Combining Deep Learning and Preference Learning for Object Tracking

Shuchao Pang[1,2], Juan José del Coz[2], Zhezhou Yu[1], Oscar Luaces[2], and Jorge Díez[2]

[1] Coll. Computer Science and Technology, Jilin University, Changchun, 130012, China
pangshuchao1212@sina.com, yuzz@jlu.edu.cn
[2] Artificial Intelligence Center, University of Oviedo at Gijón, 33204, Spain
juanjo@uniovi.es, oluaces@uniovi.es, jdiez@uniovi.es

Abstract. Object tracking is nowadays a hot topic in computer vision. Generally speaking, its aim is to find a target object in every frame of a video sequence. In order to build a tracking system, this paper proposes to combine two different learning frameworks: deep learning and preference learning. On the one hand, deep learning is used to automatically extract latent features for describing the multi-dimensional raw images. Previous research has shown that deep learning has been successfully applied in different computer vision applications. On the other hand, object tracking can be seen as a ranking problem, in the sense that the regions of an image can be ranked according to their level of overlapping with the target object. Preference learning is used to build the ranking function. The experimental results of our method, called DPL^2 (Deep & Preference Learning), are competitive with respect to the state-of-the-art algorithms.

Keywords: deep learning, preference learning, object tracking

1 Introduction

The goal of object tracking systems is to follow the trajectory of a moving object in a video. This task has attracted substantial research attention because it tackles several interesting applications, including surveillance, vehicle navigation, augmented reality, human-computer interactions and medical imaging, among others [10]. In the past decade, the research of object tracking has made significant progress but it is still a challenging task because a robust tracker must deal with difficult situations in real world environments such as illumination variation or full and partial occlusion of the target object, just to cite a couple of them. In fact, no actual tracker is able to be successful in all possible scenarios and most of them simplify the problem by imposing constraints usually based on prior knowledge about the actual application.

Trackers usually have three main components: object representation including feature selection methods, an object detection mechanism and an update strategy. Objects can be represented using different characteristics, for instance, their shape, appearance, color, texture, etcetera. These aspects are described

by a set of features that can be manually chosen by the user depending on the application domain, automatically using a feature selection algorithm or by the combination of both. The object detection mechanism is responsible for detecting the area in the image occupied by the target object in every frame. Its prediction can be only based on the information of the current frame, but there are also several approaches that take advantage of using the temporal information from previous frames. Due to the unexpected changes in the appearance of the target object, an update strategy is usually required to obtain a robust tracker.

This paper presents a method for object tracking, called DPL^2, that is based on applying two learning frameworks that have been successfully used in several computer vision systems. To the best of our knowledge, they have not been used together in the context of object tracking. Firstly, DPL^2 applies a deep learning architecture to represent the images. Secondly, the model to detect and track the target object is obtained using a supervised ranking algorithm. This selection is motivated by the fact that object tracking is indeed a ranking problem: all the possible bounding boxes of an image could be ranked by their level of overlapping with the target object. Notice that it is not necessary to obtain a perfect total ranking, it is sufficient that the areas very close to the target object rank higher than the rest. Preference learning perfectly fits this goal.

The most related work regarding the use of deep learning in the context of object tracking is [8]. The difference with respect to DPL^2 is that a classifier is used instead of a ranker, thus the representation strategy is the same, but the model follows a completely different approach. In [2] the authors present a tracking system that is based on Laplacian ranking SVM. The tracker incorporates the labeled information of the object in the initial and final frames to deal with drift changes, like occlusion, and the weakly labeled information from the current frame to adapt to substantial changes in appearance. Another difference with our approach is that they use Haar-like features to represent each image patch and we employ a deep learning network. The method described in [3] it is also based on learning to rank, where the authors propose an online learning ranking algorithm in the co-training framework. Two ranking models are built with different types of features and are fused into a semi-supervised learning process. Other references are omitted due to the lack of space.

The rest of the paper is organized as follows. Our proposal based on deep learning and preference learning is presented in Section 2. Then, we report a thoroughly experimentation devised to show the benefits of our approach. The paper ends drawing some conclusions.

2 Deep & Preference Learning Tracker

This section is devoted to describe the main components of our proposal. Figure 1 depicts the structure of DPL^2 algorithm that has four main steps. Firstly, according to the position of the target object in the frame, p positive boxes near to the target object are selected as positive examples and n negative boxes away from the object as negative examples. Then, DPL^2 uses a deep learning network

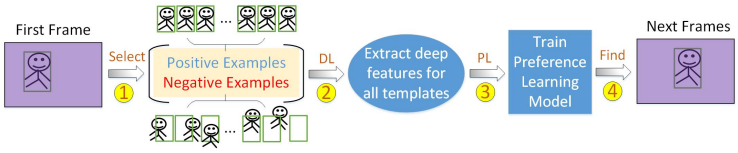

Fig. 1. The overall framework of our proposed DPL^2 algorithm

to extract deep features to describe each example selected in the previous phase. In the third step, DPL^2 applies preference learning to build a ranking model to detect the object. Such model is learned using a set of preferences judgements, formed by all possible pairs between positive and negative examples. Finally, to detect the object in the next frames, DPL^2 uses particle filter to select several particle images around the position of the target object in the last frame. Then, the model outputs the ranking of all particle images; the one that ranks higher is selected as the best position and size of target object in the frame. This last step (number 4 in Figure 1) is repeated to track the object across each frame of the video sequence. The rest of steps (1-3) are only re-executed when a model update is required as it shall be described in Section 2.3.

2.1 Deep Learning

A deep learning method is a representation-learning technique that produces computational models using raw data, obtaining the representation that will be used in the subsequent learning processes. These models, composed of l processing layers ($l > 1$), learn data representations with multiple levels of abstraction; each level transforms the representation at previous level into a more abstract representation [6]. Very complex functions can be learned when l is sufficiently large. The highest layer captures discriminative variations, suppressing the irrelevant ones [11].

In our algorithm we use the SDAE architecture [8] that is composed of two parts, an encoder and a decoder. Each part contains a network, both are usually connected (the last layer of the encoder is the first layer of the decoder). The building block of SDAE is the denoising autoencoder (DAE), a variant of conventional autoencoder. The main property of a DAE network is that it is able to encode the image in a smaller feature space and then recover the original image from the encoded version.

The encoder part is codified by a nonlinear activation function $f_\theta(x)$. The aim of f_θ is to transform the original image vector $\boldsymbol{x}_i$ into a different representation $\boldsymbol{y}_i$. Here, $\theta = \{\boldsymbol{W}, \boldsymbol{b}\}$ and $\boldsymbol{W}$ denote the weights matrix of the encoder part, whose size depends on the number on layers l and the units of each layer, and $\boldsymbol{b}$ is the bias term. The decoder follows the same formulation and it is described by a function $g_{\theta'}(y_i)$, being $\theta' = \{\boldsymbol{W}', \boldsymbol{b}'\}$, $\boldsymbol{W}'$ the weights matrix of the decoder part and $\boldsymbol{b}'$ its bias term. The goal of the decoder is to map the hidden representation

back to a reconstructed input $\boldsymbol{z}_i$. Obviously, the decoding process is the inverse process of encoding. But the key element is that both networks are trained together trying to minimize the difference between the reconstructed $\boldsymbol{z}_i$ and the original input image $\boldsymbol{x}_i$. The mathematical formulation is:

$$\min_{\boldsymbol{W},\boldsymbol{W}',\boldsymbol{b},\boldsymbol{b}'} \sum_{i=1}^{k} \|\boldsymbol{x}_i - \boldsymbol{z}_i\|_2^2 + \alpha(\|\boldsymbol{W}\|_F^2 + \|\boldsymbol{W}'\|_F^2). \tag{1}$$

Here, the number of original images is k, $\boldsymbol{y}_i = f_\theta(\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{b})$, and $\boldsymbol{z}_i = g_{\theta'}(\boldsymbol{W}'\boldsymbol{y}_i + \boldsymbol{b}')$ represents the outputs of both networks (encoder and decoder) for a given example $\boldsymbol{x}_i$. α is the regularization parameter for trading off between the error and the complexity of both networks, which is measured by means of the Frobenius norm $\|\cdot\|_F$.

Following the training approach described in [8], *DPL*^{2} trains the SDAE model offline. The network architecture used has five layers $l = 5$ with 2560 units in the first hidden layer (overcomplete filters). Then, the number of units is reduced by a half in each layer, so the final layer has just 160 units. A logistic sigmoid activation function is used for f_θ and g_θ.

2.2 Preference Learning

Although there are other approaches to learn preferences, following [1] we will try to induce a real *preference* or *ranking function* $r : \mathbb{R}^d \to \mathbb{R}$, that maps the object from the input space (images represented using SDAE with d=160) to $\mathbb{R}$. r should be learned in such a way that it maximizes the probability of having $r(\boldsymbol{v}) > r(\boldsymbol{u})$ whenever $\boldsymbol{v}$ is preferable to $\boldsymbol{u}$ ($\boldsymbol{v} \succ \boldsymbol{u}$). In our case, $\boldsymbol{v}$ is preferable whenever its level of overlapping with the target object is greater than the one of $\boldsymbol{u}$.

To learn such function r we start from the set of positive and negative examples extracted from the first frame. This set of objects is endowed with a partial order that can be expressed by a collection of preference judgments considering all pairs of positive and negative examples:

$$PJ = \{\boldsymbol{v}_i \succ \boldsymbol{u}_j : i = 1..p,\ j = 1..n\}. \tag{2}$$

In order to induce the ranking function, we look for a function $R : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that

$$\forall \boldsymbol{v}_i, \boldsymbol{u}_j \in \mathbb{R}^d, R(\boldsymbol{v}_i, \boldsymbol{u}_j) > 0 \Leftrightarrow R(\boldsymbol{v}_i, \boldsymbol{0}) > R(\boldsymbol{u}_j, \boldsymbol{0}). \tag{3}$$

Then, the ranking function r can be simply defined by

$$\forall \boldsymbol{x} \in \mathbb{R}^d, r(\boldsymbol{x}) = R(\boldsymbol{x}, 0). \tag{4}$$

Given the set of preference judgments PJ (2), we can specify R by means of the constraints

$$R(\boldsymbol{v}_i, \boldsymbol{u}_j) > 0 \text{ and } R(\boldsymbol{u}_j, \boldsymbol{v}_i) < 0, \quad \forall (\boldsymbol{v}_i, \boldsymbol{u}_j) \in PJ. \tag{5}$$

Therefore, we can apply any binary classifier. *DPL*^{2} uses Support Vector Machines (SVM) [7].

2.3 Model update

Updating the mechanism to detect the object is crucial for obtaining a robust tracking system [5]. The goal of the update method is to take into account drifting situations that happen along the video sequence, obtaining a final tracking system that is capable of following the object even when there are important variations in the appearance of the object.

DPL^2 updates the model using two different strategies. First, when the description of the predicted region for the current frame is significantly different from the average description of the predicted region of the 10 previous frames. If this situation occurs the preference judgments (2) used to update the model are obtained combining the 10 previous predicted regions (as positive examples) and $n = 30$ negative examples generated randomly. The second rule is even simpler: every t frames the model is updated following the same procedure just described.

3 Experiments

The aim of the experiments reported in this section is to compare the performance of DPL^2 with the best current tracking systems. The selection of such trackers was based on the results of the experimental study recently published in [9]. The best top 14 object tracking algorithms were selected: SCM, Struck, ASLA, L1APG, CT, TLD, IVT, MTT, CSK, MIL, LSK, VTS, VTD and CXT.

In order to fairly compare all the trackers, we used the same datasets, evaluation metrics and the original implementations and results of these trackers reported in [9] but we had to limit the number of experiments due to space restrictions. For the datasets we randomly selected 14 videos of different difficulty degree: hard (Bolt, Coke, Soccer and Woman), middle (David, Deer, Shaking and Trellis) and easy (Car4, CarDark, Fish, Jogging2, Singer1 and Walking). The evaluation method used was OPE (One-Pass Evaluation), that is, the algorithm is initialized with the ground-truth object state in the first frame and the results for the rest of the frames are computed. The scores reported here correspond to two commonly used performance metrics: Success Rate and Precision. Additionally, we also compute Area Under Curve (AUC) scores to measure the overall performance of the trackers.

The parameters used to execute DPL^2 were the following. The number of positive and negative examples were $p = 10$ and $n = 30$ respectively. The examples were generated moving the left top corner of the ground-truth region of the first frame. The positive examples are just moved ± 1 pixels, while in the case of the negatives this distance is always greater than $\pm width/4$ in the X-axis and $\pm height/4$ (Y-axis); $width$ and $heigth$ represent the size of the target object in the first frame. When the model is updated, the same process is applied but instead of using the ground-truth region of each frame, the predicted regions are employed. Finally, to detect the object DPL^2 generates randomly 100 particles around the position of the predicted region in previous frame and ranks them; the one that is ranked in first position is returned.

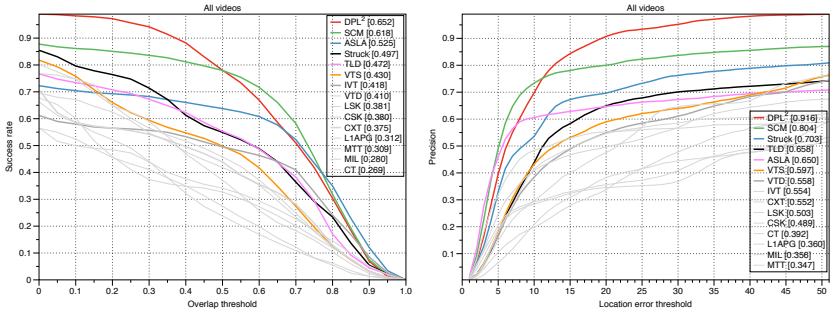

Fig. 2. Average Success Rate (left) and Precision (right) plots of all video frames. Ranks in the legend are computed using AUC of Success Rate and the percentage Precision for 20 pixels

In order to analyze the overall performance of the tracking algorithms considered, we collected together the results of all trackers over the 14 video sequences, a total of 5704 frames, obtaining average values for Success Rate and Precision.

Success Rate measures the overlap rate of each frame. Calculating the percentage of successful frames whose overlap rate is larger than 0.5, we can measure the ability of the trackers to capture most of the area occupied by the target object. However, using just a specific threshold value, it cannot depict the overall performance of each tracker. So we draw the success plot of each tracker by varying the threshold value from 0 to 1. Precision computes the average euclidean distance between the center locations of the predicted regions and the ground-truth positions in all the frames. The Precision percentage is the percentage of frames in which the error of the tracker is less than a given number of pixels. As we can see (Figure 2), DPL^2 performs quite well for both metrics, its scores are similar to those of SCM, one of the best trackers according to [9]. Although DPL^2 ranks the best for Success Rate when threshold is 0.5, it seems that its performance decays with higher thresholds. Actually, this means that DPL^2 is the best to detect at least part of the object, but it is a little bit worse for capturing a bigger area. This result suggest one of the possible directions to improve DPL^2. Looking at the Precision plots of all videos, we can conclude that when such threshold is smaller than 12 pixels, the Precision scores of SCM and ASLA are slightly better than those of DPL^2. However, when the location error threshold becomes large, DPL^2 clearly outperforms the rest of the trackers. This means that it is a quite robust tracker that rarely losses the track of the object.

In addition, for comprehensively analyzing the overall performance of each tracker from a statistical point of view, Table 1 reports the scores of the Area Under Curve (AUC) of the Success Rate and Precision percentage for all systems over all videos. To sum up the results, the average rank of each tracker is the last row of each table. Following [4], a two-step statistical test procedure was carried out. The first step consists of a Friedman test of the null hypothesis that states

Table 1. Scores and ranking position using two different metrics: AUC of Success Rate (top) and Precision (bottom). The average rank is in the last row of each table

Video	ASLA	L1APG	CT	TLD	IVT	MTT	CSK	SCM	Struck	MIL	LSK	VTS	VTD	CXT	DPL²
Bolt	.011 (11.5)	.012 (10)	.010 (14)	.159 (5)	.010 (14)	.011 (11.5)	.019 (6)	.016 (7.5)	.014 (9)	.010 (14)	.494 (2)	.173 (4)	.372 (3)	.016 (7.5)	.585 (1)
Car4	.741 (4)	.246 (13)	.213 (14)	.626 (5)	.857 (1)	.448 (8)	.468 (7)	.745 (3)	.490 (6)	.265 (12)	.157 (15)	.366 (9)	.365 (10)	.312 (11)	.791 (2)
CarDark	.832 (3)	.864 (2)	.003 (15)	.443 (13)	.653 (10)	.811 (7)	.744 (8)	.828 (4)	.872 (1)	.198 (14)	.821 (5)	.739 (9)	.534 (12)	.557 (11)	.820 (6)
Coke	.173 (13)	.176 (12)	.239 (8)	.399 (6)	.116 (15)	.442 (4)	.565 (3)	.325 (7)	.665 (2)	.212 (9)	.200 (10)	.181 (11)	.148 (14)	.423 (5)	.669 (1)
David	.735 (1)	.534 (10)	.497 (11)	.707 (3)	.637 (5)	.301 (14)	.408 (13)	.711 (2)	.249 (15)	.432 (12)	.558 (8)	.582 (7)	.556 (9)	.641 (4)	.618 (6)
Deer	.032 (15)	.596 (6)	.039 (13)	.590 (7)	.033 (14)	.604 (5)	.733 (1)	.074 (10)	.730 (2)	.129 (9)	.270 (8)	.046 (12)	.059 (11)	.690 (4)	.715 (3)
Fish	.833 (2)	.349 (12)	.705 (7)	.796 (3)	.758 (5)	.181 (15)	.222 (14)	.736 (6)	.840 (1)	.451 (11)	.313 (13)	.684 (8.5)	.553 (10)	.770 (4)	.684 (8.5)
Jogging2	.141 (8)	.147 (6)	.104 (15)	.647 (3)	.142 (7)	.125 (13)	.139 (9)	.721 (2)	.199 (5)	.134 (10)	.336 (4)	.125 (13)	.125 (13)	.126 (11)	.765 (1)
Shaking	.465 (6)	.081 (13)	.109 (12)	.394 (9)	.037 (15)	.049 (14)	.572 (5)	.680 (3)	.356 (10)	.430 (8)	.459 (7)	.698 (2)	.700 (1)	.126 (11)	.652 (4)
Singer1	.778 (3)	.284 (15)	.355 (12)	.714 (4)	.571 (5)	.344 (14)	.364 (10)	.852 (1)	.365 (9)	.362 (11)	.347 (13)	.488 (8)	.489 (7)	.491 (6)	.843 (2)
Soccer	.112 (14)	.168 (8.5)	.168 (8.5)	.128 (13)	.162 (10)	.184 (6)	.146 (11)	.239 (4)	.188 (5)	.175 (7)	.099 (15)	.332 (3)	.333 (2)	.145 (12)	.522 (1)
Trellis	.788 (1)	.212 (15)	.341 (11)	.481 (8)	.251 (12.5)	.220 (14)	.479 (9)	.665 (2.5)	.610 (5)	.251 (12.5)	.665 (2.5)	.494 (6)	.451 (10)	.649 (4)	.488 (7)
Walking	.758 (1)	.741 (3)	.519 (12)	.447 (14)	.752 (2)	.658 (6)	.534 (11)	.701 (4)	.569 (9)	.543 (10)	.453 (13)	.614 (7)	.603 (8)	.168 (15)	.670 (5)
Woman	.146 (10)	.159 (7)	.129 (15)	.131 (13)	.144 (11)	.164 (6)	.193 (5)	.653 (2)	.721 (1)	.154 (8)	.150 (9)	.130 (14)	.142 (12)	.199 (4)	.477 (3)
Avg. rank	6.6	9.5	12	7.6	9	9.8	8	4.1	5.7	10.5	8.9	8.1	8.7	7.8	3.6

Video	ASLA	L1APG	CT	TLD	IVT	MTT	CSK	SCM	Struck	MIL	LSK	VTS	VTD	CXT	DPL²
Bolt	.017 (11)	.017 (11)	.011 (15)	.306 (3.5)	.014 (13.5)	.017 (11)	.034 (6)	.031 (7)	.020 (9)	.014 (13.5)	.977 (1)	.089 (5)	.306 (3.5)	.026 (8)	.951 (2)
Car4	1.000 (2)	.302 (13)	.281 (14)	.874 (6)	1.000 (2)	.361 (10)	.355 (11)	.974 (5)	.992 (4)	.354 (12)	.077 (15)	.363 (9)	.364 (8)	.382 (7)	1.000 (2)
CarDark	1.000 (5)	1.000 (5)	.005 (15)	.639 (13)	.807 (10)	1.000 (5)	1.000 (5)	1.000 (5)	1.000 (5)	.379 (14)	1.000 (5)	1.000 (5)	.743 (11)	.728 (12)	1.000 (5)
Coke	.165 (11)	.265 (8)	.113 (15)	.684 (4)	.131 (14)	.660 (5)	.873 (3)	.430 (7)	.948 (1)	.151 (12)	.258 (9)	.189 (10)	.148 (13)	.653 (6)	.928 (2)
David	1.000 (3)	.805 (10)	.815 (9)	1.000 (3)	1.000 (3)	.333 (14)	.499 (13)	1.000 (3)	.329 (15)	.699 (11.5)	.699 (11.5)	.962 (7)	.943 (8)	1.000 (3)	.981 (6)
Deer	.028 (14)	.718 (7)	.042 (11)	.732 (6)	.028 (14)	.887 (5)	1.000 (2)	.028 (14)	1.000 (2)	.127 (9)	.338 (8)	.042 (11)	.042 (11)	1.000 (2)	.972 (4)
Fish	1.000 (3)	.055 (13)	.882 (7)	1.000 (3)	1.000 (3)	.042 (14.5)	.042 (14.5)	.863 (8)	1.000 (3)	.387 (11)	.332 (12)	.992 (6)	.649 (10)	1.000 (3)	.811 (9)
Jogging2	.182 (12)	.186 (9)	.166 (14)	.857 (3)	.199 (6)	.173 (13)	.186 (9)	1.000 (1.5)	.254 (5)	.186 (9)	.544 (4)	.186 (9)	.186 (9)	.163 (15)	1.000 (1.5)
Shaking	.485 (6)	.041 (13)	.047 (12)	.405 (8)	.011 (15)	.014 (14)	.564 (5)	.814 (4)	.192 (10)	.282 (9)	.466 (7)	.921 (2)	.934 (1)	.126 (11)	.841 (3)
Singer1	1.000 (3.5)	.379 (14)	.840 (9)	1.000 (3.5)	.963 (8)	.339 (15)	.670 (10)	1.000 (3.5)	.641 (11)	.501 (12)	.481 (13)	1.000 (3.5)	1.000 (3.5)	.966 (7)	1.000 (3.5)
Soccer	.122 (13)	.207 (8)	.219 (7)	.115 (15)	.173 (11)	.184 (10)	.135 (12)	.268 (4)	.253 (5)	.191 (9)	.120 (14)	.505 (2)	.452 (3)	.232 (6)	.798 (1)
Trellis	.861 (5)	.176 (15)	.387 (11)	.529 (8)	.332 (12)	.220 (14)	.810 (6)	.873 (4)	.877 (3)	.230 (13)	.967 (2)	.503 (9)	.497 (10)	.970 (1)	.729 (7)
Walking	1.000 (6)	1.000 (6)	1.000 (6)	.964 (13)	1.000 (6)	1.000 (6)	1.000 (6)	1.000 (6)	1.000 (6)	1.000 (6)	.658 (14)	1.000 (6)	1.000 (6)	.235 (15)	.968 (12)
Woman	.203 (11.5)	.204 (8.5)	.204 (8.5)	.191 (15)	.201 (13)	.204 (8.5)	.250 (5)	.940 (2)	1.000 (1)	.206 (6)	.204 (8.5)	.198 (14)	.203 (11.5)	.367 (4)	.938 (3)
Avg. rank	7.6	10	11	7.4	9.3	10.4	7.7	5.3	5.7	10.5	8.9	7	7.8	7.1	4.4

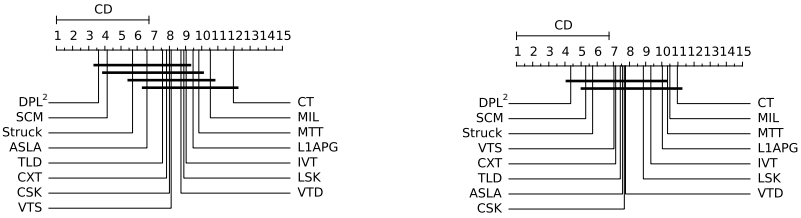

Fig. 3. Comparison using AUC of success rate (left) and precision (right) of all algorithms against each other using the Nemenyi test. Groups of classifiers that are not significantly different at $p = 0.05$ are connected (CD greater than 5.732)

that the trackers perform equally. Such hypothesis is rejected. Then, a Nemenyi test is performed to compare the methods in a pairwise way. Both tests are based on average ranks. The comparison includes 15 trackers over 14 videos, so the critical difference (CD) in the Nemenyi test is 5.732 for significance level of 5%. The results are in Figure 3. Notice that DPL^2 ranks higher in both cases. Moreover, only DPL^2 and SCM are significantly better according to a Nemenyi test than the worst trackers (MIL and CT). However, as we can observe in Figure 3, most of the differences are not statistically significant. This is due in part to the fact that the number of algorithms compared is greater than the number of video sequences, so the critical difference 5.7 is quite difficult to reach. In any case, the overall results discussed in this section are quite promising, supporting the main hypothesis of this work which is that preference learning and deep learning are both well-tailored to tackle object tracking tasks.

4 Conclusions

This paper analyzes the behavior of two very popular techniques, deep learning and preferences learning, working together in the context of object tracking. The performance of the proposed method, DPL^2 is quite competitive with respect to state-of-the-art methods, and sometimes it is even better, both from a quantitative and qualitative point of view. However, one of the most interesting aspects of this study is that it seems that there is still room to improve the accuracy of a tracking system based on the combination of deep learning and preference learning. Moreover, we have introduced a new point of view to approach object tracking tasks: the use of preference learning. The application of this paradigm opens up a new research line that can eventually be explored in the future.

Acknowledgments

This work was funded by Ministerio de Economía y Competitividad de España (grant TIN2015-65069-C2-2-R), Specialized Research Fund for the Doctoral Program of Higher Education of China (grant 20120061110045) and the Project of Science and Technology Development Plan of Jilin Province, China (grant 20150204007GX). The paper was written while Shuchao Pang was visiting the University of Oviedo at Gijón.

References

1. Bahamonde, A., Bayón, G.F., Díez, J., Quevedo, J.R., Luaces, O., del Coz, J.J., Alonso, J., Goyache, F.: Feature subset selection for learning preferences: A case study. In: ACM ICML (2004)
2. Bai, Y., Tang, M.: Robust tracking via weakly supervised ranking svm. In: IEEE CCVPR (2012)
3. Dai, P., Liu, K., Xie, Y., Li, C.: Online co-training ranking svm for visual tracking. In: IEEE ICASSP (2014)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. JMLR 7, 1–30 (2006)
5. Jepson, A.D., Fleet, D.J., El-Maraghi, T.F.: Robust online appearance models for visual tracking. IEEE PAMI 25(10), 1296–1311 (2003)
6. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553),436–444(2015)
7. Vapnik, V.: Statistical Learning Theory. John Wiley, New York, NY (1998)
8. Wang, N., Yeung, D.Y.: Learning a deep compact image representation for visual tracking. In: NIPS (2013)
9. Wu, Y., Lim, J., Yang, M.: Object tracking benchmark. IEEE PAMI 37(9), 1834–1848 (2015)
10. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Computing Surveys (CSUR) 38(4), 13 (2006)
11. Zhao, L., Hu, Q., Zhou, Y.: Heterogeneous features integration via semi-supervised multi-modal deep networks. In: ICONIP (2015)