Semantic tracking: Single-target tracking with inter-supervised convolutional networks

Jingjing Xiao, Member, IEEE, Qiang Lan, Linbo Qiao, Aleš Leonardis, Member, IEEE

Abstract—This article presents a semantic tracker which simultaneously tracks a single target and recognises its category. In general, it is hard to design a tracking model suitable for all object categories, e.g., a rigid tracker for a car is not suitable for a deformable gymnast. Category-based trackers usually achieve superior tracking performance for the objects of that specific category, but have difficulties being generalised. Therefore, we propose a novel unified robust tracking framework which explicitly encodes both generic features and category-based features. The tracker consists of a shared convolutional network (NetS), which feeds into two parallel networks, NetC for classification and NetT for tracking. NetS is pre-trained on ImageNet to serve as a generic feature extractor across the different object categories for NetC and NetT. NetC utilises those features within fully connected layers to classify the object category. NetT has multiple branches, corresponding to multiple categories, to distinguish the tracked object from the background. Since each branch in NetT is trained by the videos of a specific category or groups of similar categories, NetT encodes category-based features for tracking. During online tracking, NetC and NetT jointly determine the target regions with the right category and foreground labels for target estimation. To improve the robustness and precision, NetC and NetT inter-supervise each other and trigger network adaptation when their outputs are ambiguous for the same image regions (i.e., when the category label contradicts the foreground/background classification). We have compared the performance of our tracker to other state-of-the-art trackers on a large-scale tracking benchmark [39] (100 sequences)—the obtained results demonstrate the effectiveness of our proposed tracker as it outperformed other 38 state-of-the-art tracking algorithms.

Index Terms—Single-target tracking, convolutional networks, semantic tracking

1 INTRODUCTION

Visual object tracking has actively been researched for several decades. Depending on the prior information about the target category, the tracking algorithms are usually classified as category-free methods, like KCF [14], Struck [13], LGT [30], and category-based methods, like human tracking [32], vehicle tracking [2], hand tracking [26]. The category-free tracking methods are acknowledged for their simple initialisation (a single bounding box) and easy generalisation across different object categories. They have extensively been studied and compared [39], [15]. However, as those methods have no prior information about the target inside the bounding box, the tracking performance heavily depends on the heuristic assumptions of image regions, i.e., appearance consistency [42] and motion consistency [5], which fail when those assumptions are not met. In contrast, the category-based methods benefit from the prior information about the target and can better adjust the target model and predict its dynamics or appearance variations during tracking. Those category-based methods can achieve superior performance on a specific category but usually have difficulties being generalised to other object categories. As many sophisticated machine learning algorithms have recently been adopted for tracking [21], [35], [38], an interesting question is whether we can build a semantic tracker, based on those methods, to bridge the gap between the category-free tracking methods and category-based tracking methods (see Tab. 1). Early attempts to track and recognise the objects simultaneously were done by [19], [9], [43]. However, the aforementioned works were developed using conventional hand-crafted features, which have difficulties of being scaled up. Inspired by the recent success of convolutional networks [16], we propose, in this article, a semantic tracker with a unified convolutional framework which encodes generic features across different object categories while also captures category-based features for model adaptation during tracking. With the help of the category-classification network, the semantic tracker can avoid heuristic assumptions about the tracked objects.

The proposed semantic tracker comprises three stages: off-line training, online tracking, and network adaptation. It consists of a shared convolutional network (NetS), a classification network (NetC) and a tracking network (NetT), see Fig. 1. In the off-line training stage, NetS is pre-trained from ImageNet to extract generic features across different object categories. Those features are then fed into NetC for classification and NetT for tracking. Note that NetT has multiple branches to distinguish the tracked objects.

- J. Xiao and A. Leonardis are with the University of Birmingham, United Kingdom, E-mail: shine636363@sina.com, a.leonardis@cs.bham.ac.uk
- L. Qiao and Q. Lan are with the College of Computer, National University of Defense Technology, China, E-mail: {lanqiang, qiao.linbo}@nudt.edu.cn
object from the background. Since each branch is trained by the videos of a specific object category, this enables each branch in NetT to learn the category-specific features related to both foreground and background, e.g., when tracking a pedestrian, it is more likely to learn the features of a car in the background than features of a fish. During online tracking, NetC first recognises the object category and activates the corresponding branch in NetT. Then, NetT is automatically fine-tuned for that particular tracking video by exploiting the foreground and the background sample regions in the first frame. When a new image frame arrives, the algorithm samples a set of image regions and each sample is fed through both NetC and NetT. The regions with the right category and the foreground label are used for target estimation (i.e., the location and the size of the target bounding box). Note that the target appearance often changes during the tracking, therefore it is extremely crucial for a tracker to adapt the model accordingly. To improve the robustness and precision, NetC and NetT inter-supervise each other and trigger network adaptation when their outputs are ambiguous (i.e., not consistent) for several image regions, e.g., when an image region is classified as a non-target category from NetC but as foreground from NetT or as a target category from NetC and background from NetT. The samples with consistent labellings are used to update the networks which also results in a reduced number of ambiguous sample regions. We have evaluated the contribution of each key component to the overall performance on OTB tracking benchmark [39] (100 sequences), and also compared the whole algorithm to the other state-of-the-art single-target tracking algorithms. The experimental results demonstrate the effectiveness of our algorithm as it outperformed other 38 state-of-the-art tracking algorithms not only overall, but also on the sub-datasets annotated with specific attributes.

Different from conventional category-free and category-based trackers, the main contributions of our semantic tracker can be summarised as:

1) Our tracker simultaneously tracks a single target and recognises its category using convolutional networks, which alleviates the problems with heuristic assumptions about the targets;
2) A novel unified framework with NetS network, which extracts generic features across different object categories, combined with NetC and NetT networks which encode category-based features;
3) NetC and NetT jointly determine image samples for estimation of the target, and inter-supervise each other by triggering network adaptation to improve robustness and precision.

The rest of the paper is organised as follows. We first review related work in Sec. 2. The details of the proposed method are provided in Sec. 3. Sec. 4 presents and discusses the experimental results on a tracking benchmark [39]. Sec. 5 provides concluding remarks.

2 RELATED WORK

Conventional tracking algorithms can be classified as category-based trackers and category-free trackers. Category-based tracking is targeted at some particular applications, e.g., Vondrak et al. [32] tracked a human body by considering physical plausibility, Oikonomidis et al. [26] tracked a hand with 26-DOF hand model, where Newtonian physics was applied to approximate the rigid-body motion dynamics. The mentioned works demonstrate that prior information about the target can significantly help the tracking algorithms to achieve more accurate and robust results. However, the existing category-based (articulate/rigid/dynamic) models and corresponding (physical/common-sense) constraints often suit that particular category and have difficulties being generalised. In contrast, category-free tracking is acknowledged for its simple initialisation (one bounding box) and easy generalisation across different object categories, as has extensively been demonstrated in [39], [15]. Early category-free trackers [25], [23], [6], [1] built the methods on a single feature, which is prone to failure when the applied feature endures large variations. To alleviate the problems of using a single feature, later works [40], [33], [42], [20] adaptively fused multiple features using sophisticated machine learning algorithms to build a target model to achieve robust tracking. However, in general, it is hard to design a model suitable for all different object categories, e.g., a rigid tracker for a car is not suitable for a deformable gymnast. Therefore, semantic information about the target category becomes essential to enable a tracker to optimize the model during tracking.

Recent works [35], [18], [21] began to exploit intrinsic information about the tracked objects, with an attempt to overcome the semantic gap and assist in developing robust tracking algorithms. Lee et al. [19], Fan et al. [9] and Yun and Jing [43] tried to track and recognise the objects simultaneously, however, these works were based on hand-crafted features, which hampered them to be scaled-up.

Inspired by the recent success of convolutional networks, Wang et al. [35] conducted an in-depth study on the properties of convolutional neural network features (CNN) [16] which showed that the top layers encode more semantic features and serve as category detectors, while lower layers carry more fine-grained details and can better discriminate the target from the background. Therefore, [35] jointly used those layers with a switch mechanism during the tracking. A similar work was done by Ma et al. [21], where they exploited CNN features [28] trained on ImageNet [8] to improve tracking accuracy and robustness. Different from [35], where the tracking algorithm was switching between the layers
with semantic information and fine-grained information, [21] fused features from hierarchical layers to conduct a coarse-to-fine tracking strategy. However, both trackers, [21], [35], were off-line pre-trained on ImageNet images [8] and then directly used for on-line tracking, without any online fine-tuning of the network structure for a specific tracking task. The realisation that purely using target images for training is not optimal since a target in one video can be part of the background in another, let to the use of videos to train the trackers. Wang et al. [34] pre-trained a two-layer CNN based tracker from video sequences, and proposed a domain adaptation method which effectively adapted the pre-learned features according to the specific target during online training. Wang et al [36] also proposed a sequence-trained network with generic feature extraction layers from VGG network [28] and two-layer adaptation network. A similar work was done by Nam et al. [24], who also proposed a video-trained CNN network with a shared network and multi-branches to distinguish the object from the background. However, all the mentioned video-trained trackers [34], [36], [24] did not explicitly exploit the semantic information of the target, i.e., object category. Without knowing the category of the object, it is highly probable that the tracker will learn false positives, and will have difficulties recovering from the failures. In addition, the afore mentioned trackers triggered the network adaptation in a heuristic way with pre-defined time intervals, causing inadequate adaptation which potentially resulted in either model drifting or outdated models. In contrast, our proposed semantic tracker significantly deviates from the aforementioned related works in several aspects including the network structure, initialisation procedure, target estimation and network online adaptation, summarised as: 1) we clearly define the shared network NetS for extraction of generic features, followed by the networks NetT and NetC for category-based features extraction. This also brings more intuitive understanding about what we have learnt in each network part; 2) NetT is explicitly trained with multiple branches encoding category-based features, where the corresponding branch is activated by classification network NetC; 3) the samples for the target estimation are jointly decided by the outputs from both NetC and NetT; 4) the network adaptation of NetC and NetT is conducted in an inter-supervised manner when their outputs for the same image region are in contradiction, i.e., a sample is classified by NetT as foreground but not correctly recognised by NetC or vice-versa; this step ensures a proper network updating pace, avoiding heuristics; 5) the proposed work simultaneously tracks the target and recognises its category.

3 THE PROPOSED TRACKER

In this section, we first introduce the structure of the proposed tracker model (Sec. 3.1). Then, we explain the off-line training process, which constructs the tracker using ImageNet [8] and tracking videos [15] (Sec. 3.2). The network initialisation, target estimation and network online adaptation are explained in Sec. 3.3.

3.1 Tracker model

Recent research has shown the relationship between the human vision system and deep hierarchies in computer vision [17]. CNNs, being partly inspired by these ideas, are acknowledged for their outstanding representation power and have extensively been studied in [16], [28]. Therefore, we also build our semantic tracker based on CNN components, but propose a new architecture illustrated in Fig. 2.

Recent research [21] has shown that shallow layers in CNN contain more generic information while deep layers are more related to semantic information. Thus, our tracker consists of shared convolutional layers to extract generic features in the shallow network (NetS), followed by NetC network for classification and NetT network for extracting category-based features for tracking. Note that NetS extracts generic features across different object categories, where those features have some common properties, e.g., robustness to scale and orientation changes, and illumination variations [24], which can be useful for other higher level tasks. Therefore, those extracted generic features are fed into NetC and NetT for more semantic related tasks. NetC is a multi-classification network to recognise the object category. NetT, which is a binary classification network, aims at distinguishing foreground region (target) from the background. Considering that the images of tracked objects of the same category often contain characteristic features both in terms of the foreground as well as the background, but which are different from other categories, e.g., when tracking a pedestrian it is more likely to have cars in the background than fish, NetT comprises multiple category-based branches, and each branch is particularly trained from the videos that contain the same object category. During on-line tracking, NetC and NetT inter-supervise each other by triggering network adaptation to improve robustness and precision, shown in Fig. 1. The details of the network structure are shown in Tab. 2.

3.2 Off-line training

NetS for generic features extraction. With extensive CNN-based studies for object classification, several representative models have been proposed and made publicly available, e.g., AlexNet [16], GoogleNet [29], VGGNet [28] etc. Rather than training the model from scratch, we transfer knowledge from a pre-trained model into NetS to extract generic features. A pre-trained model VGG-f [4] is explicitly chosen, because 1) it is trained from a tremendous dataset ImageNet [8]; 2) it achieves comparable performance with the fastest speed [31]. Our NetS has the same structure as the first three convolutional layers in VGG-f [4] except that the input image size is adapted (107*107). Since our training dataset is substantially smaller than ImageNet, the shared convolutional layers (NetS) are kept fixed to avoid the over-fitting problem.

NetC for classification. NetC aims at recognising the object’s category with two fully connected layers. When training NetC with our dataset, NetS first extracts generic features and those features are then fed into NetC network for fine-tuning. Note that the object in the video often undergoes significant deformations and suffers from a poor field of view and partial occlusions. In addition, the generated image samples during tracking might only cover the target partially or the target is not centralised inside the bounding boxes. Therefore, to improve the performance of our classification network NetC, we also prepared training samples with noisy bounding boxes, denoted as:

\[ X_{c,k}^n = X_k + \Delta X_{c,k}^n \]  

where \( X_k \) is the target ground truth at \( k \)-th frame, and \( \Delta X_{c,k}^n \) is the perturbation of the \( n \)-th sampled region \( X_{c,k}^n \). Specifically, we generated 50 object samples with significant overlap ratio (0.8) with the ground truth bounding boxes from each frame. To balance the distribution of different target status, those samples are shuffled during training. Note that NetC is trained as a multi-classification network to classify the object regions into different categories by
The structure of the proposed semantic tracker. In the convolutional layers, the first number indicates the receptive field size as “num x size x size”, followed by the convolution stride “str.”, spatial padding “pad”, local response normalisation “lrn”, and the max-pooling down-sampling factor.

| Structure | NetS shared network | NetC network |
|-----------|---------------------|--------------|
|            | conv1               | conv2         | conv3         |
| Input images (107*107) | 64*11*11, str.4, pad 0, lrn, *2 pool | 256*5*5, str.1, pad 2, lrn, *2 pool | 256*3*3, str.1, pad 1 |
|            | fc4 (64)           | fc4 (256)    | fc4 (256)    |
|            | fc5 (128)          | fc5 (64)     | fc5 (64)     |
|            | softmax            | softmax      | softmax      |

3.3 Online tracking

During the online tracking stage, the algorithm first takes several image regions around the target’s position in the previous frame, and feeds them into our network to estimate the target’s bounding box. NetS extracts the low-level generic features for NetC and NetT. Then NetC and NetT jointly determine the image regions for target estimation, and inter-supervise each other while updating.

**Initialisation.** Given a bounding box in the first frame, we apply the pre-trained NetS and NetC to assign the content of the bounding box to the corresponding NetT branch.

The whole process of the training procedure is explained below:

**Algorithm 1: off-line training**

1. **Input:** the categorised training sequences from VOT benchmark [15].
2. Prepare the training dataset \( \{ X_{n,k} \}_{n=1}^{N} \) for NetC (50 positive samples each frame) and \( \{ X_{n,k} \}_{n=1}^{N} \) for NetT (50 positive samples and 200 negative samples per frame).
3. Shuffle the whole NetC training dataset, and the NetT training datasets.
4. Train the NetC with the NetC training dataset by SGD, where the low level features are extracted from NetS.
5. Train the multi-branch NetT network with the NetT training datasets by SGD, where the low level features are also extracted from NetS.
6. **Output:** the weights and bias \( \langle W_t, B_t \rangle \) for the trained NetC network and \( \langle W_t, B_t \rangle \) for the NetT network.
in NetT. Note that the same type of the target (e.g., a car) can appear different in different videos, thus we need to fine-tune the activated branch in NetT for a particular tracking video. Therefore, the algorithm samples the image regions around the target for training based on the overlap with the ground truth. For positive (foreground) samples, we initially select 500 image regions with the overlap over 0.8 in the first frame. For negative (background) samples, we initially select 5000 image regions with the overlap below 0.2. Those samples, classified as other categories, will be treated as negative samples. The generated foreground and background samples are used to fine-tune NetT at the first frame through 30 iterations with the learning rate 0.001.

To improve the tracking accuracy, we need to train the model to estimate the size of the target and adjust the bounding box scale. This is achieved by learning the correspondence between the extracted features and the target size. Recent detection works [12], [27] have explored the regression capabilities of the rich hierarchical features, which separate the tasks of associating category and scale. This is achieved by learning the correspondence between the extracted features and the target size. Recent detection works [12], [27] have explored the regression capabilities of the rich hierarchical features, which separate the tasks of associating category and scale.

where \( \Delta \) position, denoted as:

\[
\begin{align*}
X^n = \hat{X}_{k-1} + \Delta X^n_k
\end{align*}
\]

where \( \hat{X}_{k-1} \) is the estimated target position at \( k \) - 1 frame, and \( \Delta X^n_k \) is the perturbation of the sampled region \( X^n_k \). \( \Delta X^n_k \sim \mathcal{N}(0, R) \) is a zero-mean Gaussian noise with a constant variance-covariance matrix \( R \).

Then, the tracker extracts generic features from each sample by NetS, and feeds those features into NetC for the classification (to determine the category) and NetT for the tracking (to determine foreground/background), denoted as:

\[
\begin{align*}
f_c(X^n_k) & : \text{NetS}(X^n_k) \rightarrow \text{NetC} \\
\end{align*}
\]

and

\[
\begin{align*}
f_t(X^n_k) & : \text{NetS}(X^n_k) \rightarrow \text{NetT} \\
\end{align*}
\]

where \( f_c(X^n_k) \) is the output of the image sample \( X^n_k \) from NetC network, and \( f_t(X^n_k) \) is the output of NetT network. Note that no matter how the target appearance changes, the category of the object should remain the same. Therefore, after NetC classifies the samples and assigns them category labels, only the samples labelled as the original category will be treated as potential target samples. The value of \( f_c(X^n_k) \) is 1 when the recognised content of the bounding box is consistent with the active branch in NetT. If not, the value becomes 0. The value of \( f_t(X^n_k) \) ranges between 0 and 1, which denotes the likelihood of the sample being a foreground sample. Since NetC and NetT simultaneously classify each sample, there are four different combinations of labels which guide the further process, shown in Tab. 3.

Samples classified as the original category from NetC and foreground from NetT are regarded as type I samples. Since type I samples obtain consistent (positive) labellings from NetC and NetT, they are regarded as highly trustable target samples and are used to estimate the target, defined as:

\[
\hat{X}^n_k = \arg \max f(X^n_k), \quad f(X^n_k) = f_c(X^n_k)f_t(X^n_k)
\]

Note that, to improve the robustness of the tracker, instead of using the sample with the highest score in Eq. 6, we choose \( N_{top} \) samples with highest scores for bounding boxes regression. The bounding box regressors learnt in the initialization stage (Eq. 3) are applied to estimate the object scale from selected \( n \)-th image region \( X^n_k \):

\[
\begin{align*}
\tilde{X}^{n}_{x,y} &= g_f(\text{NetS}(\hat{X}^{n}_{k,x})), \tilde{X}^{n}_{k,y} = g_f(\text{NetS}(\hat{X}^{n}_{k,y})), \cr
\tilde{X}^{n}_{k,w} &= \exp(g_f(\text{NetS}(\hat{X}^{n}_{k,w}))) \ast \hat{X}^{n}_{k,w}, \cr
\tilde{X}^{n}_{k,h} &= \exp(g_f(\text{NetS}(\hat{X}^{n}_{k,h}))) \ast \hat{X}^{n}_{k,h} \\
\end{align*}
\]

where subscripts \( x, y, w, h \) have the same meaning as in Eq. 3 for the selected bounding box \( \hat{X}^n_k \) at frame \( k \). The final estimation of the target \( \hat{X}^n_k \) utilises the expectation operator over the rescaled samples \( \tilde{X}^n_k \) computed by Eq. 7, denoted as:

\[
\hat{X}^n_k = \frac{1}{N_{top}} \sum_{n=1}^{N_{top}} f(X^n_k)\tilde{X}^n_k
\]

where \( f(\hat{X}^n_k) \) is the score computed from Eq. 6. \( N_{top} \) is the number of selected Type I samples with highest scores.

### Inter-supervised network adaptation

To handle appearance variations of the target during tracking, it is important to be able to update the NetC and NetT networks accordingly. There are two essential questions about the network adaptation: 1) when to update and 2) how to update. Ideally, NetC and NetT should obtain consistent conclusions about the same image region, that means that a foreground region should also have the right category label. If not, such ambiguous situations indicate that NetC and NetT need to be re-trained with the newest samples, at which point the network adaptation is triggered.

Note that the type IV samples (the same as the type I samples in Tab. 3) also obtain consistent labellings (in the case of the type IV they are negative) from both networks. Those samples with consistent labellings are used for later network adaptation when ambiguities occur as a result of NetC and NetT outputting contradictory results (type II and type III samples). As shown in Tab. 3, the algorithm detects ambiguous samples (AS) when inconsistent labellings arise from the outputs of NetC and NetT, i.e., type II and type III samples. An increasing number of AS
indicates that the current networks have difficulties consistently classifying the incoming samples and should be updated. Since NetC is not thoroughly pre-trained with fine-grained information, it may misclassify the object under some (new) conditions. Also, the initially trained foreground/background boundary of NetT may not be reliable any more. Therefore, both NetC and NetT need to be updated with the most recent consistent samples. To update the networks, NetC and NetT use the consistent samples during the process, i.e., type I and type IV samples. While it is straightforward to use type I and type IV samples to update NetT, type IV samples do not have a validated category label to train a category X (unknown category, explained in Sec. 4.1.1) to update NetC, denoted as:

\[
\begin{align*}
< W_c, \hat{B}_c > &= \operatorname{arg\,min} \frac{1}{N_{tr}} \sum_{n=1}^{N_u} ||f_c(X_{tr,k}^n) - l_{c,k}^n||_2 \\
< W_t, \hat{B}_t > &= \operatorname{arg\,min} \frac{1}{N_{tr}} \sum_{n=1}^{N_u} ||f_t(X_{tr,k}^n) - l_{t,k}^n||_2
\end{align*}
\]

(9)

where \( < W_c, \hat{B}_c > \) and \( < W_t, \hat{B}_t > \) are the weights and biases of NetC and NetT, \{\( X_{tr,k}^n \)\}\( _{n=1,...,N_{tr}} \) are the type I and type IV samples used for training, \( l_{c,k}^n \) and \( l_{t,k}^n \) are the corresponding ground truth labels. After one round of adaptation, the updated NetC and NetT will jointly be used to classify the ambiguous samples again. The newly classified type I or IV samples originating from previous AS will be added into the training pool for the next training iteration. It is expected that the newly trained networks NetC and NetT will produce increasingly consistent labellings for the image regions, which indeed happens, as the number of ambiguous samples is reduced by updated networks. Therefore, we use this as the stopping criterion for the adaptation, i.e., when the number of AS stops decreasing or is sufficiently small (0.2 in practice). The process of online tracking is explained below:

Algorithm 2: online tracking
1: **Input**: the ground truth of the target in the first frame.  
2: Initialise the tracker by recognising the target’s category with NetC, activating corresponding branch in NetT and fine-tuning the NetT network with image regions.  
3: Train the bounding box regressors, Eq. 3.  
4: **For** frame = 2: \( N_f \)  
5: Generate candidate images samples with respect with Eq. 4  
6: Categorise each sample with NetC network and classify the samples into the foreground and background with NetT network.  
7: Choose image samples in terms of Eq. 6 for estimation.  
8: Estimate the target position and scale, Eq. 7, Eq. 8.  
9: Calculate the number of AS samples \( N_{AS} \).  
10: **While** \( N_{AS} > \) threshold  
11: Fine-tune the NetC and NetT with type I and type IV samples.  
12: Categorise each sample with NetC network and NetT network.  
13: Calculate the number of AS samples \( N_{AS} \).  
14: **End**  
15: **End**  
16: **Output**: the estimated object position and scale.

### 4 Experimental Results

In this section, we first explain the implementation details of the tracker. Then, we evaluate the tracker from four aspects: the effectiveness of the tracker sub-components, the qualitative performance compared to other CNN-based trackers, the quantitative performance compared to all other state-of-the-art trackers and the failure cases of the proposed tracker. Finally, we present some ideas for future work 1.

#### 4.1 Implementation details

In this section, we provide the details about the datasets, evaluation metrics, as well as training and running speed.

##### 4.1.1 Datasets

**Training dataset** - To train the tracker we use the sequences from VOT [15], explicitly excluding the sequences that also appear in OTB [39], which is used as the test dataset. The training dataset was, for the purpose of constructing NetT branches, classified into 8 categories according to the tracked objects, namely, pedestrians, faces, cars, animals, balls, motorbikes, dolls and a category X (which comprises of the targets that do not fall into any of the 7 categories).

**Test dataset** - The algorithm is tested on a large scale tracking benchmark OTB [39] which has 100 sequences, and each sequence has several tracking attributes to facilitate evaluation. The features of the training dataset and the test dataset are listed in Tab. 4.

##### 4.1.2 Evaluation metrics

We report the results of one pass evaluation (OPE) based on the evaluation protocol proposed in OTB [39]. Note that there are two criteria used in the OTB, namely overlap and centre-error. In our experiment, we only use the overlap (success plot) rather than the centre-error (precision plot) in tracking evaluation since the centre distance is: 1) susceptible to subjective bounding box annotations; 2) unreliable in cases when a tracker completely loses a target [3]. Therefore, we use the area under curve (AUC) of the success plot to rank the trackers.

The overlap \( \varphi_k \) at frame \( k \) is defined by using the tracker-output bounding box \( \hat{X}_k \) and ground-truth bounding box \( X_k^G \)

in Eq. 10:

\[
\varphi_k = \frac{|\hat{X}_k \cap X_k^G|}{|\hat{X}_k \cup X_k^G|}
\]

where \( \cap \) and \( \cup \) represent the intersection and union of two regions and \( |\bullet| \) is the region size measured by pixels number.

In the success plot, the x-axis depicts a set of thresholds for the overlap to indicate the tracking success. The success ratio is the

1. The code will be released upon acceptance of the paper.
number of correctly tracked frames divided by the total number of frames for a more comparable evaluation, Eq. 11.

\[ P_\tau(\hat{X}_k, X^G_k) = \frac{\#\{k|\varphi_k > \tau \}^N_{k=1}}{N_f} \]  

where \( \tau \) denotes the threshold of the overlap, and \( N_f \) is the total number of frames. A failure is detected when the overlap (computed in Eq. 10) is below the defined threshold \( \tau \).

### 4.1.3 Speed
The proposed algorithm was implemented in Matlab2014a (linked to some C components) using an Intel i7-4710MQ CPU and Nvidia Quadro K1100M GPU, giving the average training speed of 289.5 bbps (bounding boxes per second) and the test speed of 189.2 bbps.

### 4.2 Evaluation of the sub-components of the tracker
In this section, we describe how we evaluated the contributions of the key components of the proposed method (i.e., NetC, NetT branches and adaptation) to the overall performance. In the first experiment, we designed our baseline algorithm to only apply the shared network NetS which fed into one branch of NetT. Since NetC was not used to classify the tracked category, the branch of pedestrian category in NetT was manually chosen as the pedestrian category dominates the test dataset. Note that the baseline algorithm fine-tunes NetT based on the initial bounding box. In the second experiment, we combined the baseline model with NetC to activate the corresponding (category-based) branch in NetT. In this stage, we also adapt the triggered NetT in the first frame while no inter-supervised adaptation takes place between the networks during tracking. This experiment shows how much the semantic (category) information can improve the performance. Finally, we performed the experiment with enabled inter-supervision between NetC and NetT to observe further improvements of the performance, as shown in Fig. 3.

It is interesting to note that the baseline algorithm which uses the pedestrian branch of the NetT network for all testing videos (64% of the sequences, in fact, belong to other, non-pedestrian categories) still shows a relatively strong performance. For example, despite using a non-optimal NetT branch (i.e., pedestrian) for most of the sequences, it still performs favourably compared to DST [41] (0.498, ranked 6th) and DSST [7] (0.475, ranked 7th) in the overall evaluation. This relatively strong performance can be attributed to the NetT fine-tuning initialisation step which adapts the branch for a particular tracking video. Adding NetC to the baseline algorithm results in significant improvements, which demonstrates the effectiveness of the semantic information. This can also be observed in Fig. 3 (bottom), which shows that for a deforming target, the baseline tracker gradually drifts to the background while both NetC enhanced baseline algorithm and the full algorithm can track the diver robustly. The adaptation process, by inter-supervision between NetC and NetT further advances the overall performance (shown in Fig. 3 plots).

### 4.3 Qualitative comparison among CNN-based trackers
We compare our tracker to other methods [21], [37] which also have the same major component, i.e., CNN, as our proposed semantic tracker. Ma et al. [21] utilised the pre-trained VGG model (from ImageNet) as a feature extractor, together with the kernelised correlation filter tracking framework. Since HCF tracker [21] only utilised the off-line trained model, a comparison between our work and HCF demonstrates the effectiveness of the online learning part for the proposed tracker. Note that the scale of HCF tracker [21] is not adapted, thus this comparison also shows the advantages of applying the bounding box adaptation for our tracker. Different from HCF [21], Wang et al. [37] utilised the

---

**TABLE 4**
The features of the training dataset and the test dataset. The training dataset is obtained from VOT [15], explicitly excluding test sequences. The test dataset [39] consists of 100 sequences.

| Categories | Pedestrians | Faces | Cars | Animals | Balls | Motorbikes | Dolls | Category X |
|------------|-------------|-------|------|---------|-------|------------|-------|------------|
| Training set | No.of Seq | 17 | 3 | 6 | 13 | 4 | 3 | 1 | 11 |
| No. of Frames | 5975 | 441 | 3216 | 5412 | 949 | 695 | 326 | 3110 |
| Test set | No.of Seq | 36 | 23 | 12 | 5 | None | 2 | 7 | 15 |
| No. of Frames | 16258 | 11306 | 11223 | 1705 | None | 392 | 8893 | 9263 |
CNN for online learning which also distinguished the foreground target from a background like our NetT network. A comparison to DLT [37] (its performance is shown in Fig. 4) demonstrates a superior performance of our tracker due to the semantic information and inter-supervised network adaptation jointly from NetC and NetT.

In the sequences containing objects with significant scale variations, e.g. freeman4, doll, HCF [21] tracker fails in tracking the object accurately. This is because HCF tracker cannot adapt the scale of the template. In contrast, our approach which implements scale adaptation can successfully deal with this problem. Note that HCF still exhibits the advantage of using the sophisticated features learned from ImageNet in the sequence skiing, compared to the online trained DLT [37] tracker. This is because DTL tracker online trains the network purely based on the tracking results without additional supervision. When the target appearance changes dramatically, e.g., significant illumination conditions in sequence singer2 and a partial occlusion in lemming, DTL tracker will gradually learn the background information and incorporate it into the model which will finally result in a failure. In contrast, our tracker benefits from the semantic knowledge about the target category, which provides more reliable training data to update the network in a robust way.

4.4 Overall performance comparison
We evaluated our proposed tracker by comparing it to 29 original trackers in OTB [39] and additional 9 recently published trackers, namely, CCT [45], LCT [22], KCF [14], MEEM [44], DSST [7], TGPR [11], DST [41], and CNN-based trackers HCF [21] and DLT [37]. The AUC score of the top 10 trackers in terms of the success plots are shown in Tab. 5, which shows the results obtained on 1) the whole dataset and 2) sub-datasets annotated with specific attributes, i.e., deformation (39 sequences), scale variation (61 sequences), illumination variation (35 sequences), low resolution (9 sequences), out-of-view (14 sequences) and fast motion (37 sequences). As shown in Tab. 5, the proposed semantic tracker outperforms all other 38 state-of-the-art trackers, not only overall, but also on the sub-datasets annotated with specific attributes, namely IV, SV, DEF, FM, OV and LR.

4.5 Failure cases
It is also important to identify and analyse the failure cases of the designed algorithm. We show two such examples in Fig. 5. Even though our tracker has achieved superior performance both overall and on the sub-sequences with annotated attributes, it still has difficulties tracking objects in scenes with camouflage. In such cases, semantic information only about the target itself is not sufficient to distinguish the object from the background which has an identical appearance as the target. To tackle these types of problems, the tracker should also exploit the semantic information contained in the scene [41].

5 Conclusions
In this paper, we proposed a new single target semantic tracker which intertwines the processes of target classification and target tracking. This is achieved by a novel network structure which comprises of different CNNs, i.e., a shared convolutional network (NetS), a classification network (NetC) and a tracking network (NetT). These networks are trained to encompass both generic features and category-specific features. During online tracking, consistent outputs of NetC and NetT jointly determine the sample regions with the right category and foreground labels for target estimation, while inconsistencies in the outputs of NetC and NetT trigger adaptation of the networks. The extensive experiments have shown that our tracker outperforms 38 state-of-the-art tracking algorithms tested on a large scale tracking benchmark OTB [39] with 100 sequences. Note that our current work only considers the semantic information of the objects, and that a lack of contextual semantic information may cause tracking difficulties/failures in highly cluttered scenes or when tracking objects without distinguishing features, such as translucent objects, as mentioned in [41]. Therefore, in future, we will also exploit contextual semantic information and improve the performance of the tracker in cases of camouflage. In addition, our ongoing work will also focus on scaling up the proposed semantic tracker to a larger number of categories. This requires the tracker to construct multi-branches of NetT network in a more automatic, self-organised, way.

ACKNOWLEDGEMENT
We acknowledge MoD/Dstl and EPSRC for providing the grant to support the UK academics (Ales Leonardis) involvement in a Department of Defense funded MURI project. This work was also supported by EU H2020 RoMaNS 645582 and EPSRC EP/M026477/1.

REFERENCES
[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In CVPR, 2006.
[2] X. Cao, C. Gao, J. Lan, Y. Yuan, and P. Yan. Ego motion guided particle filter for vehicle tracking in airborne videos. Neurocomputing, 2014.
[3] L. Čehovin, A. Leonardis, and M. Kristan. Visual object tracking performance measures revisited. IEEE TIP, 2016.
[4] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In BMVC, 2014.
[5] J.-N. Chi, C. Qian, P. Zhang, W. Xiao, and L. Xie. A novel elm based adaptive kalman filter tracking algorithm. Neurocomputing, 2014.
[6] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In CVPR, 2000.
[7] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In BMVC, 2014.
[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In CVPR, 2009.
[9] J. Fan, X. Shen, and Y. Wu. What are we tracking: a unified approach of tracking and recognition. IEEE TIP, 22(2):549–560, 2013.
[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. IEEE TPAMI, 32(9):1627–1645, 2010.
[11] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with gaussian processes regression. In ECCV, 2014.
[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.
[13] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In ICCV, 2011.
[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. IEEE TPAMI, 2015.
[15] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin. A novel performance evaluation methodology for single-target trackers. IEEE TPAMI, 2014.
[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.
[17] N. Krueger, P. Janssen, S. Kalkan, M. Laptev, A. Leonardis, J. Piater, A. J. Rodriguez-Sanchez, and L. Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? IEEE TPAMI, 35(8):1847–1871, 2013.
Fig. 4. Qualitative results of the CNN based trackers (red: ours; yellow: HCF [21]; blue: DLT [37].) on sequences: (a) freeman4; (b) doll; (c) skiing; (d) singer2; (e) lemming.

Fig. 5. Two examples of failure cases of the semantic tracker. The blue bounding boxes indicate the (annotated) ground truth, while the red bounding boxes were output by our semantic tracker.

[18] K. Lebeda, S. Hadfield, and R. Bowden. Exploring causal relationships in visual object tracking. In ICCV, 2015.
[19] K.-C. Lee, J. Ho, M.-H. Yang, and D. Kriegman. Visual tracking and recognition using probabilistic appearance manifolds. Computer Vision and Image Understanding, 99(3):303–331, 2005.
[20] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. IEEE TIP, 25(4):1834–1848, 2016.
[21] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In ICCV, 2015.
[22] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In CVPR, 2015.
[23] X. Mei and H. Ling. Robust visual tracking using L1 minimization. In ICCV, 2009.
[24] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In CVPR, 2016.
[25] K. Nummiaro, E. Koller-Meier, and L. Van Gool. An adaptive color-based particle filter. Image and vision computing, 2003.
[26] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Tracking the articulated motion of two strongly interacting hands. In CVPR, 2012.
[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. 2016.
[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2014.
TABLE 5
The AUC score of OPE [39] success plots for the top 10 compared trackers. The best tracker is in bold, while the second best is denoted with *. IV: illumination variation; OPR: out-of-plane rotation; SV: scale variation; OCC: occlusion; DEF: deformation; MB: motion blur; FM: fast motion; IPR: in-plane rotation; OV: out of view; BC: background clutter; LR: low resolution.

| Tracker | Overall | IV | OPR | SV | OCC | DEF | MB | FM | IPR | OV | BC | LR |
|---------|---------|----|-----|----|-----|-----|----|----|-----|----|----|----|
| Ours    | 0.572   | 0.577 | 0.544* | 0.562 | 0.514 | 0.573 | 0.577* | 0.589 | 0.532 | 0.516 | 0.516 | 0.568 |
| HCF     | 0.562*  | 0.540 | 0.531 | 0.486 | 0.520 | 0.525* | 0.585 | 0.578* | 0.559 | 0.474 | 0.585 | 0.388 |
| LCT     | 0.562   | 0.556* | 0.547 | 0.500* | 0.515* | 0.507 | 0.533 | 0.560 | 0.557* | 0.452 | 0.530* | 0.399 |
| CCT     | 0.549   | 0.533 | 0.519 | 0.486 | 0.482 | 0.514 | 0.541 | 0.571 | 0.516 | 0.430 | 0.521 | 0.432* |
| MEEM    | 0.530   | 0.521 | 0.530 | 0.479 | 0.512 | 0.496 | 0.556 | 0.557 | 0.529 | 0.488* | 0.519 | 0.382 |
| DST     | 0.498   | 0.456 | 0.461 | 0.402 | 0.461 | 0.498 | 0.512 | 0.513 | 0.491 | 0.437 | 0.487 | 0.318 |
| DSSST   | 0.475   | 0.486 | 0.465 | 0.414 | 0.446 | 0.333 | 0.407* | 0.452 | 0.485 | 0.374 | 0.477 | 0.314 |
| KCF     | 0.475   | 0.474 | 0.463 | 0.396 | 0.456 | 0.455 | 0.459 | 0.465 | 0.465 | 0.393 | 0.497 | 0.290 |
| Struck  | 0.459   | 0.430 | 0.435 | 0.406 | 0.405 | 0.403 | 0.403 | 0.406 | 0.409 | 0.447 | 0.359 | 0.327 |
| TGPR    | 0.458   | 0.448 | 0.456 | 0.405 | 0.400 | 0.405 | 0.403 | 0.429 | 0.421 | 0.462 | 0.373 | 0.428 |

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015.
[30] L. Čehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. IEEE TPAMI, 2013.
[31] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In Proceedings of the 23rd ACM International Conference on Multimedia, pages 689–692. ACM, 2015.
[32] M. Vondrak, L. Sigal, and O. C. Jenkins. Dynamical simulation priors for human motion tracking. IEEE TPAMI, 2013.
[33] D. Wang, H. Lu, Z. Xiao, and M.-H. Yang. Inverse sparse tracker with a locally weighted distance metric. IEEE TIP, 24(9):2646–2657, 2015.
[34] L. Wang, T. Liu, G. Wang, K. L. Chan, and Q. Yang. Video tracking using learned hierarchical features. IEEE TIP, 24(4):1424–1435, 2015.
[35] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In ICCV, 2015.
[36] L. Wang, W. Ouyang, X. Wang, and H. Lu. Stct: Sequentially training convolutional networks for visual tracking. CVPR, 2016.
[37] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In NIPS, 2013.
[38] X. Wang, M. Valstar, B. Martinez, M. H. Khan, and T. Pridmore. Trictrack: Tracking by regression with incrementally learned cascades. In ICCV, 2015.
[39] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. IEEE TPAMI, 2015.
[40] Y. Wu, M. Pei, M. Yang, J. Yuan, and Y. Jia. Robust discriminative tracking via landmark-based label propagation. IEEE TIP, 24(5):1510–1523, 2015.
[41] J. Xiao, L. Qiao, R. Stolkin, and A. Leonardis. Distractor-supported single target tracking in extremely cluttered scenes. In ECCV, 2016.
[42] J. Xiao, R. Stolkin, and A. Leonardis. Single target tracking using adaptive clustered decision trees and dynamic multi-level appearance models. In CVPR, 2015.
[43] X. Yun and Z.-L. Jing. Kernel joint visual tracking and recognition based on structured sparse representation. Neurocomputing, 193:181–192, 2016.
[44] J. Zhang, S. Ma, and S. Sclaroff. Meem: Robust tracking via multiple experts using entropy minimization. In ECCV, 2014.
[45] G. Zhu, J. Wang, Y. Wu, and H. Lu. Collaborative correlation tracking. In BMVC, 2015.

Qiang Lan received the M.S. and B.S. degrees in computer science from the National University of Defense Technology (NUDT), Changsha, China. He continues his PhD degree in computer science and technology in NUDT. His research topics are about high performance computing and computation optimization in convolutional neural network.

Linbo Qiao received the M.S. and B.S. degrees in computer science from the National University of Defense Technology (NUDT), Changsha, China, where he is currently pursuing the PhD degree in computer science and technology. His research interests include machine learning, online and distributed computation.

Aleš Leonardis is Professor at the School of Computer Science, University of Birmingham and co-Director of the Centre for Computational Neuroscience and Cognitive Robotics. He is also Professor at the FCS, University of Ljubljana and adjunct professor at the FCS, TU-Graz. His research interests include robust and adaptive methods for computer vision, object and scene recognition and categorization, statistical visual learning, 3D object modelling, and biologically motivated vision.

Jingjing Xiao received her Bachelor and Master degree from College of Mechatronics Engineering and Automation, National University of Defense Technology, China, in 2010 and 2012, respectively. She holds a PhD degree from University of Birmingham in 2016. She received the best poster award in the BMVA summer school in 2014. Currently, she is a research fellow in the University of Birmingham, U.K.. Her research interests include single object tracking, multi-object tracking with computer vision.