Learning a Visual Tracker from a Single Movie without Annotation

Lingxiao Yang, David Zhang, Lei Zhang
Department of Computing, The Hong Kong Polytechnic University, Hong Kong
{cslyang,csdzhang,cslzhang}@comp.polyu.edu.hk

Abstract

The recent success of deep network in visual trackers learning largely relies on human labeled data, which are however expensive to annotate. Recently, some unsupervised methods have been proposed to explore the learning of visual trackers without labeled data, while their performance lags far behind the supervised methods. We identify the main bottleneck of these methods as inconsistent objectives between off-line training and online tracking stages. To address this problem, we propose a novel unsupervised learning pipeline which is based on the discriminative correlation filter network. Our method iteratively updates the tracker by alternating between target localization and network optimization. In particular, we propose to learn the network from a single movie, which could be easily obtained other than collecting thousands of video clips or millions of images. Extensive experiments demonstrate that our approach is insensitive to the employed movies, and the trained visual tracker achieves leading performance among existing unsupervised learning approaches. Even compared with the same network trained with human labeled bounding boxes, our tracker achieves similar results on many tracking benchmarks. Code is available at: *https://github.com/ZjjConan/UL-Tracker-AAAI2019*.

Introduction

Generic object tracking has attracted considerable attention in computer vision since it has been successfully applied into many domains, including human-computer interaction, video surveillance, and unmanned aerial vehicle, to name a few. This paper focuses on the problem of single object tracking, whose goal is to automatically estimate a trajectory of a single target over a video. The target is often annotated as a rectangle in the first frame.

Recent advances in object tracking are driven by the development of Discriminative Correlation Filters (DCFs) (Bolme et al. 2010; Henriques et al. 2015; Danelljan et al. 2015; Fan and Xiang 2017). DCF based methods begin by learning a template of target object and then apply the template to detect target location in subsequent frames. The template is usually updated over time to adapt the appearance changes of the tracked object. DCFs are very suitable for fast tracking because they only need to solve a ridge regression problem, whose solution can be efficiently obtained in the Fourier domain with a few element-wise operations.

Early DCF trackers often use hand-crafted descriptors to describe the tracked target. Inspired by the great success of deep neural networks (Krizhevsky, Sutskever, and Hinton 2012; Chatfield et al. 2014; Simonyan and Zisserman 2014) in many vision tasks, hand-crafted descriptors have been replaced by pre-trained deep features to improve the tracking performance of DCFs (Danelljan et al. 2017). More recently, some works (Valmadre et al. 2017; Wang et al. 2017) reformulated DCFs as a particular layer in deep networks, resulting in an end-to-end learning framework. Highly competitive tracking results have been reported on tracking datasets such as OTB (Wu, Lim, and Yang 2013; 2015) and VOT (Kristan, Matas, and Leonardis 2015).

The pre-trained network plays the key role for achieving robust and accurate tracking performance. A well trained network is expected to mitigate the overfitting issue in visual tracking, for which only a single labeled data is available. However, this often requires a significant amount of human labeled data for training, which are expensive to collect. For example, the widely utilized networks, VGG-M (Chatfield et al. 2014) and VGG-16 (Simonyan and Zisserman 2014), are both pre-trained on large scale static image database, *i.e.,* ImageNet (Deng et al. 2009), with human annotated object labels, while the Siamese networks (Bertinetto et al. 2016; Wang et al. 2017; Valmadre et al. 2017; Li et al. 2018) are pre-trained on ILSVRC2015 videos (Russakovsky et al. 2015) with given object bounding boxes.

It is interesting to investigate whether we can train a good visual tracking network without using human annotations. In recent years, some unsupervised learning approaches (Wang and Yeung 2013; Ma et al. 2015b; Wang et al. 2012) have been reported to achieve this goal. These trackers, however, do not exhibit comparable performance with those trackers trained by supervised methods. The main problem is that these trackers are optimized from an auxiliary task, in which the main objective is not the same as that in the online tracking process. More specifically, all these methods learn to reconstruct input signals, and then transfer the learned network to visual tracking as either a binary classification (Wang and Yeung 2013; Wang et al. 2012) task or a target localization (Ma et al. 2015b) task. Clearly, those trackers trained from different tasks are suboptimal for the visual tracking task.

In this paper, we explore the question that *whether a robust visual tracker can be pre-trained in an unsupervised manner using an objective similar to online tracking process with data that can be easily accessed?* We propose to off-line learn a visual tracker using a self-tracking strategy from a single movie. Our method is based on discriminative correlation filter network (Wang et al. 2017) and starts with extracting object-like region proposals (Uijlings et al. 2013) from each frame. Secondly, we automatically segmented frames into shot clips by detecting scene cuts over a movie. The detection is based on cross correlation score between two consecutive frames. Thirdly, we incrementally optimize the network by repeatedly alternating between tracking and learning, leading to a stable and accurate off-line learned network. We train our tracker from a single movie because movies contain many static and motion cues which are very helpful for visual understanding. In addition, a movie could be easily obtained compared with preparing thousands of videos (Russakovsky et al. 2015) or millions of images (Deng et al. 2009). The learnt network can be easily transferred for online tracking as standard DCF trackers.

Related Work

In this section, we briefly discuss the related work, including supervised and unsupervised pre-training for tracking, as well as using movies for vision applications. For a completed reviews, we kindly refer readers to the recent surveys (Smeulders et al. 2014; Wu, Lim, and Yang 2015).

Supervised pre-training. The pre-trained deep networks such as VGG-M (Chatfield et al. 2014) and VGG-16 (Simonyan and Zisserman 2014) have been widely adopted for visual tracking. Usually, the networks are pre-trained on some large scale well-labeled datasets (Deng et al. 2009; Russakovsky et al. 2015) using supervised learning. For example, CNN-SVM (Hong et al. 2015) takes output from the first fully-connected layer of a pre-trained CNN (Girshick et al. 2014) as inputs, and builds a SVM on those features to distinguish between target object and background. Instead of using SVM on top of CNN features for visual tracking, DCFs (Ma et al. 2015a; Danelljan et al. 2017) have also been coupled with CNN features, as well as fully convolutional networks (Wang et al. 2015). Furthermore, many works focus on the end-to-end learning pipeline to improve tracking accuracy and robustness. Typical examples include pre-training Siamese networks for object verification (Tao, Gavves, and Smeulders 2016), target localization (Bertinetto et al. 2016; Valmadre et al. 2017; Wang et al. 2017) and axis prediction (Held, Thrun, and Savarese 2016). Our work differs from these approaches in that we train the visual tracker without human annotations.

Unsupervised pre-training. Some works explore unsupervised pre-training for visual tracking. For example, The work (Wang et al. 2012) learns an overcomplete dictionary to represent visual prior for tracking, while the works (Wang and Yeung 2013) and (Ma et al. 2015b) learn neural networks from natural images (Torralba, Fergus, and Freeman 2008) and movies (Cadieu and Olshausen 2009) respectively. Our work shares a similar philosophy by pre-training a network without human annotations, but has following two differences: (1) we formulate the procedure as a tracking and learning task, ensuring that the learning objective is similar to that of the online tracking process; (2) we learn the network from a single movie, and obtain state-of-the-art results compared with previous unsupervised learning methods.

Movies for vision applications. Movies have been utilized for various vision researches. For example, Cadieu et al. (Cadieu and Olshausen 2009) demonstrated that the usual movies can be used to learn visual transformations. Ivan et al. (Laptev et al. 2008) propose to recognize human action in natural movies. Besides, several applications for movie understanding have been proposed, including movie description (Rohrbach et al. 2015) and movie question answering (Tapaswi et al. 2016). Contrary to these works, we aim at training a tracker from a single movie.

Discriminative Correlation Filter Network

We base our approach on DCFNet, which has shown to provide a good balance between accuracy and speed and is publicly available. To be self-contained, we briefly introduce DCFNet (Wang et al. 2017) as follows.

Standard DCFs learn a group of convolutional filters $\boldsymbol{w}$ from training samples $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ by:

$$arg\min_{\boldsymbol{w}} \Sigma_{i=1}^{N} ||\boldsymbol{y}_i - \Sigma_{d=1}^{D} \boldsymbol{w}^d \star \phi^d(\boldsymbol{x}_i)||^2 + \lambda \Sigma_{d=1}^{D} ||\boldsymbol{w}^d||^2 \tag{1}$$

where ϕ is a feature function such as a CNN. D is the feature channels, and $\star$ is circular correlation. Label $\boldsymbol{y}_i$ is a Gaussian shaped matrix, with the true object location corresponding to the highest value. The solution to Eq. (1) can be efficiently gained in the Fourier domain with a few Fast Fourier Transform and element-wise operations, as shown below:

$$\boldsymbol{w}^d = \mathcal{F}^{-1}\left(\frac{\hat{\boldsymbol{y}}^* \odot \hat{\phi}^d(\boldsymbol{x})}{\Sigma_{d=1}^{D} \hat{\phi}^d(\boldsymbol{x}) \odot (\hat{\phi}^d(\boldsymbol{x}))^* + \lambda}\right) \tag{2}$$

where the hat symbol and $*$ represent the discrete Fourier transform $\mathcal{F}$ and complex conjugate of given variables, respectively. $\odot$ is the Hadamard product. The learned filters $\boldsymbol{w}$ are applied to a new frame to detect a target by:

$$\boldsymbol{g} = \mathcal{F}^{-1}(\Sigma_{d=1}^{D} \hat{\phi}^d(\boldsymbol{z}) \odot \hat{\boldsymbol{w}}^{d*}) \tag{3}$$

DCFNet optimizes all parameters by the following loss:

$$L = ||\tilde{\boldsymbol{g}} - \boldsymbol{g}||^2 + \gamma ||\boldsymbol{\theta}||^2 \tag{4}$$

where $\tilde{\boldsymbol{g}}$ is the same to $\boldsymbol{y}$. $\boldsymbol{\theta}$ groups all parameters in ϕ. $\boldsymbol{\theta}$ can be trained in an end-to-end manner if the above loss L can be back propagated to the feature extractor ϕ. Fortunately, all components in DCFNet are differentiable. The following two formulations show the back-propagation processes with respect to $\phi(\boldsymbol{x})$ and $\phi(\boldsymbol{z})$:

$$\frac{\partial L}{\partial \phi^d(\boldsymbol{x})} = \mathcal{F}^{-1}\left(\frac{\partial L}{\partial (\hat{\phi}^d(\boldsymbol{x}))^*} + \left(\frac{\partial L}{\partial (\hat{\phi}^d(\boldsymbol{x}))}\right)^*\right) \tag{5}$$

$$\frac{\partial L}{\partial \phi^d(\boldsymbol{z})} = \mathcal{F}^{-1}\left(\frac{\partial L}{\partial (\hat{\phi}^d(\boldsymbol{z}))^*}\right) \tag{6}$$

Figure 1: Examples of (a) two removed frames, (b) two grouped short clips, and (c) some region proposals in two frames.

Unsupervised Learning of DCFNet

Denote $\mathcal{D} := (\boldsymbol{x}_i, \boldsymbol{z}_i)_{i=1}^{N}$ as a series of pair regions to train DCFNet, where N is the number of pairs. In the case of supervised learning, the training database $\mathcal{D}$ can be easily obtained by cropping regions from image pairs according to the human provided bounding boxes. Other labeled information such as instance ID or occlusion can also be used to refine $\mathcal{D}$ by removing some low quality region pairs caused by disappeared or partial occlusion. In the case of unsupervised learning, however, none of above mentioned labeling information exist. Therefore, the key challenges to learn a DCFNet in an unsupervised manner are how to generate region pairs to construct $\mathcal{D}$ and how to utilize them to train the network. In this section, we first present the data preparation steps and then detail our unsupervised learning scheme.

Data Preparation

Though some well labeled video datasets (Russakovsky et al. 2015; Real et al. 2017) have been established for vision research, it is always expensive and time-consuming to acquire large-scale videos as well as human annotated labels. Our goal is to explore whether a DCFNet can be trained from a data source which can be easily obtained without human annotation. To achieve this goal, we propose to off-line train the network from a single movie.

To validate that the proposed method does not depend on some specific movies, we select top 5 colored movies from *https://www.imdb.com/chart/top* in our experiments. These movies are *The Shawshank Redemption (1994)*, *The Godfather (1972)*, *The Godfather: Part II (1974)*, *The Dark Knight (2008)* and *The Lord of the Rings: The Return of the King (2003)*. We use color movies because color information plays an important role for improving tracking performance in modern benchmarks (Wu, Lim, and Yang 2015; Kristan, Matas, and Leonardis 2015). For each movie, the following three steps are adopted to ensure robust learning.

Removing frames. Most of movies have many frames for prologue and epilogue which do not contain meaningful objects. These frames often come from the first 2 and the last 6 minutes and thus are removed. Figure 1 (a) shows two removed frames from prologue and epilogue, respectively.

Grouping frames. One movie contains a lot of scenes for depicting different stories. However, most of changes between two consecutive scenes are not smooth. Region pairs discovered from two different scenarios are likely to contain different instances, leading to unstable tracking performance. To mitigate this problem, we segment a long movie into several shot clips by automatically detecting scene cuts between two adjacent frames. Specifically, we first convert all frames into gray images and then downsample them with a factor of 10 for fast processing. We calculate cross-correlation scores for all adjacent frames. A low correlation score between two frames often means scene cuts. In this paper, we assign two consecutive frames into two individual clips when the correlation score is below 0.3. Two examples of segmented shot clips are shown as two rows in Figure 1 (b). As one can see, frames in the same clip depict similar instances and backgrounds.

Extracting regions. To learn a robust tracker for generic object tracking, we need to generate many region pairs which contain objects. Here, we adopt Selective Search (Uijlings et al. 2013) method to generate hundreds of object proposals for each frame and retain at most 100 object proposals according to their objectness scores. Figure 1 (c) shows some object proposals of two example frames.

Off-line Incremental Learning

A naive solution to construct database $\mathcal{D}$ is to initialize the DCFNet and then apply it to obtain region pairs using Eq. (3), and then optimize DCFNet using such $\mathcal{D}$. However, region pairs discovered by an unoptimized DCFNet often contain many false matchings, resulting in poor tracking performance. To improve tracking robustness and accuracy, we formulate the unsupervised DCFNet optimization as an incremental learning problem, which mainly consists of two iteratively phases: **target tracking** and **tracker update**.

Our training pipeline is shown in Figure 2. More specifically, we use current DCFNet to track many regions from the sampled frame pairs. Some of those discovered pairs

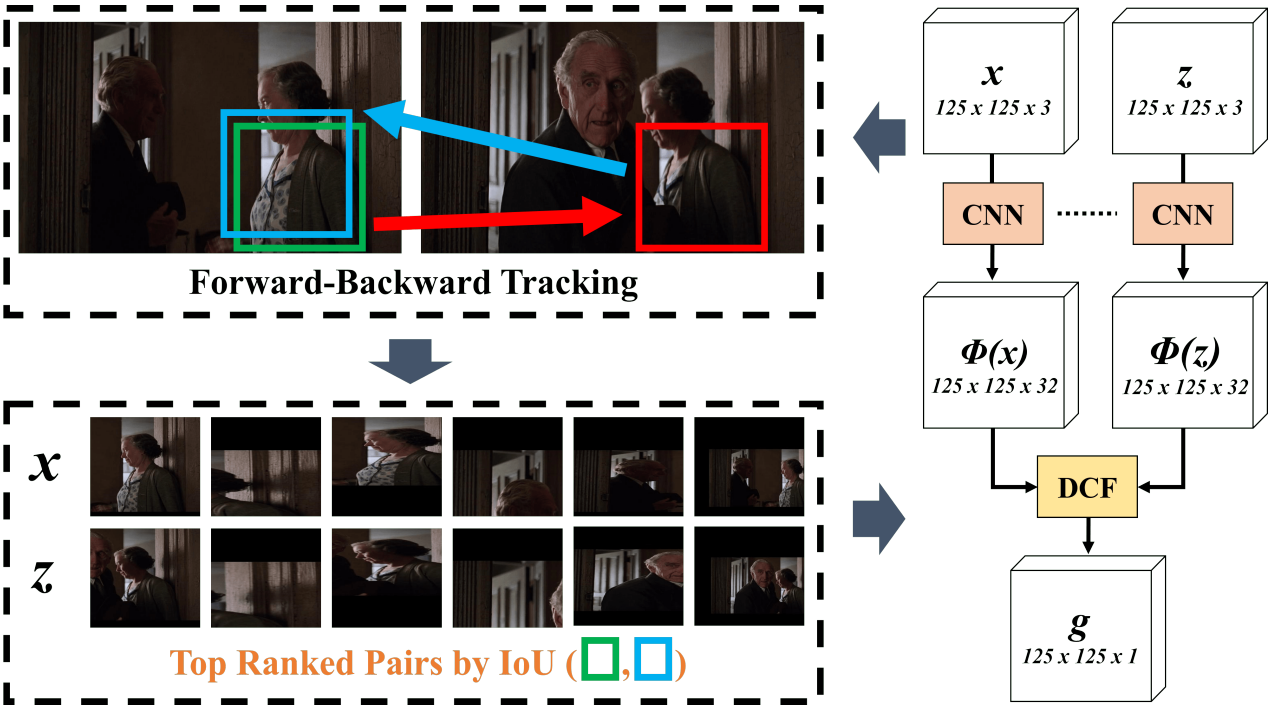

Figure 2: The pipeline of our training process. The input is a set of frame pairs, where each pair are sampled from the same clip. During training we alternate between: (1) using current DCFNet to track region proposals between two frames in Forward-Backward mode, as shown in Red and Blue arrows, respectively, (2) constructing a database $\mathcal{D}$ by Forward-Backward Analysis. (3) optimizing the network using current database $\mathcal{D}$. The process is iterated over a movie. Best viewed in PDF.

will be removed so that the database $\mathcal{D}$ can be generated to ensure region diversity and quality. With a given database $\mathcal{D}$, the DCFNet can be trained. We alternatively implement these two steps until a pre-defined condition is satisfied. To demonstrate the effect of the proposed unsupervised learning scheme, we adopt the same DCFNet (Wang et al. 2017) in our method. The structure of DCFNet is shown in the right part of Figure 2. It has two convolutional layers and a DCF layer. Each convolutional layer has convolutional kernels of size 3×3 with 1 padding and outputs 32 feature maps. Note that the convolutional layers are shared between $\boldsymbol{x}$ and $\boldsymbol{z}$ (dash line). The DCF layer will generate a response map $\boldsymbol{g}$ by Eq. (3). In the following, we present the details of the two major phases of our learning method.

Stage 1: target tracking. For each turn, the tracking phase is evaluated on V segmented clips from a single movie. For each clip, we sample I frame pairs to discover useful regions. Since these frames are grouped by their correlation scores, the sampled frame pairs are likely to contain similar instances or background contexts. In addition, it is found that instances in the same clip tend to move smoothly and thus their spatial locations will not change significantly. Based on these observations,the pair regions can be discovered in a local searching area. For each frame pair, let's denote by (F^t, F^{t+l}) as a pair of sampled frame, where $l \leq L$ is a random number for frame interval. For each proposal p_i^t in F^t, the current DCFNet is used to find a most likely location p_i^{t+l} in F^{t+l} by Eq. (3). (p_i^t, p_i^{t+l}) is the discovered region pair which will be used for network optimization. However, some of the pairs can be highly overlapped with each other, and some of them are low quality matching results, which will degrade the region extraction accuracy and consequently the tracking performance.

To address these issues, we propose to use Forward-Backward Analysis (FBA) (Kalal, Mikolajczyk, and Matas 2010) to select matching pairs. More detail, each tracked region p_i^{t+l} in F^{t+l} is treated as a new start point to obtain a *backward* result p_{ib}^t in F^t, where b means *backward*. To analyze each proposal's trajectory, we define a score over p_i^t and p_{ib}^t using $IoU(p_i^t, p_{ib}^t)$ and adopt Non-Maximum-Suppression (NMS) to remove highly overlapped regions. The top K regions with highest IoU scores are selected from each frame. We do not use NMS to remove overlapped regions according their objectness scores in the off-line data preparation phase since many overlapped regions are also helpful for tracker optimization. More importantly, it is hard to determine which region can be easily tracked between two sampled frames. Another merit of FBA is that it can automatically detect many less quality matching pairs. For example when the motion of an target is faster than expected. Tracker is likely to produce a trajectory which is to some extent random. Similar observation can be found in backward tracking. In addition, considering an object is disappeared in F^{t+l}, its appearance at start point p_i^t and p_i^{t+l} are difference, which will yield unstable trajectories. After FBA based selection, training pairs $(x_i, z_i)_{i=1}^K$ are cropped at positions

$(p_i^t, p_i^{t+l})_{i=1}^K$ from (F^t, F^{t+l}) respectively. Then all cropped regions are resized to 125×125 and added into $\mathcal{D}$. Current $\mathcal{D}$ is constructed by checking all $I \times V$ frame pairs.

Stage 2: DCFNet update. Given a discovered $\mathcal{D}$, we can update DCFNet in a similar way as supervised learning (Wang et al. 2017). For each iteration, we randomly sample a batch of $(\boldsymbol{x}_i, \boldsymbol{z}_i)_{i=1}^M$ from current database $\mathcal{D}$ and fed them into DCFNet to generate M response map $\boldsymbol{g}_i$. Batch loss is computed according to Eq. (4) by comparing each output $\boldsymbol{g}_i$ with the desired output $\tilde{\boldsymbol{g}}$, which is generated by a Gaussian distribution with spatial bandwidth 0.1. Gradients of ϕ can be gained from Eq. (5) and Eq. (6). We train network $[N/M]$ times, where $[\cdot]$ is rounding operator. Then the update DCFNet is utilized to construct a new $\mathcal{D}$ as illustrated in **Stage 1**, which will be used for the next round optimization of DCFNet. This incremental learning iteratively refines the network and the database $\mathcal{D}$, resulting in a more stable tracker.

Online Tracking

Online tracking is the same as the original paper (Wang et al. 2017). Specifically, DCF layer is firstly replaced by standard DCFs and filters $\boldsymbol{w}$ are learned using Eq (1) in the initial frame. For current frame, a searching area is cropped centered at the previously estimated position and then represented as feature map by the learned network ϕ. The template $\boldsymbol{w}$ is compared that feature to generate a response map $\boldsymbol{g}$ using Eq (3). The new position is obtained by finding the maximal score in that of response map. To cope with scale changes, the maximum score is searched from a pyramid of patch with 3 scales $\{1.015^s, s = -1, 0, 1\}$. Finally, filters $\boldsymbol{w}$ are updated with a parameter α_t to control the importance of sample $\boldsymbol{x}$ in time t, which is shown as following:

$$\boldsymbol{w}^k = \mathcal{F}^{-1}\left(\frac{\Sigma_{t=1}^T \alpha_t \hat{\phi}(\boldsymbol{x}_t) \odot \hat{\boldsymbol{y}}_t^*}{\Sigma_{t=1}^T \alpha_t \Sigma_{k=1}^K \hat{\phi}^k(\boldsymbol{x}_t) \odot (\hat{\phi}^k(\boldsymbol{x}_t))^* + \lambda}\right) \quad (7)$$

Experiments

In this section, we firstly conduct several experiments to study the effect of various options to train a DCFNet. Secondly, we will compare the learned DCFNet to variants where the networks are trained from different movies. Finally, we demonstrate that our unsupervised learning based tracker achieves state-of-the-art results in comparison to the ones based unsupervised pre-training approaches.

Testing datasets

The experiments are evaluated on OTB-2013 (Wu, Lim, and Yang 2013), OTB-2015 (Wu, Lim, and Yang 2015) and VOT-2015 (Kristan, Matas, and Leonardis 2015) datasets. We follow the standard evaluations adopted in both datasets. For OTB testing, the performance is measured by one-pass evaluation (OPE) with precision and success plots metrics. The precision metric measures the rate of predicted locations within a certain threshold (20) distance from those of human annotated ground truth, while the success plots measures the overlap ratio between those two kinds of bounding boxes. For success plots, we mainly report area-under-the-curve (AUC) scores. For VOT testing, we use the official toolkit[1] to evaluate and report expected average overlap (EAO), robustness (failure rate, FRT) and accuracy (ACC). EAO is the main evaluation metric and takes into account both the per-frame accuracy and the number of failures.

Implementation

Parameters for DCFNet. For fairly comparison to supervised learning, all parameters in DCFNet are the same as (Wang et al. 2017) to tease apart that effect. In detail, the maximal interval L for frame pair sampling is 10. λ and γ in Eq. (2) and Eq. (4) are set to $1e-4$ and $5e-4$ respectively. For online tracking, the λ is the same to the off-line learning, and the importance factor α_t is set to 0.01.

Parameters for off-line learning. We set $I = 4$ and $V = 400$ for $\mathcal{D}$ construction. For each sampled frame pair, we track all object proposals in the first frame and then use FBA to select at most $K = 16$ examples to construct database $\mathcal{D}$. The NMS threshold in FBA is set to 0.3, which is the same to most of object detection system. Usually, $\mathcal{D}$ consists of around $N = 25,000$ pair regions for each optimization round. For network optimization, the initial weights are randomly generated using improved "Xavier" technique (He et al. 2015). We train the network using Stochastic Gradient Descent. For each iteration, the mini-batch size is $M = 32$. Momentum rate and weight decay is set to 0.9 and $5e-4$ respectively. We repeat above steps until all clips from a movie are reviewed and then start a new epoch. The network is trained for 10 epochs with a learning rate exponentially decaying from $1e-2$ to $1e-3$. All above parameters are fine-tuned on the movie *The Shawshank Redemption*, and reused across all training movies. We implement on MATLAB using MatConvNet toolbox (Vedaldi and Lenc 2015).

Variant options for network training

Table 1 shows many results on OTB-2013 and OTB-2015 by off-line learning DCFNet with different configurations. The adopted movie in this experiment is *The Shawshank Redemption*. Because of the randomness involved in our learning algorithm, we repeat each configuration 5 times with different random seeds, and report the mean results and standard derivations over 5 trials. For the sake of fair comparison, we fix the number of total discovered region pairs in various options. More detail, in settings of training without FBA and without incremental learning, denoted as w/o FBA and w/o IL in Table 1 respectively, we randomly select at most 16 region pairs from each frame pair to construct database $\mathcal{D}$. Note that, in case of training without incremental manner, all regions are discovered by the DCFNet with random initialized weights. As shown in Table 1, our full learning pipeline achieves best results in both OTB-2013 and OTB-2015 datasets, significantly outperforming those results obtained without any off-line pre-training ($+10\%$), denoted as **RW** in this table. As one can see, frame grouping **(FG)** plays an very important role in learning a robust visual tracker.

[1] https://github.com/votchallenge/vot-toolkit

Table 1: Comparison results by training DCFNet on *The Shawshank Redemption* using variant options. We report mean and (std) over 5 trials. **Full** is our full unsupervised learning pipeline including frame grouping (**FG**), Forward-Backward Analysis (**FBA**), incremental learning (**IL**). **P-NMS**: using NMS for proposal removing in data preparation stage. **RW**: DCFNet without off-line learning. **Bold** and *Italic* represent the best and second best results, respectively.

		Full	w/o FG	w/o FBA	w/o IL	w P-NMS	RW
OTB-2013	AUC	**62.47 (1.37)**	57.11 (4.54)	59.13 (2.21)	60.44 (1.32)	*60.81 (1.42)*	47.11 (1.21)
	Prec@20	**81.03 (2.68)**	75.12 (2.79)	77.81 (1.99)	78.56 (2.14)	*79.32 (2.17)*	61.64 (1.98)
OTB-2015	AUC	**57.45 (0.64)**	51.92 (3.67)	55.31 (1.64)	56.22 (0.79)	*56.14 (1.44)*	46.73 (1.03)
	Prec@20	**75.19 (1.25)**	67.94 (2.19)	70.94 (1.01)	72.11 (1.04)	*72.20 (0.98)*	59.93 (1.77)

Table 2: Comparison results by using different movies. **SR**: *The Shawshank Redemption*, **GF**: *The Godfather*, **GF-II**: *The Godfather: Part II*, **DK**: *The Dark Knight*, **LR**: *The Lord of the Rings: The Return of the King*. **RW**: DCFNet without off-line learning. We report mean and (std) over 5 trials. **Bold** and *Italic* represent the best and second best results, respectively.

		GF	GF-II	SR	LR	DK	RW
OTB-2013	AUC	**62.57 (1.40)**	61.54 (1.12)	*62.47 (1.37)*	61.41 (1.07)	60.89 (0.92)	47.11 (1.21)
	Prec@20	*80.62 (2.00)*	80.16 (1.45)	**81.03 (2.68)**	79.04 (1.45)	79.01 (0.47)	61.64 (1.98)
OTB-2015	AUC	57.23 (0.89)	56.47 (0.59)	**57.45 (0.64)**	*57.41 (0.75)*	57.08 (1.32)	46.73 (1.03)
	Prec@20	74.30 (0.76)	73.44 (1.64)	**75.19 (1.25)**	*74.67 (0.97)*	73.85 (0.85)	59.93 (1.77)

For example, without FG, tracker obtains the highest standard derivations. This is mainly because the network is optimized with many regions, which are likely contain different instances or objects. Another important factor in our method is Forward-Backward Analysis, which helps remove many low quality regions. Some removed boxes are denoted using red color in Figure 3. Green boxes are used for constructing database $\mathcal{D}$. The third picture in Figure 3 shows the statistic of those regions whose IoU scores are below 0.5 according to FBA. We report the percentage of those region pairs over the total number of object proposals in each epoch. The values are calculated over 5 trials. As one can see, for each epoch, there have a large part of regions (around 8%) which cannot be well matched. Our FBA removes such regions and thus increases the final tracking performance as demonstrated in Table 1. This table also proves that the network trained with our incremental manner performs better $+2\%$ than that network trained without IL ($(w/o$ **IL**$)$). The main reason is that DCFNet with random weights cannot well track all regions, building a low quality of database for network optimization. In the next comparison, we only report network trained with our full scheme for analysis.

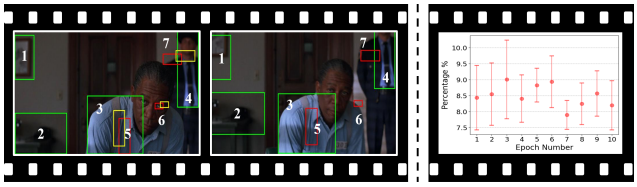

Figure 3: Left: Example of some matching pairs. Red and yellow boxes are removed regions and their *backward* results. Green boxes are used for network optimization. Each pair is denoted using the same number in different frames. **Right:** Statistics of such removed regions in various training epochs over 5 trials. Best viewed in PDF.

Different movies for network training

The second experiments are conducted on different movies. We also repeat 5 times to reduce the impact of randomness and show all results in Table 2. Visual trackers with pre-training perform significantly better (around $+10\%$) than the one without any off-line optimization (denoted as **RW**). Note that, we do not fine-tune the off-line learning parameters across different movies. Therefore, results showed in this table demonstrate that the proposed unsupervised learning scheme is insensitive to the employed movie. For example, visual trackers achieve very similar tracking results by training on the movie **SR** and **GF-II**, where the latter one is recorded in 40 years ago and frames of that movie reveal very different image qualities to videos in modern tracking benchmarks. In the next section, we use the best trained model from **GF** to compare with other methods, and denote our model as **UL-DCFNet** for simplicity.

Comparison with state-of-the-arts

OTB Dataset. The goal of our work is to explore whether a visual tracker can be trained without human supervision, we mainly compare our trained model with closely related works. The examples include Siamese structure based trackers, including CFNet (Valmadre et al. 2017), SiamFc (Bertinetto et al. 2016), SINT (Tao, Gavves, and Smeulders 2016) and the basic model – DCFNet (Wang et al. 2017), (2) DCF based trackers including KCF (Henriques et al. 2015), DSST (Danelljan et al. 2014), HCFT (Ma et al. 2015a), (3) Two representative unsupervised learning based methods including DLT (Wang and Yeung 2013) and TIR (Ma et al. 2015b) and (4) other trackers such as CNN-SVM (Hong et al. 2015) and FCNT (Wang et al. 2015). Note that

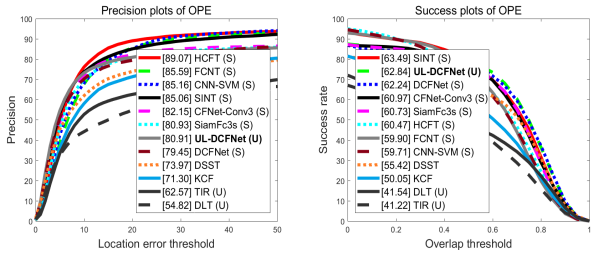

Figure 4: Results on OTB-2013 dataset.

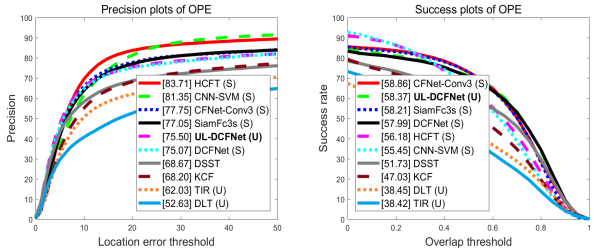

Figure 5: Results on OTB-2015 dataset.

all methods in (1) are trained with human supervision from hundreds or thousands of videos (Smeulders et al. 2014; Russakovsky et al. 2015). Among all compared methods, TIR (Ma et al. 2015b) is the mostly related work, which off-line trained a neural network to reconstruct input signals, and then transferred the optimized network for online tracking in combination with DCFs. Note that we employ the publicity available codes or results provided by the authors for fairly comparisons.

Figure 4 and 5 show all compared results. In these figures, the values in the legend represent the precisions at a threshold of 20 pixels for precision plots, and the AUC scores for success plots. **U** and **S** in the legend stand for trackers pre-trained in unsupervised or supervised manners, respectively. As we can see, our UL-DCFNet achieves 62.84 and 58.37 AUC scores on OTB-2013 and OTB-2015, respectively, which are similar as that results obtained by training DCFNet using supervised learning (DCFNet (S) *v.s* UL-DCFNet (U)). When compared with trackers based on supervised pre-training, such as SINT, CFNet-Conv3, SiamFc3s, FCNT, HCFT and CNN-SVM. UL-DCFNet also shows very competitive results on both testing datasets. In this comparison, our UL-DCFNet does not show good performance in precision scores as compared with some trackers such as HCFT, which utilizes the pre-trained VGG16 (Simonyan and Zisserman 2014) model as their feature extractors, and ensembles 3 DCFs for tracking. However, UL-DCFNet obtains better AUC scores than such trackers in both datasets.

In comparison to trackers based on unsupervised pre-training, UL-DCNet (U) attains the best results. In addition, we convert all frames in the movie *The Godfather* into gray-scale images and train DCFNet on that images using our unsupervised learning pipeline for fairly comparison to TIR and DLT. Our UL-DCFNet (Gray) achieves 55.34 (1.04) / 71.11 (1.78) scores on OTB-2015 dataset, outperforming TIR and DLT by a large margin on both AUC (+17) and precision plots (+8), respectively. This clearly demonstrates that *trackers trained using an objective which is similar to the online tracking process is more suitable for visual tracking task*. Our method provides a way to implement this idea without human annotations. More importantly, our method is just trained from a single movie, which can be obtained much easier than thousands of videos or millions of images utilized in all above mentioned methods.

Table 3: Results on VOT-2015 datasets.

Method	Acc	FRT	EAO
SiamFc3s	0.54	1.65	0.2547
SODLT	0.56	1.81	0.2329
DCFNet	0.53	1.68	0.2174
CFNet-Conv3	0.53	2.17	0.2038
UL-DCFNet (Ours)	**0.53**	**1.70**	**0.2234**

VOT-2015 Dataset. We mainly compared with 5 trackers based on supervised learning. Results are shown in Table 3. Our UL-DCFNet shows very competitive results on this dataset as compared with those trackers which are pre-trained using supervised learning. Interestingly, our UL-DCFNet shows slightly better performance than the same network trained on many tracking videos using human annotated bounding boxes, denoted as DCFNet. This might because our self-tracking strategy discovers region pairs which contain not only the primal objects, but also some part of objects, or even some stuff for network learning.

Quantitative Results. We visualize some tracking results in the **supplementary material**.

Discussion and Conclusion

We proposed a novel approach to train a visual tracker with similar pipeline to the online tracking process without human supervision. Our tracker is trained from a single movie by iteratively alternating between tracking and updating processes. In our method, we proposed to group frames and utilized the Forward-Backward method to select valid training data, leading to a better off-line learned tracker. The experiment results showed that the network trained by our unsupervised learning scheme achieves state-of-the-art performance among existing unsupervised learning approaches. Their performance is comparable with those trackers learned with full supervision from very large scale databases. Some issues still remain. As it stands, our training pipeline needs several tracking and update rounds, resulting in a longer time than supervised based methods. Therefore, it cannot be easily scaled to many videos, such as combining all movies for training. The discovered region pairs based on simple FBA still contain a few low quality matches. We will address these issues in the future work.

Acknowledgments

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. This work

is supported by China NSFC grant (no. 61672446) and Hong Kong RGC GRF grant (PolyU 152240/15E).

References

Bertinetto, L.; Valmadre, J.; Henriques, J. F.; Vedaldi, A.; and Torr, P. H. 2016. Fully-convolutional siamese networks for object tracking. In *ECCV*, 850–865. Springer.

Bolme, D. S.; Beveridge, J. R.; Draper, B. A.; and Lui, Y. M. 2010. Visual object tracking using adaptive correlation filters. In *CVPR*, 2544–2550. IEEE.

Cadieu, C., and Olshausen, B. A. 2009. Learning transformational invariants from natural movies. In *NIPS*, 209–216.

Chatfield, K.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2014. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*.

Danelljan, M.; Häger, G.; Khan, F.; and Felsberg, M. 2014. Accurate scale estimation for robust visual tracking. In *BMVC*. BMVA Press.

Danelljan, M.; Hager, G.; Shahbaz Khan, F.; and Felsberg, M. 2015. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 4310–4318.

Danelljan, M.; Bhat, G.; Khan, F. S.; and Felsberg, M. 2017. Eco: Efficient convolution operators for tracking. In *CVPR*, 21–26.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255. IEEE.

Fan, H., and Xiang, J. 2017. Robust visual tracking via local-global correlation filter. In *AAAI*, 4025–4031.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 580–587.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 1026–1034.

Held, D.; Thrun, S.; and Savarese, S. 2016. Learning to track at 100 fps with deep regression networks. In *ECCV*.

Henriques, J. F.; Caseiro, R.; Martins, P.; and Batista, J. 2015. High-speed tracking with kernelized correlation filters. *in TPAMI* 37(3):583–596.

Hong, S.; You, T.; Kwak, S.; and Han, B. 2015. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, 597–606.

Kalal, Z.; Mikolajczyk, K.; and Matas, J. 2010. Forward-backward error: Automatic detection of tracking failures. In *ICPR*, 2756–2759. IEEE.

Kristan, M.; Matas, J.; and Leonardis, A. e. a. 2015. The visual object tracking vot2015 challenge results. In *ICCVW*, 1–23.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

Laptev, I.; Marszalek, M.; Schmid, C.; and Rozenfeld, B. 2008. Learning realistic human actions from movies. In *CVPR*, 1–8. IEEE.

Li, B.; Yan, J.; Wu, W.; Zhu, Z.; and Hu, X. 2018. High performance visual tracking with siamese region proposal network. In *CVPR*, 8971–8980.

Ma, C.; Huang, J.-B.; Yang, X.; and Yang, M.-H. 2015a. Hierarchical convolutional features for visual tracking. In *ICCV*, 3074–3082.

Ma, C.; Yang, X.; Zhang, C.; and Yang, M.-H. 2015b. Learning a temporally invariant representation for visual tracking. In *ICIP*, 857–861. IEEE.

Real, E.; Shlens, J.; Mazzocchi, S.; Pan, X.; and Vanhoucke, V. 2017. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *CVPR*, 7464–7473. IEEE.

Rohrbach, A.; Rohrbach, M.; Tandon, N.; and Schiele, B. 2015. A dataset for movie description. In *CVPR*, 3202–3212.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *in IJCV* 115(3):211–252.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

Smeulders, A. W.; Chu, D. M.; Cucchiara, R.; Calderara, S.; Dehghan, A.; and Shah, M. 2014. Visual tracking: An experimental survey. *in TPAMI* 36(7):1442–1468.

Tao, R.; Gavves, E.; and Smeulders, A. W. 2016. Siamese instance search for tracking. In *CVPR*, 1420–1429. IEEE.

Tapaswi, M.; Zhu, Y.; Stiefelhagen, R.; Torralba, A.; Urtasun, R.; and Fidler, S. 2016. Movieqa: Understanding stories in movies through question-answering. In *CVPR*, 4631–4640.

Torralba, A.; Fergus, R.; and Freeman, W. T. 2008. 80 million tiny images: A large data set for nonparametric object and scene recognition. *in TAPMI* 30(11):1958–1970.

Uijlings, J. R.; Van De Sande, K. E.; Gevers, T.; and Smeulders, A. W. 2013. Selective search for object recognition. *in IJCV* 104(2):154–171.

Valmadre, J.; Bertinetto, L.; Henriques, J.; Vedaldi, A.; and Torr, P. H. S. 2017. End-to-end representation learning for correlation filter based tracking. In *CVPR*.

Vedaldi, A., and Lenc, K. 2015. Matconvnet: Convolutional neural networks for matlab. In *ACM MM*, 689–692. ACM.

Wang, N., and Yeung, D.-Y. 2013. Learning a deep compact image representation for visual tracking. In *NIPS*, 809–817.

Wang, Q.; Chen, F.; Yang, J.; Xu, W.; and Yang, M.-H. 2012. Transferring visual prior for online object tracking. *in TIP* 21(7):3296–3305.

Wang, L.; Ouyang, W.; Wang, X.; and Lu, H. 2015. Visual tracking with fully convolutional networks. In *ICCV*, 3119–3127.

Wang, Q.; Gao, J.; Xing, J.; Zhang, M.; and Hu, W. 2017. Dcfnet: Discriminant correlation filters network for visual tracking. *arXiv preprint arXiv:1704.04057*.

Wu, Y.; Lim, J.; and Yang, M.-H. 2013. Online object tracking: A benchmark. In *CVPR*, 2411–2418. Ieee.

Wu, Y.; Lim, J.; and Yang, M.-H. 2015. Object tracking benchmark. *in TPAMI* 37(9):1834–1848.