Visual Object Tracking by Using Ranking Loss

Hakan Cevikalp[1], Hasan Saribas[2], Burak Benligiray[2], Sinem Kahvecioglu[2]
[1]Eskisehir Osmangazi University, [2]Eskisehir Technical University
Electrical and Electronics Engineering Department, Eskisehir, Turkey
hakan.cevikalp@gmail.com, {hasansaribas,burakbenligiray,skahvecioglu}@eskisehir.edu.tr

Abstract

This paper introduces a novel deep neural network tracker for robust object tracking. To this end, we employ a ranking loss which provides a fine-tuning of the target object position and returns more precise bounding boxes framing the target object. This is achieved by systematically learning to give higher scores to the candidate regions better framing the target object than the regions that frame the object with less accuracy. As a result, the risk of tracking error accumulation and drifts are largely mitigated, and the object is tracked more successfully. When the proposed network is used with a simple yet effective model update rule, our proposed tracker achieves the state-of-the-art results on all tested challenging tracking datasets. Especially, our results on the OTB (Object Tracking Benchmark) datasets are very promising. The proposed tracker outperforms both deep neural network and correlation filter based trackers, MDNet and ECO, by about 2%, which is a significant improvement over the previous state-of-the-art.

1. Introduction

The visual object tracking aims at tracking an unknown object in the subsequent frames specified mostly by a bounding box in the first frame. It is one of the most widely studied problems in computer vision and it has numerous applications in many domains including surveillance, video and activity analysis, human-machine interactions, and robotics. Despite great research efforts, object tracking is still a challenging problem since target objects often undergo significant appearance changes caused by deformation, pose variation, abrupt motion, cluttered background and occlusion. Severe illumination changes and similar objects close to the tracked object make the problem even harder. The main difficulty arises from the fact that we have access only to the bounding box of the target object in the first frame and typically no other appearance model or prior information are available. The tracker has to learn all variations of the tracked object in the subsequent frames based on a very limited amount of information, therefore most trackers easily drift away from the target.

Related Work: Generally, earlier tracking algorithms belong to either one of two categories: generative and discriminative trackers. Generative tracking methods are built based on the assumption that target appearances can be represented by using generative models, and hence these trackers search for the target regions that fit those generative models best. Some representative methods use sparse representation [31,51,42], density estimation [15,21], subspace learning [34,42], dictionary learning [50], and regression networks [18]. In contrast, discriminative trackers treat the problem as binary classification and learn a classifier to separate the foreground from the background. To this end, various classifier learning methods have been used including multiple instance learning [3], linear SVM (Support Vector Machine) [1], structured output SVM [16], online boosting [12], ensemble of classifiers [2,22], etc. For a more complete list of earlier tracking methods and their comparisons, the keen reader is referred to papers [49,48].

The most recent state-of-the-art tracking methods can be roughly divided into two categories: CNN (Convolutional Neural Network) based trackers and correlation filter based trackers. CNN based trackers [33,30,44,27,43,45,20] usually use small deep neural networks since it has been shown in many studies that the last layers of large CNN networks are more effective in capturing semantics, and they are not suitable for tracking because of their insufficiency in capturing fine-grained spatial details that are important for tracking. For example, same kind of object instances such as basketball players or pedestrians can be considered as both the target and background in a given frame, thus the features learned for separating the human category will not work well in this case. On the other hand, earlier layers are more sensitive to appearance changes of objects, and this makes those layers to be more precise in localization of targets. Small networks are also necessary for speed issues since updating parameters of large networks is slow and it cannot be accomplished in real-time for online visual tracking. One of the earliest deep neural network learning tracker [45] used a stacked de-noising auto-encoder to learn generic image features from a large dataset as auxiliary data and then transferred the learned features to the online tracking task. Hong et al. [20] used CNNs and SVM classifier to learn target-specific saliency maps. Li et al. [27] introduced an online learning method based on a CNN network using multiple image cues. Wang et al. [44] pre-trained a large CNN network offline and transferred the learned features to the online tracking as in [45]. Ma et al. [43] used CNN layers of a large deep neural network to learn multiple adaptive correlation filters for encoding target appearances. Nam and Han [33] introduced the Multi-Domain Network (MDNet) for object tracking. Beside some online training tricks such as long-term and short-term update strategies, the main novelty of the paper was to show how to transfer rich and effective features for tracking. To this end, Nam and Han [33] treated each video sequence as a separate domain and trained a network that includes shared layers and multiple branches of domain-specific layers offline. Once the features are learned, all existing branches used in the training phase are removed, and a new network using a single branch is fine-tuned online to adapt to the new target during tracking.

Correlation filter (CF) based trackers on the other hand learn a correlation filter to localize the target object in consecutive frames by solving a ridge regression problem efficiently in the Fourier frequency domain. The learned filter is applied to a region of interest in the next frame, and the maximum correlation filter response determines the object location in the new frame. Then, the filter is updated by using this new object location. Since the problem is solved in the frequency domain, CF based trackers are extremely fast compared to the deep neural network based trackers. Bolme et al. [6] introduced a very fast correlation filter based method using the minimum output of squared error (MOSSE) for visual tracking. Kernelized correlation filters using circulant matrices and multi-channel features have been proposed in [19]. Danelljan et al. [9] introduced a sophisticated formulation using continuous convolutional operators, which paved the way for efficiently integrating multi-resolution deep feature maps into the convolution filter based trackers. In [7], this method has been improved more by introduction of factorized convolutional operators. Initially, CF trackers used gray levels or hand-crafted features such as histogram of oriented gradients (HOGs) [19], color names [10] or color histograms [4]. On the other hand, CF trackers proposed in [30, 8] utilized CNN features extracted from pre-trained CNN networks. The most recent studies introduced methods to learn both deep CNN features and correlation filters simultaneously [39, 47, 13]. The best performing tracker [13] of the VOT2017 [24] challenge uses this methodology. There are also different variations of CF trackers that have been proposed to improve tracking performance [32, 28, 26].

There are also studies using distance (or similarity) metric learning methods for visual object tracking [38, 5, 41, 14]. Fully-convolutional Siamese network models are utilized for this purpose. Initially, these methods [38, 5] are trained on larger datasets such as ILSVRC15 and the learned metric (i.e., matching function) is simply evaluated online during tracking, and these methods did not have the ability to update the previously learned distance function during tracking. Later, Guo et al. [14] and Wang et al. [41] proposed methods that update the learned distance function online during tracking.

Motivation and Contributions: Existing deep learning based trackers typically draw positive and negative training samples around the estimated target location based on Intersection over Union (IoU) ratio, and then update the model online by using these samples. As a typical rule, samples whose IoU ratios are greater than 70% are regarded as positive samples and samples are taken as negative samples if the ratios are smaller than 50% or 30%. In general, this is not a good strategy in the sense that it does not encourage to obtain a sharper and more precise target localization. As a result, even slight inaccuracies affect the classifier and the learned model, and gradually cause the trackers to drift and eventually fail to track the target object. The success of correlation filters on the other hand is mostly because of the precise localization of the target object center (but not the whole bounding box framing the target). This is due to the efficient and approximate dense sampling performed by circularly shifting training samples. In case of homogeneous backgrounds and when the target object does not move much, the circular shifts correspond to actual translation of object and this framework works well. However, in addition to the boundary effects, the major problem with the correlation filter based trackers is that it is not straightforward to estimate aspect ratios of the target object bounding boxes (To the best of our knowledge, there is only a single study [25] that addresses this issue for CF trackers). As a result, despite the precise localization of the object center, the returned bounding box of the target object is not satisfactory especially in the case of abrupt aspect ratio changes.

To avoid these limitations, we propose a light-weight deep neural network based tracker that not only estimates the position of the target object but also provides a fine-tuning of the position and returns more precise bounding boxes framing the target object. This is achieved by using a novel classification loss adopted at the last stage of the deep neural network and employing simple but effective model update strategies during online learning. More precisely, we introduce a loss function that includes two terms where the first term aims to separate the positive sample instances from the negative ones whereas the second term encourages to return more precise location of the tar-

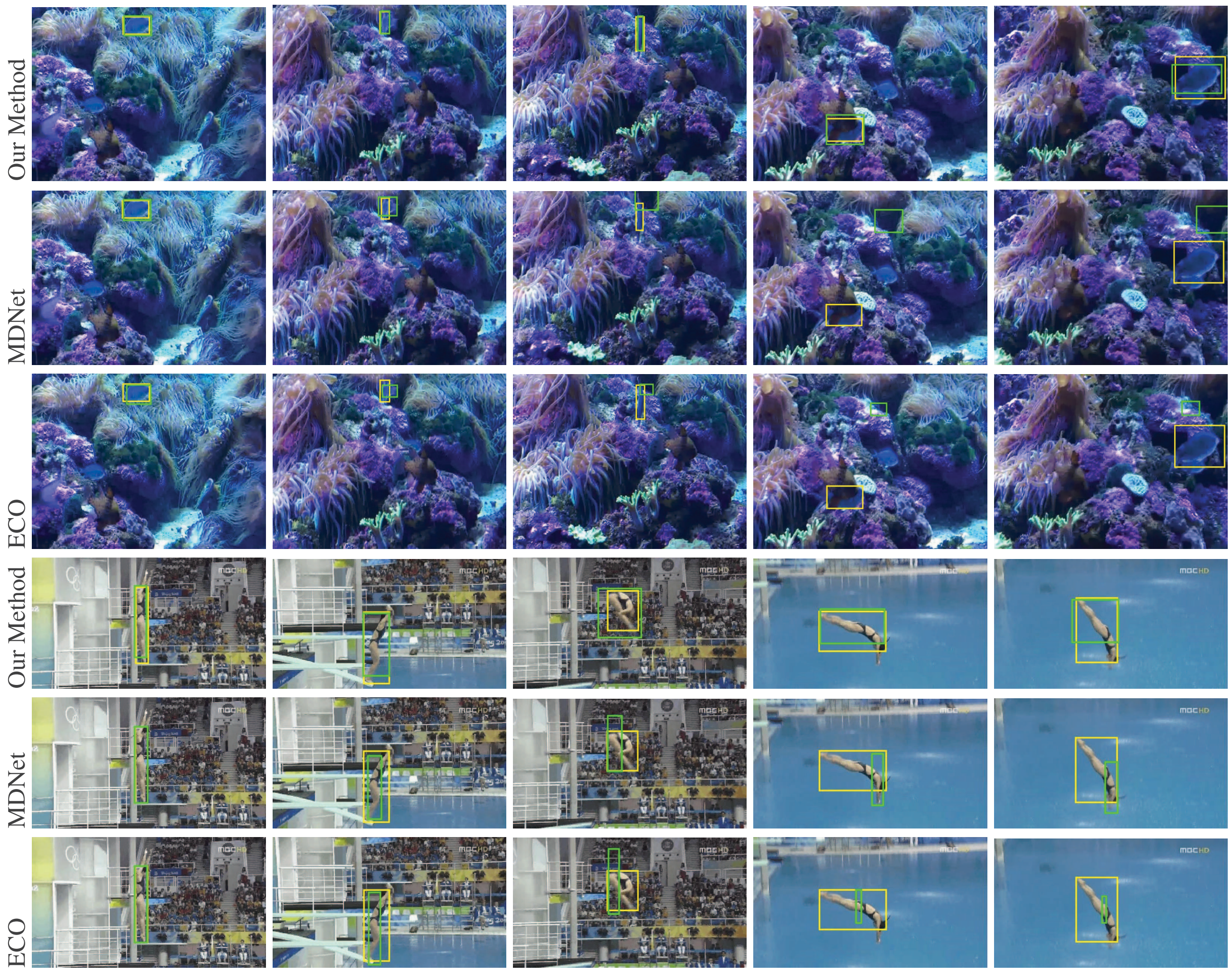

Figure 1: Visual comparison of the proposed tracker to the state-of-the-art trackers, MDNet and ECO. Yellow rectangles are the ground-truth and the green bounding boxes are the tracker outputs. The proposed tracker returns much better bounding boxes framing the target. In fact, the returned bounding boxes for the most of the fish video frames are even better than the ground-truth. In contrast, both MDNet and ECO fail to track the fish and return less accurate bounding boxes for the swimmer in the diving video.

get object. The latter objective is achieved by using a ranking loss. This loss term can also be interpreted as a way to learn a distance metric in a more principled way with Siamese networks. Regarding the model update, we follow a simple yet effective update rule: We update the model in each frame as long as scores of the best target position returned by the current model and a cache model (obtained from previous frames and updated at every 10 frames) are larger than a pre-defined threshold. This is in contrast to the update strategy of the state-of-the-art deep neural network tracker MDNet [33], where the model is updated only in some pre-defined time periods or whenever potential tracking failures are detected. However, our studies show that it is crucial to update the model in each frame to adapt to appearance and scale/aspect ratio changes and to circumvent possible drifts. Fig. 1 qualitatively compares our method to the state-of-the-art deep neural network tracker (MDNet) [33] and correlation filter tracker ECO [7] on two challenging video sequences. Object appearances and aspect ratios of bounding boxes framing the objects dramatically change in these video frames. The first and the last images in the figure correspond to the second and the last images of the video sequences, whereas the other images in the figure correspond to intermediate frames. As seen in the figure, the proposed tracker returns much better bounding boxes framing the target object compared to MDNet and ECO, whereas both MDNet and ECO fail to track the fish. MDNet and ECO can estimate scale changes well, but they fail to adapt to aspect ratio changes. In contrast, the proposed methodology adapts to both scale and aspect ratio changes successfully.

Briefly, our contributions are summarized as follows:

- We introduce a deep neural network tracker that uses a novel ranking loss function. Using ranking loss causes the tracker to return more precise bounding boxes framing the target object. Therefore, the tracker successfully adapts itself to appearance, scale and aspect ratio changes.
- A cache model is utilized to make the tracker more robust against occlusion.
- A simple yet effective model update rule which is compatible with the proposed tracker is implemented to take full advantage of the ranking loss.

2. Method

Our proposed tracker uses a light-weight deep neural network classifier for online learning as in [33]. The deep neural network classifier includes 5 hidden layers with 3 convolutional layers and 2 fully connected layers. As opposed to the classical soft-max loss used in many deep neural networks, we use a novel classification loss function that is more suitable for tracking and more robust to potential drifts from the tracked targets.

For offline learning, we used the same strategy as in [33]. More precisely, we treat each training video sequence as a separate domain, and train the network with K branches of domain-specific layers. Each domain (video sequence) is trained separately and iteratively while the shared layers are updated in each iteration. Once the network is trained with K training video sequences, the multiple branches of domain-specific layers are removed from the network and they are replaced with a single branch for the new test sequence. Then, this network with a single branch is fine-tuned online to adapt to the new target object. During online learning, we employ completely different model update strategies compared to [33]. Next, we will first explain the proposed novel loss function and then describe the overall online tracking system and the model update tricks used during online learning.

2.1. Robust Ranking Loss for Object Tracking

During online learning, the deep learning classifier is first used to estimate the best target location in a given frame and then positive and negative samples are created based on the IoU ratios with this estimated target location. Finally, these positive and negative samples are used to update the deep neural network classifier. Therefore, we also have IoU ratio belonging to each selected sample, and these ratios play a crucial role in our learning algorithm.

Let $\mathbf{x}_i$, $(i = 1, \ldots, n)$ represent the visual feature of a sampled region during tracking and $y_i \in \{-1, 1\}$ is the corresponding label. Our proposed deep neural network classifier uses the following loss function,

$$\min_{\mathbf{w}} \frac{\lambda}{2}\operatorname{trace}(\mathbf{w}^\top\mathbf{w}) + \sum_{r=1}^{n_r} L(r) H_1(\mathbf{w}^\top\mathbf{x}_{r1} - \mathbf{w}^\top\mathbf{x}_{r2}) + \kappa \sum_{i=1}^{n} H_1(y_i(\mathbf{w}^\top\mathbf{x}_i)), \quad (1)$$

where $L(.)$ is a weighting function for different ranks, $H_1(t) = \max(0, 1-t)$ is the classical hinge loss, and λ and κ are the parameters that must be set by the user. The first term in the optimization problem (1) is the regularization term and various regularization methods are already implemented in deep learning frameworks. Therefore, we will focus on the other terms in (1). The last term in the optimization problem aims to separate the positive samples from the negatives, and this term ensures that the resulting classifier returns positive scores (≥ 1) for the positive samples and negative scores (≤ -1) for the negative ones. In fact, this term along with the first term in the optimization just implement a linear SVM classifier, and it is better suited for object tracking settings where there are very limited training samples. As given in the experiments, the tracker with the classifier using the first and the last terms even performs slightly better than MDNet, which uses soft-max loss. This is very natural since soft-max classifier is a probabilistic and nonlinear function which needs many training samples. The similar findings are obtained in [11], where the authors achieve better accuracies with linear SVM compared to soft-max classifier in the context of visual object detection. For labeling, the samples whose IoU ratios are greater than 0.7 are regarded as positive samples, and samples whose IoU ratios smaller than 0.3 are treated as negative samples.

The second loss term in (1) is the major novelty in the proposed loss function, and it is specifically introduced for online object tracking. The main goal of this term is to ensure that the classifier gives higher scores to the sampled regions that frame the target object better. To this end, we use the samples whose IoU ratios are larger than or equal to 0.1. We randomly select pairs and use the ones where the difference between IoU ratios are greater than or equal to 0.4 as illustrated in Fig. 2. $\mathbf{x}_{r1}$ is the visual feature of the region that frames the target object better, and $\mathbf{x}_{r2}$ is the visual feature of the region that frames the object with less accuracy. If the score difference, $(\mathbf{w}^\top\mathbf{x}_{r1} - \mathbf{w}^\top\mathbf{x}_{r2})$ is larger than 1, there is no loss; otherwise there exists a cost determined by $L(.)$. To compute $L(.)$, we simply use the

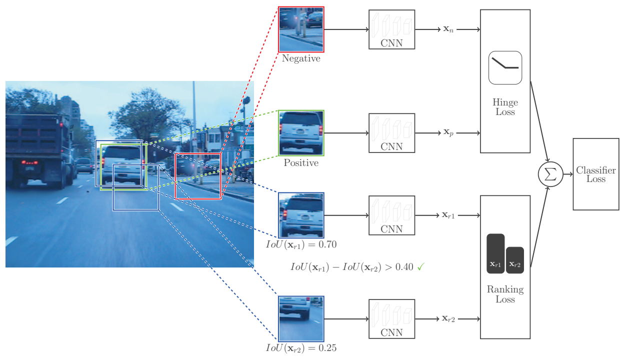

Figure 2: Selection of candidate regions for the proposed classifier loss function. For ranking loss, we use the candidate regions where difference between IoU ratios are greater than or equal to 0.4.

following formula,

$$L(r) \;=\; [IoU(\mathbf{x}_{r1}) - IoU(\mathbf{x}_{r2})]\,\gamma, \qquad (2)$$

where $IoU(\mathbf{x}_{r1})$ denotes IoU ratio of the sample $\mathbf{x}_{r1}$, and γ is a parameter that must be set by the user. Thus, $L(r)$ can take values between 0.4γ and 0.9γ. In each frame, the classifier learns to give higher scores for the sampled regions that frame the target object better since the classifier produces the highest score for the best target position. As a result, the tracking failures caused by accumulation of errors due to the small drifts are largely reduced. There are other advantages of using this loss term: Although we do not directly target to use the context information, this term also takes advantage of context information since the samples with IoU ratios between 0.1 and 0.3 include regions mainly coming from the background. The proposed loss term can also be interpreted as a more systematic way of using distance learning controlled by the $L(.)$ function. Therefore, the proposed tracker can also be regarded as a more advanced and specialized Siamese network tracker.

2.2. Implementation Details

During online tracking, we draw 256 samples in each frame with varying scales and aspect ratios near the target object position found in the previous frame. To this end, we create a ROI (Region of Interest) whose size and aspect ratio depend on the current bounding box of the target. The bounding box of the target appears in the middle of the ROI, and the size of ROI is approximately 7 times of the target bounding box size. Both random sampling and an efficient region proposal method (Edge Boxes) [53] are used to create candidate regions. The candidate region which has the maximum classifier score is temporarily assigned as the new target position, then we also make another local search around this position. For local search, we densely draw another 256 candidates around the temporary target position and obtain classifier scores. If a region produces a higher score than the score of the temporary target position, the final target object position in the current frame is assigned to this position, otherwise the temporary target position becomes the final target position. This local search is crucial for the proposed tracker since our network classifier is very sensitive to even small target position changes because of the ranking loss used in its optimization.

Once we determine the new position of the target object in a current frame, we need to update the deep neural network model. We only update the weights of the classifier layer and the fully connected layers online due to speed issues. To update the network, we draw 64 positive samples, 256 negative samples and 48 sample pairs where the difference between IoU ratios are larger than 0.4 except for the first frame, in which we used 750 positives, 7000 negatives and 200 sample pairs to fine-tune the pre-trained network. To decide whether to update the model, we also keep a network model in cache which is quite similar to the current active model, and this model is updated in every 10 frames. We update the current network model as long as the scores of the best target position for both models are larger than a pre-defined threshold which is set to 0. This model update strategy contrasts with the one proposed in [33], where short-term updates are conducted only whenever potential tracking failures are detected. However, updating the model in each frame (unless scores are lower than 0, which indicates that the target has disappeared probably because of an occlusion) is very important to keep track of scale and aspect ratio changes of the bounding boxes framing the target. The maximum number of iterations is set to 7 during model updates.

We needed to maintain another model in cache because the active model rapidly adopts itself to smooth appearance changes since it is updated in every frame. As a result, when the object undergoes an occlusion in a gradual way, the active model sometimes learns the background instead of the object and cannot recover even if the object appears again in the scene. To avoid such incidents, we keep a model that ensures that the active model is only updated when the object is mostly visible. We currently use only one cache model, but one may use multiple models for different purposes based on different applications. For instance, consider aerial tracking of a car from a UAV (Unmanned Aerial Vehicle). Since both the target and the camera move, there will be various aerial car views in different orientations. When the car disappears from the scene because of an occlusion, and then it reappears in the scene with a completely different view, the active model may struggle to locate the target. For such cases, cache models trained with different views will be helpful to relocate the target easily.

Table 1: Comparisons with the state-of-the-art trackers on the OTB datasets. Red, green, and blue fonts indicate 1st, 2nd, 3rd performance, respectively.

Tracker	**OTB-2013 AUC**	**OTB-2013 Prec.**	**OTB-50 AUC**	**OTB-50 Prec.**	**OTB-100 AUC**	**OTB-100 Prec.**
RankingT$_{\text{VOT}}$	72.8	94.8	67.3	91.0	70.1	91.8
RankingT$_{\text{VOT-NoRanking}}$	70.7	93.4	64.9	89.7	68.1	90.6
RankingT$_{\text{VOT-NoCache}}$	71.9	94.1	67.1	90.8	69.5	91.3
RankingT$_{\text{ILSVRC15}}$	71.3	93.9	65.9	89.5	68.3	91.2
RankingT$_{\text{ILSVRC15-NoRanking}}$	69.5	91.6	64.1	87.8	66.4	89.4
Vital	71.0	95.0	65.7	90.5	68.2	91.7
LSART	-	-	-	-	67.2	92.3
DRT	72.0	95.3	-	-	69.9	92.3
MCCT	71.4	92.8	-	-	69.5	91.4
CFCF	69.2	92.2	62.6	86.3	67.8	89.9
MDNet	70.8	94.8	64.5	89.0	67.8	90.9
ECO	70.9	93.0	64.3	87.3	69.1	91.0
CCOT	67.2	89.9	61.4	84.3	67.1	89.8

Setting Design Parameters: There are three parameters of the proposed method that must be set by the user: the regularization parameter λ, the weight γ in $L(.)$, and the weight κ of the ranking loss that enforces the classifier to return positive scores for positive class samples and negative scores for negative class samples, as seen in Eq. (1). For deep neural networks, the regularization parameter is commonly called as the weight decay. We set the weight decay parameter to 0.0005 as in [33]. To determine the best values of κ and γ, we randomly selected 40 videos from the ImageNet dataset and evaluated accuracies for different values. The best accuracy is obtained for $\kappa = 0.5$ and $\gamma = 4.5$ values.

3. Experiments

We tested the proposed tracker, RankingTracker (RankingT), on the challenging OTB (Object Tracking Benchmark) and VOT2017 (Visual Object Tracking 2017) datasets. We experimented with 2 different trackers which are initialized from different pre-trained models. For pretraining our first network model, RankingT$_{\text{VOT}}$, we used the same 58 video sequences collected from different VOT datasets as in [33]. Although, the videos from the same domains were used for both training and testing in different trackers, this is not a fair strategy as pointed out by [4] and the VOT committee because of concerns related to overfitting to the scenes and objects in the benchmarks. To address this issue, we also used 2156 ILSVRC 2015 video sequences to pre-train our network, RankingT$_{\text{ILSVRC15}}$, as in [4].

The proposed tracker is implemented using MatConvNet toolbox [40] on a GPU server including eight cores of 2.1 GHz Intel Xeon E5-2600 and 16 GB Quadro P5000 GPU. The average speed of the proposed tracker with a single GPU is approximately 3 FPS. We compared our trackers to other state-of-the-art and recently published trackers, including Vital (CVPR 2018) [35], LSART (CVPR 2018) [37], DRT (CVPR 2018) [36], MCCT (CVPR 2018) [46], CFCF (T-IP 2018) [13], MDNet (CVPR 2016) [33], ECO (CVPR 2017) [7], CCOT (ECCV 2016) [9], CFWCR (ICCVW 2017) [17], Gnet (ICIP 2017) [23], CSRDCF (CVPR 2017) [29] and MCPF (CVPR 2017) [52].

3.1. Results on OTB

OTB is a very common tracking benchmark that includes 100 videos sequences with 11 different attributes such as fast motion, background clutter, deformation, in-plane rotation, etc. We tested our tracker on the OTB-2013, OTB-50 and OTB-100 benchmarks which contain 50, 50 and 100 fully annotated videos, respectively. The standard OTB evaluation protocol is used to compare the tracking methods. According to this, we report the results in one-pass evaluation (OPE) protocol with both precision and success plots, and the OTB evaluation toolkit is used to generate the plots and performance scores. The precision plot calculates the Euclidean distance of center locations between ground-truth and estimated target position then reports the percentage of frame locations that are within 20 pixels distance from those of the ground truth. The success measures IoU ratios of the ground-truth and predicted target bounding boxes, and the success plot shows the rate of bounding boxes whose IoU ratios are larger than a given threshold. Tracking algorithms are ranked based on the area under curve (AUC) scores obtained from these plots.

The results are given in Table 1. The red, green, and blue fonts in the table respectively indicate the best, the second best and the third best accuracy scores. The corresponding precision and success plots are given in Fig. 3. As seen in the results, the proposed RankingT$_{\text{VOT}}$ achieves the best success rates on all tested datasets, and achieves the best precision score on the OTB-50 dataset, the second best precision score on the OTB-100 dataset, and the third best precision score on the OTB-2013 dataset. Another proposed tracker fine-tuned from the model using ILSVRC15, RankingT$_{\text{ILSVRC15}}$, obtains the third best success rate on the OTB-50 dataset. As seen in the results, the proposed trackers achieve the best success rates but precision scores are not always the best. This is due to the fact that the correlation filter based trackers are very good at estimating the center of the target object position, but the predicted bounding boxes are not very good because of scale/aspect ratio changes. Proposed trackers on the other hand estimate the bounding boxes with a very high precision, but their centers are not necessarily very accurate compared to correlation filter based trackers. In general, the proposed tracker, RankingT$_{\text{VOT}}$, which is fine-tuned from the model pre-trained by using VOT dataset videos, achieves the best accuracy followed by the one that is fine-tuned from the model pre-trained by using ILSVRC15 videos. Both of our proposed trackers using different pre-trained models outperform MDNet which also uses a deep neural network for tracking.

To demonstrate the importance of the ranking loss term introduced in optimization problem (1), we also conducted some tests by removing this term from the optimization. The resulting trackers, RankingT$_{\text{VOT(ILSVRC15)-NoRanking}}$, just use SVM hinge loss and these trackers are also pre-trained by using videos from VOT(ILSVRC15) datasets. Therefore, they are directly comparable to the RankingT$_{\text{VOT(ILSVRC15)}}$ trackers. As seen in the results, including the ranking loss term in the classifier improves success rates around 2% for both trackers. In addition, we also conducted tests to measure the effect of the cache model on the accuracy. Removing the cache model from the tracker slightly decreased the performances between 0.2% and 0.9% in terms of AUC.

Lastly, we report the results on various attributes in the OTB-100 dataset. The proposed tracker, RankingT$_{\text{VOT}}$, achieves the best success rates on 9 of 11 attributes in the dataset and it obtains the second best accuracies on the remaining 2 attributes. Fig. 4 visualizes the 4 attributes in which the proposed tracker achieves the best accuracies. This plot clearly shows that the proposed tracker successfully handles all kinds of challenging tracking situations.

3.2. Results on VOT2017

The VOT2017 [24] benchmark consists of 60 challenging video sequences selected from a pool of about 390 sequences, with 6 different attributes, such as occlusion, illumination change, camera motion and so on. The VOT challenge protocol uses reset-based methodology in which a tracker is re-initialized whenever the tracking fails. The accuracy is measured in terms of expected average overlap (EAO), which quantitatively reflects both bounding box overlap ratio (accuracy-A) and re-initialization times (robustness-R). In addition, the VOT2017 evaluation protocol also applies no-reset based OTB protocol where the performance is measured in terms of the average overlap

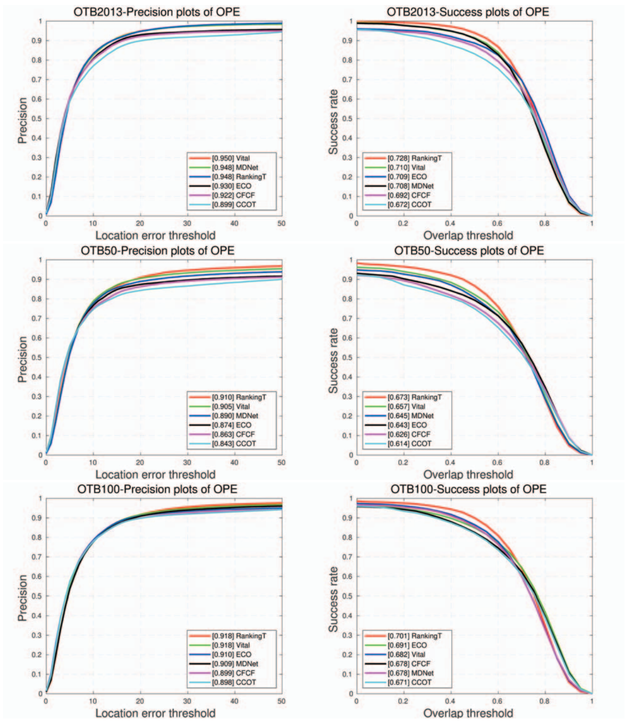

Figure 3: Precision and success plots on the OTB datasets (RankingT corresponds to RankingT$_{\text{VOT}}$ in the figure).

Table 2: Results of the tested trackers on the VOT2017.

Tracker	**EAO**	**A**	**R**	**AO**
RankingT$_{\text{VOT}}$	0.33	0.55	13.32	0.48
RankingT$_{\text{VOT-NoRanking}}$	0.31	0.54	15.04	0.46
RankingT$_{\text{VOT-NoCache}}$	0.32	0.54	14.21	0.46
RankingT$_{\text{ILSVRC15}}$	0.27	0.50	19.84	0.46
LSART	0.32	0.48	11.20	0.44
CFWCR	0.30	0.47	14.33	0.37
CFCF	0.29	0.49	15.17	0.38
ECO	0.28	0.46	15.33	0.40
Gnet	0.27	0.49	13.33	0.42
MCCT	0.27	0.52	14.0	0.43
CCOT	0.27	0.46	17.67	0.39
CSRDCF	0.26	0.49	18.50	0.34
MCPF	0.25	0.51	21.29	0.44

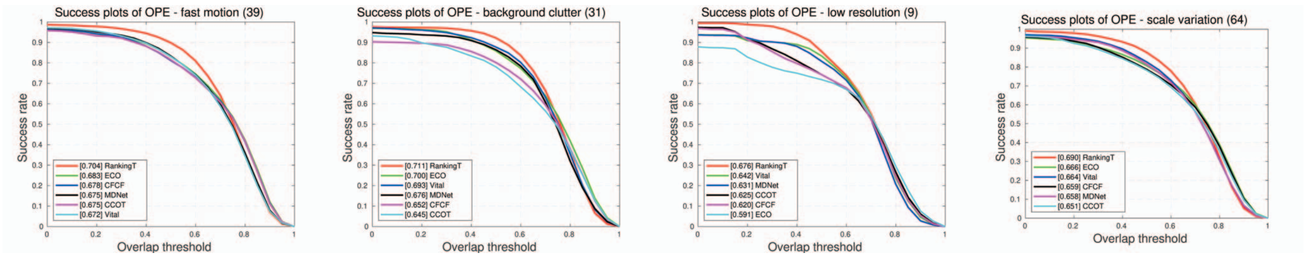

Figure 4: Comparison of performances on different attributes of the OTB-100 dataset.

Table 3: Comparison of performances for the challenging attributes on the VOT2017 dataset.

Tracker	**Camera Motion**		**Illum. Changes**		**Motion Changes**		**Occlusion**		**Size Changes**	
	A	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**
RankingT$_{\text{VOT}}$	0.55	19.93	0.56	0.80	0.56	8.33	0.52	18.33	0.56	4.93
RankingT$_{\text{ILSVRC15}}$	0.53	30.13	0.46	6.07	0.53	21.20	0.47	25.67	0.51	10.47
LSART	0.51	19.73	0.48	0.53	0.50	8.67	0.44	20.80	0.46	9.00
CFWCR	0.54	28.00	0.48	3.00	0.49	12.00	0.39	19.00	0.45	12.00
CFCF	0.54	29.00	0.44	0.00	0.51	15.00	0.44	20.00	0.49	8.00
ECO	0.51	25.00	0.50	4.00	0.48	18.00	0.35	22.00	0.45	9.00
Gnet	0.55	23.00	0.50	1.00	0.50	14.00	0.41	14.00	0.49	9.00
CCOT	0.52	26.00	0.45	6.00	0.48	19.00	0.39	22.00	0.45	14.00

(AO).

The results are given in Table 2, and all results are again obtained by the provided VOT evaluation toolkit. We tested our RankingT$_{\text{VOT}}$ and RankingT$_{\text{ILSVRC15}}$ trackers on this dataset. The proposed tracker, RankingT$_{\text{VOT}}$, achieves the best results in terms of accuracy (A), expected average overlap (EAO) and average overlap (AO), and it significantly outperforms other state-of-the-art trackers including the VOT2017 challenge winner, CFCF. However, its robustness performance is not the best, and it comes the second after the score of LSART. The other proposed tracker, RankingT$_{\text{ILSVRC15}}$, obtains the second best AO score after RankingT$_{\text{VOT}}$. As in OTB results, removing the ranking loss term from the tracker decreases the results by 2% in terms of EAO whereas the results decrease by 1% if the cache model is not utilized in the proposed tracker. Table 3 gives the results for different attributes on the VOT2017 dataset. The proposed tracker, RankingT$_{\text{VOT}}$, again obtains the best accuracies for all tested attributes and obtains the best robustness scores for *Size Changes* and *Motion Changes*. Moreover, it obtains the second best results for *Camera Motion* and *Occlusion*. Obtaining the best scores of both accuracy and robustness for the *Size Changes* attribute clearly demonstrates that the proposed tracker better adapts itself to scale/aspect ratio changes compared to other state-of-the-art trackers. The other proposed tracker, RankingT$_{\text{ILSVRC15}}$, achieves the second best accuracies for the three of the five tested attributes.

4. Summary and Conclusions

This paper introduces a deep neural network classifier for visual tracking. In contrast to other deep neural network based trackers, we use a new classification loss specifically designed for tracking settings. To this end, we use hinge loss which is better suited for separation of data samples when there is a limited amount of training data for learning. Moreover, we also introduce a novel ranking loss which ensures that the classifier gives much higher scores to the candidate regions that frame the target object better. This reduces the risk of tracking error error accumulation and avoids possible drifts from the tracked object. The proposed classifier achieves the state-of-the-art results on the challenging OTB and VOT datasets when used with simple online classifier update strategies. Experimental results show that the proposed tracker successfully estimates the best target positions as long as the object appearance changes are not very harsh. Especially, the estimation of the aspect ratios of target bounding boxes is very successful whereas the majority of state-of-the-art detectors fail to do so only with a very few exceptions.

Acknowledgments: This work was supported by the Scientific and Technological Research Council of Turkey (TUBİTAK) under grant number EEEAG-116E080. The authors also would like to thank NVIDIA for GPU donation used in this study.

References

[1] Shai Avidan. Support vector tracking. *IEEE Transactions on PAMI*, 26:1064–1072, 2004. 1

[2] Shai Avidan. Ensemble tracking. In *CVPR*, 2005. 1

[3] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on PAMI*, 33:1619–1632, 2011. 1

[4] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip HS Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, 2016. 2, 6

[5] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *arXiv:1606.09549*, 2016. 2

[6] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010. 2

[7] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *CVPR*, 2017. 2, 3, 6

[8] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Convolutional features for correlation filter based visual tracking. In *ICCV Workshops*, 2015. 2

[9] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016. 2, 6

[10] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost Van de Weijer. Adaptive color attributes for real-time visual tracking. In *CVPR*, 2014. 2

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 4

[12] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *BMVC*, 2006. 1

[13] Erhan Gundogdu and A Aydın Alatan. Good features to correlate for visual tracking. *IEEE Transactions on Image Processing*, 27:2526–2540, 2018. 2, 6

[14] Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. Learning dynamic siamese network for visual object tracking. In *ICCV*, 2017. 2

[15] Bohyung Han, Dorin Comaniciu, Ying Zhu, and Larry S Davis. Sequential kernel density approximation and its application to real-time visual object tracking. *IEEE Transactions on PAMI*, 30:1186–1197, 2008. 1

[16] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2096–2109, 2015. 1

[17] Zhiqun He, Yingruo Fan, Junfei Zhuang, Yuan Dong, and HongLiang Bai. Correlation filters with weighted convolution responses. In *ICCV Workshops*, 2017. 6

[18] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with regression networks. In *ECCV*, 2016. 1

[19] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on PAMI*, 37:583–596, 2015. 2

[20] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. Online tracking by learning discriminative saliency map with convolutional network. In *ICML*, 2015. 1, 2

[21] Allan D Jepson, David J Fleet, and Thomas F El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transactions on PAMI*, 25:1296–1311, 2003. 1

[22] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on PAMI*, 34:1409–1422, 2012. 1

[23] Thanikasalam Kokul, Clinton Fookes, Sridha Sridharan, Amirthalingam Ramanan, and UAJ Pinidiyaarachchi. Gate connected convolutional neural network for object tracking. In *Image Processing (ICIP), 2017 IEEE International Conference on*. IEEE, 2017. 6

[24] Matej Kristan, Aleš Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Čehovin Zajc, Tomas Vojir, Gustav Häger, Alan Lukežič, Abdelrahman Eldesokey, and Gustavo Fernandez. The visual object tracking vot2017 challenge results, 2017. 2, 7

[25] Feng Li, Yingjie Yao, Peihua Li, David Zhang, Wangmeng Zuo, and Ming-Hsuan Yang. Integrating boundary and center correlation filters for visual tracking with aspect ratio variation. In *ACM International Conference on Multimedia*, 2015. 2

[26] Feng Li, Yingjie Yao, Peihua Li, David Zhang, Wangmeng Zuo, and Ming-Hsuan Yang. Integrating boundary and center correlation filters for visual tracking with aspect ratio variation. In *ICCV Workshops*, 2017. 2

[27] Hanxi Li, Yi Li, and Fatih Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *IEEE Transactions on Image Processing*, 25(4):1834–1848, 2015. 1, 2

[28] Si Liu, Tianzhu Zhang, Xiaochun Cao, and Changsheng Xu. Structural correlation filter for robust visual tracking. In *CVPR*, 2016. 2

[29] Alan Lukezic, Tomas Vojir, Luka Cehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *CVPR*, 2017. 6

[30] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015. 1, 2

[31] Xue Mei and Haibin Ling. Robust visual tracking using l_1 minimization. In *ICCV*, 2009. 1

[32] Matthias Mueller, Neil Smith, and Bernard Ghanem. Context-aware correlation filter tracking. In *CVPR*, 2017. 2

[33] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 1, 2, 3, 4, 5, 6

[34] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77:125–141, 2008. 1

[35] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson WH Lau, and Ming-Hsuan Yang. Vital: Visual tracking via adversarial learning. In *CVPR*, 2018. 6

[36] Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Correlation tracking via joint discrimination and reliability learning. In *CVPR*, 2018. 6

[37] Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Learning spatial-aware regressions for visual tracking. In *CVPR*, 2018. 6

[38] Ran Tao, Efstratios Gavves, and Arnold WM Smeulders. Siamese instance search for tracking. In *CVPR*, 2016. 2

[39] Jack Valmadre, Luca Bertinetto, João Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017. 2

[40] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *ICCV Workshops*, 2017. 6

[41] Bing Wang, Li Wang, Bing Shuai, Zhen Zuo, Ting Liu, Kap Luk Chan, and Gang Wang. Joint learning of convolutional networks temporally constrained metrics for tracklet association. In *CVPR Workshops*, 2016. 2

[42] Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Online object tracking with sparse prototypes. *IEEE Transactions on Image Processing*, 22:314–325, 2013. 1

[43] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015. 1, 2

[44] Naiyan Wang, Siyi Li, Abhinav Gupta, and Dit-Yan Yeung. Transferring rich feature hierarchies for robust visual tracking. In *arXiv:1501.04587*, 2015. 1, 2

[45] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013. 1, 2

[46] Ning Wang, Wengang Zhou, Qi Tian, Richang Hong, Meng Wang, and Houqiang Li. Multi-cue correlation filters for robust visual tracking. In *CVPR*, 2018. 6

[47] Qiang Wang, Jin Gao, Junliang Xing, Mengdan Zhang, and Weiming Hu. Dcfnet: Discriminant correlation filters network for visual tracking. In *arXiv:1704.04057*, 2017. 2

[48] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. In *CVPR*, 2013. 1

[49] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: a survey. *ACM Computing Surveys*, 38:1–45, 2006. 1

[50] Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja. Low-rank sparse learning for robust visual tracking. In *ECCV*, 2012. 1

[51] Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja. Robust visual tracking via multi-task sparse learning. In *CVPR*, 2012. 1

[52] Tianzhu Zhang, Changsheng Xu, and Ming-Hsuan Yang. Multi-task correlation particle filter for robust object tracking. In *CVPR*, 2017. 6

[53] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 5