Progressive Unsupervised Learning for Visual Object Tracking

Qiangqiang Wu Jia Wan Antoni B. Chan
Department of Computer Science, City University of Hong Kong
qiangqwu2-c@my.cityu.edu.hk, jiawan1998@gmail.com, abchan@cityu.edu.hk

Abstract

In this paper, we propose a progressive unsupervised learning (PUL) framework, which entirely removes the need for annotated training videos in visual tracking. Specifically, we first learn a background discrimination (BD) model that effectively distinguishes an object from background in a contrastive learning way. We then employ the BD model to progressively mine temporal corresponding patches (i.e., patches connected by a track) in sequential frames. As the BD model is imperfect and thus the mined patch pairs are noisy, we propose a noise-robust loss function to more effectively learn temporal correspondences from this noisy data. We use the proposed noise robust loss to train backbone networks of Siamese trackers. Without online fine-tuning or adaptation, our unsupervised real-time Siamese trackers can outperform state-of-the-art unsupervised deep trackers and achieve competitive results to the supervised baselines.

1. Introduction

Visual object tracking (VOT) is one of the fundamental tasks in computer vision, which has gained much attention in recent years due to its wide usage in many practical applications, such as robotics [37], video surveillance [25] and eye-tracking applications [24]. Although much progress has been made on VOT, it is still far from being fully solved due to the numerous real-world challenges, e.g., background cluster, rotation, occlusion and illumination variation.

To handle with the above challenges, the existing end-to-end trainable deep trackers [62, 65] learn rich feature representations from large-scale annotated training videos based on supervised learning. The representative works are Siamese trackers [29, 30], which learn feature representations in an offline manner, and achieve favorable performance even without online fine-tuning or adaptation. The success of Siamese-based trackers demonstrate that the numerous annotated data is the key to feature learning of deep trackers. For example, the pioneering deep tracker SiamFC [4] is trained with the ILSVRC [43] dataset, which contains about 2 million labeled frames. Some extensions

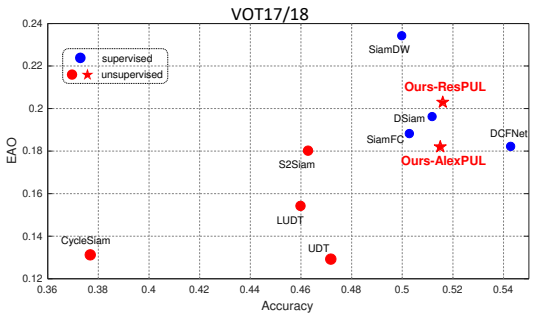

Figure 1: Tracking accuracy and EAO obtained by supervised (denoted by blue color) and unsupervised trackers (red color) on the VOT17/18 dataset. Without online fine-tuning or adaptation, our unsupervised AlexPUL and ResPUL trackers outperform state-of-the-art online updating-based unsupervised trackers, meanwhile performing favorably against supervised deep trackers, which shows the potential of our unsupervised method.

like SiamRPN++ [30] commonly need more labeled video datasets for training, e.g., using Got-10k [18], LaSOT [12], Youtube-BB [42]. These supervised deep trackers assume that a robust tracking model can be learned by using large-scale annotated video datasets. However, they usually ignore the fact that annotating such large scale video datasets can be prohibitively expensive and time-consuming.

Unlabeled videos or images are a fertile source for unsupervised learning, since they are easily collected from online services. The main problem is how to construct a supervisory signal from these sources for learning a feature map representation that is useful for visual tracking. The supervision in visual tracking generally comes from two aspects: spatial supervision in a static video frame, and sequential supervision across multiple frames. Spatial supervision [45] can be achieved by learning template correspondence in a single frame or image. The training cost is reduced by using static images, but the learned features lack invariance to visual changes in sequential frames. On the other hand, sequential supervision is commonly achieved by finding temporal correspondence in multiple frames. The recent cycle learning [2, 51] is based on sequential supervision, but the tracking performance is still limited without online adaptation [53]. To achieve favorable performance, an online updated correlation filter is required [8]. In this paper, we propose to use both spatial and sequential supervision for un-

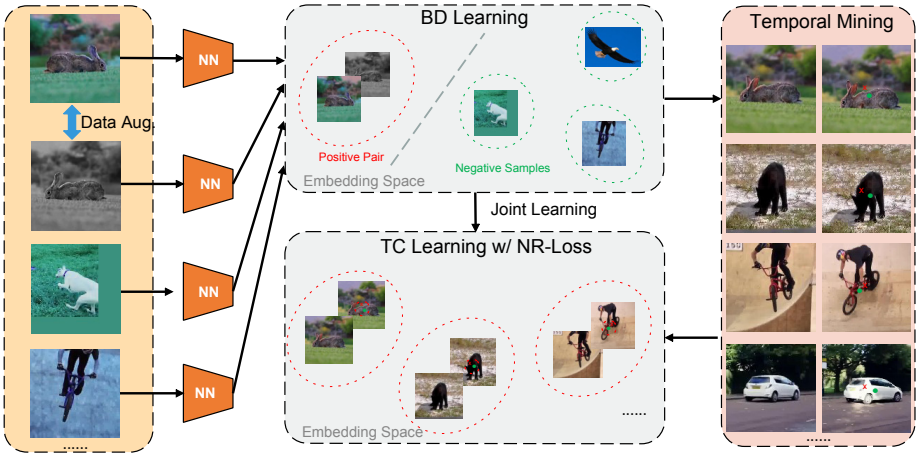

Figure 2: Overview of Progressive Unsupervised Learning (PUL) for learning feature representations for tracking. Given various instances sampled from unlabeled videos, we first use contrastive learning to learn a background discrimination (BD) model, applying anchor-based hard negative mining. In order to learn temporal correspondence (TC), we then apply the BD model to mine temporal corresponding patches. Since the mined patch pairs are noisy (i.e., they lack exact spatial correspondence), we propose a noise-robust (NR) loss function for TC learning. In the temporally-mined patches, the estimated target center is the red "x", while the true target center is the green circle.

supervised representation learning in visual tracking. Without applying online fine-tuning or updating, our method can achieve favorable unsupervised performance.

In this paper, our main motivation is that a robust tracking model should suppress background distractors and identify temporal correspondences across multiple frames simultaneously. To achieve this, we propose a progressive unsupervised learning (PUL) framework (see Fig. 2). The key idea is to progressively learn an unsupervised tracking model based on the two parts of learning: background discrimination (BD) and temporal correspondence (TC) learning. For the former, we formulate BD learning as a contrastive learning task [7, 16], which aims to separate different identity instances in the deep embedding space. To learn more discriminative features for visual tracking, we extend the original contrastive learning framework [7] by proposing an anchor-based hard negative mining (AHM) strategy. Our AHM further boosts the feature discrimination by learning to suppress hard negative distractors.

For temporal correspondence learning, we employ the BD model to progressively mine corresponding-pairs of patches in short video clips. Following [4], a common practice is to use pixel-wise binary-cross entropy (BCE) loss between the tracker and GT response maps for learning the temporal correspondence across multiple frames in a video. However, the mined patch-pairs that are used as the pseudo ground-truth (GT) are not well matched, due to an imperfect BD model (e.g., see right side of Fig. 2). That is, there exists non-negligible spatial noise in the GT locations, whereas BCE assumes no spatial noise. This mismatch between the actual noise in the GT and the assumed noise by the loss will limit the generalization and localization ability of the trained model. To alleviate problem, we propose a new likelihood function for tracking response maps, which is based on a generative model of the GT spatial noise. Notably, our likelihood function models correlations between pixels in the response map, whereas BCE does not consider these correlations. We use the negative log likelihood as our new noise-robust (NR) loss function, to more effectively learn temporal correspondence from noisy patch pairs.

We apply the proposed PUL framework to train backbone networks used in two Siamese trackers [4, 65]. The offline-learned backbone networks (i.e., AlexNet [28] and CIResNet-22 [65]) can be directly applied for online Siamese tracking. We show that without using online fine-tuning or adaptation, our unsupervised trackers outperform state-of-the-art unsupervised trackers on five benchmarks. In summary, the main contributions of our work are:

- We propose a progressive unsupervised learning (PUL) framework, which formulates visual tracking as a combination of background discrimination and temporal correspondence learning.
- We propose to learn a background discrimination model through contrastive learning and anchor-based hard negative mining.
- We propose a noise-robust loss to effectively learn temporal correspondences from noisy patch-pairs.

2. Related Work

Supervised Deep Tracking. Existing deep trackers commonly follow a one-stage or two-stage supervised learning framework. The representative one-stage learning framework is built on a Siamese network [46]. Based on SiamFC [4], further improvements are made, including scale regression [30, 29], online adaptation [64], architecture design [60] and discriminative feature learning [54, 11]. The two-stage framework generally trains both an offline-

learned model and a complementary online model for improvement, e.g., as with deep feature-based correlation filters [6, 8]. To improve the offline-learned Siamese networks, some extensions [5, 64] learn to update the model in an meta-learning fashion. In this paper, we apply our PUL framework to the one-stage learning framework, mainly because we aim to make direct comparisons with other offline-learned feature representations for tracking.

Contrastive Unsupervised Learning. Contrastive learning aims to contrast positive pairs against negative pairs for unsupervised learning of representations [7]. Based on this idea, [58] proposes to use a memory bank to store the pre-computed instance features for contrast, while other extensions have been developed, e.g., CPC [35], CMC [48], MoCo [16]. Instead of using a memory bank, another type of contrastive learning method [7, 47] focuses on using in-batch samples for augmented positive or negative sampling, which aims to maximize/minimize the similarity between a positive/negative sample pair. Although much progress has been achieved by contrastive learning in some downstream applications (e.g., detection and segmentation), its application to visual tracking is not well-explored. In this work, we show that an effective background discrimination model can be learned using contrastive learning.

Unsupervised Learning from Video. The most related work to ours is [55], which employs an off-the-shell tracker (e.g., KCF [17]) to find temporal correspondence in multiple frames, and then uses this supervisory signal for triplet ranking learning. There are two main differences between our work and [55]: 1) we show that an off-the-shell tracker is not necessarily required – a robust tracker can be learned with unsupervised contrastive learning, which can provide favorable temporal supervision for further learning; 2) we further model the noise in mined patches to make the temporal learning more robust.

Recent works leverage various pretext tasks constructed in video, including temporal order verification [36], frame reconstruction [13] and displacement prediction [34]. A typical tracker TLD [20] uses unlabeled data with estimated pseudo labels to boost the online training while our PUL further considers temporal correspondence noise for the better offline learning. Several unsupervised VOT methods use cycle learning in video. UDT [51] performs forward tracking and backward verification in a correlation filter tracking framework, by using a cycle consistence loss. [33] proposes to learn dense correspondence from videos. CycleSiam [2] further extends the cycle learning to a Siamese tracking framework. Finally, S^2SiamFC [45] only uses spatial supervision, which learns template correspondence in a static video frame. In contrast, our approach uses both spatial and sequential supervision, achieving better performance via progressive learning.

Learning from Noisy Labels. Most works on learning from noisy labels focus on image classifiers, where large datasets are crawled from the web or annotated by crowdsourcing, via methods such as robust loss functions [15, 56], label cleaning or noise-modeling [59, 49], sample selection [41], and improved training schemes [19, 32]. Similarly, [61, 44] learns from noisy class labels for instance segmentation and webly supervised object detection.

Besides classification problems, previous works focus on various types of noisy labels specific to computer vision. To handle noisy annotated segmentations, [66] modifies the loss to allow multiple class labels near segment boundaries, while [1] aligns predicted boundaries with edges in the image. [40] uses a regression model [22] to predict uncertainty in dense pose correspondences. Several methods perform label correction to clean noisy labels. In human pose estimation, [21] performs label correction for missing annotations of joint points. In object detection, [31] addresses noisy positive/negative labels of the anchors, via reweighting based on a cleanliness score computed using a trained detector. In crowd-counting, [3] uses an EM-like algorithm to correct the point-wise person annotations while learning the crowd density map estimator. In VOT, [10] trains the response map predictor by minimizing KL divergence (KLD) to a Gaussian conditional distribution arising from label noise. In contrast, our method explicitly integrates out the label noise to obtain the response map likelihood Eq. (7). KLD in [10] is a per-pixel loss, whereas ours models the correlations between pixels in the response map.

In contrast to these previous methods, we propose a loss function that is robust to spatial annotation noise, based on a generative noise model. To the best of our knowledge, our work is the first to consider spatial noise in the patch-pairs used for learning in VOT.

3. Proposed Method

An overview of our proposed *progressive unsupervised learning* (PUL) framework is shown in Fig. 2, which consists of three main steps: background discrimination learning, temporal mining, and temporal correspondence learning. Our goal is to learn a deep tracking model *without using any annotated videos*. Moreover, the learned unsupervised representation should be effective for visual tracking even *without online fine-tuning or adaptation*. Our framework is compatible with standard Siamese trackers.

3.1. Revisiting Siamese Tracking

SiamFC [4] is a classic end-to-end trainable deep tracking framework. The key idea in SiamFC is to formulate visual tracking as a deep similarity learning problem, thus enabling the deep network to be learned with large-scale annotated data in an offline manner. Formally, given a template image $\mathbf{z}$ and a search image $\mathcal{I}$, a response map is cal-

culated via cross-correlation:

$$f(\mathcal{I}, \mathbf{z}) = \varphi(\mathcal{I}) * \varphi(\mathbf{z}) + b, \quad (1)$$

where $\varphi(\cdot)$ is a fully-convolutional embedding function, b is a bias parameter and $*$ indicates the cross-correlation operation. Each element in the response map represents the similarity between the corresponding subregion in $\mathcal{I}$ and the template $\mathbf{z}$. In the training stage, elements in the ground-truth (GT) label map $\mathbf{m}$ are assigned to a positive label if they are within radius R of the true object location $\mathbf{y}$:

$$\mathbf{m}(\mathbf{x}) = B_R(\mathbf{x}|\mathbf{y}) = \begin{cases} 1, & k||\mathbf{x} - \mathbf{y}||_1 \leq R, \\ 0, & \text{otherwise}, \end{cases} \quad (2)$$

where $\mathbf{x} \in \mathbb{R}^2$ is a pixel location in the response map, and k is the total stride of network. The loss between the GT label map $\mathbf{m}$ and the predicted response map $\mathbf{f} = f(\mathcal{I}, \mathbf{z})$ is the pixel-wise weighted binary cross entropy (BCE) loss,

$$L(\mathbf{f}, \mathbf{m}) = \sum_{\mathbf{x} \in \mathcal{X}} -\mathbf{w}(\mathbf{x}) \log[\mathbf{f}(\mathbf{x})^{\mathbf{m}(\mathbf{x})}(1 - \mathbf{f}(\mathbf{x}))^{(1-\mathbf{m}(\mathbf{x}))}], \quad (3)$$

where $\mathbf{w}$ is pixel-wise weight map that alleviates the class imbalance problem.

3.2. Background Discrimination Learning

We next describe our proposed method for learning a background discrimination (BD) model from static video frames, which learns to suppress background distractors.

3.2.1 Data Preprocessing

The basic Siamese trackers commonly use the ILSVRC-2015 dataset [43] as the training data, which contains about 4,300 videos in total. For fair comparison, we also use the ILSVRC-2015 dataset for learning, but we do not use the bounding box annotations for supervision. To keep the diversity of the training data, we sample a training frame in every 10 frames of a video, resulting in the training set $\mathbb{F}$. Since we assume the bounding box annotations are not available, we generate a set of object proposals as candidate objects for learning the BD model. In particular, we apply EdgeBox [67] to generate 500 object proposals for each frame in $\mathcal{F}$. Note that other unsupervised object proposal methods could be used in our framework. Here, we use EdgeBox due to its efficiency and high object recall.

Since the generated 500 object proposals in each frame of $\mathcal{F}$ might be highly overlapped, we next apply non-maximum suppression with an overlap threshold of 0.6 to filter out the highly overlapped proposals with low objectness score, finally yielding 64 proposals for each frame (denoted as $\mathcal{P}$). For each training epoch, we randomly sample a patch-based training set $\mathcal{E}$ from $\mathcal{F}$, by randomly selecting 12 proposals in one random frame of each video.

Algorithm 1: Anchor-based Hard Negative Mining

Input: Frame set $\mathcal{F}$, rounds R, iterations per round T, batch size N and the initial model $\mathcal{M}_0$.
Output: Background discrimination model $\mathcal{M}_b$.

```
for i=1:R do
    Sample patch set E from F;
    Randomly select candidate anchors P_i from E;
    Candidate anchor evaluation using (5);
    Get anchor set {A_i}_{i=1}^T;
    for t=1:T do
        Select N-1 nearest neighbors N(A_t) of A_t;
        Use N(A_t) and A_t for one mini-batch
          contrastive learning with (4);
        Update the model M_i;
    end
end
```

3.2.2 Contrastive Learning

Inspired by recent contrastive learning algorithms [7, 16], we propose to learn a background discrimination (BD) tracking model in a contrastive learning (CL) way. Given a tracked instance $\mathbf{z}$, we first create two augmented views of $\mathbf{z}$ through data augmentation operations. Assume that we randomly sample N object instances from $\mathcal{E}$ per iteration, through data augmentation we obtain $2N$ augmented tracked instances in one mini-batch. A contrastive loss [7] is calculated in a softmax formulation:

$$\mathcal{L}_c(i, j) = -\log \frac{\exp(\varphi(\mathbf{z}_i) \cdot \varphi(\mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{I}_{[i \neq k]} \exp(\varphi(\mathbf{z}_i) \cdot \varphi(\mathbf{z}_k)/\tau)}, \quad (4)$$

where $\mathbf{z}_i$ and $\mathbf{z}_j$ $(i \neq j)$ are a positive pair of the augmented instances, τ is a temperature hyper-parameter, φ is a fully-convolutional embedding function, and $\mathbb{I}_{[i \neq k]}$ is an indicator function that returns 1 iff $i \neq k$ and 0 otherwise. Minimizing the contrastive loss $\mathcal{L}_c(i, j)$ encourages the positive response between $\mathbf{z}_i$ and $\mathbf{z}_j$ to be significantly larger than the sum of negative responses. That is, the learned embedding function φ should concentrate the positive pairs together, while separating negative pairs.

Although the above CL method only learns a limited transformation invariance with data augmentation, it can effectively facilitate our model to learn features to discriminate positive and background patches. We denote this background discrimination (BD) model as $\mathcal{M}_b$. In the optimization of $\mathcal{L}_c(i, j)$ in (4), a large denominator value can facilitate effective feature learning. Following [7], a straightforward idea is to use a brute force solution, i.e., using an extremely large mini-batch size (e.g., 8192), such that many different negative samples can be used for learning. However, using an extremely large batch size is not memory-friendly. We next propose a strategy to alleviate this issue.

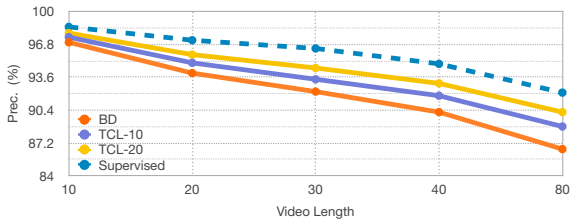

Figure 3: Distance precision rate (DPR@20) obtained by our unsupervised SiamFC using background discrimination (BD) learning and temporal correspondence learning with frame gaps 10 and 20 (TCL-10, TCL-20). We compare with supervised SiamFC for various lengths of video clips sampled from the OTB-13 dataset.

3.2.3 Anchor-based Hard Negative Mining

To effectively facilitate feature learning in CL, we propose anchor-based hard negative mining (AHM), which selects neighboring instances that are close to each other in the learned feature space of φ. The selected instances are constructed as one mini-batch samples for learning, which can lead to a large sum in the denominator of (4), even for a relatively small batch size N. To achieve this, we assume that an instance that lies in the region with dense samples is more like to have close nearest neighbors.

Let T denote the iterations per epoch, R be the number of epochs and $\mathcal{N}(\mathbf{z}_i)$ be the N-1 nearest instances to $\mathbf{z}_i$, where the similarity is measured with the dot product in the deep embedding space. For $(t+1)$-th epoch of training, the anchor instances are selected based on the following accumulated similarity score:

$$s_i = \sum_{j=1}^{N-1} \varphi_t(\mathbf{z}_i) \cdot \varphi_t(\mathbf{z}_j), \mathbf{z}_i \in \mathcal{P}_t, \mathbf{z}_j \in \mathcal{N}(\mathbf{z}_i) \quad (5)$$

where φ_t is the deep embedding function learned in the t-th epoch, $\mathcal{P}_t = \{\mathbf{z}_i\}_{i=1}^{N \times E}$ contains various candidate anchor instances randomly selected from $\mathcal{E}$ with a scale parameter $E > 1$. Since we have T iterations per epoch, we finally select T anchor instances $\mathcal{A} = \{A_i\}_{i=1}^{T}$, which have top-ranked accumulated similarities on their nearest neighbors. The learning pipeline with our AHM is shown in Alg. 1.

For the initial deep embedding model, we first randomly select mini-batch samples for CL using (4) for 20 epochs, then the proposed AHM is added for learning with 40 epochs. This can guarantee that our the model can select close NNs for more effective learning. For AHM, to speed up the computation of the nearest-neighbors, we add a global average pooling layer to the end of the deep embedding φ to reduce the extracted feature size. For ResPUL, the overall size of extracted features in $\mathcal{E}$ is 108.5MB, which is stored in main memory. Additionally, different from SiamRCNN [50], our AHM employs the self-updated backbone to adaptively select hard mini-batch samples for contrastive learning during each training epoch.

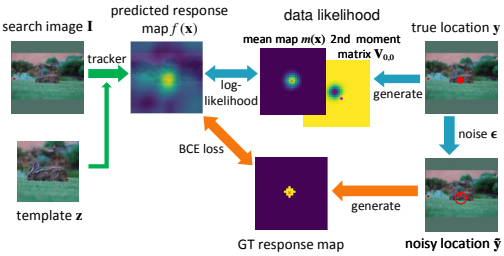

Figure 4: Modeling observation noises for mined patch pairs. The orange arrows show a traditional pipeline that uses the estimated center (red x) to generate a binary response map for training with binary cross-entropy (BCE) loss. Our approach (the blue arrows) infers a data log-likelihood (approximated by mean map $\bar{\mathbf{m}}$ and 2nd moment matrix $\mathbf{V}$) for training.

3.3. Temporal Mining

By integrating our learned BD model $\mathcal{M}_b$ to a SiamFC [4] tracker, we find that our BD-SiamFC can achieve competitive results on short video clips (see Fig. 3), but its accuracy drops significantly for longer video clips. This is mainly because our BD-SiamFC cannot well track objects with large appearance variations in longer videos, which is caused by the lack of temporal correspondence (TC) learning. To further improve the feature discrimination, we perform temporal mining to collect temporal corresponding patches for better learning.

We employ our BD-SiamFC for temporal mining, and the mining procedure follows the same tracking steps used in SiamFC without any modifications. In the mining process, we first randomly select a start frame in $\mathcal{F}$. Then a proposal patch is randomly picked from this frame to start mining. After tracking along the sampled continuous frames in the video, we obtained a corresponding tracked patch in the last frame, which has temporal appearance variations. The first and last patches are then collected as patch-pairs for training, denoted as $\mathcal{S}$. Note that the whole mining process is fully unsupervised and performed in an offline manner.

3.4. Temporal Correspondence Learning

Given the mined set of patch pairs $\mathcal{S}$, the original Siamese trackers commonly use a binary cross entropy (BCE) loss in (3) to learn temporal correspondence. However, the BCE loss is not suitable in our case since the mined set $\mathbb{S}$ is noisy (i.e., a patch pair may not be center-aligned). Thus, there exists a mismatch between the actual noise distribution in the response maps, caused by the practical mining procedure, and the no-noise assumption of the used BCE loss function (which assumes aligned patches). To alleviate this, we propose a noise-robust (NR) loss to model the mining or observation noise in order to more effectively learn temporal correspondence.

3.4.1 Modeling Observation Noise

The overall pipeline of our observation noise model is shown in Fig. 4. Formally, given mined patch pair com-

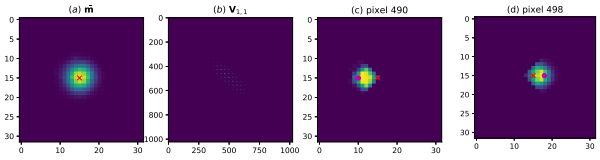

Figure 5: 2nd-order approximation of response map likelihood $p(\mathbf{m})$: (a) mean map $\bar{\mathbf{m}}$; (b) 2nd moment matrix $\mathbf{V}_{1,1}$; (c-d) moment maps for different pixels (marked with the circle), corresponding to one row (reshaped to 32×32) in the 2nd moment matrix. The red "x" is the noisy target center.

prising a template $\mathbf{z}$ and a larger searching image $\mathcal{I}$, the learning of Siamese trackers aims to predict the target map $\mathbf{m}(\mathbf{x}) = B_R(\mathbf{x}|\mathbf{y})$, where $\mathbf{x}$ is a pixel location and $\mathbf{y}$ is the true target center position in the map.

Now we consider a noisy patch pair where $\tilde{\mathbf{y}}$ is the observed or estimated target center location obtained by temporal mining. Let $\tilde{\mathbf{y}} = \mathbf{y} - \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \alpha\mathbf{I})$ is the observation noise with variance α, and $\mathbf{y}$ is the true center location. We have $\mathbf{y} = \tilde{\mathbf{y}} + \boldsymbol{\epsilon}$. Note that $\mathbf{y}$ is a random variable, and thus the target label map is also a random variable, $\mathbf{m}(\mathbf{x}) = B_R(\mathbf{x}|\tilde{\mathbf{y}} + \boldsymbol{\epsilon})$. Given the noise $\boldsymbol{\epsilon}$, the likelihood of the target map is a product of independent Bernoulli distributions (equivalent to pixel-wise BCE),

$$p(\mathbf{m}|\boldsymbol{\epsilon}) = \prod_{\mathbf{x}\in\mathcal{X}} \mathbf{f}(\mathbf{x})^{\mathbf{m}(\mathbf{x})}(1-\mathbf{f}(\mathbf{x}))^{(1-\mathbf{m}(\mathbf{x}))}, \tag{6}$$

where $\mathbf{f}(\mathbf{x}) \in [0,1]$ is the label probability (predicted by a NN using a sigmoid activation). To estimate the likelihood of the map $\mathbf{m}$, we integrate over the noise $\boldsymbol{\epsilon}$, yielding

$$\begin{aligned} p(\mathbf{m}) &= \int_{\boldsymbol{\epsilon}} p(\mathbf{m}|\boldsymbol{\epsilon})p(\boldsymbol{\epsilon})d\boldsymbol{\epsilon} \\ &= \int_{\boldsymbol{\epsilon}} \Big[\prod_{\mathbf{x}\in\mathcal{X}} \mathbf{f}(\mathbf{x})^{\mathbf{m}(\mathbf{x})}(1-\mathbf{f}(\mathbf{x}))^{(1-\mathbf{m}(\mathbf{x}))}\Big] p(\boldsymbol{\epsilon})d\boldsymbol{\epsilon}. \end{aligned} \tag{7}$$

Note that the integration over $\boldsymbol{\epsilon}$ induces dependence between locations in the label map. This integration has no closed-form solution, and thus we derive efficient approximations in the next subsection.

3.4.2 2nd-order Taylor Approximation

We first define the term $g(\boldsymbol{\epsilon}) = \log p(\mathbf{m}|\boldsymbol{\epsilon})$:

$$g(\boldsymbol{\epsilon}) = \log \prod_{\mathbf{x}} \mathbf{f}(\mathbf{x})^{\mathbf{m}(\mathbf{x})}(1-\mathbf{f}(\mathbf{x}))^{(1-\mathbf{m}(\mathbf{x}))}. \tag{8}$$

Then the log-likelihood of (7) can be rewritten as

$$\log p(\mathbf{m}) = \log \int_{\boldsymbol{\epsilon}} e^{g(\boldsymbol{\epsilon})} p(\boldsymbol{\epsilon})d\boldsymbol{\epsilon} = \log \mathbb{E}_{\boldsymbol{\epsilon}}[e^{g(\boldsymbol{\epsilon})}]. \tag{9}$$

To approximate the likelihood, we apply 2nd-order Taylor approximation, $e^g = 1 + g + \frac{1}{2}g^2$, yielding

$$\log p(\mathbf{m}) \approx \log(1 + \mathbb{E}_{\boldsymbol{\epsilon}}[g(\boldsymbol{\epsilon})] + \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}}[g(\boldsymbol{\epsilon})^2]). \tag{10}$$

Next we substitute $g(\boldsymbol{\epsilon})$ and compute the expectations. For convenience, we define the notation:

$$\begin{aligned} \mathbf{m}_1(\mathbf{x}) &= \mathbf{m}(\mathbf{x}), \quad & \mathbf{m}_0(\mathbf{x}) &= 1 - \mathbf{m}(\mathbf{x}), \\ \mathbf{h}_1(\mathbf{x}) &= \log \mathbf{f}(\mathbf{x}), \quad & \mathbf{h}_0(\mathbf{x}) &= \log(1 - \mathbf{f}(\mathbf{x})), \end{aligned} \tag{11}$$

and thus $g(\boldsymbol{\epsilon}) = \sum_{\mathbf{x}} \sum_{b\in\{0,1\}} \mathbf{m}_b(\mathbf{x})\mathbf{h}_b(\mathbf{x})$. Substituting into the first expectation in (10) (see derivations in Supp. B),

$$\mathbb{E}_{\boldsymbol{\epsilon}}[g(\boldsymbol{\epsilon})] = \sum_{\mathbf{x}} \sum_{b\in\{0,1\}} \bar{\mathbf{m}}_b(\mathbf{x})\mathbf{h}_b(\mathbf{x}) \tag{12}$$

where $\bar{\mathbf{m}}_1(\mathbf{x})$ denotes the mean map (first moment),

$$\bar{\mathbf{m}}_1(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\epsilon}}[\mathbf{m}(\mathbf{x})] = \int p(\boldsymbol{\epsilon})\mathbf{m}(\mathbf{x})d\boldsymbol{\epsilon}, \tag{13}$$

and $\bar{\mathbf{m}}_0(\mathbf{x}) = 1 - \bar{\mathbf{m}}_1(\mathbf{x})$. For the 2nd expectation in (10),

$$\mathbb{E}_{\boldsymbol{\epsilon}}[g(\boldsymbol{\epsilon})^2] = \sum_{\mathbf{x},\mathbf{x}',b,b'} \mathbf{V}_{b,b'}(\mathbf{x},\mathbf{x}')\mathbf{h}_b(\mathbf{x})\mathbf{h}_{b'}(\mathbf{x}'), \tag{14}$$

where $\mathbf{V}_{b,b'}$ is a 2nd-moment matrix,

$$\mathbf{V}_{b,b'}(\mathbf{x},\mathbf{x}') = \mathbb{E}_{\boldsymbol{\epsilon}}[\mathbf{m}_b(\mathbf{x})\mathbf{m}_{b'}(\mathbf{x}')]. \tag{15}$$

We compute the mean map $\bar{\mathbf{m}}_b$ and 2nd moment matrix $\mathbf{V}_{b,b'}$ via sampling to approximate the expectation. Note that these only need to be computed once for a given noise variance α. An example is shown in Fig. 5.

3.4.3 Noise-Robust Loss

By substituting for the 1st and 2nd-order terms in (12) and (14) into (10), and removing the log which is monotonically increasing and does not affect the optimization, we obtain our noise-robust (NR) loss function:

$$\mathcal{L}_{NR}(\mathbf{f}) = -\sum_{b} \bar{\mathbf{m}}_b^T\mathbf{h}_b - \frac{1}{2}\sum_{b,b'} \mathbf{h}_b^T\mathbf{V}_{b,b'}\mathbf{h}_{b'}, \tag{16}$$

where $\mathbf{h}_b = [\mathbf{h}_b(\mathbf{x})]_{\mathbf{x}}$ and $\bar{\mathbf{m}}_b = [\bar{\mathbf{m}}_b(\mathbf{x})]_{\mathbf{x}}$ are the vectorized maps, and $\mathbf{V}_{b,b'} = [\mathbf{V}_{b,b'}(\mathbf{x},\mathbf{x}')]_{\mathbf{x},\mathbf{x}'}$ is the matrix of the 2nd moments. In the temporal correspondence learning stage, the mined patch pairs (including templates and searching images) are used to train the previously learned BD model using the proposed NR loss. In addition, the contrastive loss in (4) is jointly used for training. The final loss for one mini-batch training is:

$$\mathcal{L}_{all} = \mathcal{L}_{NR}(\mathbf{f}) + \lambda\mathcal{L}_c(\mathbf{f}), \tag{17}$$

where λ is a weighting parameter.

3.5. Online Tracking

After training backbone networks using the proposed PUL, the learned backbone networks can be naturally integrated to a Siamese tracker without any modifications. For online tracking, to better measure the learned unsupervised representation, we do not apply any online adaptation or fine-tuning, keeping the same tracking steps used in the original trackers, e.g., SiamFC and SiamDW.

Table 1: Consistent improvements of AUCs and EAOs (AUC/EAO) achieved by BD learning (BDL), AHM and TC learning (TCL) with different mining lengths (10 and 20 frames) on OTB-13 and VOT16.

BDL	AHM	TCL-10	TCL-20	AlexPUL	ResPUL
✓				0.491/0.151	0.580/0.202
✓	✓			0.515/0.165	0.598/0.237
✓	✓	✓		0.567/0.204	-
✓	✓	✓	✓	**0.574/0.219**	**0.620/0.263**

4. Experiments

In this section, we compare PUL-based trackers (PULT) with state-of-the-art trackers, including both unsupervised and supervised trackers, and present an ablation study.

4.1. Implementation Details

We use two backbone networks: the Alex-net-like backbone network in SiamFC [4], and a CIResNet-22 backbone network in SiamDW [65]. For fair comparison with the supervised baselines, we use the same network initial parameters as SiamFC and SiamDW, and further train the backbone networks using our proposed PUL. We denote our PUL-based trackers as AlexPUL and ResPUL. For BD learning, we set temperature $\tau = 0.5$ and scale $E = 5$. We use the AlexPUL for temporal mining, setting the initial maximum mining length to 10 frames, and generating 1 million patch pairs. Each patch is resized to 127×127. Note that the mining only needs to be performed once offline. For TC learning, we set weight $\lambda = 1$ and further train BD-SiamFC. We then set a larger mining length of 20 frames to repeat. For ResPUL, we directly use all the patch pairs from both mining rounds for training, in order to avoid overfitting on easy patch pairs. In NR loss, we set variance $\alpha = 0.5$. We use mini-batches of 196, and Adam optimizer [23] with learning rates 1e-4 and 1e-5 for AlexPUL and ResPUL. The weight decay is 5e-4.

4.2. Ablation Study

We first conduct ablation studies on the components of PUL and the NR loss on the OTB-13 and VOT16 datasets.

4.2.1 Components of PUL

The results of incrementally using the components of PUL on OTB-13 and VOT-16 are presented in Table 1. In the first stage, BD learning obtains good initial unsupervised representations for both AlexPUL and ResPUL from only static images. ResPUL obtains competitive AUC (0.580) and EAO (0.202) due to its larger model capacity. Although AlexPUL is worse than ResPUL, it still has comparable performance to supervised SiamFC on short video clips (see Fig. 3), which shows its suitability as a temporal miner.

Next, using the proposed AHM further boosts the feature discrimination in AlexPUL and ResPUL without introducing new training data, yielding gains of 2.4% and 1.8% on AUC. We also visualize the mini-batch training samples selected by our AHM and the original random selection in the Supp. C. The visualization suggests that our AHM can select similar samples for contrastive learning, which learns more discriminative features for visual tracking.

Finally, adding TC learning with the proposed NR loss further improves the performance. For AlexPUL, training on mined patch-pairs with frame gap 10 (TCL-10) yields large improvement (5.2%) over the previous stage (BD+AHM). Enlarging the mining length to 20 frames (TCL-20) improves AlexPUL to AUC 0.574 and EAO 0.219, which is comparable to the supervised SiamFC (0.608/0.235). Note that SiamFC selects training patch pairs with a much larger frame gap (100 frames). Fig. 3 plots the performance of TCL-10 and TCL-20 versus video clip length, and shows that using TC learning steadily improves tracking in longer video clips. The deeper backbone of ResPUL also benefits from TC learning, improving the AUC 2.2% and EAO 2.6% compared to BD+AHM. These results show that, through TC learning, our model successfully learns the temporal variations of the target for tracking, allowing longer successful tracks.

Table 2: The distance precision and AUC (DP/AUC) obtained by AlexPUL trained using our NR loss or the traditional BCE loss for temporal correspondence learning on OTB-13.

Loss	Baseline	TCL-10	TCL-20
BCE	0.693 / 0.515	0.714 / 0.530	0.710 / 0.526
Ours	0.693 / 0.515	**0.742 / 0.567**	**0.758 / 0.574**

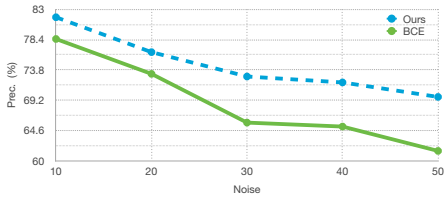

Figure 6: Tracking results when training with spatial annotation noise using traditional binary cross entropy (BCE) loss and our noise-robust loss.

4.2.2 Robustness to spatial annotation noise

We conduct two experiments to demonstrate our proposed NR loss can well handle noisy spatial annotations. First, we use our NR loss and traditional BCE to train the baseline BD model of AlexPUL on the mined patch pairs. The results are presented in Table 2. For TCL-10, our NR loss improves the BD model with large gains of 4.9% and 5.2% in terms of precision and AUC, while the BCE loss achieves very limited improvements. Increasing to TCL-20 yields further improvements when training with our NR loss, even though there is likely more spatial noise. Training with BCE loss suffers performance degradation in the presence of noise.

In the second experiment, we compare our NR loss to BCE on supervised training of SiamFC with noisy annotations. We generate a noisy dataset by randomly moving the annotated target center by $\{10, 20, 30, 40, 50\}$ pixels in the

Table 3: Comparison with state-of-the-art unsupervised and supervised deep trackers on four tracking benchmarks. The best unsupervised performance is highlighted. U, S, O indicate frameworks using **U**nsupervised, **S**upervised, and **O**nline-updating.

Trackers	Year	U	S	O	OTB '13	OTB '15	VOT16 A↑	VOT16 R↓	VOT16 EAO↑	VOT17/18 A↑	VOT17/18 R↓	VOT17/18 EAO↑
SiamFC [4]	2016	✗	✓	✗	0.609	0.582	0.530	0.461	0.235	0.503	0.585	0.188
SiamDW [65]	2019	✗	✓	✗	0.663	0.644	0.540	0.380	0.303	0.500	0.490	0.234
SiamRPN++ [29]	2019	✗	✓	✗	-	0.696	0.637	0.177	0.478	0.600	0.234	0.414
ROAM [63]	2020	✗	✓	✓	-	0.681	0.543	0.195	0.380	0.599	0.174	0.441
UDT [51]	2019	✓	✗	✓	0.620	0.594	0.539	0.475	0.225	0.472	0.932	0.129
LUDT [52]	2020	✓	✗	✓	**0.641**	**0.602**	0.540	-	0.231	0.460	-	0.154
S_2SiamFC [45]	2020	✓	✗	✗	-	-	0.493	-	0.215	0.463	0.782	0.180
CycleSiam [2]	2020	✓	✗	✗	-	-	0.540	0.735	0.191	0.377	0.750	0.131
AlexPUL	**Ours**	✓	✗	✗	0.574	0.551	0.548	0.545	0.219	0.515	0.693	0.182
ResPUL	**Ours**	✓	✗	✗	0.620	0.584	**0.554**	**0.405**	**0.263**	**0.516**	**0.660**	**0.203**

image space. Note that the total stride in the backbone of SiamFC is 8. Here we use the original SiamFC framework and only change the loss function. The results on OTB-13 are plotted in Fig. 6. Our NR loss better handles the spatial annotation noise, achieving higher AUC scores compared to the BCE loss. This demonstrates the robustness of our NR loss to various levels of spatial annotation noise.

4.3. Comparison with Unsupervised Deep Trackers

We compare our AlexPUL and ResPUL trackers with state-of-the-art unsupervised deep trackers, UDT [51], LUDT [52], CycleSiam [2] and S_2SiamFC [45]. We evaluate on OTB13/15 [57] and VOT16/17/18 [26, 25] datasets, since most of the unsupervised trackers are evaluated on these datasets. Performance is measured with AUC on OTB, and Accuracy (A), Robustness (R) and expected average overlap (EAO) on VOT. For completeness, we also compare with the supervised baselines (SiamFC and SiamDW) and two recently proposed supervised deep trackers, i.e., SiamRPN++ [29] and ROAM [63].

The test results are presented in Table 3. Compared with existing state-of-the-art deep unsupervised trackers, our ResPUL tracker can obtain the leading performance on these VOT16/17/18. Note that UDT is a recently proposed deep unsupervised tracker, which uses a correlation filter for online updating. LUDT further improves UDT by trajectory combination and training sample selection. Without applying any online fine-tuning or adaptation steps, our trackers ResPUL and AlexPUL achieve competitive results to UDT and LUDT on the OTB datasets, and better performance than UDT/LUDT on VOT16/17/18, which demonstrates the generalization ability of our offline learned unsupervised representations. Our AlexPUL and ResPUL significantly outperform S_2Siam and CycleSiam, which do not apply online updating, on the four benchmarks.

Our unsupervised AlexPUL and ResPUL achieve comparable results to supervised baselines SiamFC and SiamDW. Specifically, our AlexPUL tracker can obtain higher accuracy than SiamFC on VOT2016 (0.548 vs. 0.530). ResPUL also outperforms SiamDW in terms of accuracy on VOT16/17/18 datasets. However the EAOs of our trackers are still inferior to the supervised baselines, which is mainly caused by more tracking failures. The learned temporal representations of our trackers are still limited, compared to SiamFC and SiamDW, which are trained on larger frame gaps between patches, resulting larger appearance variations. Our trackers have potential to be improved by using more temporal mined samples.

Table 4: Comparison with state-of-the-art trackers on TrackingNet [38] and VOT19 [27]. Supervised methods are italicized.

TrackingNet	*MDNet* [39]	ECO [8]	ECO-HC [8]	*SiamFC* [4]	BACF [14]	LUDT [52]	**ResPUL**
Precision	0.565	0.492	0.476	0.533	0.461	0.469	0.485
Norm Prec.	0.705	0.618	0.608	0.663	0.580	0.593	0.630
Succes	0.606	0.554	0.541	0.571	0.523	0.543	0.546

VOT19	*DRNet* [27]	*SiamFCOSP* [27]	*ATOM* [9]	*SiamRPN* [30]	*RSiamFC* [27]	**ResPUL**
EAO↑	0.395	0.171	0.292	0.224	0.163	0.198
R↓	0.261	1.194	0.411	0.552	0.958	0.828
A↑	0.605	0.508	0.603	0.517	0.470	0.515

4.4. Comparison with State-of-the-art Trackers

In this subsection, we compare our unsupervised ResPUL tracker with state-of-the-art tracker on the TrackingNet [38] and VOT19 [27] datasets in Table 4.

TrackingNet is a large-scale tracking dataset with 511 test videos. Each test video only gives the ground-truth at the first frame, and our results are obtained via the official online evaluation server. Even without online updating, our ResPUL can outperform the state-of-the-art online updated unsupervised tracker (LUDT) in terms of all the three metrics on TrackingNet. Meanwhile, our ResPUL performs favorably against supervised ECO and SiamFC.

VOT19. Our ResPUL achieves favorable performance on VOT19 compared to supervised trackers, e.g., RSiamFC and SiamFCOSP, which are based on fully supervised SiamFC. Our ResPUL outperforms them in terms of all the three metrics, which shows the effectiveness of our unsupervised representation learning.

5. Conclusion

In this paper, we propose a progressive unsupervised learning (PUL) framework for unsupervised representation learning in visual tracking. The proposed PUL framework formulates the representation learning problem in visual tracking as a combination of background discrimination and temporal correspondence learning. We propose a noise-robust loss function to more effectively learn from the noisy patch-pairs mined for TCL. We show that the proposed PUL framework can effectively learn unsupervised representations in Siamese tracking frmeworks. Our Siamese unsupervised trackers can achieve state-of-the-art unsupervised tracking performance without applying any online adaptation or fine-tuning steps.

References

[1] David Acuna, Amlan Kar, and Sanja Fidler. Devil is in the edges: Learning semantic boundaries from noisy annotations. In *CVPR*, pages 11075–11083, 2019.

[2] W. Yuan adn Y. Micheal and Q. Chen. Self-supervised object tracking with cycle-consistent siamese networks. In *arxiv:2008.00637*, 2020.

[3] Shuai Bai, Zhiqun He, Yu Qiao, Hanzhe Hu, Wei Wu, and Junjie Yan. Adaptive dilated network with self-correction supervision for counting. In *CVPR*, pages 4594–4603, 2020.

[4] L. Bertinetto, J. Valmadre, J.F. Henriques, A. Vedaldi, and P.H.S. Vedaldi. Fully-convolutional siamese networks for object tracking. In *ECCVW*, pages 850–865, 2016.

[5] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte. Leanring discriminative model prediction for tracking. In *ICCV*, pages 6182–6191, 2019.

[6] G. Bhat, J. Johnander, M. Danelljan, F. S. Khan, and M. Felsberg. Unveiling the power of deep tracking. In *ECCV*, pages 483–498, 2018.

[7] T. Chen, S. Kornblith, and M. Norouzi. A simple framework for contrastive learning of visual representations. In *arXiv:2002.05709*, 2020.

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

[9] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. Atom: Accurate tracking by overlap maximization. In *CVPR*, pages 4660–4669, 2019.

[10] M. Danelljan, L.V. Gool, and R. Timofte. Probabilistic regression for visual tracking. In *CVPR*, pages 7183–7192, 2020.

[11] X. Dong and J. Shen. Triplet loss in siamese network for object tracking. In *ECCV*, 2018.

[12] H. Fan, L. Lin, and F. Yang. Lasot: A high-quality benchmark for large-scale single object tracking. In *CVPR*, pages 5374–5383, 2019.

[13] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *arXiv:1605.07157*, 2016.

[14] H. Galoogahi, A. Fagg, and S. Lucey. Learning background-aware correlation filters for visual tracking. In *ICCV*, 2017.

[15] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *AAAI*, pages 1919–1925, 2017.

[16] K. He and H. Fan adn Y. Wu. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020.

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[18] L. Huang, X. Zhao, and K. Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[19] Lu Jiang, Di Huang, Mason Liu, and Weilong Yang. Beyond synthetic noise: Deep learning on controlled noisy labels. In *ICML*, 2020.

[20] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

[21] Naoki Kato, Tianqi Li, Kohei Nishino, and Yusuke Uchida. Improving multi-person pose estimation using label correction. *arXiv preprint arXiv:1811.03331*, 2018.

[22] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*, pages 5574–5584, 2017.

[23] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *arXiv:1412.6980*, 2014.

[24] K. Krafka and A. Kellnhofer. Eye tracking for everyone. In *CVPR*, 2016.

[25] M. Kristan, A. Leonardis, J. Matas, and M. Felsberg. The sixth visual object tracking vot2018 challenge results. In *ECCVW*, 2018.

[26] M. Kristan, A. Leonardis, J. Metas, M. Felsberg, R. Pflugfelder, and L. Cehovin. The visual object tracking vot2016 challenge results. In *ICCVW*, pages 777–823, 2016.

[27] M. Kristan, J. Matas, and A. Leonardis. The seventh visual object tracking vot2019 challenge results. In *ICCVW*, 2019.

[28] A. Krizhevsky, S. Ilya, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Communications of the ACM*, pages 84–90, 2017.

[29] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019.

[30] B. Li, W. Wu, Z. Zhu, and J. Yan. High performance visual tracking with siamese region proposal network. In *CVPR*, pages 8971–8980, 2018.

[31] Hengduo Li, Zuxuan Wu, Chen Zhu, Caiming Xiong, Richard Socher, and Larry S Davis. Learning from noisy anchors for one-stage object detection. In *CVPR*, pages 10588–10597, 2020.

[32] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Learning to learn from noisy labeled data. In *CVPR*, pages 5051–5059, 2019.

[33] X. Li, S. Liu, and S. De. Joint-task self-supervised learning for temporal correspondence. In *International Conference on Neural Information Processing*, pages 318–328, 2019.

[34] Z. Liu, R. Yeh, and X. Tang. Video frame synthesis using deep voxel flow. In *ICCV*, pages 4463–4471, 2017.

[35] G. Lorre, J. Rabarisoa, and A. Orcesi. Temporal contrastive pretraining for video action recognition. In *WACV*, 2020.

[36] I. Misra, C. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, pages 527–544, 2016.

[37] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for uav tracking. In *Proceedings of the ECCV*, pages 445–461, 2016.

[38] M. Muller, A. Bibi, and Giancola S. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, pages 300–317, 2018.

[39] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, pages 4293–4302, 2016.

[40] Natalia Neverova, David Novotny, and Andrea Vedaldi. Correlated uncertainty for learning dense correspondences from noisy labels. In *NeurIPS*, pages 920–928, 2019.

[41] Duc Tam Nguyen, Thi-Phuong-Nhung Ngo, Zhongyu Lou, Michael Klar, Laura Beggel, and Thomas Brox. Robust learning under label noise with iterative noise-filtering. *arXiv preprint arXiv:1906.00216*, 2019.

[42] E. Real, J. Shlens, and S. Mazzocchi. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *CVPR*, pages 5296–5305, 2017.

[43] O. Russakovsky, J. Deng, and et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[44] Yunhang Shen, Rongrong Ji, Zhiwei Chen, Xiaopeng Hong, Feng Zheng, Jianzhuang Liu, Mingliang Xu, and Qi Tian. Noise-aware fully webly supervised object detection. In *CVPR*, pages 11326–11335, 2020.

[45] C. Sio and Y. Ma. S2siamfc: Self-supervised fully convolutional siamese network for visual tracking. In *ACM Multimedia*, pages 1948–1957, 2020.

[46] R. Tao, E. Gavves, and A. W.M. Smeulders. Siamese instance search for tracking. In *CVPR*, pages 1420–1429, 2016.

[47] Y. Tian, S. Kornblity, and K. Swersky. Big self-supervised models are strong semi-supervised learners. In *Neurips*, 2020.

[48] Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding. In *arXiv:1906.05849*, 2019.

[49] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, pages 839–847, 2017.

[50] P. Voigtlaender, J. Luiten, and P.H.S. Torr. Visual tracking by re-detection. In *CVPR*, pages 6578–6588, 2020.

[51] N. Wang, Y. Song, and C. Ma. Unsupervised deep tracking. In *CVPR*, pages 3708–1317, 2019.

[52] N. Wang, Y. Song, and C. Ma. Unsupervised deep representation learning for real-time tracking. *IJCV*, pages 1–19, 2020.

[53] Q. Wang, J. Gao, and J. Xing. Dcfnet: Discriminant correlation filters network for visual tracking. In *arXiv:1704.04057*, 2017.

[54] Q. Wang, M. Zhang, J. Xing, J. Gao, W. Hu, and S. Maybank. Do not lose the details: Reinforced representation learning for high performance visual tracking. In *IEEE Conference on International Jont Conference on Artifical Intelligence*, pages 985–991, 2018.

[55] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, pages 2794–2802, 2020.

[56] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *ICCV*, pages 322–330, 2019.

[57] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[58] Z. Wu, Y. Xiong, and S. Yu. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, pages 3733–3742, 2018.

[59] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, pages 2691–2699, 2015.

[60] Y. Xu and Z. Wang adn Z. Li. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, pages 12549–12556, 2020.

[61] Longrong Yang, Fanman Meng, Hongliang Li, Qingbo Wu, and Qishang Cheng. Learning with noisy class labels for instance segmentation. In *ECCV*, 2020.

[62] T. Yang and A. B. Chan. Learning dynamic memory networks for object tracking. In *ECCV*, pages 152–167, 2018.

[63] T. Yang, P. Xu, and R. Hu. Roam: Recurrently optimizing tracking model. In *CVPR*, pages 6718–6727, 2020.

[64] Lichao Zhang, Abel G.-Garcia, Joost van de Weijer, Martin Danelljan, Fahad Shahbaz, and Fahad Shahbaz Khan. Learning the model update for siamese trackers. In *arXiv:1908.00855*, 2019.

[65] Z. Zhang and H. Peng. Deeper and wider siamese networks for raal-time visual tracking. In *CVPR*, 2017.

[66] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *CVPR*, 2019.

[67] C. Zitnick and P. Dollar. Edge boxes: Locating object proposals from edges. In *ECCV*, pages 391–405, 2014.