

Quadruplet Network with One-Shot Learning for Fast Visual Object Tracking

Xingping Dong¹, Jianbing Shen¹, Yu Liu¹, Wenguan Wang¹, and Fatih Porikli²

¹ School of Computer Science,
Beijing Institute of Technology, China
{dongxingping, shenjianbing, liuyu}@bit.edu.cn,
wenguanwang.ai@gmail.com

² Research School of Engineering,
Australian National University, Australia
fatih.porikli@anu.edu.au

Abstract. In the same vein of discriminative one-shot learning, Siamese networks allow recognizing an object from a single exemplar with the same class label. However, they do not take advantage of the underlying structure of the data and the relationship among the multitude of samples as they only rely on pairs of instances for training. In this paper, we propose a new quadruplet deep network to examine the potential connections among the training instances, aiming to achieve a more powerful representation. We design four shared networks that receive multi-tuple of instances as inputs and are connected by a novel loss function consisting of pair-loss and triplet-loss. According to the similarity metric, we select the most similar and the most dissimilar instances as the positive and negative inputs of triplet loss from each multi-tuple. We show that this scheme improves the training performance. Furthermore, we introduce a new weight layer to automatically select suitable combination weights, which will avoid the conflict between triplet and pair loss leading to worse performance. We evaluate our quadruplet framework by model-free tracking-by-detection of objects from a single initial exemplar in several Visual Object Tracking benchmarks. Our extensive experimental analysis demonstrates that our tracker achieves superior performance with a real-time processing speed of 78 frames-per-second (fps).

Keywords: Quadruplet deep network; Visual object tracking; Siamese networks.

1 Introduction

Deep learning models have attracted significant attention thanks to their powerful regression capacity in a spectrum of applications from speech recognition to natural language processing and computer vision. It is recognized that training of deep neural networks requires a large corpus of labeled data to attain generality of the learned models and robustness of the feature maps. Such networks seem less useful for one-shot learning tasks where the objective is to learn a model, often in an online fashion, from a single exemplar (or a few). One exemption is the embedding with Siamese networks [1,2,3] since it is not necessary to retrain the deep model for a newly given object or class.

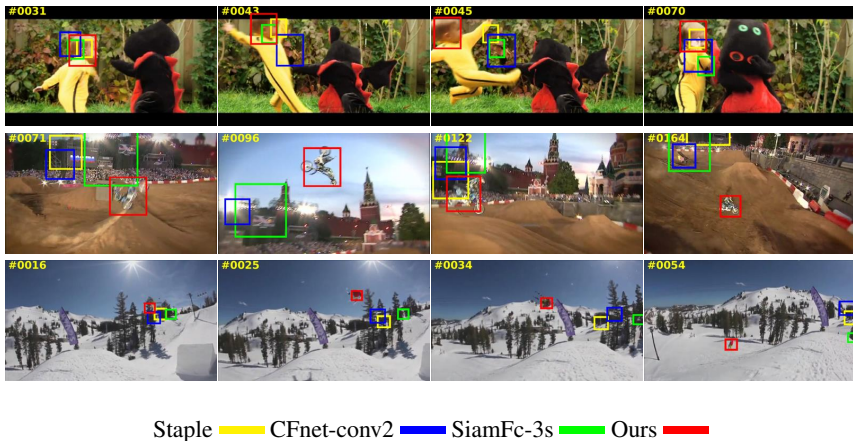


Fig. 1. Sample results of our quadruplet network for visual tracking. By only changing the training loss on base Siamese network [3] to get more powerful feature, we achieve a significant improvement on the object tracking task. For instance, our method improves the distance precision [4] from 30.1% to a higher score of 86.7% on *DragonBaby* (top row). It also outperforms the referenced real-time trackers Staple [5] and CFnet-conv2 [6].

Siamese architectures can identify other instances of the target class from its original exemplar using a fixed model.

Conventional Siamese networks use tuples of two labeled instances for training [1,7,8,2,3]. They are sensitive to calibration and the adopted notion of similarity vs. dissimilarity depending on the given context [9]. The requirement of calibration can be removed by applying triplets for training with a contrastive loss, which favors a small distance between pairs of exemplars labeled as similar, and a large distance for pairs labeled dissimilar [9]. At the same time, using triplets enables utilization of the underlying connections among more than two instances. Our intuition is that more instances (larger tuples) lead to better performance in the learning process. Therefore, we design a new network structure adding as many instances into a tuple as possible (including a triplet and multiple pairs) and connect them with a novel loss combining a pair-loss and a triplet based contractive-loss. Existing Siamese [3] or triplet networks [9] do not use the full potential of the training instances since they take randomly sampled pairs or triplets to construct the training batches. In contrast, our framework aims to select the triplets that would lead to a powerful representation for a stronger deep network.

In this paper, we introduce a novel quadruplet network for one-shot learning. Our quadruplet network is a discriminative model to one-shot learning. Given a single exemplar of a new object class, its learned model can recognize objects of the same class. To capture the underlying relationships of data samples, we design four branch networks with shared weights for different inputs such as instances, exemplar, positive and negative branches. We randomly sample a set of positive and negative instances as inputs of instances branch. A pair of loss function is designed to connect the exemplar and instances branch to utilize the underlying relationships of pairs. To achieve the triplet

of representation, we select the positive instance that is most similar to the exemplar and the negative instance that is most dissimilar as the inputs of positive and negative branches, respectively. Then, we use a contractive loss function to measure the similarity of this triplet. Finally, the weighted average of the pair loss and the contractive loss is assigned as the final loss.

The triplet loss is the key to utilize the underlying connections among instances to achieve improved performance. To combine it and pair loss, a simple solution is to apply a weighted average with prior weights between these two losses. However, directly applying prior weights maybe not improve even reduce performance. For example, we test our approach with prior weights for visual object tracking on OTB-2013 benchmark [4] while the distance precision is reduced from 0.809 to 0.800. Thus, we propose a weight layer to automatically choose suitable combination weights during training to solve this problem. We evaluate the quadruplet network with one-shot learning for visual object tracking. SiamFc-3s [3] is our baseline tracker. We apply our quadruplet network instead of its Siamese network to train the shared net and adapt the same mechanism for online tracking. As shown in Fig. 1, our training method achieves better tracking accuracy, which demonstrates the more powerful representation of our framework. In several popular tracking benchmarks, our experimental results show that the tracker runs at high real-time speed (78 frames-per-second on OTB-2013) and achieves excellent results compared with recent state-of-the-art real-time trackers.

The main contributions of this work are summarized as:

- We propose a novel quadruplet network for one-shot learning that utilizes the inherent connections among multiple instances and apply it to visual tracking. To the best of our knowledge, we are the first to introduce quadruplet network into single object tracking.
- A weight layer is proposed for selecting suitable combination weights between triplet and pair loss. It may take a lot of time to manually choose appropriate combination weights while our weight layer is able to automatically adjust these weights during each training iteration to achieve powerful features.
- By applying the proposed quadruplet network on training to get more representable features, our detection-by-tracking method can achieve state-of-the-art results even without online updating during tracking. Furthermore, our tracker runs beyond real-time with high speed of 78 fps.

2 Related Work

Our work is related to a wide array of literature, in particular, one-shot learning with generative models, learning an embedding space for one-shot learning, and visual object tracking.

Many previous studies focus on the context of the generative models for one-shot learning, which is different from our formulation of the problem as a discriminative task. One early approach [10] uses probabilistic generative models to present object categories and applies a variational Bayesian framework for learning useful information from a handful of training samples. In [11], a recurrent spatial attention generative model

is proposed to generate images using a variational Bayesian inference. Having seen samples once, this method generates a set of diverse samples.

The most common discriminative approach to one-shot learning is embedding learning, where the goal is to learn an embedding space and then perform classification by applying a simple rule in the embedding space such as finding the nearest-neighbor of an exemplar of a novel category. Learning embeddings of objects is to represent each object as a low-dimensional vector. It is essential in unsupervised learning and in data preprocessing of supervised learning. A typical model for embedding learning is Siamese networks [1]. Recently, several techniques [2,3,9] are presented to improve the embedding performance. In [2] a learning-to-learn approach is proposed to determine the shared weights of Siamese network. It applies a number of factorizations of the parameters to make the construction feasible. In [3], a fully convolutional Siamese network is designed for visual object tracking. As an alternative to using pairs in Siamese network, [9] employs triplets for training by distance comparisons. In comparison, our method combines the pairs and triplets for training to take the advantage of the underlying useful connections (e.g. the manifold structure) among samples.

The task of visual object tracking can be viewed as an application of one-shot learning [2]. We only provide a brief introduction of recent object tracking methods such as correlation filter based trackers and deep neural networks based solutions. After Kernelized Correlation Filters [12] had reported state-of-the-art performance at hundreds of frame-per-second speed, more investigation focused on correlation filters [13,14,15,16,5,17,18,19,20,21]. The tracking performance improved consistently, and DSST [13] and C-COT [17] obtained the top ranks at the VOT 2014 [22] and the VOT 2016 challenges [23], respectively. However, the speed became lower and lower, i.e. the speed of C-COT is only 1 fps. The computational load remains as the same problem in deep neural networks for tracking. The online learning network [24] provided superior tracking performance, albeit its prohibitive load (near 1 fps on GPU) limits its practicality. To accelerate deep network for tracking, some studies proposed off-line training of models, such as DeepTrack [25], GOTURN [26] and SiamFc-3s [3], and directly applied them for online tracking to avoid the cost of online retraining. All of these achieve a speed of more than real-time (30 fps) with a comparable performance. Our work is similar to SiamFc-3s, while we focus on improving off-line training phase to obtain more robust feature representation. Thus, our method runs at a comparable speed with SiamFc-3s yet achieves better performance.

3 Siamese Networks for Tracking

Here, we briefly review recent work on Siamese networks for object tracking [3]. When applying one-shot learning scheme for visual tracking, the object patch in the first frame is used as an exemplar, and the patches in the search regions within the consecutive frames are employed as the candidate instances. The aim is to find the most similar instance from each frame in an embedding space, where the object is represented as a low-dimensional vector. Learning an embedding function with a powerful and discriminative representation is a critical step in this task.

To this end, Bertinetto *et al.* [3] apply a deep learning method to learn the embedding function and design a fully convolution Siamese network to reduce computation for real-time speed. This network includes two network branches processing different inputs. One is the exemplar branch used to receive the object bounding box in the first frame. The other is the instances branch applied to process the patches in searching regions of the following frames. These two network branches share the parameters; thus they can be seen as an identical transformation ϕ for different inputs. Accordingly, the similar function for an exemplar z and an instance x is defined as $f(z, x) = g(\phi(z), \phi(x))$, where g is a simple similarity metric such as vectorial angle and cross-correlation. In [3], they use the cross-correlation for g , and the formulation of function f is transferred as follows:

$$f(z, x) = \phi(z) * \phi(x) + b. \quad (1)$$

using the convolution $*$ operator. Then, a logistical loss is applied to the pair-wise loss function, which is formulated as follows:

$$L_p(y, v) = \sum_{u \in \mathcal{D}} \log(1 + e^{(-y[u] \cdot v[u])}). \quad (2)$$

where \mathcal{D} is the set of inputs for instances branch, $y[u] \in \{+1, -1\}$ is the ground-truth label of a single exemplar-instance pair (z, u) , $v[u]$ is the similarity score of (z, u) i.e. $v[u] = f(z, u)$.

4 Quadruplet Network

A quadruplet network (inspired by Siamese networks [1,3] and triplet networks [9]) contains four branches of the same network with shared parameters, while building a direct connection between them through a common loss function. Our proposed quadruplet network structure is shown in Fig. 2.

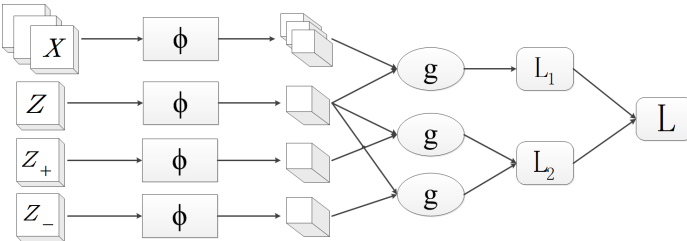


Fig. 2. The structure of our quadruplet network. Here, x, z, z_+, z_- are different inputs of four shared branch networks respectively corresponding to instances, exemplar, positive and negative branches. ϕ represents a convolution network with shared weights. g is a function of similarity metric. L_1, L_2 , and L are loss functions.

4.1 Network branches

These four network branches are respectively named as exemplar, instances, positive, and negative branches according to their inputs. We first introduce the input of exemplar branch since other inputs are depended on it. An instance is randomly chosen as an exemplar z . Then the instances with the same class are denoted as positive instances and the others are denoted as negative instances. The inputs of instances branches (denoted as a set \mathcal{D}) consist of some positive instances and negative instances, which are used for two aspects.

Firstly, these instances are used to learn an function of similarity metric $f(z, x)$ that compares an exemplar z to an instance x and returns a high score with a positive instance and a low score with a negative instance. Secondly, according to these scores, we select the instance with lowest score from positive instances as the input of positive branch z_+ and select the one with highest score from negative instances as the input of negative branch z_- . z_+ and z_- have the strongest representation among the inputs of instances branch, since they are on the boundaries of classification in current small sample space (defined on the set \mathcal{D}). If we decrease the distance between z_+ and z , and increase it between z_- and z , the corresponding positive boundary will be closer to z and negative one will keep away from z . In other words, the margin between boundaries of classification will be increased and it will reduce the error of classification. To achieve this purpose, we design a new loss function for these three branches. More details are shown in the next subsection.

4.2 Loss function

The loss function of a quadruplet network consists of two kinds of loss function. The first one is constructed between the outputs of exemplar branch and instances branch, and the second one connects positive, negative and exemplar branches. To give the definition of these two loss functions, we first simplify the notation of branches. In fact, these 4 branches are an identical transformation to different inputs. Thus, we denote them as a transformation function ϕ . Then the function f is transfered as $f(z, x) = g(\phi(z), \phi(x))$, where g is a simple similarity metric such as vectorial angle and cross correlation. In this paper, we use the cross correlation for g as the same as Siamese network [3]. And the formulation of function f is defined as the same as the function in Eq. (1).

Then, we apply a weighted averaged logistical loss to the first pair-wise loss function, which is formulated as follows:

$$L_1(y, v) = \sum_{u \in \mathcal{D}} w[u] \log(1 + e^{(-y[u] \cdot v[u])}). \quad (3)$$

where \mathcal{D} is the set of inputs for instances branch, $y[u] \in \{+1, -1\}$ is the ground-truth label of a single exemplar-instance pair (z, u) , $v[u]$ is the similarity score of (z, u) i.e. $v[u] = f(z, u)$, $w[u]$ is the weight for an instance u , and $\sum_{u \in \mathcal{D}} w[u] = 1, w[u] > 0, u \in \mathcal{D}$. $w[u]$ is a dynamic weight that is changed during the training process. Different to general training method for deep learning, we first do forward computation once to get the similarity score of each exemplar instance. If the similarity score of a negative instance is larger than the minimum score of positive instances, it will violate the

assumption of similarity, i.e. the score of the positive instance should be larger than the negative instance. Thus, we amplify the penalty for this situation by increasing the corresponding weights. In this work, we set $w[u] = 2w[u]$ for the cases violating the assumption, and then normalize the weights by dividing their summation. These adapted weights are applied for forward computation and backward computation to finish a training process.

Inspired by the triplet network [9], the second loss function is constructed by a mean square error on the soft-max results of similarity scores between different branches. Comparing soft-max results to (0, 1) vector, we can get the triplet loss function:

$$L_2(s_+, s_-) = \|(s_+ - 1, s_-)\|_2^2, \quad (4)$$

where

$$s_+ = \frac{e^{f(z, z_+)}}{e^{f(z, z_+)} + e^{f(z, z_-)}} \quad (5)$$

and

$$s_- = \frac{e^{f(z, z_-)}}{e^{f(z, z_+)} + e^{f(z, z_-)}}. \quad (6)$$

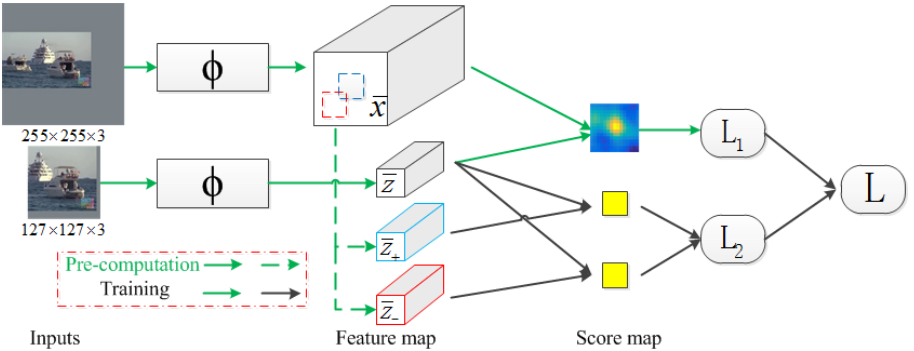


Fig. 3. The framework of a single training iteration during the tracking process, which consists of a precomputing phase and a training phase. The green arrows represent the precomputation, which aims to select the powerful positive and negative feature maps (\bar{z}_+ and \bar{z}_-) from feature map of instances \bar{x} . They produce the adapted weights for pairs loss L_1 . The solid arrows indicate the training phase include the forward and backward computations. The definitions of ϕ , L_1 , L_2 , L are the same as Fig. 2.

The final loss function is the weighted sum of these two loss functions:

$$L = \frac{1}{\sum_{i=1}^2 \bar{w}[i]} \sum_{i=1}^2 \bar{w}[i] L_i, \quad (7)$$

where $\bar{w}[1], \bar{w}[2] > 0$ are the weights to balance the loss L_1 and L_2 . They are not prior constants but learned during training. The parameters of branch network θ and the

weights of loss \bar{w} are obtained by applying Stochastic Gradient Descent (SGD) to the problem:

$$\arg \min_{\theta, \bar{w}} \mathbb{E} L(z, z_+, z_-, \mathcal{D}, y; \theta, \bar{w}). \quad (8)$$

For our neural networks, we propose three new layers using L_1 , L_2 and L losses. The backward computation of L_1 loss is similar to the logistic loss since the weights are fixed during backward computing. We only need to add the weights into the derivatives of logistical loss. The other two are not similar to the common loss. Thus, we give the gradients of these two losses for backward computation. L_2 loss layer can be decomposed as a square error layer (Eq. 4) and a soft-max layer (Eq. 5, 6). We get the derivatives of the square error as:

$$\frac{\partial L_2}{\partial s_+} = 2(s_+ - 1) = -2s_-, \quad \frac{\partial L_2}{\partial s_-} = 2s_-. \quad (9)$$

To avoid big gradients in the soft-max layer, we reformulate Eq. 5 as follows:

$$s_+ = \frac{e^{f_+}/e^m}{(e^{f_+} + e^{f_-})/e^m} = \frac{e^{f_+ - m}}{e^{f_+ - m} + e^{f_- - m}}. \quad (10)$$

Similarly, the reformulation of Eq. 6 becomes

$$s_- = \frac{e^{f_- - m}}{e^{f_+ - m} + e^{f_- - m}}. \quad (11)$$

Where f_+ and f_- are similarity scores $f(z, z_+)$ and $f(z, z_-)$, m is the maximal between f_+ and f_- i.e. $m = \max(f_+, f_-)$. Then we can get Jacobian matrix \mathbf{J} :

$$\mathbf{J} = \begin{bmatrix} \frac{\partial s_+}{\partial f_+} & \frac{\partial s_+}{\partial f_-} \\ \frac{\partial s_-}{\partial f_+} & \frac{\partial s_-}{\partial f_-} \end{bmatrix} = \begin{bmatrix} s_+(1 - s_+) & -s_+s_- \\ -s_+s_- & s_-(1 - s_-) \end{bmatrix} \quad (12)$$

According to the chain rule, we can get the partial derivatives as follows:

$$\begin{bmatrix} \frac{\partial L_2}{\partial f_+} \\ \frac{\partial L_2}{\partial f_-} \end{bmatrix} = \mathbf{J}^T \begin{bmatrix} \frac{\partial L_2}{\partial s_+} \\ \frac{\partial L_2}{\partial s_-} \end{bmatrix} = \begin{bmatrix} -4s_+s_-^2 \\ 4s_+s_-^2 \end{bmatrix} \quad (13)$$

The loss layer L contains two inputs (L_1 and L_2) and two parameters ($\bar{w}[1]$ and $\bar{w}[2]$), which partial derivatives are formulated as:

$$\frac{\partial L}{\partial L_j} = w[j] \quad \text{and} \quad \frac{\partial L}{\partial \bar{w}[j]} = \frac{1}{\bar{w}_s^2} (\bar{w}_s L_j - \sum_{i=1}^2 \bar{w}[i] L_i) \quad (14)$$

where $j = 1, 2$ and $\bar{w}_s = \sum_{i=1}^2 \bar{w}[i]$. In practice, we set a small threshold T , \bar{w} as 0.01, for $\bar{w}[i]$ to insure $\bar{w}[i] > 0$ by imposing $\bar{w}[i] = \max(T, \bar{w}[i])$.

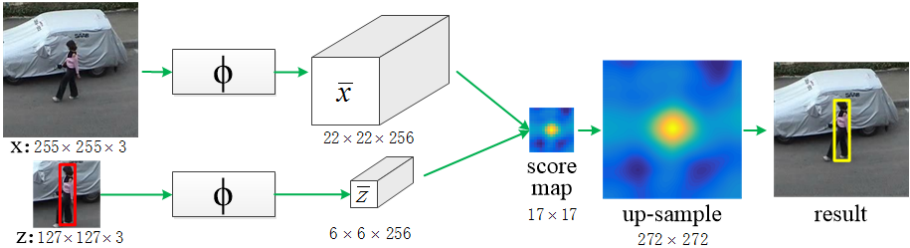


Fig. 4. The framework of online tracking. Above, z and x represent the input exemplar image and the search image, and \bar{z} and \bar{x} are the corresponding embedding features. The score map is calculated by the similarity between the sub-windows of \bar{x} and \bar{z} . After upsampling the score map, we find the location of the object in the search image according to the location of maximum score in the upsampled map.

4.3 Framework of our tracker

We apply the proposed Quadruplet network on an one-shot learning problem in visual object tracking. Given an object of interest on the first video frame, the task of single object tracking is to find the object on the following frames. In the opinion of one-shot learning, a sub-window enclosing the object is seen as an exemplar and the sub-windows of each frame are viewed as a candidate of instances. The goal is to find the most similar one with exemplar in each frame. Our tracking method consists of off-line training and online testing stages using different network structures.

In off-line training, we use quadruplet network to achieve powerful embedding representation. The architecture of shared convolutional network is the same as SiamFc-3s [3], which is a variation of the network of Krizhevsky et al. [27].

As shown in Fig. 3, we simplify the quadruplet network by selecting powerful feature patches at the last convolutional layer of instances branch instead of positive and negative branch. Pairs of an exemplar image and a larger search image are applied to inputs of exemplar and instances branch, where each sub-window of the same size with the exemplar image is an candidate instance. The scores v in equation (3) will become a score map as shown in Fig. 3. It's label map is designed according the location i.e. we set positive label +1 to points in the central area within radius R ($R = 2$ in our experiments), which is denoted as positive area \mathcal{D}_p , and ones in other area (negative area \mathcal{D}_n) are set as -1 . Before each iteration training, we first precompute the score map, and respectively select the instance in the center and the one with highest score in negative area as the powerful positive and negative instance. Their last convolutional features are set as the inputs of triplet loss. Otherwise, the precomputed score map is also used to construct the weighted pair loss. The initial weights are defined as balance weights to balancing the number of positive and negative instances. The formulation is defined as:

$$w[u] = \begin{cases} \frac{1}{2|\mathcal{D}_p|}, & u \in \mathcal{D}_p \\ \frac{1}{2|\mathcal{D}_n|}, & u \in \mathcal{D}_n \end{cases} \quad (15)$$

If a negative score is more than the minimal of positive scores, we increase the weight as $w[u] = 2w[u]$ and then normalize all weights as $w[u] = w[u] / \sum w[u]$. After precomputation, we do forward and backward computation to finish a training iteration.

In online testing (tracking) phase, only the exemplar branch and the instances branch are used. We crop an exemplar image in the first frame and also larger search images (search region) in the consecutive frames. The search images center on the location of the object in the previous frames. The exemplar and the search images are resized to 127×127 and 255×255 , respectively. Using these inputs, our tracking network calculates a score map as shown in Fig. 4. Then, the score map is upsampled by bicubic interpolation from 17×17 to 272×272 to achieve a higher location accuracy. The location of the object is determined by the maximum score in the upsampled map.

5 Experimental Results

5.1 Implementation details

Training. We use MatConvNet [28] to train the parameters of the shared network θ and the weights of loss \bar{w} by minimizing Eq. (8) with SGD. The initial values of the shared network are copied from the trained model in SiamFc-3s [3] and the weights are set as (0.9, 0.1). We use the same training and validation sets with [3]. They are randomly sampled from the ‘new object detection from video challenge’ in the 2015 edition of the ImageNet Large Scale Visual Recognition Challenge [10] (ILSVRC). The dataset contains almost 4500 videos with 30 different classes of animals and vehicles. Training is performed over 10 epochs, each consisting of 53,200 sampled pairs. We randomly select 10% pairs as the validation set at each epoch, and the final network used for testing is determined by the minimal mean error of distance (presented in [3]) 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^{-2} to 10^{-5} . To handle the gray videos in benchmarks, 25% of the pairs are converted to grayscale during training.

Tracking. As mentioned before, only the initial object is selected as the exemplar image. Thus, we compute the embedding feature $\phi(z)$ once and compare it with the searching images of the subsequent frames. To handle scale variations, three scales $1.0375^{\{-1,0,1\}}$ are searched for the object, and the scale is updated by linear interpolation with a factor of 0.59 to avoid huge variation of scale. Our Intel Core i7-6700 at 3.4 GHz machine is equipped with a single NVIDIA GeForce 1080, and our online tracking method runs at 78 frames-per-second.

5.2 Benchmarks and evaluation metric

We evaluate our tracking method, which we call as ‘Quad’ with the recent state-of-the-art trackers in popular benchmarks including OTB-2013 [4], OTB-50, OTB-100 [29], and VOT-2015 [30].

The OTB-2013 benchmark contains 50 challenging sequences and uses different metrics to evaluate tracking methods. The authors expand the OTB-2013 to OTB-100 including 100 sequences and select 50 more challenging sequences denoted OTB-50 as

a small benchmark. In this paper, we use the overlap success rate and distance precision metrics [4] 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 threshold 0.5 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 trackers. The VOT-2015 benchmark is a challenging dataset to evaluate the short-term tracking performance since in VOT-2015, a tracker is restarted in the case of a failure, where there is no overlap between the predicted bounding box and ground truth. It contains 60 sequences collected from some previous tracking benchmarks (356 sequences in total). For this dataset, we evaluated tracking performance in terms of accuracy (overlap with the ground-truth), robustness (failure rate), and Expected Average Overlap (EAO which is principled combination of accuracy and robustness) [30].

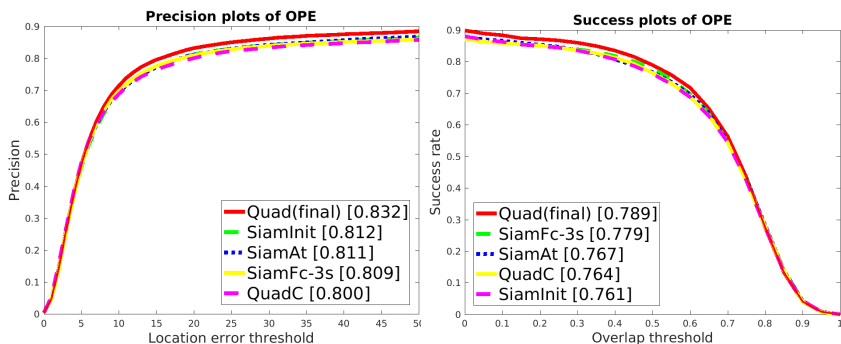


Fig. 5. Results of self-comparison with different variants of our tracker. The plots show the precision and overlap success rate on OTB-2013[4] in terms of OPE.

5.3 Ablation study

To evaluate the effect of different components in our method, we compare the tracker SiamFc-3s [3] against different variants of our tracker: SiamInit, SiamAt, QuadC, and Quad. SiamFc-3s-3s can be seen as the baseline of our method. Since our online tracking method is the same with it. The difference is the training model. SiamInit is the variant of SiamFc-3s, which is initialized with the final network parameters in SiamFc-3s and trained again over 10 epochs. SiamAt is also a Siamese network but trained with the proposed adapt weighted pairs loss. QuadC is the version that combines the weighted pairs loss and the triplet loss with constant weights $\bar{w} = [0.9, 0.1]$, and Quad is the final version with the learned weights.

We evaluate these trackers with one-pass evaluation (OPE) while running them throughout a test sequence with an initialization from the ground-truth position in the first frame. As shown in Fig. 5, directly training more epochs (SiamInit) will improve

precision but reduce overlap success rate compared with the baseline SiamFc-3s. This indicates that representation power of the original network was close to its limit. More training may not improve more performance. Thus, we seek for an alternative solution for improvement. Firstly, we try an adapt weighted pairs loss and achieve slightly overall improvement with increasing overlap while slightly reducing precision compared with SiamInit. Secondly, we want to add triplet loss to mine the potential relationships among samples further. However, directly adding triplet loss into a Siamese net with prior weights may reduce the performance (see QuadC in Fig. 5) since the prior weights may not be optimal for the combination of triplet loss and pair loss. To solve this problem, we design a weight layer to choose suitable combination weights during training and achieve better precision and overlap success rate as shown with Quad in Fig. 5.

5.4 Results on OTB-2013 benchmark

On OTB-2013 benchmark, we compare our Quad tracker against several state-of-the-art trackers that can operate in real-time: CFnet-conv2 [6], SiamFc-3s [3], Staple [5], CN [13], and KCF [12]. For reference, we also compare with recent trackers: DSST [13], MEEM [31], SAMF [32], DLSSVM [33].

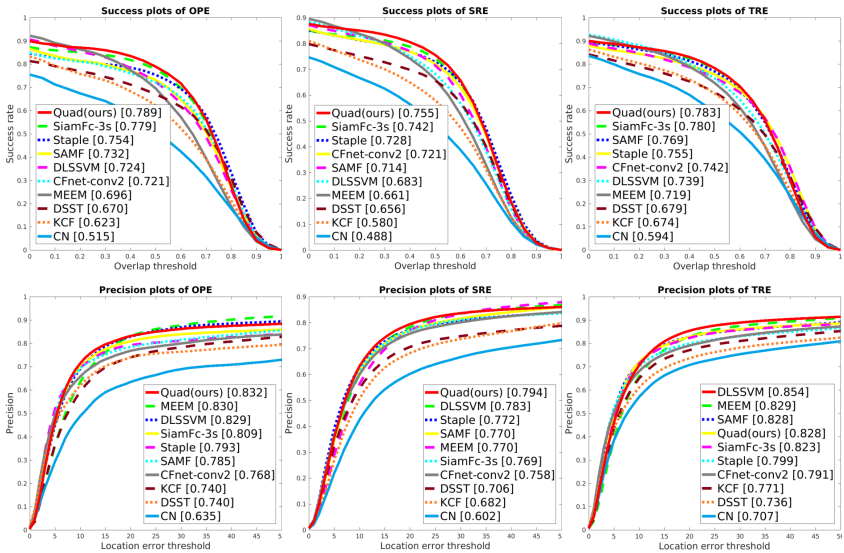


Fig. 6. Results of OTB-2013 [4] benchmark. Success and precision plots for OPE, SRE and TRE, which are one-pass evaluation, spatial robustness evaluation, and temporal robustness, respectively.

Overall comparison. A conventional way to evaluate trackers is one-pass evaluation (OPE). However, some trackers are sensitive to the initialization errors. To measure the

performance with different initializations, we use spatial robustness evaluation (SRE) and temporal robustness evaluation (TRE). SRE uses different bounding boxes in first frame and TRE starts at different frames for initialization. As shown in Fig. 6, our method outperforms over recent state-of-the-art real-time trackers in terms of the overlap success rate and precision for OPE, SRE, and TRE, respectively. Compared with our baseline SiamFc-3s, the results demonstrate that our training method is able to generate more robust and powerful features for tracking. Among all trackers, our tracker outperforms in five evaluation metric except the precision for TRE. In this metric, our method achieves the 4th rank (0.828), which is in fact very close to the 3rd SANF (0.828) and the 2nd MEEM (0.829). SAMF ranks 1st in precision while it ranks 6th in success for TRE.

5.5 Results on OTB-50 and OTB-100

On OTB-50 and OTB-100 benchmarks, we also compare the recent trackers mentioned on OTB-2013 comparison.

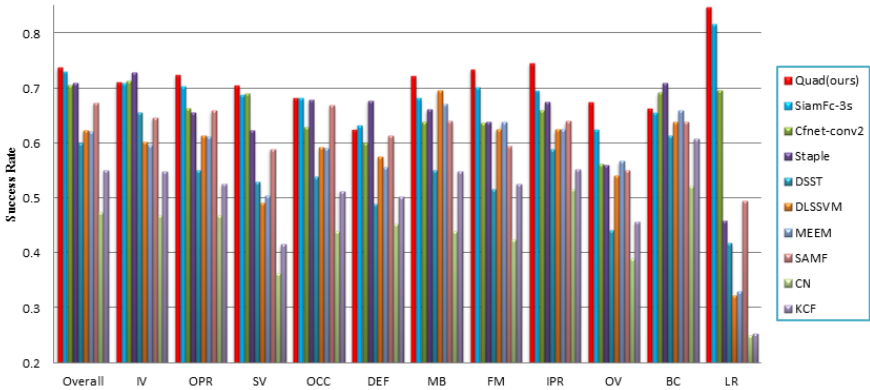
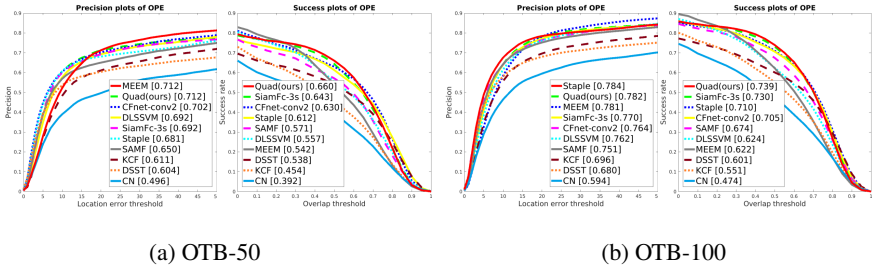


Fig. 7. Overlap success rates of OPE (with overlap threshold 0.5) for 11 attributes on OTB-100.

Attribute-based Performance Analysis. In OTB-100 benchmark, the sequences are annotated with 11 attributes for different challenging scenarios 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).

We evaluate our method on these different attributes as shown in Fig. 7 where the overlap success rates of OPE for overlap threshold is set to 0.5. As visible, our method outperforms all other trackers in 8 categories: SV, OCC, LR, MB, FM, IPR, OPR, and OV, especially in LR and IPR. In 3 categories, Staple performs the best while our approach ranks the third best. Staple tracker is based on the correlation filter and applies Fourier transform. We believe, integrating Fourier transform into our method will further improve the performance our method. Compared to the baseline SiamFc-3s, our tracker

has superior performance in all categories except DEF. The reason might be that the triplet loss has to be customized to handle this situation.



(a) OTB-50

(b) OTB-100

Fig. 8. Precision and success plots ranked with overlap threshold 0.5 for OPE on OTB-50 and OTB-100 [29] benchmark.

Overall comparison. Both precision and success metrics are reported for OPE. Fig. 8 shows that our tracker also achieve improvement compared with our baseline SiamFc-3s in these two benchmarks in terms of precision and success metrics. In success metric, our method performs better than all trackers on these two benchmarks. In precision metric, our tracker achieves second best performance on OTB-50 following the first one MEEM with slight reducing from 0.7122 to 0.7117 while we get significant improvement from 0.542 to 0.660. Similarly, on OTB-100, our tracker ranks second (0.782) slightly lower than the first one Staple (0.784) in precision while increases the success rate from 0.710 to 0.739.

Table 1. Evaluation on VOT-2015 by EAO, the weighted means of accuracy, robustness and speed. (*) values from the VOT-2015 results [30] in EFO units, which roughly correspond to FPS. The first and second best scores are highlighted in color.

	Quad(ours)	SiamFc-3s	BDF	NCC	FOT	ASMS	FCT	matFlow	SKCF	PKLTF	sumShift
EAO	0.261	0.248	0.153	0.080	0.139	0.212	0.151	0.150	0.162	0.152	0.234
Acc.	0.553	0.550	0.401	0.500	0.432	0.507	0.431	0.420	0.485	0.453	0.517
Rob.	1.791	1.818	3.106	11.345	4.360	1.846	3.338	3.121	2.681	2.721	1.682
FPS	78	78	175*	135*	126*	101*	73*	71*	58*	26*	15*

5.6 Results on VOT-2015

In our evaluations, we use the Visual Object Tracking 2015 (VOT-2015) toolkit, which contains the evaluation in short-term visual object tracking tasks.

Fast speed: We compare our tracker with SiamFc-3s [3] and 9 top participants in the VOT-2015 including BDF [34], FOT [35], ASMS [36], NCC, FCT, matFlow, SKCF, PKLTF [30], and sumShift [37]. Table 1 shows that our tracker achieves the best

Expected Average Overlap (EAO) and the highest accuracy among the most accurate trackers with speed more than 15 fps. Among the fast trackers, the highest robustness (1.682) belongs to sumShift followed by ours Quad (1.791). Our tracker significantly improves the accuracy and robustness of most participants with top speed in VOT-2015 (BDF, FOT, ASMS, NCC, FCT, matFlow, SKCF, PKLTF) and SiamFc-3s. For reference, the comparisons of other trackers are provided in supplementary material.

6 Conclusion

We demonstrated that the proposed quadruplet network using multi-tuples for training allows accurate mining of the potential connections among instances and achieves more robust representations for one-shot learning. We showed that by automatically adjusting the combinational weights between triplet and pair loss, we improve the training performance. We analyzed the feasibility of our quadruplet network in visual object tracking. Our results indicate that our tracking method outperforms others while maintaining beyond real-time speed. As future work, we employ different loss functions and extend the quadruplet networks to other computer vision tasks.

References

1. Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., Shah, R.: Signature verification using a siamese time delay neural network. *IJPRAI* 7(4) (1993) 669–688
2. Bertinetto, L., Henriques, J.F., Valmadre, J., Torr, P., Vedaldi, A.: Learning feed-forward one-shot learners. In: *NIPS*. (2016) 523–531
3. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: *ECCV*, Springer (2016) 850–865
4. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: *IEEE CVPR*. (2013) 2411–2418
5. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.: Staple: Complementary learners for real-time tracking. In: *IEEE CVPR*. (2016) 1401–1409
6. 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*. (2017) 5000–5008
7. Fan, H., Cao, Z., Jiang, Y., Yin, Q., Doudou, C.: Learning deep face representation. *arXiv preprint arXiv:1403.2802* (2014)
8. Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear cnn models for fine-grained visual recognition. In: *IEEE ICCV*. (2015) 1449–1457
9. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: *International Workshop on Similarity-Based Pattern Recognition*, Springer (2015) 84–92
10. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 28(4) (2006) 594–611
11. Rezende, D.J., Mohamed, S., Danihelka, I., Gregor, K., Wierstra, D.: One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106* (2016)
12. Henriques, J.F., Rui, C., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 37(3) (2015) 583–596

13. Danelljan, M., Häger, G., Khan, F., Felsberg, M.: Accurate scale estimation for robust visual tracking. In: *BMVC*. (2014)
14. Ma, C., Yang, X., Zhang, C., Yang, M.H.: Long-term correlation tracking. In: *IEEE CVPR*. (2015) 5388–5396
15. Liu, T., Wang, G., Yang, Q.: Real-time part-based visual tracking via adaptive correlation filters. In: *IEEE CVPR*. (2015) 4902–4912
16. Hong, Z., Chen, Z., Wang, C., Mei, X., Prokhorov, D., Tao, D.: Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In: *IEEE CVPR*. (2015) 749–758
17. Danelljan, M., Robinson, A., Shahbaz Khan, F., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: *ECCV*. (2016)
18. Lukežić, A., Vojír, T., Zajc, L.C., Matas, J., Kristan, M.: Discriminative correlation filter with channel and spatial reliability. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Volume 2. (2017)
19. Mueller, M., Smith, N., Ghanem, B.: Context-aware correlation filter tracking. In: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*. (2017) 1396–1404
20. 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. (2017) 21–26
21. Sun, X., Cheung, N.M., Yao, H., Guo, Y.: Non-rigid object tracking via deformable patches using shape-preserved kcf and level sets. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. (2017) 5495–5503
22. Kristan, M., Pflugfelder, R.P., Leonardis, A., Matas, J., Cehovin, L., et al.: The visual object tracking vot2014 challenge results (2014)
23. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., et al.: The visual object tracking vot2016 challenge results. *Springer* (Oct 2016)
24. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: *IEEE CVPR*. (2016)
25. Hanxi, L., Yi, L., Fatih, P.: Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In: *BMVC*. (2014)
26. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 fps with deep regression networks. In: *ECCV*, Springer (2016) 749–765
27. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: *NIPS*. (2012) 1097–1105
28. Vedaldi, A., Lenc, K.: Matconvnet: Convolutional neural networks for matlab. In: *Proceedings of the 23rd ACM international conference on Multimedia*, ACM (2015) 689–692
29. Yi, W., Jongwoo, L., Yang, M.H.: Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **37**(9) (2015) 1834–1848
30. 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 ICCV2015*. (2015)
31. Zhang, J., Ma, S., Sclaroff, S.: Meem: Robust tracking via multiple experts using entropy minimization. In: *ECCV*. (2014) 188–203
32. Li, Y., Zhu, J.: A scale adaptive kernel correlation filter tracker with feature integration. In: *ECCV Workshops*. (2014) 254–265
33. 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*. (2016) 4266–4274
34. Maresca, M.E., Petrosino, A.: Clustering local motion estimates for robust and efficient object tracking. In: *ECCV Workshops*. (2014) 244–253
35. Vojír, T., Matas, J.: The enhanced flock of trackers. In: *Registration and Recognition in Images and Videos*, Springer (2014) 113–136

36. Vojir, T., Neskova, J., Matas, J.: Robust scale-adaptive mean-shift for tracking. In: Pattern Recognition Letters. (2014) 250–258
37. Lee, J.Y., Yu, W.: Visual tracking by partition-based histogram backprojection and maximum support criteria. In: IEEE ROBIO. (2011) 2860–2865