Neural Processing Letters manuscript No.
(will be inserted by the editor)

Deep Learning and Preference Learning for Object Tracking: A combined approach

Shuchao Pang · Juan José del Coz · Zhezhou Yu · Oscar Luaces · Jorge Díez

the date of receipt and acceptance should be inserted later

Abstract Object tracking is one of the most important processes for object recognition in the field of computer vision. The aim is to find accurately a target object in every frame of a video sequence. In this paper we propose a combination technique of two algorithms well-known among machine learning practitioners. Firstly, we propose a *deep learning* approach to automatically extract the features that will be used to represent the original images. *Deep learning* has been successfully applied in different computer vision applications. Secondly, object tracking can be seen as a ranking problem, since the regions of an image can be ranked according to their level of overlapping with the target object (ground truth in each video frame). During object tracking, the target position and size can change, so the algorithms have to propose several candidate regions in which the target can be found. We propose to use a preference learning approach to build a ranking function which will be used to select the bounding box that ranks higher, i.e., that will likely enclose the target object. The experimental results obtained by our method, called DPL^2 (Deep & Preference Learning), are competitive with respect to other algorithms.

Keywords Deep learning · Preference learning · Object tracking

1 Introduction

The goal of object tracking systems is to precisely follow the trajectory of a moving object in a video. This task attracts attention because it tackles several interesting applications, including surveillance, vehicle navigation, augmented reality, human-computer interactions and medical imaging, among others [33]. In the past decade,

S. Pang · Z. Yu
Coll. Computer Science and Technology, Jilin University, Changchun, 130012, China
E-mail: pangshuchao1212@sina.com, yuzz@jlu.edu.cn

J.J. del Coz · O. Luaces and J. Díez
Artificial Intelligence Center, University of Oviedo at Gijón, 33204, Spain
E-mail: juanjo@uniovi.es, oluaces@uniovi.es, jdiez@uniovi.es

the research of object tracking makes significant progress and some algorithms have been proposed [9,13,36,24]. However, object tracking is still a challenging task because a robust tracker must deal with difficult situations in real world environments such as illumination variations, full or partial occlusions of the target object, scale changes, deformation, fast motion and background clutter, just to cite the most important ones. In fact, no actual tracker is able to be successful in all possible scenarios and most of them simplify the problem by imposing constraints (e.g. assuming that the object motion is smooth) usually based on prior knowledge about the actual application.

Object tracking systems can conventionally be classified into two categories: *generative* and *discriminative* methods. The former usually learns a model that represents the appearance of the target object and then try to find the most similar object in each frame. Discriminative trackers train a classifier to separate the target object from the complex background.

Trackers usually have three main components: object representation and feature selection methods [21], an object detection mechanism [5] and an update strategy [12]. Objects can be represented by a set of features such as the shape, appearance, color, texture, etc. These features can be i) manually chosen by an expert, depending on the application domain, ii) automatically chosen, using a feature selection algorithm or iii) by the combination of both. In order to accurately track the target object, the best target object representation needs to be selected. 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. For instance, in the case of following a supervised discriminative tracking approach, the model must be updated to deal with the drifting situations described above. Trackers proposed so far differ in at least some of these components. See [33] for a complete survey of object tracking approaches.

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. They have not been used together in the context of object tracking. Firstly, DPL^2 applies a deep learning architecture to represent the images. Thus, we extract invariant features with a multi-layer network. These features will be used instead of raw pixels or other features obtained by means of image analysis techniques which are sensible to, for instance, illumination variations. This process has also the advantage of being computationally efficient because such network is trained beforehand. Secondly, the model to detect and track the target object is obtained using a supervised ranking algorithm. All the possible bounding boxes of an image could be ranked by their degree of overlapping with the target object. Thus, there are some regions that are *preferable* for tracking the object. This reason leads us to use preference learning to build and keep updated a ranking model that allows DPL^2 to select the more promising region. 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 rest of the paper is organized as follows. After a brief discussion of the related work in next section, our proposal based on deep learning and preference learning is presented in Section 3. Then, we report a thoroughly experimentation devised to show the benefits of our approach. The paper ends drawing some conclusions.

This work is an extended version of the research published at ICONIP-16 [25], which includes a more detailed analysis of related works, a more exhaustive experimentation as well as a deeper analysis of the results.

2 Related work

In the work of Wu et al. [31] several trackers are experimentally compared and an object tracking benchmark is presented. We will use this benchmark in our experiments. Jia et al. [13] propose a tracking method based on a structural local sparse appearance model. Such representation combines spatial information and partial information of the target based on a alignment-pooling method. Additionally, incremental subspace learning and sparse representation are used to update the templates to handle drifting situations. Zhong et al. [36] employ a sparsity-collaborative model that exploits both holistic templates and local representations. The method combines a discriminative classifier and a generative model. Its update scheme takes into account the latest observations and the original template. The method presented in [9] is based on structured output prediction by the definition of an appropriate output space to represent the needs of the tracker. Then, a kernelized structured output support vector machine is learned to obtain an adaptive tracking system. The object model is updated using online learning techniques. Kwon and Lee [17] search for the appropriate trackers in each frame. Since the sequence may vary over time, the trackers should be modified or replaced. To take this decision, the system obtains samples of the state of the target and of the own trackers. The trackers are sampled using a Markov Chain Monte Carlo method from a predefined tracker space composed of different building blocks, for instance, motion models. The sampled trackers run in parallel, interacting and covering different target variations. The work of Kalal et al. [15] is based on a binary classifier trained with labeled and unlabeled examples. It uses an iterative learning process guided by positive and negative constraints that restrict the labeling of the unlabeled data. The method evaluates the classifier on the unlabeled examples, identifying those that are in contradiction with structural constraints. Those examples are added to the training set and the classifier is retrained.

The work done by Wang et al. [30] is the most comparable to our work within the framework of deep learning in the context of object tracking. 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 strategy. In [3] a tracking system based on Laplacian ranking SVM is presented. The tracker incorporates the labeled information of the object in the initial and the latest frames to deal with drift changes, like occlusion, and the weakly labeled information from the current frame to adapt to substantial changes of the appearance. Another difference with our approach is that Bai and Tang use Haar-like features to represent each image patch and we employ a deep learning network. The method described in [6] is also based on preference learning, but the authors

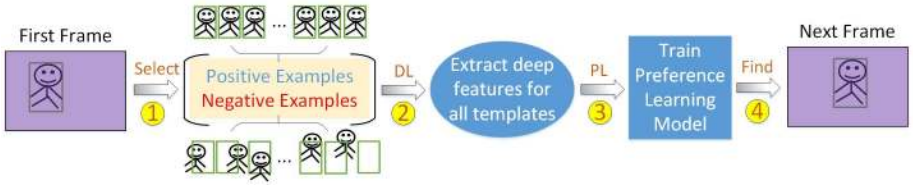

Fig. 1 The overall framework of our proposed DPL^2 algorithm. 1) Positive boxes near to the target object and negative boxes away from the object are selected; 2) deep features to describe each example are extracted; 3) preference learning is applied in order to build a ranking model; and 4) the learned model ranks several patches and the best one is selected as the object in the next frame.

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.

3 Deep & Preference Learning Tracker

This section is devoted to describe in detail 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 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. The positive examples are obtained by moving the target box just ± 1 pixel from the true position, while the negative ones are obtained by displacing the box a distance greater than $\pm width/4$ in both axis ($width$ and $height$ represent the size of the target object in the first frame). Then, DPL^2 uses a deep learning network to extract deep features to describe each example selected in the previous step. The goal of this step is to obtain invariant features to make tracking more robust against variations in the video sequence. In the third step, the deep learning model is then integrated with preference learning to build a ranking model to detect the object. Such a model is learned using a set of preferences judgements. This set is formed by all possible pairs of a positive example and a negative example. The model should learn that the positive examples are preferable to the negative ones. 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 previous 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 3.3.

3.1 Deep Learning

Many tracker systems use the raw pixels or handcrafted features, such as pixel histogram features, Haar-like features or Surf features, as the methods to represent

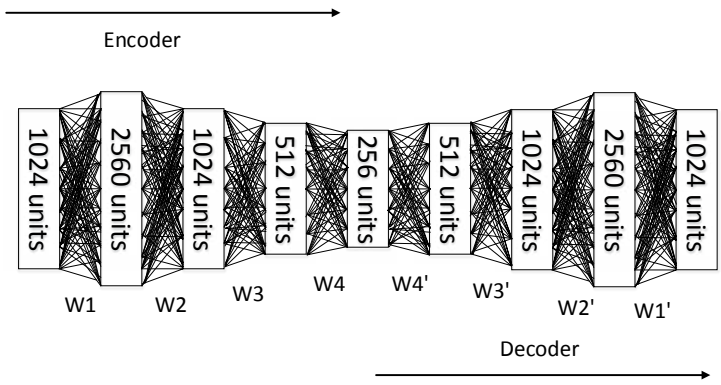

Fig. 2 Structure of the Stacking Denoising Auto-Encoder (SDAE). During the offline training phase SDAE learns the weights that allow to encode (W1. . . W4) and decode (W4'. . . W1') each image. Once the weights are learned, during the tracking phase each patch is encoded in a 256 dimensional space. This representation is used as a vector of features in the preference learning algorithm

the tracked object. However, all these low-level and superficial features do not usually work for difficult video sequences.

Although there are many variants of deep learning architectures, all of them are based on the same main idea. A deep learning method is a representation-learning technique that using raw data produces computational models to discover the representations needed for subsequent learning processes (like classification or detection in our case). These models, composed of l processing layers ($l > 1$), learn data representations with multiple degrees of abstraction; each degree transforms the representation at previous level into more abstract representation. Very complex functions can be learned when l is sufficiently large. The higher layer captures discriminative variations, suppressing irrelevant ones. Another important aspect is that these networks are not designed by practitioners, but learned using general-purpose procedures.

In recent years, deep learning architectures have succeeded to be implemented in numerous practical applications in many research areas, especially in computer vision, like face recognition and image classification, because deep learning architectures are able to learn more invariant and inherent features via the multi-layer learning described before. Deep learning yields high-level image features that are more accurate and stable in order to represent the essence of the images. Even in the case of complex videos, this approach provides better features of the moving object to prevent drifting away from the target. For all these reasons, deep learning is used in our tracking system for learning an object representation which is robust against variations.

Examples of deep learning methods include Deep Belief Networks (DBN), Convolutional Neural Networks (CNN) [20,22,37], Deep Boltzmann Machines (DBM) and Stacked Denoising Auto-Encoders (SDAE). However, lately the most popular deep learning architectures are probably CNN [18,19] and SDAE [26,30]. Here, we choose the SDAE architecture, shown in Figure 2, just as an example to study how

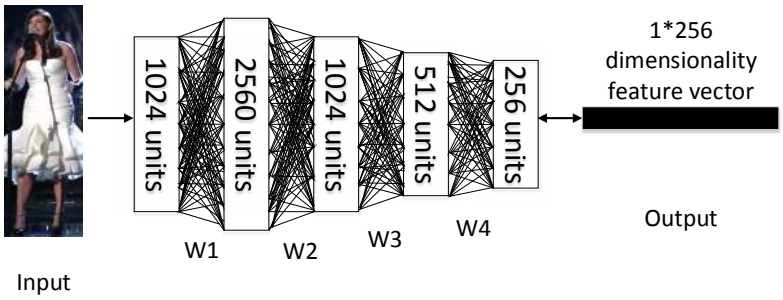

Fig. 3 Deep feature extraction using the encoder substructure of the previously trained SDAE

current deep learning techniques perform in combination with preference learning to tackle object tracking tasks.

A SDAE model is composed of two parts, an encoder and a decoder. Each part contains a network that 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) that is 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:

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

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 [30], DPL^2 uses the Tiny Images dataset [28] to train the SDAE model offline. Such dataset contains 80 million images of 32x32 pixels. The idea is to obtain a generic representation model for any real world image. The network architecture 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 256 units. A logistic

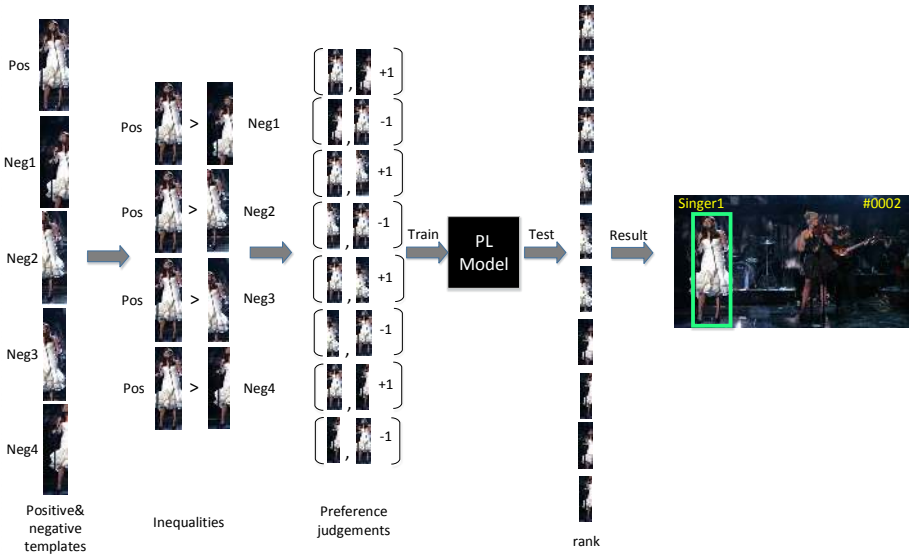

Fig. 4 Schematic view of the training and use of the preference learning module in DPL^2. In this figure we represented image patches for illustrative purposes but, in fact, we use their encoded version given by the trained SDAE. The examples used to train the preference learning process (PL model) are triplets of the form (image1, image2, label), where label can be either +1, when image1 is preferred to image2, or -1, when image2 is preferred to image1

sigmoid activation function, $\rho(x) = \frac{1}{1+e^{-x}}$, is used for f_θ and g_θ. Figure 3 shows an example of use of the encoder to obtain the deep features of an image.

3.2 Preference Learning

Although there are other approaches to learn preferences, following [2,11,14] we will try to induce a real *preference* or *ranking function* $r : \mathbb{R}^d \to \mathbb{R}$, that is, from the space of objects considered, our images represented using SDAE ($d = 256$), to the reals. 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 patches extracted from the first frame, as shown in Figure 4. This set of objects is endowed with a partial order that can be expressed by a collection of preference judgments where any positive patch is always preferred to any negative patch:

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

In order to induce the ranking function, we can use the approach presented by Herbrich et al. in [11]. So, 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 construct a dataset of preference judgements, PJ, using pairs of image patches so that a good (positive) patch which includes the target in the image is preferred to a bad (negative) patch which only includes partially (or not at all) the target. These pairs give rise to a binary classification problem, where a pair of correctly ordered patches (good, bad) belongs to class +1 and a pair of incorrectly ordered patches (bad, good) belongs to class -1. Then, DPL^2 uses Support Vector Machines (SVM) [29] to address this classification task, so we need to solve the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{w}, \boldsymbol{\xi}} \quad & \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_{i,j} \xi_{ij}, \\ \text{s.t.} \quad & \langle \boldsymbol{w}, \boldsymbol{v}_i - \boldsymbol{u}_j \rangle \geq 1 - \xi_{ij}, \\ & \xi_{ij} \geq 0, \forall (\boldsymbol{v}_i, \boldsymbol{u}_j) \in PJ. \end{aligned} \tag{6}$$

The final ranking function is:

$$r(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle = \sum_{i,j} \beta_{ij} \langle \boldsymbol{v}_i - \boldsymbol{u}_j, \boldsymbol{x} \rangle, \tag{7}$$

in which $\beta_{ij} \neq 0$ represents the Lagrange multipliers of the so-called support vectors. Notice that there is not bias term because the positives and negatives pairs in PJ are symmetric with respect to the origin.

3.3 Model update

Previous studies [1,9,12,13,15] have shown that updating the mechanism to detect the object is crucial for obtaining robust tracking. For instance, when the illumination changes suddenly or when the target object is partially occluded, it is easy that the tracker losses the position or the size of the object. The goal of the update method is to take into account for such drifting situations that happen in the video sequence, obtaining a final tracking after learning that is able to smoothly follow the object even when there are important variations in the appearance of the object.

In the case of discriminative tracking algorithms, like DPL^2, the model used for detection must be updated. DPL^2 updates the model learned solving the optimization problem in (6) 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. The new model replaces the previous one and it is used to predict the position of the object in the next frames.

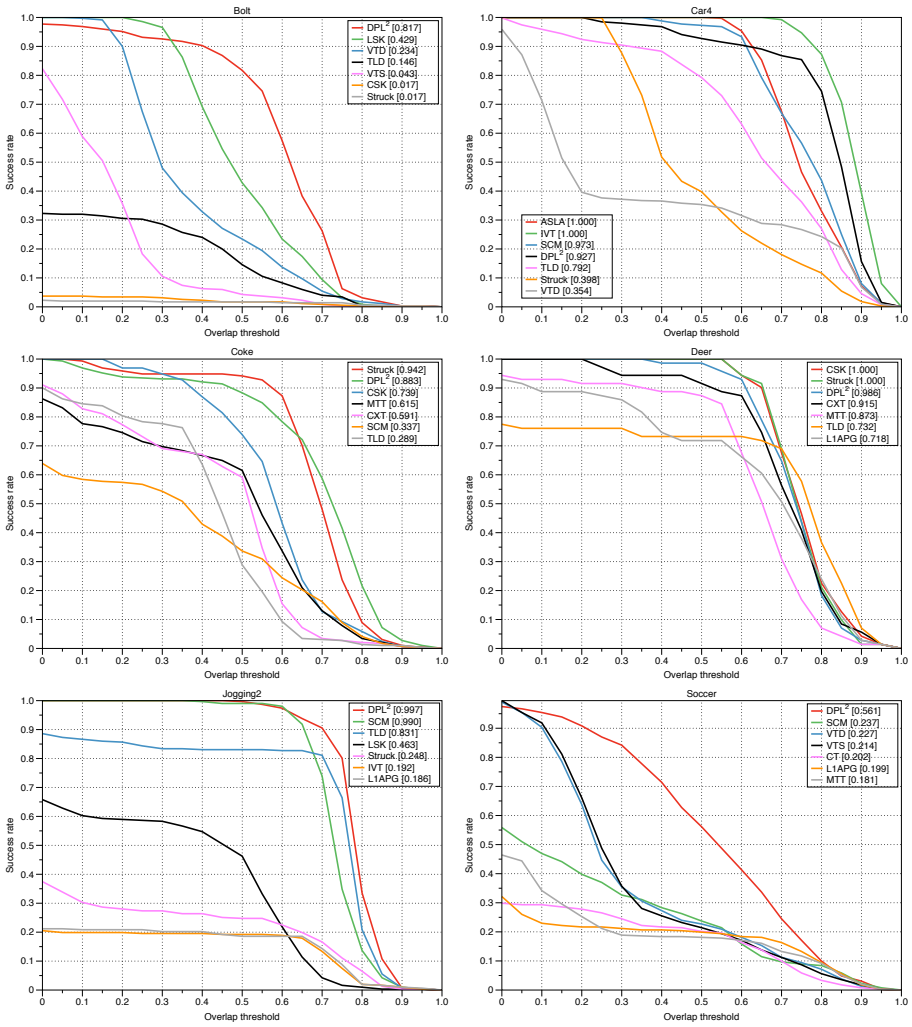

Fig. 5 Success plots with different threshold values. For each video, only the results of the top seven trackers are depicted. The trackers systems are ranked in the legend according to their success rate at the overlap threshold 0.5. Following [31,32], colors in the graphs are related to the position of the algorithms in the performance ranking.

4 Experimental results and discussion

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 [31,32]. The best top 14 object tracking algorithms were selected, concretely: SCM [36], Struck [9], ASLA [13], L1APG [4], CT [34], TLD [15], IVT [27], MTT [35], CSK [10], MIL [1], LSK [23], VTS [17], VTD [16] and CXT [8].

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 [31,32] but we had to limit the number of experiments due to space restrictions. For the datasets we randomly selected 14 videos with different difficulty degree: hard videos (Bolt, Coke, Soccer and Woman), medium-difficulty videos (David, Deer, Shaking and Trellis) and easy videos (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 stated 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, thus the PJ set (2) used to learn the tracking model had always 300 positive preference judgments (and their respective 300 negative preference judgements). 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, see Section 3.3, 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 100 particles randomly around the position of the predicted region in the previous frame; the one that obtains a higher value applying the ranking function (7) is returned.

4.1 Quantitative Comparison

The first metric to compare the performance of object tracking algorithms is the Success Rate which measures the overlap rate of each frame. This overlap is defined as the intersection area between the predicted tracking box (r_t) and the ground-truth area (r_0) divided by the union area of these two regions. In symbols, $S = r_t \cap r_0 / r_t \cup r_0$. Calculating the percentage of the number 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 comprehensively depict the overall performance of each tracker. So we draw the success plots of each tracker by making the threshold value vary from 0 to 1. Due to the limited space, only the results of six videos are shown in Figure 5. The rest can be found in the supplementary material [1]. As we can see,

[1] https://doi.org/10.13140/RG.2.2.18362.18881

the performance of DPL^2 in terms of Success Rate is rather good. DPL^2 is clearly the best in two of the sequences (bolt and soccer), is similar to best performers in other three videos (coke, deer, jogging2) and it is only worst in one of them (car4).

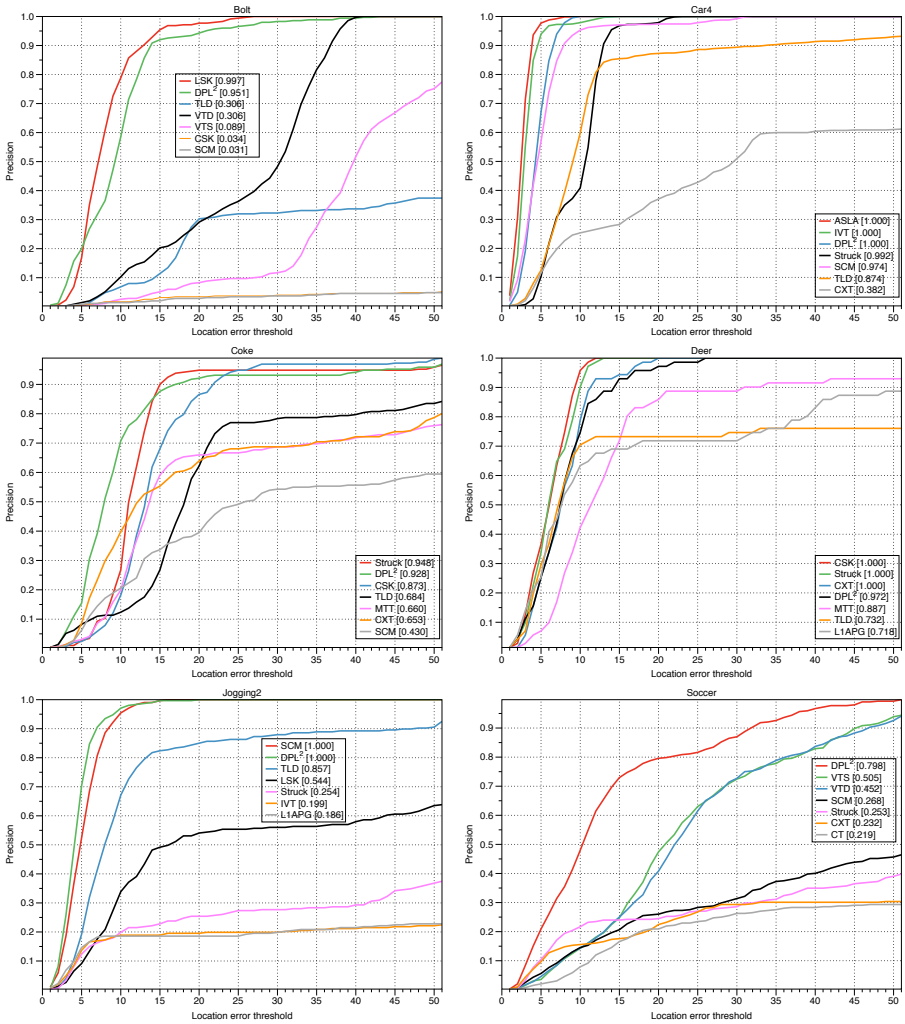

Fig. 6 Precision percentage plots with different location error thresholds for the top seven trackers. The trackers systems are ranked in the legend according to the percentage of frames in which the error of the tracker is less than 20 pixels. Following [31,32], colors in the graphs are related to the position of the algorithms in the performance ranking.

Another widely used evaluation metric is Precision that computes the average euclidean distance between the center locations of the predicted regions and the ground-truth positions of all the frames. Figure 6 depicts the Precision percentage, that is the percentage of frames in which the error of the tracker is less than a given

number of pixels. The scores of DPL^2 are competitive for all the video sequences, and in one of them is clearly the best tracker (soccer).

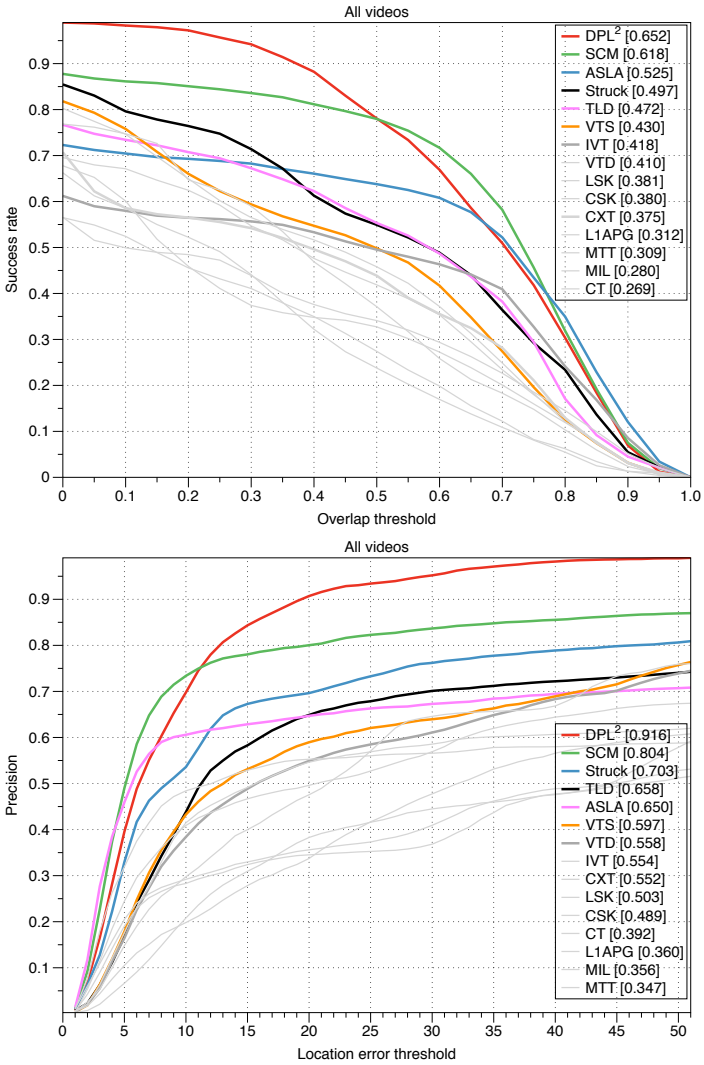

Fig. 7 Average Success Rate (top) and Precision (bottom) plots of all video frames. Ranks in the legend are computed using the rea under the curve of Success Rate and the percentage Precision for 20 pixels. Following [31, 32], colors in the graphs are related to the position of the algorithms in the performance ranking.

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, totally 5704 frames, obtaining the average values for Success Rate and Precision. These plots are in Figure 7. As we can see, DPL^2 performs quite well for both

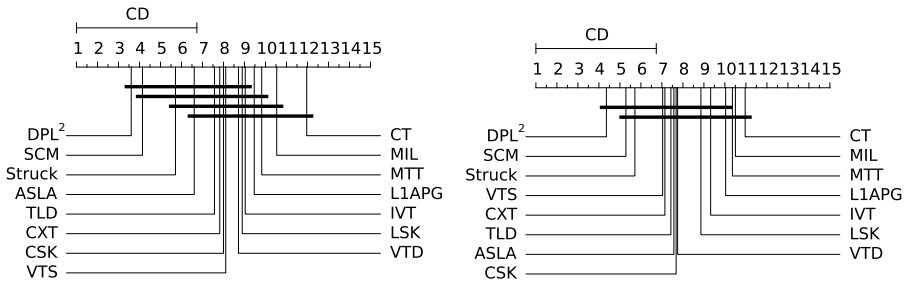

Fig. 8 Comparison using the area under the curve of Success Rate (left) and precision (right) of all algorithms against each other using the Nemenyi test. Groups of classifiers that are not significantly different at $p = 0.05$ are connected (critical difference, CD, greater than 5.732)

metrics, its scores are similar to those of SCM which is one of the best trackers according to [31]. Although DPL^2 ranks the best for Success Rate when threshold is 0.5, it seems that its performance decays when the value of the threshold increases. 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 is a quite robust tracker that rarely losses the track of the object.

Besides, for comprehensively analyzing the overall performance of each tracker from a statistical point of view, Table 1 reports the scores of the Area Under the Curve (AUC) of the Success Rate and the results for Precision percentage for all the tracking systems over all the video sequences. To sum up the results, the average rank of each tracker is the last row of each table. Following [7], a two-step statistical test procedure was carried out. The first step consists of a Friedman test of the null hypothesis that states 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 8.

It is worth noting that DPL^2 ranks higher in both cases followed by SCM and Struck algorithms. 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 8, 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.

Video	ASLA	L1APG	CT	TLD	IVT	MTT	CSK	SCM	Struck	MIL	LSK	VTS	VTD	CXT	DPL^2
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	DPL^2
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

Table 1 Scores and ranking position of each tracker on each video using two different metrics: area under the curve (AUC) of Success Rate (top) and Precision (bottom). The average rank for each system is in the last row of each table

4.2 Qualitative Comparison

To compare the methods considered form a qualitative point of view, we run all the 15 object tracking approaches on the 14 video sequences, drawing the predicted bounding boxes of each tracker on every frame. All these videos are included as supplementary material.

Furthermore, according to the average ranks based on Success Rate, Precision and the AUC of Success Rate (see Table 1), we can obtain the top 5 methods in each ranking. To perform a qualitative comparison we take the union of all these rankings, resulting in a group of seven trackers (ASLA, TLD, SCM, Struck, VTS, CXT and DPL^2). These systems were analyzed frame-by-frame. The idea was to study the robustness of each method, particularly looking if the trackers loss the target object. Figure 9 contains the results for six video sequences. For each tracking system the figures show the overlap rate on each frame. The rest of the graphs can be found in the supplementary material (that will be available once the paper will be published).

Bolt video is a rather difficult tracking sequence because the target (Usain Bolt) moves quite fast from frame to frame and also there are some interferences (the rest of the athletes). We can observe that DPL^2 is the overall best tracker. Although at the beginning there are several frames in which DPL^2 can not successfully track the target, then it recovers the object using its update process. After that DPL^2 maintains acceptable overlap rates for the rest of the video. At the same time, we can see that other trackers only can track Bolt in the first 50 or 100 frames and then fail to track him. Although the overlap rate of VTS tracker is very low, it may slightly better than the rest trackers. But after 200 frames it also fails. Considering the total of 350 frames, it seems that DPL^2 outperforms the rest of the methods.

In the Car4 sequence, the object being tracked is a moving car on a broad road. The main challenges of this video are the illumination variations before and after crossing a bridge and scale changes of the car. As the plots of Car4 show, DPL^2 tracks the car in a similar way than ASLA, TLD and SCM. And the average overlap rate is better than those of other methods. However, the scores of DPL^2 decreases and are relatively low when the car is under the bridge. This is caused by the sudden illumination variation. However, in these 50 dark frames, DPL^2 still tracks the car and does not totally loss it. After the car passes under the bridge, DPL^2 rapidly recovers the size of tracked bounding box according to the scale of the car. It seems that is able to handle reasonably well these drifting situations.

In Coke sequence there are continuous part and full occlusions of the object, light variations, pose changes and in-plane rotations. The results show that only DPL^2 and Struck can accurately track the object despite these interference factors. In particular, both loss the object at the end of the sequence but they recover the track quite fast.

Deer is a short but very challenging video sequence due to the interference of fast motion, similar background and pose and expression changes. In the Jogging2 sequence, the main challenging is the full occlusion produced by the big telegraph pole. The behavior of DPL^2 is similar in both cases, obtaining pretty good scores. DPL^2 maintains a high overlap rate, greater that 0.5 almost during all the sequences. Other methods obtain also good results, like Struck and CXT in the case of Deer video and SCM in Jogging2 video. Interestingly, the success plots of DPL^2

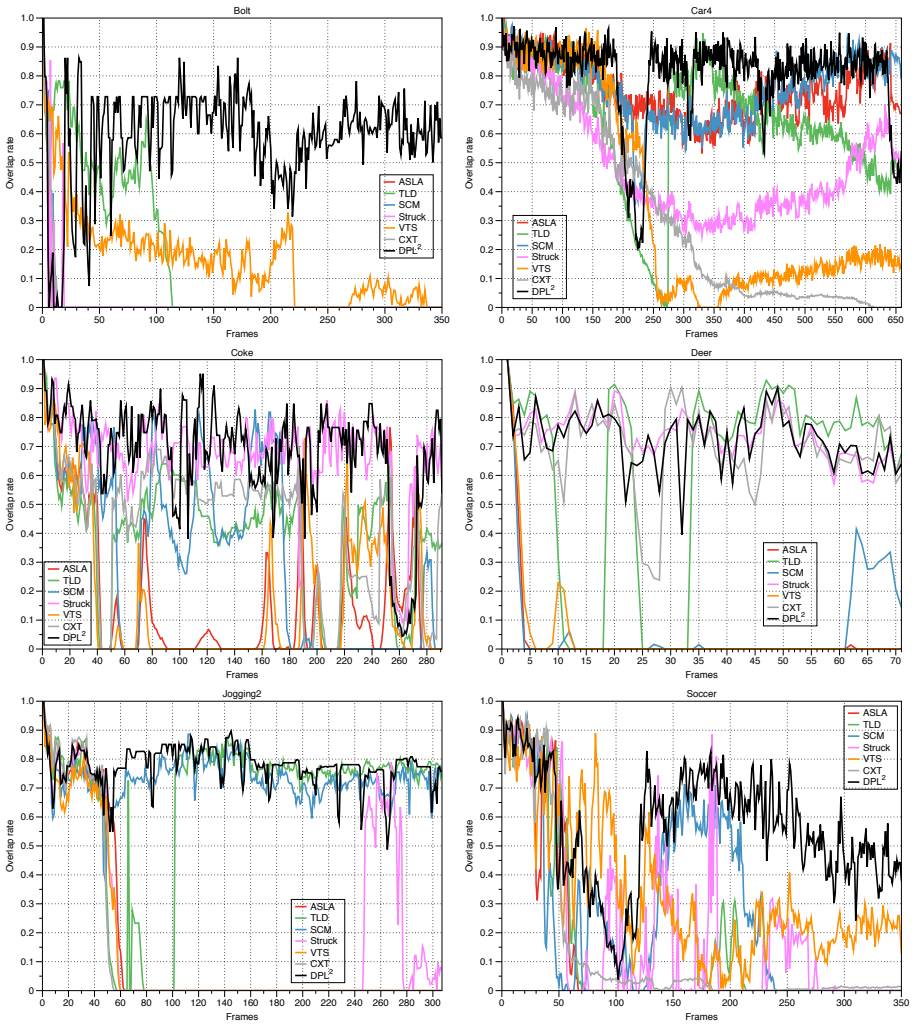

Fig. 9 Qualitative comparison frame-by-frame with the best trackers. For each tracking system the graphics show the overlap rate on each frame

are very sharp and narrow. This implies that when DPL^2 losses the object, the recovery is particularly fast.

Finally, Soccer sequence is one of the hardest video to track. It not only includes severe occlusion, scale variation, motion blur and fast motion, but exists the phenomenon of background clutters in the whole process and illumination variation. DPL^2 obtains worse results than in previous cases, specially at the beginning of the sequence. However, it is the best method at the end of the video. Other methods only accurately track the first 50 frames and then fail.

5 Conclusions

The work presented in this paper explores the possibility of combining deep learning with the paradigm of preference learning to build an object tracking system. The performance of the proposed method, DPL^2, is quite competitive with respect to state-of-the-art methods, and sometimes 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.

The main conclusion that can be extracted from the qualitative frame-by-frame comparison is that DPL^2 is more stable than other trackers, it maintains an acceptable overlap rate for most of the frames, and when it losses the object it is able to recover it quite reliably.

Additionally, we have also outlined the object tracking task from a new point of view: the preference learning. This will allow us to explore the development of new algorithms that may increase performance. 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. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09), pp. 983–990 (2009) 8, 10
2. 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: Proceedings of ICML'04. ACM (2004) 7
3. Bai, Y., Tang, M.: Robust tracking via weakly supervised ranking svm. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1854–1861. IEEE (2012) 3
4. Bao, C., Wu, Y., Ling, H., Ji, H.: Real time robust l1 tracker using accelerated proximal gradient approach. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12), pp. 1830–1837 (2012) 10
5. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **25**(5), 564–577 (2003) 2
6. Dai, P., Liu, K., Xie, Y., Li, C.: Online co-training ranking svm for visual tracking. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 6568–6572 (2014) 3
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**, 1–30 (2006) 13
8. Dinh, T.B., Vo, N., Medioni, G.: Context tracker: Exploring supporters and distracters in unconstrained environments. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'11), pp. 1177–1184 (2011) 10

9. Hare, S., Saffari, A., Torr, P.H.: Struck: Structured output tracking with kernels. In: IEEE International Conference on Computer Vision (ICCV'11), pp. 263–270 (2011) 2, 3, 8, 10
10. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: Exploiting the circulant structure of tracking-by-detection with kernels. In: Computer Vision–ECCV 2012, pp. 702–715. Springer (2012) 10
11. Herbrich, R., Graepel, T., Obermayer, K.: Support vector learning for ordinal regression. In: In International Conference on Artificial Neural Networks, pp. 97–102 (1999) 7
12. Jepson, A.D., Fleet, D.J., El-Maraghi, T.F.: Robust online appearance models for visual tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **25**(10), 1296–1311 (2003) 2, 8
13. Jia, X., Lu, H., Yang, M.H.: Visual tracking via adaptive structural local sparse appearance model. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12), pp. 1822–1829 (2012) 2, 3, 8, 10
14. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02, pp. 133–142 (2002) 7
15. Kalal, Z., Matas, J., Mikolajczyk, K.: Pn learning: Bootstrapping binary classifiers by structural constraints. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10), pp. 49–56 (2010) 3, 8, 10
16. Kwon, J., Lee, K.M.: Visual tracking decomposition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10), pp. 1269–1276 (2010) 10
17. Kwon, J., Lee, K.M.: Tracking by sampling trackers. In: IEEE International Conference on Computer Vision (ICCV'11), pp. 1195–1202 (2011) 3, 10
18. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: A convolutional neural-network approach. IEEE Trans. on Neural Networks **8**(1), 98–113 (1997) 5
19. LeCun, Y., Bengio, Y.: Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks **3361**(10) (1995) 5
20. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015) 5
21. Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A., Hengel, A.V.D.: A survey of appearance models in visual object tracking. ACM Transactions on Intelligent Systems and Technology (TIST) **4**(4), 58 (2013) 2
22. Lin, T.Y., Cui, Y., Belongie, S., Hays, J., Tech, C.: Learning deep representations for ground-to-aerial geolocalization. In: Proc. of the IEEE CVPR'15, pp. 5007–5015 (2015) 5
23. Liu, B., Huang, J., Yang, L., Kulikowsk, C.: Robust tracking using local sparse appearance model and k-selection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'11), pp. 1313–1320 (2011) 10
24. Luque-Baena, R.M., Ortiz-de Lazcano-Lobato, J.M., López-Rubio, E., Domínguez, E., J. Palomo, E.: A competitive neural network for multiple object tracking in video sequence analysis. Neural Processing Letters **37**(1), 47–67 (2013). 2
25. Pang, S., del Coz, J.J., Yu, Z., Luaces, O., Díez, J.: Combining deep learning and preference learning for object tracking. In: A. Hirose, S. Ozawa, K. Doya, K. Ikeda, M. Lee, D. Liu (eds.) Neural Information Processing: 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part III, pp. 70–77. Springer International Publishing (2016). 3
26. Pang, S., Yu, Z.: Face recognition: a novel deep learning approach. J. of Optical Technology **82**(4), 237–245 (2015) 5
27. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. International Journal of Computer Vision **77**(1-3), 125–141 (2008) 10
28. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: A large data set for nonparametric object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**(11), 1958–1970 (2008) 6
29. Vapnik, V.: Statistical Learning Theory. John Wiley, New York, NY (1998) 8
30. Wang, N., Yeung, D.Y.: Learning a deep compact image representation for visual tracking. In: Advances in Neural Information Processing Systems, pp. 809–817 (2013) 3, 5, 6
31. Wu, Y., Lim, J., Yang, M.: Object tracking benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(9), 1834–1848 (2015) 3, 9, 10, 11, 12, 13
32. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13), pp. 2411–2418 (2013) 9, 10, 11, 12
33. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Computing Surveys (CSUR) **38**(4), 13 (2006) 1, 2

34. Zhang, K., Zhang, L., Yang, M.H.: Real-time compressive tracking. In: Computer Vision–ECCV 2012, pp. 864–877. Springer (2012) 10
35. Zhang, T., Ghanem, B., Liu, S., Ahuja, N.: Robust visual tracking via multi-task sparse learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12), pp. 2042–2049 (2012) 10
36. Zhong, W., Lu, H., Yang, M.H.: Robust object tracking via sparsity-based collaborative model. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12), pp. 1838–1845 (2012) 2, 3, 10
37. Zhou, S., Chen, Q., Wang, X.: Convolutional deep networks for visual data classification. Neural Processing Letters **38**(1), 17–27 (2013). 5