuples using the inputs of SiamFC i.e. a tuple contains exemplar, positive instance and negative instance. However, SiamFC only utilizes the pairwise loss and ignores the underlying relation between the positive instance and the negative instance. Based on this consideration, we design a new triplet loss to mine the potential relation among the inputs as much as possible. As splitting the instances set $\mathcal{X}$, the similarity score set $\mathcal{V}$ of exemplar-instance pairs can also be split as a positive score set $\mathcal{V}_p$ and a negative score set $\mathcal{V}_n$. Then, we can directly define the triplet loss on these score-pairs. To measure each score-pair, we apply a matching probability i.e. the probability assigning positive instance to exemplar by using a soft-max function. The formulation of this matching probability is defined as follows.

$$prob(vp_i, vn_j) = \frac{e^{vp_i}}{e^{vp_i} + e^{vn_j}}. \tag{5}$$

In the explanation of probability theory, our goal is to maximize the joint probability among all score-pairs i.e. the product of all probabilities. By using its negative logarithm, we can get the loss formulation as follows.

$$L_t(\mathcal{V}_p, \mathcal{V}_n) = -\frac{1}{MN} \sum_{i}^{M} \sum_{j}^{N} \log prob(vp_i, vn_j), \tag{6}$$

where the balance weight $\frac{1}{MN}$ is used to keep the loss with the same scale for different number of instance sets.

Compared with the original pairwise logistic loss L_l in Eq. 3, our triplet loss L_t will capture more underlying information to achieve more powerful representation with little extra computation during training. Firstly, our triplet loss contains more elements (i.e. single losses), which can mine more underlying relationship among exemplar, positive instance, and negative instance. In more detail, L_l only includes $M + N$ varied losses while our L_t is the weighted average of MN variates. The more variates in the loss function means the more powerful representation, since it can capture more information by these variates. More detailed analysis is shown in next section. Secondly, our loss is defined on the original scores by using their combination between positive scores and negative scores. Thus, we use the same inputs to feed the network. It means we do not need extra computation for feature extraction with deep network during training. The only adding time cost is taken for computing the new loss, which is occupied small part of time cost during training.

5 Relationship Between Logistic Loss and Triplet Loss

As mentioned before, our triplet loss in Eq. 6 contains MN elements while the number in the logistic loss in Eq. 3 is $M + N$. If we want to compare these two losses, we have to keep the number consistent. Therefore, we manage to transform Eq. 3 for comparison. To keep the same input of instances, no additional instance is imported for increasing element number during the transformation. The only change is the increased frequency of usage of exemplar-instance pairs. We also add constant weight to make it become equivalent transformation. For a set of instances $\mathcal{X}$, the logistic loss can be reformulated as follows.

$$\begin{aligned} L_l &= \sum_i^M \frac{1}{2M} \log(1+e^{-vp_i}) + \sum_j^N \frac{1}{2N} \log(1+e^{vn_j}) \\ &= \frac{1}{N} \sum_j^N \sum_i^M \frac{1}{2M} \log(1+e^{-vp_i}) + \frac{1}{M} \sum_i^M \sum_j^N \frac{1}{2N} \log(1+e^{vn_j}) \\ &= \frac{1}{MN} \sum_i^M \sum_j^N \frac{1}{2} \left(\log(1+e^{-vp_i}) + \log(1+e^{vn_j}) \right). \end{aligned} \tag{7}$$

This equation is similar with Eq. 6. We need to simplify Eq. 6 for further analysis. By submitting Eq. 5 to Eq. 6, we can get the following formulation.

$$L_t = -\frac{1}{MN} \sum_i^M \sum_j^N \log \frac{e^{vp_i}}{e^{vp_i} + e^{vn_j}} = \frac{1}{MN} \sum_i^M \sum_j^N \log(1+e^{vn_j - vp_i}). \tag{8}$$

From Eqs. 7 and 8, we can find the main difference is their terms inside summation. Thus, we only need to further analyze these two terms to achieve the difference between two losses. Their formulation can be denoted as follows.

$$T_l = \frac{1}{2} \left(\log(1+e^{-vp}) + \log(1+e^{vn}) \right), \quad T_t = \log(1+e^{vn-vp}). \tag{9}$$

For simplification, we omit the subscripts i and j to focus on the difference on these terms.

5.1 Comparison on the Gradients

The gradients play important role during deep learning training since they are directly involved in the back-propagation stage. Thus, they are used to point out the characteristics of different terms. Firstly, we give their gradients. For the logistic term, the gradients are derived as:

$$\frac{\partial T_l}{\partial vp} = -\frac{1}{2(1+e^{vp})}, \qquad \frac{\partial T_l}{\partial vn} = \frac{1}{2(1+e^{-vn})}. \tag{10}$$

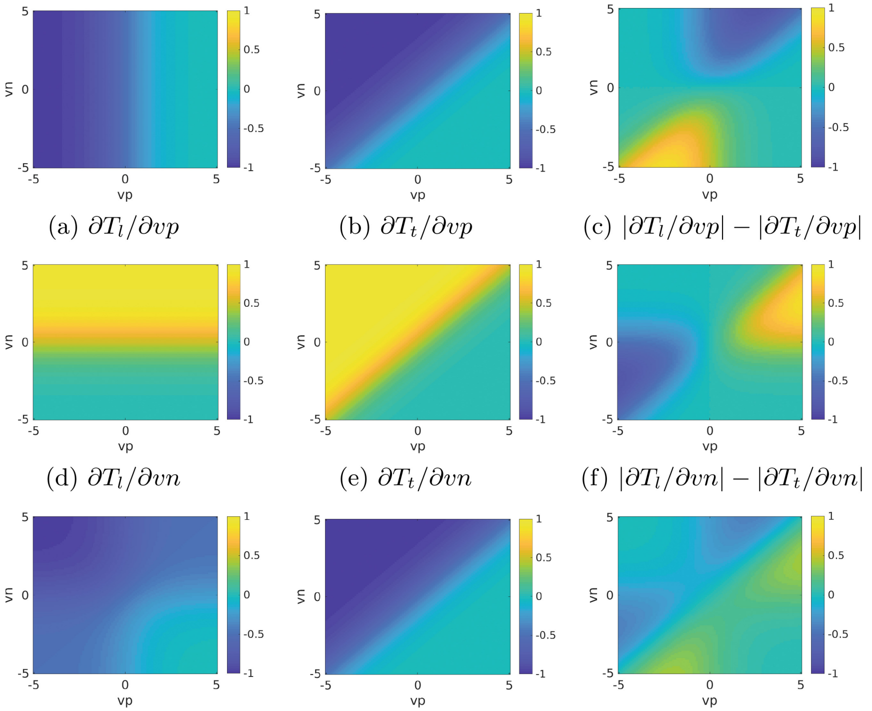
For our triplet loss, the gradients of its term are given as:

$$\frac{\partial T_t}{\partial vp} = -\frac{1}{1+e^{vp-vn}}, \qquad \frac{\partial T_t}{\partial vn} = \frac{1}{1+e^{vp-vn}}. \tag{11}$$

From Eq. 10, we can find $\partial T_l / \partial vp$ and $\partial T_l / \partial vn$ in logistic term only depend on vp and vn respectively, while our $\partial T_t / \partial vp$ considers both vp and vn. It means the logistic term can not take full advantage of information offered by vp and vn. In the other words, $\partial T_l / \partial vp$ can not utilize the information from vn and

$\partial T_l/\partial vn$ fails to make use of the information of vp. For further analysis, visual comparison is shown in Fig. 2 by using the color maps of different gradients. Figure 2(a) and (d) also show that $\partial T_l/\partial vp$ and $\partial T_l/\partial vn$ are independent for vn and vp, respectively.

In the tracking task, we should keep the important constraint condition $vp > vn$ to reduce the tracking error. $vp \leq vn$ means the similarity score of positive instance is less than or equal to the negative instance, and the negative instance will be regarded as the object leading to tracking failure. Thus, we should pay more attention for $vp \leq vn$ during training. Now we will analyze the gradients of positive instance of two losses.



(a) $\partial T_l/\partial vp$ (b) $\partial T_t/\partial vp$ (c) $|\partial T_l/\partial vp| - |\partial T_t/\partial vp|$

(d) $\partial T_l/\partial vn$ (e) $\partial T_t/\partial vn$ (f) $|\partial T_l/\partial vn| - |\partial T_t/\partial vn|$

(g) $\partial T_l/\partial y$ (h) $\partial T_t/\partial y$ (i) $|\partial T_l/\partial y| - |\partial T_t/\partial y|$

Fig. 2. Comparison of gradients for logistic loss and triplet loss. (a) and (b) are the gradients on positive instance of logistic loss and triplet loss, respectively. (c) is the differences between their absolute values. Similarly, (d), (e), and (f) are corresponding to the negative instance. (g), (h), and (i) are corresponding to their difference $y = vp - vn$.

As shown in Fig. 2(a) and (b), when $vp \leq vn$ (i.e. up-left-triangle region), our gradient $\partial T_t/\partial vp$ has relatively large absolute value ($|\partial T_t/\partial vp| \geq 0.5$) in this situation. While the absolute gradient $|\partial T_l/\partial vp|$ will be close to 0 even on

$vp < vn$ when vp is approaching a big value like 5. It means the gradient $\partial T_l/\partial vp$ only offers little feedback for back-propagation even vn violates the constraint $vp > vn$, when vp is large. However, our gradient $\partial T_t/\partial vp$ can give more feedback for this situation by offering larger absolute gradient. For further comparison, the color map of difference between absolute gradients $|\partial T_l/\partial vp| - |\partial T_t/\partial vp|$ is shown in Fig. 2(c). It indicates that inside the region $vp \leq vn$ and $vp > 0$, our absolute gradient is larger than logistic absolute gradient, which means our loss can offer better feedback for back-propagation in this region. In most of the resident region of $vp \leq vn$, our gradient is approximately equal to the logistic gradient. Secondly, the comparison of negative instance gradients is shown in Fig. 2(d), (e), and (f). Similarly, our gradient also gives sufficient feedback on the region $vp \leq vn$, while the logistic gradient offers fewer feedback on the region $vp \leq vn$ and $vn < 0$. For more direct comparison on two variables vp and vn, we observe the derivatives on $y = vp - vn$. It is easy to get $\partial T_l/\partial y = \partial T_l/\partial vp - \partial T_l/\partial vn$ and $\partial T_t/\partial y = \partial T_t/\partial vp$. As shown in Fig. 2(g), the gradients of logistic loss are depended on both vp and vn, which is similar with triplet loss. This comparison is more intuitive. Figure 2(i) shows that on the region $vp \leq vn$, our absolute gradients are larger than logistic loss. It means on this region we can offer better feedback. In summary, our loss will give suitable gradient feedback for back-propagation when the similarity scores violate the constraint $vp > vn$, however the gradient of logistic loss will vanish on extreme condition, such as $\partial T_l/\partial vp \rightarrow 0$ at $vp \rightarrow 5$.

6 Experimental Results

In this section, we show the experimental results on several popular tracking benchmarks including OTB-2013 [32], OTB-100 [33], and VOT-2017 [15]. Firstly, we give the details of implementation and the introduction of benchmarks and evaluation metrics. Then, various comparisons on these benchmarks are shown to evaluate the proposed triplet loss, including experiments on baselines and comparisons between our trackers and other state-of-the-art trackers.

6.1 Implementation Details

Baseline trackers. Firstly, we introduce three aforementioned baseline trackers: SiamFC, CFnet2, and SiamImp. We selected the version with 3 scales in [2] as baseline tracker denoted as SiamFC, since this version runs faster than the one with 5 scales and only performs slightly lower. In [28], a lot of variants of CFnet are proposed for experimental comparison. The one with 2 convolutional layers (CFnet2) obtains high speed and slightly lower performance than the best. Thus, it is selected as the representative of CFnet structure. This work also proposes an improved Siamese network (SiamImp) as baseline, by reducing the total stride and the number of final CNN output channels in SiamFC. The training method and training dataset are similar with the ones in [2,28] except the training loss.

Training. The deep learning toolbox MatConvNet [29] is applied to train the parameters of the shared network by minimizing the loss with SGD. The initial weights of the shared networks are set with the pre-trained models in SiamFC [2] and CFnet [28]. We randomly sample 53,200 pairs from the dataset ILSVRC15 [22] as a training epoch and perform training over 10 epochs. 10% pairs are chosen as the validation set at each epoch. And we decide the final network used for testing from the trained models at the end of each epoch, by the minimal mean error of distance (presented in [2]) on the validation set. The gradients for each iteration are estimated using mini-batches of size 8, and the learning rate is decayed geometrically after epoch from 10^{-4} to 10^{-5}. To handle the gray videos in benchmarks like [2,28], 25% of the pairs are converted to grayscale during training for SiamFC. For CFnet2 and SiamImp, a gray network is trained with all grayscale pairs to process gray videos. Similarly, all color pairs are applied to train a color network.

Tracking. In the tracking phase, we only replace the pre-trained networks with the models trained by triplet loss. The others inside online tracking, such as tracking approaches, and hyper-parameters setting, are the same with the original papers. Thus, the improved trackers can run at very similar high speed with baseline trackers. In more details, our variants: SiamFC-tri, CFnet2-tri, and SiamImp-tri achieve speeds at 86.3 fps, 55.3 fps and 55.8 fps on OTB-2013, respectively. The corresponding baseline trackers run respectively at 86.5 fps, 55.1 fps, and 55.4 fps. Our machine is equipped with a single NVIDIA GeForce 1080 and an Intel Core i7-6700 at 3.4 GHz, and our software platform is Matlab 2017a + CUDA 8.0 + cudnn v7.0.5.

6.2 Tracking Benchmarks

Our improved trackers are evaluated with recent state-of-the-art trackers in popular benchmarks: OTB-2013 [32], OTB-50, OTB-100 [33], and VOT-2017 [15].

The OTB-2013 benchmark proposes several metrics to evaluate trackers on 51 challenging sequences. OTB-100 including 100 sequences is the extension of OTB-2013 where 50 more challenging sequences are selected as a small benchmark denoted as OTB-50. In this paper, the overlap success rate and distance precision metrics [32] are used to evaluate trackers on OTB-2013, OTB-50, and OTB-100. Overlap success rate measures the intersection over union (IoU) of ground truth and predicted bounding boxes. The success plot shows the rate of bounding boxes whose IoU score is larger than a given threshold. We apply the overlap success rate in terms of Area Under Curve (AUC) to rank the trackers. The precision metric means the percentage of frame locations within a certain threshold distance from those of the ground truth. The threshold distance is set as 20 for all the trackers. VOT-2017 is the 2017 edition of Visual Object Tracking challenge [17] evaluating the short-term tracking performance. In this challenge, a tracker is restarted in the case of a failure, where there is no overlap between the predicted bounding box and ground truth. VOT-2017 updated the sequences in VOT-2016 [16] by replacing 10 easily tracking sequences with 10 more challenging videos. A new real-time challenge was proposed to evaluate trackers with the limit of real-time speed i.e. the tracker should update the tracking result for each frame at frequency higher than or equal to the video frame rate. If a new frame is available before the tracker responds, the last updated bounding box is assumed as the reported tracker output at the available frame. For this dataset, we evaluated the tracking performance under the real-time challenge in terms of Expected Average Overlap (EAO). EAO is a principled combination of accuracy (overlap with the ground-truth) and robustness (failure rate) [15].

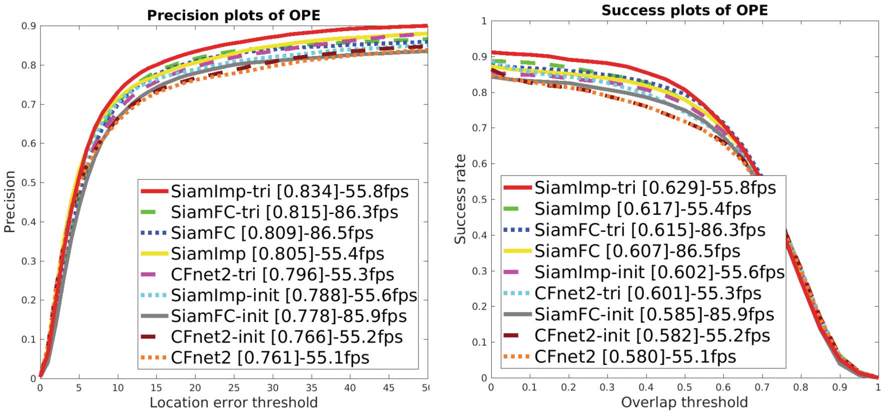
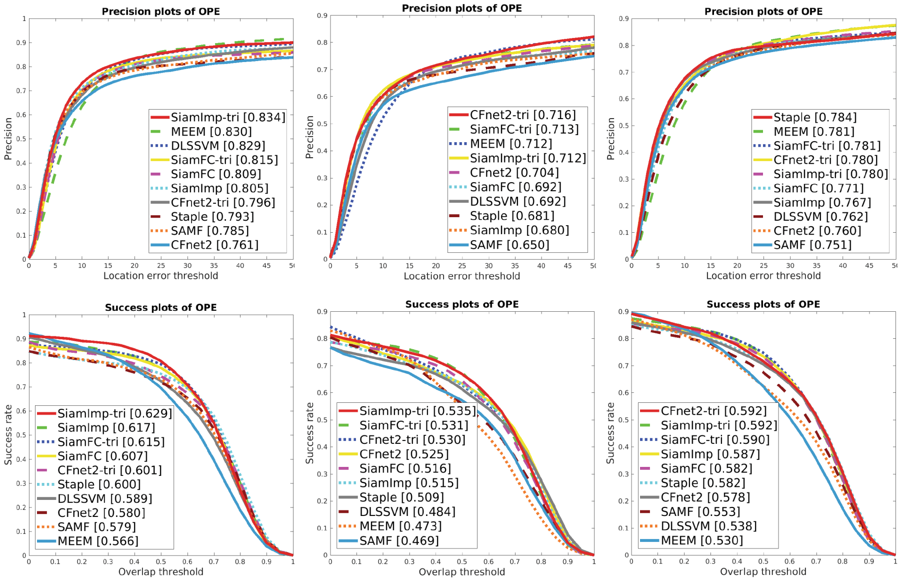


Fig. 3. Self-comparisons with variants of baseline trackers. The plots show precision and overlap success rate with AUC on OTB-2013 [32] in terms of OPE.

6.3 Experiments on Baseline Trackers

To validate the effectiveness of our triplet loss, we compare the baseline trackers (SiamFC [2], CFnet2, and SiamImp [28]) against their different variants: SiamFC-init, CFnet2-init, SiamImp-init, SiamFC-tri, CFnet2-tri, and SiamImp-tri. The postfix '-init' means the variant is initialized with the original pre-trained model and trained again with original logistic loss over 10 epochs with the aforementioned hyper-parameters. Similarly, the '-tri' represents it is trained with our triplet loss over 10 epochs with the same initialization and hyper-parameters.

These trackers are evaluated with one-pass evaluation (OPE) on OTB-2013, via running them throughout a test sequence with initialization from the ground truth position in the first frame. As shown in Fig. 3, directly training more epochs using logistic loss will reduce the precision and AUC of most baseline trackers excepting CFnet2. It indicates that the logistic loss can not enhance the representation power of original networks by training more iterations. However, the proposed triplet loss can further mine the potential of original networks to achieve more powerful representation. The corresponding results in Fig. 3 show it improves the performance in terms of both precision and overlap success rate in all the baseline trackers. It is worth mentioning all of the variants with triplet loss operate at almost the same high speed with baselines.



(a) OTB-2013 (b) OTB-50 (c) OTB-100

Fig. 4. Precision and success plots with AUC for OPE on OTB-2013 [32], OTB-50 and OTB-100 [33] benchmark. Only 10 best ranked trackers are shown.

6.4 Comparisons on OTB Benchmarks

On OTB-2013 [32], OTB-50, and OTB-100 [33] benchmarks, we compare improved trackers: SiamFC-tri, CFnet2-tri and SiamImp-tri against several state-of-the-art real-time trackers: SiamFC [2], CFnet2, SiamImp [28], Staple [1], CN [5], and KCF [12]. For reference, we also compare with recent trackers: DSST [5], MEEM [35], SAMF [19], and DLSSVM [21].

Overall comparison. Both precision and success metrics are reported for OPE. Figure 4 shows that all of our improved trackers SiamFC-tri, SiamImp-tri, and CFnet2 achieve improvement compared with their baselines in these three benchmarks in terms of both precision and success metrics. Especially on OTB-50 with success metric, SiamImp-tri achieves 3.9% improvement compared with its baseline SiamImp. In success metric, our trackers perform better than all other trackers on these three benchmarks, where our variants (SiamImp-tri and CFnet2-tri) occupy top two ranks. In precision metric, SiamImp-tri achieves the best performance on OTB-2013 as well as CFnet2-tri ranks first on OTB-50.

On OTB-100, our tracker SiamFC-tri ranks third (0.781) slightly lower than the second MEEM (0.781) and the first Staple (0.784) in precision while increases the success rate from 0.530 (MEEM) and 0.582 (Staple) to 0.590, respectively.

Attribute-based Performance Analysis. In OTB-100 benchmark, the sequences are annotated with 11 attributes for different challenging factors including Illumination Variation (IV), Scale Variation (SV), Occlusion (OCC), Deformation (DEF), Motion Blur (MB), Fast Motion (FM), In-Plane Rotation (IPR), Out-of-Plane Rotation (OPR), Out-of-View (OV), Background Clutters (BC), and Low Resolution (LR). To evaluate the proposed method in terms of each challenging factor, we compare our method to other trackers with different dominate attributes. Figure 5 shows the results of 9 main challenging attributes evaluated by the overlap success rate of OPE in terms of AUC. Our improved trackers outperform other trackers in 7 subsets, where SiamFC-tri rank first in 3 subsets: FM, OPR, and OV, SiamImp-tri performs best in OCC and IPR, CFnet2-tri achieves the best performance in SV and BC. In other subset LR, our SiamFC-tri ranks second with 0.615 in AUC slightly lower than the first SiamFC with 0.619. Similarly, in subset MB, DLSSVM performs best with 0.571 AUC slightly higher than our CFnet2-tri (0.568). Compared with baseline trackers, our trackers outperform than them in almost all subsets except for one case. In LR, SiamFC ranks higher than SiamFC-tri.

Table 1. EAO scores of VOT-2017 real-time challenge for our improved trackers: SiamFCT, CFnet2T, SiamImT, their baselines: SiamFC [2], CFnet2, SiamImp [28], recent tracker PTAV [10], and the other top 9 trackers in VOT-2017 [15].

	SiamFCT	SiamImT	CFnet2T	PTAV	DACF	ECOhc	Staple	KFebT	ASMS
EAO	**0.2125**	0.1833	0.1080	0.0654	0.2120	0.1767	0.1696	0.1693	0.1678
	SiamFC	SiamImp	CFnet2	sskcf	csrf	UCT	mosse_ca	SiamDCF	KCF
EAO	0.1966	0.1728	0.0963	0.1638	0.1585	0.1447	0.1395	0.1347	0.1336

6.5 Results on VOT-2017

Real-time challenge: We compare our improved trackers: SiamFC-tri, CFnet2-tri, SiamImp-tri, their baselines: SiamFC [2], CFnet2, SiamImp [28], recent tracker PTAV [10], and the top 9 trackers in VOT-2017 by using real-time evaluation. For simplicity, we shortened the names of our improved trackers as SiamFCT, CFnet2T and SiamImT. As shown in Table 1, all of our trackers also outperform their baseline trackers on VOT-2017 in terms of Expected Average Overlap (EAO). Especially, our SiamFCT achieves the best EAO among all these compared trackers. Another variant with our triplet loss SiamImT also occupies top position at the 4th ranking among all the trackers.

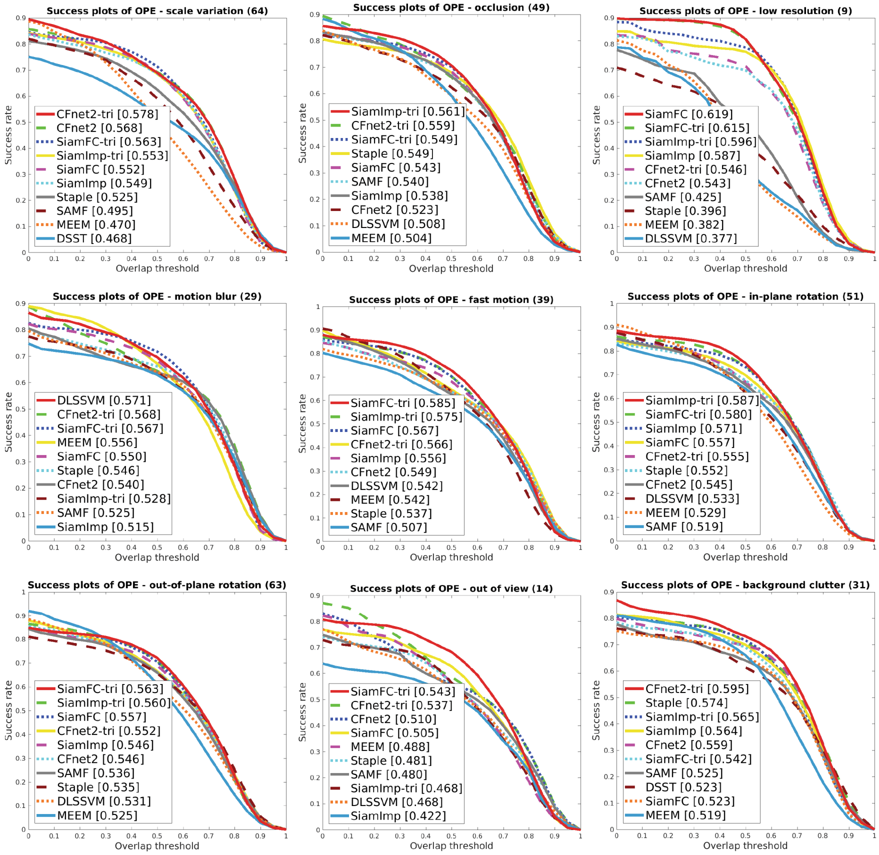


Fig. 5. Overlap success plots of OPE with AUC for 9 tracking challenges on OTB-100.

7 Conclusions

In this paper, we have proposed a novel triplet loss to achieve more powerful feature for object tracking by applying it into Siamese network. In contrast to original logistic loss, our triplet loss can further mine potential relationships among samples and utilize more elements for better training performance. We have shown the effectiveness of the proposed triplet loss in theory and experiments. In theoretical analysis, we found that when the network outputs wrong similarity scores, it gives more absolute gradients for feedback in back-propagation. We added this triplet loss into three baseline trackers based on Siamese network for experiments. The results on popular tracking benchmarks show that our triplet loss can improve the performance without reducing speed for these baselines.

Acknowledgements. This work was supported in part by the Beijing Natural Science Foundation under Grant 4182056, and the Fok Ying-Tong Education Foundation for

Young Teachers under Grant 141067. Specialized Fund for Joint Building Program of Beijing Municipal Education Commission.

References

1. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.: Staple: complementary learners for real-time tracking. In: IEEE CVPR, pp. 1401–1409 (2016)
2. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: Fully-convolutional siamese networks for object tracking. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9914, pp. 850–865. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48881-3_56
3. Cheng, D., Gong, Y., Zhou, S., Wang, J., Zheng, N.: Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1335–1344 (2016)
4. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: Eco: efficient convolution operators for tracking. In: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, pp. 21–26 (2017)
5. Danelljan, M., Häger, G., Khan, F., Felsberg, M.: Accurate scale estimation for robust visual tracking. In: BMVC (2014)
6. Danelljan, M., Häger, G., Khan, F.S., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: IEEE ICCV, pp. 4310–4318 (2015)
7. Danelljan, M., Robinson, A., Shahbaz Khan, F., Felsberg, M.: Beyond correlation filters: learning continuous convolution operators for visual tracking. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9909, pp. 472–488. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46454-1_29
8. Dong, X., Shen, J., Wang, W., Liu, Y., Shao, L., Porikli, F.: Hyperparameter optimization for tracking with continuous deep q-learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 518–527 (2018)
9. Dong, X., Shen, J., Yu, D., Wang, W., Liu, J., Huang, H.: Occlusion-aware real-time object tracking. IEEE Trans. Multimedia **19**(4), 763–771 (2017)
10. Fan, H., Ling, H.: Parallel tracking and verifying: a framework for real-time and high accuracy visual tracking. In: Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy (2017)
11. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: Exploiting the circulant structure of tracking-by-detection with Kernels. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7575, pp. 702–715. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33765-9_50
12. Henriques, J.F., Rui, C., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Trans. Pattern Anal. Mach. Intell. **37**(3), 583–596 (2015)
13. Hermans, A., Beyer, L., Leibe, B.: In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737 (2017)
14. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: Feragen, A., Pelillo, M., Loog, M. (eds.) SIMBAD 2015. LNCS, vol. 9370, pp. 84–92. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24261-3_7
15. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Čehovin Zajc, L., et al.: The visual object tracking VOT2017 challenge results. In: Visual Object Tracking Workshop 2017 at ICCV 2017 (2017)

16. Kristan, M., et al.: The visual object tracking VOT2016 challenge results. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9914, pp. 777–823. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48881-3_54
17. Kristan, M., et al.: A novel performance evaluation methodology for single-target trackers. IEEE Trans. Pattern Anal. Mach. Intell. **38**(11), 2137–2155 (2016)
18. Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Čehovin, L., et al.: The visual object tracking VOT2015 challenge results. In: Visual Object Tracking Workshop 2015 at ICCV 2015 (2015)
19. Li, Y., Zhu, J.: A scale adaptive kernel correlation filter tracker with feature integration. In: Agapito, L., Bronstein, M.M., Rother, C. (eds.) ECCV 2014. LNCS, vol. 8926, pp. 254–265. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16181-5_18
20. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: IEEE CVPR (2016)
21. Ning, J., Yang, J., Jiang, S., Zhang, L., Yang, M.H.: Object tracking via dual linear structured SVM and explicit feature map. In: IEEE CVPR, pp. 4266–4274 (2016)
22. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
23. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)
24. Shen, J., Liang, Z., Liu, J., Sun, H., Shao, L., Tao, D.: Multiobject tracking by submodular optimization. IEEE Trans. Cybern. (2018, in press). https://doi.org/10.1109/TCYB.2018.2803217
25. Shen, J., Yu, D., Deng, L., Dong, X.: Fast online tracking with detection refinement. IEEE Trans. Intell. Transp. Syst. **19**(1), 162–173 (2017)
26. Song, H.O., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4004–4012. IEEE (2016)
27. Tao, R., Gavves, E., Smeulders, A.W.: Siamese instance search for tracking. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1420–1429. IEEE (2016)
28. Valmadre, J., Bertinetto, L., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: End-to-end representation learning for correlation filter based tracking. In: IEEE CVPR, pp. 5000–5008 (2017)
29. Vedaldi, A., Lenc, K.: Matconvnet: Convolutional neural networks for matlab. In: Proceedings of the 23rd ACM International Conference on Multimedia, pp. 689–692. ACM (2015)
30. Wang, F., Zuo, W., Lin, L., Zhang, D., Zhang, L.: Joint learning of single-image and cross-image representations for person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1288–1296 (2016)
31. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: Advances in Neural Information Processing Systems, pp. 1473–1480 (2006)
32. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: IEEE CVPR, pp. 2411–2418 (2013)
33. Yi, W., Jongwoo, L., Yang, M.H.: Object tracking benchmark. IEEE Trans. Pattern Anal. Mach. Intell. **37**(9), 1834–1848 (2015)

34. Yilmaz, A., Javed, O., Shah, M.: Object tracking: a survey. ACM Comput. Surv. (CSUR) **38**(4), 13 (2006)
35. Zhang, J., Ma, S., Sclaroff, S.: MEEM: robust tracking via multiple experts using entropy minimization. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8694, pp. 188–203. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10599-4_13
36. Zhang, T., Xu, C., Yang, M.H.: Multi-task correlation particle filter for robust object tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, p. 3 (2017)
37. Zhuang, B., Lin, G., Shen, C., Reid, I.: Fast training of triplet-based deep binary embedding networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5955–5964 (2016)