Adaptive Siamese Tracking with a Compact Latent Network

Xingping Dong, Jianbing Shen, *Senior Member, IEEE*, Fatih Porikli, *Fellow, IEEE*, Jiebo Luo, *Fellow, IEEE*, and Ling Shao, *Fellow, IEEE*

Abstract—In this paper, we provide an intuitive viewing to simplify the Siamese-based trackers by converting the tracking task to a classification. Under this viewing, we perform an in-depth analysis for them through visual simulations and real tracking examples, and find that the failure cases in some challenging situations can be regarded as the issue of *missing decisive samples* in offline training. Since the samples in the initial (first) frame contain rich sequence-specific information, we can regard them as the decisive samples to represent the whole sequence. To quickly adapt the base model to new scenes, a compact latent network is presented via fully using these decisive samples. Specifically, we present a statistics-based compact latent feature for fast adjustment by efficiently extracting the sequence-specific information. Furthermore, a new diverse sample mining strategy is designed for training to further improve the discrimination ability of the proposed compact latent network. Finally, a conditional updating strategy is proposed to efficiently update the basic models to handle scene variation during the tracking phase. To evaluate the generalization ability and effectiveness and of our method, we apply it to adjust three classical Siamese-based trackers, namely SiamRPN++, SiamFC, and SiamBAN. Extensive experimental results on six recent datasets demonstrate that all three adjusted trackers obtain the superior performance in terms of the accuracy, while having high running speed.

Index Terms—Compact latent network, Siamese networks, decisive samples, visual object tracking.

1 Introduction

Visual object tracking is a critical task in computer vision, with many applications such as automated surveillance, human-computer interaction, and vehicle monitoring [2]. The objective of this task is to track a target, specified in the first few frames, throughout a video sequence. As with many other tasks of computer vision [3], [4], [5], recent years have witnessed the huge success of deep learning methods in object tracking.

In the early years, researchers in the field focused on incorporating online correlation-filter-based models with deep features pre-trained on recognition tasks [3]. With the increase in labeled video data, more and more works [6], [7], [8] have attempted to re-train deep models on large-scale video datasets to obtain tracking-specific features, with SiamFC [6] being one of the pioneering models. This work first introduced the large-scale video object detection dataset ILSVRC2015 [9], and then successfully incorporated data-driven deep learning with real-time visual tracking, attracting significant attention in the tracking community. Several works have since followed, including [10], [7], [8], [11], [12], [13]. For instance, Li *et al.* [7] introduced the Region Proposal Network (RPN) into the Siamese framework, avoiding the requirement of multiple computations for the scale estimation of SiamFC, and achieving faster speed (160 frame-per-second (FPS)) and better tracking performance. Many variants based on this have also been proposed, such as SiamRPN++ [12], which further enhances the accuracy while maintaining the high speed.

However, these Siamese-based trackers are still unable to address more challenging situations, such as the presence of similar distractors or significant deformation. We provide a new and intuitive method to analyze the underlying reason for these failures. Specifically, we simplify the Siamese-based model to a linear binary-classifier and convert the tracking task to a classification problem. For example, as shown in the tracking results in Fig. 1, a Siamese-based model (such as SiamRPN++ [12]) usually produces many boxes (yellow dotted boxes) with classification scores as candidates and then selects the box with the highest score as the final tracking result (yellow solid box). We can regard the image patches under these boxes as candidate classification samples. Then, the tracking problem is transferred to classifying these candidate samples. In this case, we observe that above failures come from the issue of *missing decisive samples*, *i.e.* some key samples, such as the samples in these challenging cases, are rare or unseen during offline training. This will lead to the poor discriminative ability of the trained model for these samples. This is a common problem with most data-driven models, since they do not capture all the data used

- *This work was supported in part by the FDCT grant SKL-IOTSC(UM)-2021-2023, the National Natural Science Foundation of China (No. 61929104), the Start-up Research Grant (SRG) of University of Macau (SRG2022-00023-IOTSC), and HPC resources in IIAI.*

- *X. Dong is with the Inception Institute of Artificial Intelligence, Abu Dhabi, UAE. (email: xingping.dong@gmail.com)*
- *J. Shen is with the State Key Laboratory of Internet of Things for Smart City, Department of Computer and Information Science, University of Macau, Macau, China. (email: shenjianbingcg@gmail.com)*
- *F. Porikli is with the Research School of Engineering, the Australian National University. (email: fatih.porikli@anu.edu.au)*
- *J. Luo is with the Department of Computer Scienece, University of Rochester. (email: jiebo.luo@gmail.com)*
- *L. Shao is with Terminus AI Lab, Terminus Group, China. (email: ling.shao@ieee.org)*

- *A preliminary version of this work has appeared in ECCV 2020 [1].*
- *Corresponding author: Jianbing Shen*

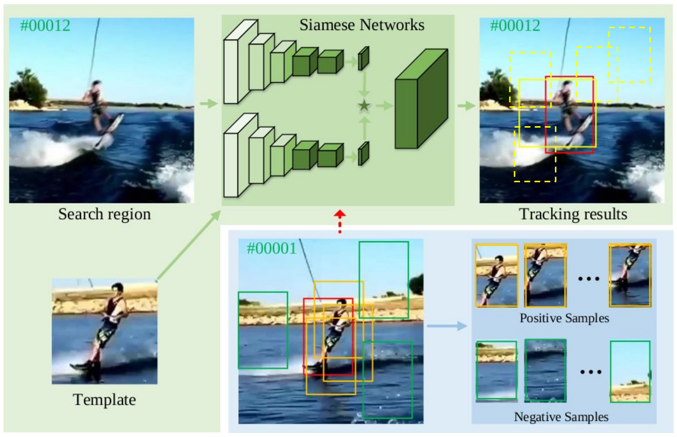

Fig. 1. **Illustration of tracking procedure (green panel) and ignoring context (blue panel) of a Siamese-based model (SiamRPN++ [12]).** In the green panel, the red solid box is the ground truth, the yellow one is the final predicted bounding box (bbox) and the yellow dotted boxes are candidate samples. The blue panel shows the sequence-specific samples, which are extracted from the first frame. Green and orange rectangles indicate the positive and negative bboxes, respectively. However, most Siamese-based approaches ignore these sequence-specific samples in the first frame, leading to failures in challenging cases, such as frame #0012 in this example.

for training. In contrast, in the tracking task, each sequence provides annotated bounding boxes (bboxes) in the first frame. These boxes can generate sequence-specific samples and improve the recognition ability of the model, since these samples include most foreground and background information in this sequence. For example, in Fig. 1, we randomly draw several bboxes in the first frame to produce the sequence-specific samples, which are labeled as positive or negative ones according to their overlap with the ground-truth. These samples, such as 'sky' or 'sea' patches in our example, can provide important context in other frames (frame #0012) to facilitate the tracking. However, most Siameses-based methods only apply annotations to extract templates, thus ignoring the rich contextual information. Therefore, these models can be tuned with neglected contextual information, improving their discriminative ability for all samples in the sequence.

However, fully utilizing sequence-specific information to tune a model trained offline is a challenging task, especially for real-time tracking, since it incurs computational load. A simple solution is to use an optimization method such as stochastic gradient descent (SGD) [14], ride regression [15], or Lagrangian multipliers [16], directly retrain the Siamese-based model. However, these methods are usually impractical and time-consuming for tracking.

To extract effective information from sequence-specific samples and meet real-time requirements, a compact latent network (CLNet) is proposed to tune recent Siamese-based trackers. The proposed CLNet consists of a feature-adjusting subnetwork, a latent encoder, and a prediction subnetwork. The first module is built by three 1×1 convolutional layers for efficiency, and provides the adjusted feature for each sequence-specific sample. The core module, the latent encoder, generates a compact feature representation for the entire set of sequence-specific samples by computing the statistical information inside the positive and negative adjusting feature sets, respectively. Next, the latent features are fed into the last subnetwork (a three-layer perceptron) to predict the adjusting parameters.

It is worth mentioning that our statistics-based latent features are compact and only with a few thousand parameters. Furthermore, they include more uncertainty information than standard features, which is beneficial for performance improvement [17]. In addition, a new diverse sample mining algorithm is proposed for training to further enhance the performance. In contrast to the previous training methods [12], we aim to effectively capture the sequence-specific information to train the latent feature. Thus, for each batch, we adopt image-pairs sampled from a sequence. To handle significant appearance changes and similar distractors, we mine the diverse samples to enhance the training performance.

To verify the effectiveness, we adopt three representative trackers, SiamRPN++ [12], SiamFC [6], and SiamBAN [18], as our baselines. To keep the generalization ability of the baseline models, our CLNet is applied to adjust the last layers in the regression and classification branches. For handling scene variation and fast adjustment, we propose a new conditional updating strategy to dynamically adjust basic models, by using classification scores of candidate boxes. Specifically, CLNet is only used on the first frame and the frames that suffer from scene changes, as determined by our conditional update strategy, and is ignored in other frames. Comprehensive evaluations on six popular benchmarks (in §4) demonstrate the strengths of the proposed algorithm.

Our key contributions are as follows:

- We provide an **in-depth analysis of Siamese-based trackers** and demonstrate that their poor discrimination ability under certain challenging scenarios is caused by *missing decisive samples* during training. We find that this problem can be alleviated by utilizing sequence-specific samples from the first frame.
- A **novel framework termed CLNet** is proposed to adjust deep Siamese-based trackers by efficiently capturing the sequence-specific information. To our knowledge, this is the first attempt to utilize statistics-based latent features to adjust deep trackers.
- We incorporate a **diverse sample mining** technique to facilitate network training. This technique can effectively enhance the discrimination ability of CLNet, by capturing the critical statistical information of a sequence.
- A **conditional updating strategy** is designed for fast and robust adjustment. This strategy can dynamically adjust basic models to efficiently utilize sequence-specific information and handle scene variation.
- Comprehensive experiments on six tracking benchmarks show the benefits of our CLNet, which leads to **promising improvement for three baseline trackers**: SiamRPN++, SiamFC, and SiamBAN. Moreover, our adjusted trackers can still remain high running speed over 38 FPS.

This paper significantly extends our previous work in [1], and presents several improvements with extensive discussions. First, we obtain additional details of our approach via an intuitive illustration of the missing decisive samples and ignoring context caused by current Siamese-based models to better illustrate our motivation. Second, we provide thorough discussions analyzing the core causes for the failures in Siamese-based models by conducting qualitative and quantitative analyses on real sequences. Third, we design a new conditional updating strategy to dynamically and efficiently adjust basic models for further performance improvement and remaining high speed. Fourth, we apply our method to two other representative Siamese-based trackers, SiamFC and SiamBAN, with further elaboration on the formulations and the implementation details. Fifth, we provide more experiments for ablation study to analyze the impact of adjusting regression branches and different weight augmentation approaches. Last but not least, we add two new datasets, VOT2020 [19] and GOT10k [20] and conduct extensive experiments to thoroughly examine the effectiveness in quickly adjusting Siamese-based trackers. The results on SiamRPN++, SiamFC, and SiamBAN demonstrate the generalization ability of our adjustment method.

2 Related Work

Although many methods have been proposed for visual object tracking, such as the regression model [24], [25], correlation filter [21], [22], [23], and sparse coding [26], [27]. Here, we pay attention to Siamese networks based trackers [6], [7], and meta-learning [28], [29], mostly related to our algorithm.

2.1 Siamese Networks Based Trackers

In early years, pre-trained CNN models were used to extract powerful features and incorporated with different online tracking methods [30], [31], [15], [32], [22], [33]. As the increment of the available datasets for tracking [9], [34], [35], [36], [20], more and more researchers have paid attention to designing new networks and retraining their models with tracking datasets [14], [24], [6], [37], [7], [12]. Among these approaches, trackers based on Siamese networks [32], [6] have become mainstream in recent years.

Bertinetto *et al.* [6], provided a new paradigm to train fully convolutional Siamese networks (SiamFC), by using the large-scale labeled video datasets. Specifically, SiamFC learns a similarity metric based on the Siamese networks, through offline training on a large-scale video detection dataset [9]. Then, the learned Siamese networks are directly applied to generate the score map between the target and search region for tracking, without any online training. This paradigm, separating offline training and online tracking, achieves a well balance between efficiency and accuracy. Since this work, more attentions are attracted to further mine the offline tracking model by presenting various Siamese networks [38], [39], [40], [41], using efficient Siamese networks [43], designing a powerful training loss [42], and so on [44], [45], [33]. Some researchers have focused on presenting various strategies for online updating, such as utilizing deep reinforcement learning [47], [48], [49] , learning a dynamic network [46], or incorporating a correlation filter [10]. Among these trackers, SiamRPN [7] successfully applies the region proposal network to Siamese networks, achieving impressive accuracy on the challenging VOT dataset [50], and with particularly high speed. Several following works also set up the new state-of-the-art on it [8], [11], [12], [13]. For instance, Zhu *et al.* [8] presented an incremental learning approach and a distractor-aware data augmentation method to enhance the online tracking mechanism and offline training, respectively. Since this, more researchers focus on improving the discriminative ability of the core network by designing more powerful architectures, such as a deeper networks [12], cascaded RPN [11], and wider networks [13]. These methods prefer to propose more discriminative offline models, utilizing more data. However, the contextual information of the first frame is ignored.

2.2 Meta-Learning

Meta-learning is usually interpolated as the inception of *learning-to-learn* [29], [52], [53], [54], or *fast weights* [28], [51], and can be applied to the few-shot recognition task. Recent meta-learning approaches are roughly grouped by three major categories, including 1) memory-based methods [58], [59], which investigate the storage of key training examples by encoding fast adaptation methods or utilizing effective memory architectures; 2) optimization-based methods [60], [61], which explore the adaptive parameter initialization to quickly tune the model for new tasks; and 3) metric-based methods [55], [56], [57], which focus on learning powerful similarity metrics to discriminate samples from the same class.

Although recent years witness the success of meta-learning in few-shot classification [60], [62], [63], only a few works are related with visual object tracking [64], [65]. An optimization-based meta-learner [64] is proposed for gradient-based online updating by learning an adaptive step. This approach has been used to speedup two online training trackers [14], [66], by decreasing the training iterations. However, it is not suitable for offline training models, such as Siamese-based trackers. Recently, a gradient-guided network [67] and meta-learner model [65] were presented to online update Siamese networks based trackers, by capturing target-specific information with the gradients. In contrast, our algorithm investigates a new direction to use the statistics-based latent feature to extract the sequence-specific information.

3 Siamese-based Tracker with Compact Latent Network

We adopt SiamRPN++ [12] as our basic tracker, which is a recent representative Siamese-based tracker. Its framework is briefly introduced in §3.1. Then, a deep analysis is conducted to explore the underlying issue of Siamese-based trackers, called as *missing decisive samples* (see §3.2). To alleviate this problem, an efficient compact latent network (CLNet) is proposed in §3.3, by adjusting the basic model. Furthermore, a new mining method is proposed to find the diverse training samples in §3.4. We show our framework in Fig. 2. Besides, we combine it to the more recent state-of-the-art SiamBAN [18] tracker, and classical SiamFC [6]

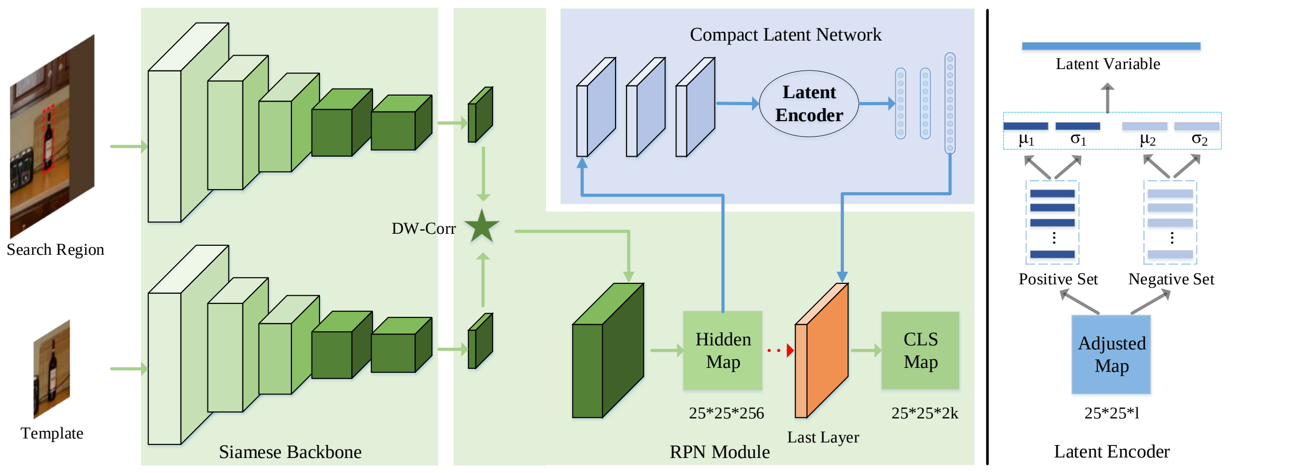

Fig. 2. **Flowchart of adjusting a Siamese-based tracker (such as SiamRPN++ [12]) via our compact latent network (CLNet).** For clarity, only the classification branch is shown. In the first frame, our CLNet predicts the weights of the last layer to adjust the basic model by using the hidden map. In the following frames, we only adjust the basic model on a few frames by using our conditional updating strategy (§3.5) to reduce computation and handle scene variation. In most frames, we omit the CLNet and apply the adjusted model for tracking. We show the latent encoder on the right, where σ and μ are the standard deviation and mean of a sample set. We generate the positive and negative sets using the annotated (predicted) bounding box in the first (previous) frame.

tracker in §3.6, to further verify the generalization ability of our approach.

3.1 Revisiting SiamRPN++ for Tracking

The pioneering SiamFC [6] regards the target patch as a template, which is given in the first frame, and the goal is to find the most similar patch from the adjacent (search) region of the following frames. SiamFC [6] construct a fully convolutional Siamese model ϕ to obtain representative features. This model includes an instance branch to process the search region $\mathbf{x}$, and a template branch to represent the target patch $\mathbf{z}$. Then the final similarity map $\mathbf{S}$ is generated by the cross-correlation (convolution) operation:

$$\mathbf{S}(\mathbf{x},\mathbf{z}) = \phi(\mathbf{x}) * \phi(\mathbf{z}) + b, \tag{1}$$

where $*$ represents the cross-correlation operation and b is the offset of the similarity value. $\mathbf{S}$[1] $\in \mathbb{R}^{w\times h}$ is the similarity metric for the searching instances and template, where w and h represent the height and width of this map. Then the target location is on the peak of $\mathbf{S}$.

SimRPN [7] incorporates Region Proposal Network (RPN) to extend SiamFC. It simultaneously makes position and scale estimation by regressing the target bounding box (bbox) with the classification and regression branches. It also uses a multi-anchors technique [4] for higher accuracy. Incorporated with Siamese networks, SiamRPN generates $w \times h \times k$ anchors, where each search position will produce k anchors on search feature map $\phi(\mathbf{x})$. The classification and regression branches provide a classification score and corresponding proposal (bbox) for each anchor. The formulations are as follows:

$$\begin{aligned}\mathbf{A}^{cls}_{w\times h\times 2k} &= conv^{fea}_{cls}(\phi(\mathbf{x})) * conv^{ker}_{cls}(\phi(\mathbf{z})),\\ \mathbf{A}^{loc}_{w\times h\times 4k} &= conv^{fea}_{loc}(\phi(\mathbf{x})) * conv^{ker}_{loc}(\phi(\mathbf{z})),\end{aligned} \tag{2}$$

1. For simplification, we remove $(\mathbf{x},\mathbf{z})$ for similar math notations.

where $conv$ represents the convolutional network providing new feature maps. Then the proposal with the highest classification score is the final target bbox. To reduce the number of parameters, SiamRPN++ [12] uses the asymmetrical depth-wise cross-correlation to provide efficient computation and also alleviates the issue caused by the distinction between the regression and classification. The formulation of new RPN block is as follows:

$$\begin{aligned}\mathbf{A}^{cls}_{w\times h\times 2k} &= head^{cls}\left(\alpha^{fea}_{cls}(\phi(\mathbf{x})) \star \alpha^{ker}_{cls}(\phi(\mathbf{z}))\right),\\ \mathbf{A}^{loc}_{w\times h\times 4k} &= head^{loc}\left(\alpha^{fea}_{loc}(\phi(\mathbf{x})) \star \alpha^{ker}_{loc}(\phi(\mathbf{z}))\right),\end{aligned} \tag{3}$$

where $\star$ is the depth-wise cross-correlation, α is an adjusted $(1\times 1$ convolutional) layer, and $head$ denotes the head block to predict maps of regression and classification.

3.2 Analysis of Siamese-Based Training Methods

Siamese-based trackers are trained by using numerous image pairs from labeled video datasets (e.g., VID [9]) as rich training samples. The generalization ability of tracking models can be effectively enhanced by these rich samples. However, the sequence-specific information is ignored, which is given by the annotation in the initial frame. Here, a simple binary-classification case is introduced to intuitively illustrate the underlying issue in the general training method. We can also regard SiamFC [6] as a binary-classification model. Specifically, the template is viewed as the classifier to classify the instance patches. Thus, our analysis is suitable for most approaches based on SiamRPN [7] or SiamFC [6].

Visualization Analysis. According to the Siamese-based training method, we assume that various negative and positive samples come from different videos in training. As shown in Fig. 3, a group of negative samples (the big green ellipse) and positive samples (the big blue ellipse) are used for training. To maintain the largest margin of the two group boundaries, the optimal decision hyperplane should be in the middle of the two groups. w^1 is assumed as the ideal decision hyperplane based on the training data in Fig. 3. In testing, we assume the testing negative and positive samples (e.g. sampled from unseen videos) do not overlap training samples, and the testing samples are usually far fewer than training samples. Therefore, the testing samples are represented by small ellipses, and the challenging samples are assumed to lie near the decision hyperplane. Some challenging samples in Fig. 3 may pass through the decision hyperplane. This indicates the trained classifier can not discriminate against these difficult samples.

If any negative sample's score is higher than some positive samples in the general classification task, an error will occur. In contrast, in the tracking task, the error occurs only when one negative sample's score is higher than all positive samples. However, there still exists the problem of an unsuitable decision hyperplane (e.g. w^1) in tracking. For example, as shown in Fig. 3, we take samples from the search region as the testing data points. Among all negative samples, we denote the point farthest away from the decision hyperplane w^1 as the red triangle. This indicates it has the highest positive score. Similarly, among all positive samples, the red star means the point with the highest score. We can find the red triangle's score is higher than that of the red star. Therefore, the red triangle is viewed as the final target, leading to tracking faults.

A simple method to alleviate the above issue is using some optimization models to retrain the classifier with the available testing data (e.g. the support vector machine). However, this method may ignore the information captured from training data and reduce the classifier's generalization, because only a few samples are available for retraining. For example, in Fig. 3, if only a few testing samples (*i.e.*, the ones in the first frame) are used for retraining, the decision hyperplane should be w^2, not the ideal $w*$ (considering testing and training data). This means that retrained classifier fails to distinguish some difficult samples, like the triangles in the up-right of ellipse (training data), even if it has learned them during training. Thus, a new solution should be explored for this issue.

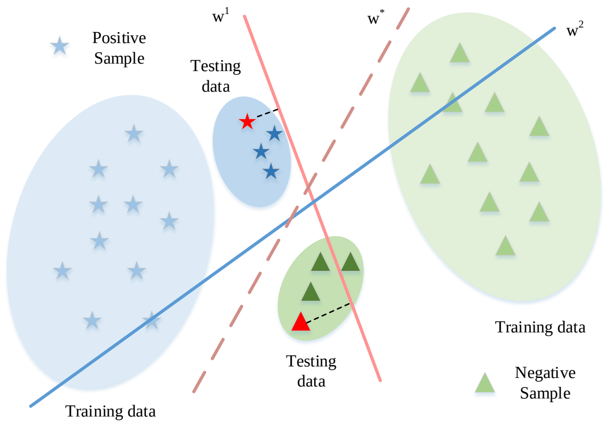

Fig. 3. **The binary-classification issue with the general Siamese-based training method.** w^1 and w^2 are the ideal decision hyperplanes based on the training and testing sets, respectively. The dotted w^* is the decision hyperplane considering both sets. The red star and triangle represent the decisive samples.

Analysis for the Tracking Case. Firstly, a metric is defined to measure the unsuitable decision hyperplane and better illustrate the issue under the tracking setting. We run the SiamRPN++ [12] tracker on the *SnowBoarding4* sequence in NfS [68], and observe the classification scores of alternative bboxes for each frame. Firstly, we group these bboxes into a positive set and a negative set. The bboxes are assigned to the positive set, if their overlap scores with ground-truth bboxes are more than or equal to 0.5. The other bboxes are classified into the negative set. Then, we select the bbox with the highest classification score as the decisive positive bbox (P-bbox) for each positive set. Similarly, we can obtain the decisive negative bbox (N-bbox) from the negative set. As shown in Fig. 4(a), we plot the P-bbox (blue rectangle) and N-bbox (red rectangle) of SiamRPN++ for frame 1 and frame 50. We also provide the classification ($P_c \in [0, 1]$ and $N_c \in [0, 1]$) and overlap (P_o and N_o) scores. When the positive set is empty, *i.e.*, the overlaps of all bboxes are less than 0.5, we set $P_c = 0$ and $N_c = 0$. This also indicates tracking faults.

Then, we can use the score difference value $D = P_c - N_c \in [-1, 1]$ to measure the classification decision hyperplane. A large D means the decision hyperplane is discriminative and can separate the samples. Conversely, a small D indicates that the hyperplane is not robust and can easily be misled by some difficult samples. When $D < 0$, it faces tracking faults, *i.e.*, the negative bbox ($N_o < 0.5$) is regarded as the target bbox. Thus, from this measurement, we find that some unexpected performance can be attributed to an unsuitable decision hyperplane. As shown in Fig. 4(c), SiamRPN++ obtains many low score differences before frame 50, resulting in accumulated error. This prevents the model from being able to track the object after frame 50 (see Fig. 4(e)).

Furthermore, we suppose the underlying reason for the unsuitable decision hyperplane is that the tracking model can not see the decisive samples in this sequence (blue and red rectangles in frame #0001 of Fig. 4(a)) during offline training, leading to high positive classification scores for negative samples (Fig. 4(c)). To verify this suppose, we adjust the tracking model by using the samples in frame #0001. As shown in Fig. 4(d), we significantly reduce the scores of negative samples and enhance the model's discriminative ability for this sequence. This indicates that the underlying reason for performance degradation can be attributed to the issue of *missing decisive samples* during offline training.

Context Ignoring. We try to alleviate the above issue and seek a reasonable decision hyperplane for tracking. Notice that in the first frame of a sequence, the target ground-truth (GT) bbox is given. It contains rich information for this sequence, which can help generate a reasonable decision hyperplane and enhance the model discrimination ability. However, most Siamese-based trackers ignore the discriminative context information and only apply GT bbox to obtain the template feature. It is vital to effectively utilize the supervision information for the model (classifier) learning. One simple strategy is using the negative and positive samples (i. e. testing data in Fig. 3) for finetuning. However, model finetuning is time-consuming because of the requirement of numerous iterations for optimization [14]. Furthermore, the limited number of

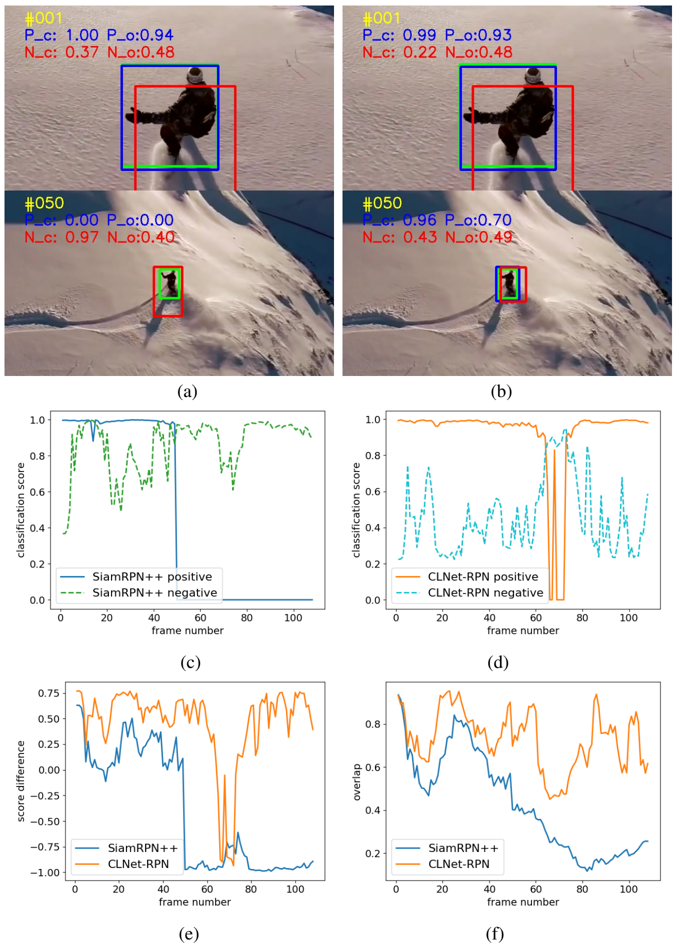

Fig. 4. **The issue in SiamRPN++ [12].** Subfigure (a) shows the decisive bounding boxes (bboxes) provided by SiamRPN++ in two frames of the *SnowBoarding4* sequence in NfS [68]. The blue rectangle represents the decisive positive bbox (P-bbox), which has the highest classification score in the positive set. The blue text shows the classification score (P_c) and overlap (P_o) with the ground-truth (green rectangle). We also obtain the decisive negative bbox (red rectangle), denoted as N-bbox, and its scores (red text). Similarly, (b) shows the decisive bboxes obtained by our CLNet-RPN. Subfigures (c) and (d) show the decisive classification scores (positive and negative) for the whole sequence, obtained by SiamRPN++ and our CLNet-RPN, respectively. Subfigure (e) plots the score differences for the two methods. Finally, (f) shows the overlap scores of the final tracking bboxes produced by SiamRPN++ and our CLNet-RPN.

samples easily results in overfitting. If only the testing data in Fig. 3 is considered, the optimal decision hyperplane will be w^2. However, it will provide the wrong predictions for some difficult training samples. In fact, it is not easy to find the ideal decision hyperplane w^* by considering both the testing and training data. We try to use a compact latent network (CLNet) to find the ideal model which can distinguish both samples in the first frame (testing data) and large-scale labeled videos (training data).

In our experiments, we find that a discriminative decision hyperplane (with large score difference D) can indeed improve the tracking performance. We employ our CLNet (see §3.3) to adjust SiamRPN++ and denote the new tracker as CLNet-RPN. As shown in Fig. 4(e)(f), our CLNet achieves larger score difference D than SiamRPN++ in many frames, and also provides better overlaps. Furthermore, we find that D is approximately proportional to the overlap, which indicates that there is an underlying connection between the discriminative decision hyperplane and the tracking performance. Besides, we also provide two visual examples in Fig. 4(b), and plot classification scores (P_c and N_c) of decisive bboxes in Fig. 4(d). Compared with Fig. 4(a)(c), we find that our CLNet significantly improves the discrimination ability for the negative samples, achieving better performance.

3.3 Compact Latent Network for SiamRPN++

The recent state-of-the-art Siamese-based model, SiamRPN++ [12], is selected as the base tracker. Then, we apply our CLNet to this basic model to demonstrate its effectiveness, denoting the new tracker as CLNet-RPN.

All layers are fixed except the last one in the regression and classification branches, to maintain the original model's generality. Note that, we only explain incorporating classification branches with our compact latent network, but the regression branches can be processed with a similar method. We omit the notation cls for simplicity. Firstly, the base model is reformulated to better explain our new module. As mentioned in §3.1, SiamRPN++ has two convolutional layers in the last head block, which can be decomposed into two components, i.e. $head = head_1(head_0)$. In the classification branch, we use a function f to denote all layers except the last one and reformulate the classification map in Eq. (3):

$$\mathbf{A} = head_1(\mathbf{M}(\mathbf{x}, \mathbf{z}); \theta_1), \tag{4}$$

where θ_1 is the parameter in $head_1$, and $\mathbf{M}(\mathbf{x}, \mathbf{z}) = head_0\left(\alpha^{fea}(\phi(\mathbf{x})) \star \alpha^{ker}(\phi(\mathbf{z}))\right)$. In fact, we view $\mathbf{M} \in \mathbb{R}^{w \times h \times c}$ as a $w \times h$ feature map, and the channel number is c.

A compact feature representation is provided to better utilize the feature map $\mathbf{M}$ via exploiting its statistical information. Firstly, the channel number c is adjusted to $\bar{c}$ to acquire the initial latent feature $\bar{\mathbf{M}}$, by using a network with three 1×1 convolutional layers, called **feature-adjusting subnetwork** g_a (see the details in §4.1). The formulation is as follows:

$$\bar{\mathbf{M}} = g_a(\mathbf{M}), \tag{5}$$

where $\bar{\mathbf{M}} \in \mathbb{R}^{w \times h \times \bar{c}}$. We set $\bar{c} \leq 2c$ to reduce the cost.

We regard this latent feature $\bar{\mathbf{M}}$ as a set containing the hidden vectors of the template-instance pair, i.e., $\bar{\mathbf{M}} = \{\bar{\mathbf{m}}_1, \bar{\mathbf{m}}_2, \cdots, \bar{\mathbf{m}}_{wh}\}$. We split this set into two groups according to the classification labels $\mathbf{Y} \in \mathbb{R}^{w \times h}$, i. e. a negative set $\mathcal{N} = \{\bar{\mathbf{m}}_1^-, \bar{\mathbf{m}}_2^-, \cdots, \bar{\mathbf{m}}_{n^-}^-\}$, and a positive set $\mathcal{P} = \{\bar{\mathbf{m}}_1^+, \bar{\mathbf{m}}_2^+, \cdots, \bar{\mathbf{m}}_{n^+}^+\}$. To capture the statistical information, the **latent encoder** operation is presented via the standard deviation σ and mean μ of these two sets:

$$\begin{aligned}\mu^\rho &= \frac{1}{n^\rho}\sum\nolimits_{i=1}^{n^\rho} \bar{\mathbf{m}}_i^\rho, \\ \sigma^\rho &= \sqrt{\frac{1}{n^\rho}\sum\nolimits_{i=1}^{n^\rho} (\bar{\mathbf{m}}_i^\rho - \mu^\rho)^2},\end{aligned} \tag{6}$$

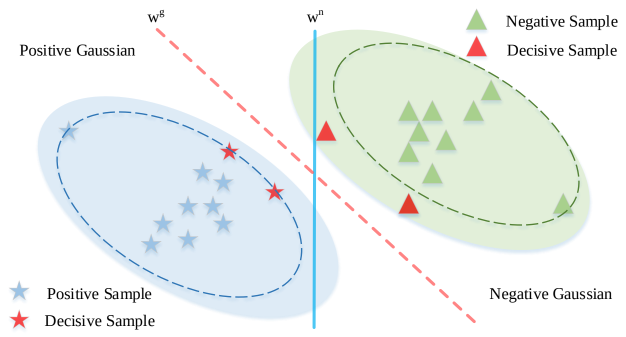

Fig. 5. **Intuitive example for the benefits of statistics-based features on the binary classification.** w^n and w^g are the decision hyperplanes decided by normal samples and statistics-based features, respectively. Dotted ellipses are the approximations of true Gaussian distributions (blue and green ellipses).

where $\rho \in \{+,-\}$, $\mu \in \mathbb{R}^l$, and $\sigma \in \mathbb{R}+^l$. We use concatenation to build the final latent feature $\mathbf{c}$, i.e.,

$$\mathbf{c} = concat(\mu^+, \sigma^+, \mu^-, \sigma^-). \tag{7}$$

In summary, we denote our latent encoder operation LE:

$$\mathbf{c} = LE(\bar{\mathbf{M}}, \mathbf{Y}). \tag{8}$$

Benefits of Statistics-Based Feature. In fact, our statistics-based feature can alleviate the *missing decisive samples* issue and is more robust than original features. An intuitive example is given in Fig. 5 to verify its powerful representative ability to reduce this issue. We assume the distributions of negative and positive sets are Gaussian distributions (represented by the blue and green ellipses), and draw several negative (green triangles) and positive (blue stars) samples. If only these drawn samples and the rule of largest margin are used to seek the decision hyperplane, a wrong hyperplane w^n will be found. In practice, w^g should be the true decision hyperplane, because the two ellipses' edges are true sample boundaries. The incorrect decision hyperplane can be attributed to *missing decisive samples*, *i.e.*, the drawn samples do not include the key samples (red triangle and star). However, if the Gaussian distribution is represented by the statistics features, *i.e.*, the standard deviation and mean of each sample set, it is easier to obtain the approximations (dotted ellipses). Then we can use them to achieve better classification performance and find out the ideal decision hyperplane, by some optimization methods based on uncertainty [17]. In summary, even we do not draw the decisive samples in this intuitive example, the correct decision hyperplane can still be found by using the statistics-based feature. Besides, note that we only require the mean and standard deviation to represent a Gaussian distribution. Thus, they can be adopted as the compact features to reduce the *missing decisive samples* issue, via unitizing the underlying distribution.

Our compact representation has two additional merits. 1) Its parameters are fewer than the original feature. Note that $\bar{c}$ has the same order of magnitude to c. In practice, the original feature's parameter number $w \times h \times c$ is usually higher than $4 \times \bar{c}$ of the compact feature $\mathbf{c}$. 2) Our compact feature is number-free. This feature can be built by a various number of negative ad positive samples. Since SiamRPN-based tracking models have different input image-pairs [7], leading to the changing number of negative or positive samples, our feature is very suitable for these models.

Given the latent compact feature $\mathbf{c}$, we use a multi-layer perceptron to build the **prediction subnetwork** g_Δ and generate a deviation for the last head layer's weights:

$$\Delta\theta_1 = g_\Delta(\mathbf{c}), \tag{9}$$

where the weight deviation predictor is g_Δ, which including three fully connected layers. We add these deviations with the weights to tune the model for various sequences:

$$\theta_a = \theta_1 + \Delta\theta_1. \tag{10}$$

Finally, we use the adjusted weight θ_a to generate the final classification map. We apply a similar weight adjustment for the regression branch, except that we construct the compact latent feature by using the positive set, because only the positive samples are available during training. Overall, the formulations of adjusted regression map $\mathbf{A}_a^{loc}$ and classification map $\mathbf{A}_a^{cls}$ are as follow:

$$\begin{aligned} \mathbf{A}_a^{cls} &= head_1^{cls}(f^{cls}(\mathbf{x},\mathbf{z});\theta_a^{cls}), \\ \mathbf{A}_a^{loc} &= head_1^{loc}(f^{loc}(\mathbf{x},\mathbf{z});\theta_a^{loc}). \end{aligned} \tag{11}$$

These maps are used to predict targets, like SiamRPN++ [12].

3.4 Training with Diverse Sample Mining

All parameters in the original model are fixed to train our compact latent networks, and the same smooth-L1 loss L^{loc} and softmax loss L^{cls} in SiamRPN++ [12] are used to adjust regression and classification maps, respectively. We formulate the final loss as follows:

$$L = L^{cls}(\mathbf{A}_a^{cls}, y^{cls}) + \lambda L^{loc}(\mathbf{A}_a^{loc}, y^{loc}), \tag{12}$$

where y^{loc} and y^{cls} represent the regression and classification ground-truths, respectively, and we set the trade-off parameter λ is as 1.2. To obtain a more discriminative feature, SiamRPN++ samples the image-pairs in one training batch from different sequences. This training method is not suitable for our adjustment. Since our goal is to predict the adaptive weights for one sequence rather than the whole dataset. We do not require the general information across sequences but aim to extract the vital information within a specific sequence during training. Besides, the local statistical information inside one sequence is more suitable for our statistics-based latent feature. Thus, several image-pairs are randomly sampled from the same sequence in each training batch .

A diverse sample mining approach is proposed to enhance the training performance, via ranking unused negative samples with their classification scores. In SiamRPN++ [12], adopts the intersection-over-union (IoU) between the ground-truth bbox and anchor are adopted to choose the negative and positive samples. When $IoU < 0.3$, the sample is negative, and when $IoU > 0.6$, it is regarded as positive. One training pair contains 64 samples, where the number of positive samples is at most 16. The diverse samples may exist in the unused negative samples, and they will benefit the discrimination ability of the model. The original score map $\mathbf{A}^{cls}$ is used to mine these diverse samples. According to the positive classification scores of $\mathbf{A}^{cls}$, the unused negative samples are sorted in descending order. We regard the first 16 negative samples as diverse samples and add them to each training pair. Therefore, the total sample number in one training pair is 80. The additional diverse samples are usually near the original model's decision hyperplane and have a high possibility to be the decisive samples. Notice that these samples are helpful for building robust compact latent features and finding the ideal decision hyperplane (See the analysis in §3.3).

3.5 Online Tracking with Conditional Updating

In our previous work [1], we only run the proposed CLNet in the first frame to capture the scene information and generate adjusting parameters for the base tracker. Then we run the adjusted tracker without any more adjustments to reduce computation load. However, this strategy cannot handle the videos with huge scene variation. Thus, we propose a simple and efficient updated method, called conditional updating (CU).

First, we need to select reliable samples as the candidates for updating, since we do not have access to the ground-truth of other frames. Specifically, a tracker usually predicts a bbox with the confidence (classification) score for each frame. If the bbox in a frame has a high score, this indicates that the target in the current frame has a similar appearance with the template (initial target) and the predicted bbox is more likely correct. If not, this means that the current target may suffer from huge appearance variation or occlusion. Extracting samples from this frame can introduce error information for the positive samples. Thus, we only use the bbox with high confidence to generate reliable samples. For simplification, we only provide the formulations for the classification branch. We give a constant threshold τ^r to select reliable frames and a candidate set $\{\mathbf{M}_c, \mathbf{b}_c\}$ to save the feature map and bbox. We denote the feature map in Eq. 4 for i-th frame as $\mathbf{M}_i$, then the candidate set is defined as

$$\{\mathbf{M}_c, \mathbf{b}_c\} = \begin{cases} \{\emptyset, \emptyset\}, & i = 1, \\ \{\mathbf{M}_c, \mathbf{b}_c\}, & i > 1, s_i \le \tau_r, \\ \{\mathbf{M}_i, \mathbf{b}_i\}, & i > 1, s_i > \tau_r, \end{cases} \tag{13}$$

where s_i is the confidence score of predicted bbox b_i.

Second, we decide when to update the tracking model. To capture the hard cases caused by huge scene variation or similar distractors, we propose a simple metric based on the classification scores. As mentioned in the visual analysis of §3.2, when the highest negative classification score is larger than the positive one, the tracking fault will occur. We utilize the margin between the highest positive and negative scores to measure the tracking difficulty for each frame. Since we do not have the access to the ground-truth bbox on other frames except the first frame, we can only use the predict bbox to generate pseudo classification labels $\bar{\mathbf{Y}}_\mathbf{i}$ (We adopt the same label generation methods from the base trackers). According to these labels, we split the classification scores into positive and negative sets. Then we can denote the margin of i-th frame as $\eta_i = s^*_{p,i} - s^*_{n,i}$, where $s^*_{p,i}$, $s^*_{n,i}$ are highest scores from positive and negative sets, respectively. Small margin means the samples in the current frame and adjacent frame are difficult to distinguish by using the current tracking model. We need to adjust the tracking model for better performance in the next frame. Thus, we set a constant threshold τ_m to decide whether updating the tracking model, *i.e.*, when $\eta_i < \tau_m$, we update the tracking model. To avoid updating frequently, we clean the candidate set, *i.e.*, $\{\mathbf{M}_c, \mathbf{b}_c\} = \{\emptyset, \emptyset\}$ after updating once. We stop to update model until we find a new reliable frame and the score margin is less than the threshold τ_m.

To distinguish the original CLNet in [1], we denote our new proposed model with conditional updating as CLNet*.

3.6 Extensions to Other Siamese-Based Trackers

We apply our CLNet* to two other representative Siamese-based trackers: SiamFC [6] and SiamBAN [18]. To distinguish the different trackers adjusted by our method, we denote them as CLNet*-RPN, CLNet*-FC, and CLNet*-BAN, which correspond to SiamRPN++, SiamFC, and SiamBAN, respectively.

CLNet*-FC. As mentioned in §3.2, for SiamFC, the template feature $\phi(\mathbf{z})$, and offset b in Eq. (1), in SiamFC can be regarded as the classifier for discriminating the instance patch in the search region. Thus, we can use our CLNet to adjust the template feature and the offset. In SiamFC, the final similarity map $\mathbf{S}$ in Eq. 1 is regarded as the feature map (like $\mathbf{M}$ in Eq. 4). This is the input of our CLNet, since we require the correlation features between the instances and template to build our CLNet and only the similarity map contains the correlation features in SiamFC. We denote the feature map as $\mathbf{M}_f = \mathbf{S}$. According to the classification labels, we can extract the latent compact feature $\mathbf{c}_f$ via Eq. 5, 6 and 7 in §3.3. A prediction subnetwork g_{Δ_f} is used to produce the deviations for the template feature and offset:

$$[\Delta\phi(\mathbf{z}), \Delta b] = g_{\Delta_f}(\mathbf{c}_f), \tag{14}$$

where g_{Δ_f} is a deviation predictor that includes three fully connected layers, similar to g_Δ in Eq. 9. We add these deviations into the corresponding template feature and offset:

$$\phi(\mathbf{z})_a = \phi(\mathbf{z}) + \Delta\phi(\mathbf{z}), \quad b_a = b + \Delta b. \tag{15}$$

Finally, the adjusted similarity map $\mathbf{S}_f$ is formulated as:

$$\mathbf{S}_f = \phi(\mathbf{x}) * \phi(\mathbf{z})_a + b_a. \tag{16}$$

Similar to SiamFC, the adjusted map is applied to predict the target. Notice that SiamFC uses all samples for training. Thus, we omit the diverse sample mining (§3.4) in our CLNet*-FC.

CLNet*-BAN. The recent SiamBAN removes the pre-defined anchors to obtain faster and better performance than its baseline SiamRPN++. Thus, the architecture of SiamBAN is very similar to SiamRPN++, and only the head part has slight differences. To obtain CLNet*-BAN, we only need to change the number of output channels in the prediction subnetwork of our CLNet. The other adjustment procedures and training method are the same as CLNet*-RPN, described in §3.3 and 3.4. The detailed structures of CLNets can be found in §4.1.

TABLE 1
Size of output feature map in each layer of our CLNets. This table provides the detailed structures of the CLNets inside our CLNet*-FC and CLNet*-RPN trackers, which are based on SiamFC-Pt [6], and SiamRPN++ [12], respectively. H, W, and C represent the height, weight, and channel number of the feature map, respectively. $\bar{c}$, $\dot{c}$, and k are set to 128, 256 and 5, respectively.

	CLNet*-FC		CLNet*-RPN			
	Classification		Classification		Regression	
	HxW	C	HxW	C	HxW	C
Input	25x25	1	25x25	256	25x25	256
Conv1	25x25	$\bar{c}$	25x25	$\bar{c}$	25x25	2*$\bar{c}$
Conv2	25x25	$\bar{c}$	25x25	$\bar{c}$	25x25	2*$\bar{c}$
Conv3	25x25	$\bar{c}$	25x25	$\bar{c}$	25x25	2*$\bar{c}$
LE	1x1	4*$\bar{c}$	1x1	4*$\bar{c}$	1x1	4*$\bar{c}$
FC1	1x1	$\dot{c}$	1x1	$\dot{c}$	1x1	$\dot{c}$
FC2	1x1	$\dot{c}$	1x1	$\dot{c}$	1x1	$\dot{c}$
FC3	1x1	25*25+1	1x1	2k*(256+1)+1	1x1	4k*(256+1)+1

4 Experimental Results

Our algorithm is implemented in Pytorch (Python 3) and evaluated on a single GPU (NVIDIA RTX 2080Ti). We use different versions of Pytorch according to the base trackers. For CLNet*-FC, we select the Pytorch-version[2] of SiamFC [6] with Pytorch 0.4.0 as the base code. This base tracker is denoted as SiamFC-Pt. For CLNet*-RPN and CLNet*-BAN, we use the official codes from SiamRPN++ [12] and SiamBAN [18], and develop our method in the recommended Pytorch 0.4.0 and Pytorch 1.3.1, respectively. We compare our trackers, CLNet*-RPN, CLNet*-FC, and CLNet*-BAN with several representative trackers on NfS [68], DTB [69], LaSOT [36], GOT10k [20], VOT2019 [70], and VOT2020 [19]. The tracking speeds of CLNet*-RPN, CLNet*-FC, and CLNet*-BAN are 38.1 FPS, 104.9 FPS, and 43.6 FPS.

4.1 Implementation Details

4.1.1 Architecture

The proposed CLNet includes a feature-adjusting subnetwork g_a, one latent encoder operation LE, and a prediction subnetwork g_Δ. g_a includes three convolutional blocks (named as Conv1, Conv2, and Conv3), which are built by a 1×1 convolution, batch-norm, and ReLU layer. g_Δ includes three fully connected layers (denoted FC1, FC2, and FC3), where the first two layers are followed by a ReLU layer and the last one is followed by a *Tanh* layer. The settings of **CLNet*-RPN** and **CLNet*-FC** are shown in Table 1. For **CLNet*-BAN**, we only need to set the anchor number $k = 1$ in Table 1.

4.1.2 Training

We follow the training protocol of the basic trackers: SiamRPN++, SiamFC-Pt, and SiamBAN, using the same training datasets. Since we want to eliminate the impact of different training data for each basic tracker, and explore whether our approach can further improve each basic tracker with the same training data. In fact, we freeze the networks in the basic trackers and only train the CLNets inside them. Thus, the number of training iterations is less than that of the basic trackers (1/10 number of SiamRPN++ and SiamBAN).

CLNet*-RPN. COCO [34], DET [9], VID [9], and YouTubeBB [35] are adopted as training datasets. We use Synchronized stochastic gradient descent (SGD) over four GPUs. The batch size is 64. Each epoch contains 60,000 training pairs. We train over 20 epochs, where the first five epochs are used for warrmup with a step learning rate from 0.001 to 0.005, and we exponentially decay the learning rate from 0.005 to 0.0005 in the last 15 epochs. The other hyperparameters are the same as for SiamRPN++.

CLNet*-FC. We use the VID [9] dataset and train our network over 30 epochs with SGD on a single GPU. In fact, the training protocol and the hyperparameters are the same as for the basic tracker, *i.e.*, the Pytorch-version of SiamFC.

CLNet*-BAN. The training datasets includes COCO [34], DET [9], VID [9], YouTubeBB [35], GOT10k [20] and LaSOT [36]. Our network is trained with SGD over four GPUs. The batch size is 112 (28 per GPU). We train over 20 epochs, each of which contains 10,000 training pairs. The other hyperparameters are the same as for SiamBAN.

4.2 Comparison on the NfS30 Dataset

Need for Speed (NfS) [68] includes 100 challenging sequences with fast-moving objects. We evaluate the basic models, SiamRPN++ [12], SiamFC-Pt [6], and SiamBAN [18], and our trackers, CLNet*-RPN, CLNet*-FC, and CLNet*-BAN, on the 30 FPS version (NfS30). We also compare with the several well-known trackers evaluated in [68], including Correlation Filter (CF) based trackers (SRDCF [71], DSST [72], KCF [21], LCT [73], SAMF [74], HCF [31], and HDT [75]), and deep trackers (MDNet [14], FCNT [30], and GOTURN [24]). As shown in Fig. 6, our CLNet*-BAN achieves the best performance in terms of precision and AUC, which are 29.8% and 31.1% higher than the best tracker, MDNet [14], reported in the original paper. Compared with the SiamBAN, our tracker also obtains promising performance improvement in both metrics. Furthermore, our CLNet*-RPN achieves significant relative gains of 10.6% and 10.2% in terms of both precision and area-under-the-curve (AUC), compared with the baseline SiamRPN++. Besides, the proposed CLNet*-FC ranks second among the original trackers in [68]. Notably, it significantly outperforms its baseline SiamFC-Pt, by large relative gains of 9.7% and 6.8% in terms of precision and AUC.

4.3 Results on the DTB70 Dataset

Our trackers are also evaluated on the DTB Dataset [69], which includes 70 videos. We report the precision and AUC plots in Fig. 7, comparing against the basic trackers, as well as other trackers reported in DTB70, such as MDNet [14], CREST [76], MEEM [77], and SRDCF [71]. Among all the trackers, our CLNet*-RPN is the best tracker, which outperforms the basic tracker, SiamRPN++, by large gains of 7.1% and 8.1% in terms of precision and AUC. In both evaluation metrics, the proposed CLNet*-BAN tracker obtains the second-best score, achieving superior performance to its

2. https://github.com/StrangerZhang/SiamFC-PyTorch

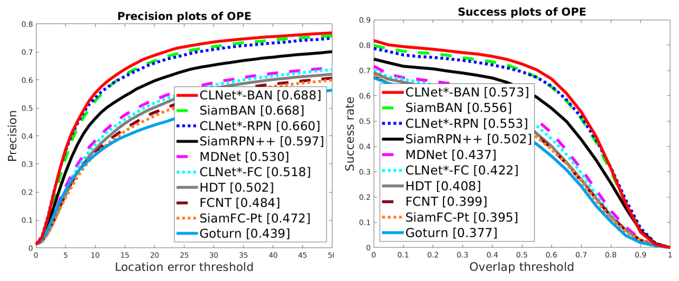

Fig. 6. **Precision and overlap success plots with AUC on the NfS30 dataset [68].** Only the top-ten trackers are shown.

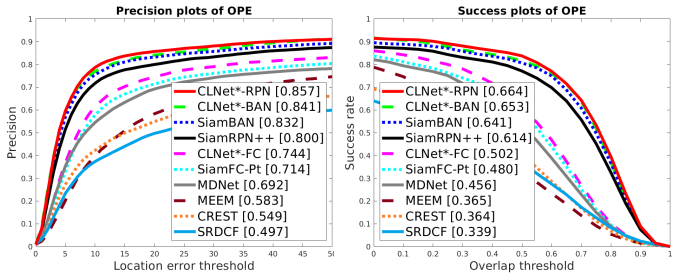

Fig. 7. **Precision and overlap success plots with AUC on the DTB70 dataset [69].** Only the top-ten trackers are shown.

baseline SiamBAN. Moreover, our CLNet*-FC also improves its baseline SiamFC-Pt with promising gains in precision and AUC, and achieves 7.5% and 10.1% higher performance than the best MDNet tracker, reported in the original benchmark [69]. We attribute the performance promotion to the proposed compact latent network.

4.4 Results on Large-Scale Datasets

LaSOT. LaSOT [36] includes 1,400 sequences with an average sequence length of 2,512 frames. We conduct experiments on the testing set of LaSOT with 280 sequences, comparing with the recent MLT [65], GradNet [67], SiamDW [13], C-RPN [11], PrDiMP [78], SiamRCNN [79], and offline Ocean [80]. Following [36], three metrics are used for evaluation, including the overlap success (AUC), precision (P), and normalized precision (P_{norm}). As shown in Table 2, all of our trackers achieve better performance than their baselines in terms of all three metrics. Especially, CLNet*-BAN outperforms the baseline SiamBAN with a large relative gain of 4.9% in terms of P_{norm}. SiamRCNN and PrDiMP outperform our best tracker CLNet*-BAN, since they have well-designed online learning approaches. While our CLNet*-BAN obtains faster speed around 43.6 FPS, compared with 4.6 FPS of SiamRCNN and 30 FPS of PrDiMP (provided by original papers).

GOT10k. GOT10k [20] contains more than 10,000 videos, convering 640+ generic classes. We conduct experiments on the testing set with 180 videos (127 frames average length), comparing with MDNet [14], GOTURN [24], MemTracker [45], PrDiMP [78], and SiamRCNN [79]. We use the average overlap (AO) and success rate (SR0.5) for evaluation. As shown in Table 3, our trackers outperform baselines in both two metrics. Notice that, CLNet*-BAN achieves significant relative gains of 5.0% and 4.5% in terms of AO and SR0.5. Our best tracker CLNet*-BAN ranks third in both AO and SR0.5, which is lower than PrDiMP and SiamRCNN. While our running speed is faster than them as mentioned before.

4.5 Comparison on the VOT Datasets

To measure the short-term tracking performance, we also evaluate our trackers on the real-time challenge provided by VOT2019 [70] and VOT2020 [19]. VOT2019 performs re-initialization of a tracker when it fails to track the target, which is the same as the previous VOT challenge. VOT2020 proposes a new anchor-based short-term tracking evaluation protocol by replacing the re-initialization with anchor-based initialization, *i.e.*, placing several anchors on each frame as initialization points. Notice that under the VOT protocol, we will re-initialize our tracker by using CLNet many times. To reduce time cost, we omit the conditional updating strategy (CU), since we have updated our model at each initialization point. We denote our trackers without CU as CLNet-FC, CLNet-RPN, and CLNet-BAN for three basic trackers.

VOT2019. We adopt the EAO, ACC and ROB under the real-time setting to compare different trackers, including our basic trackers, the top-two algorithms (SiamMargin and SiamFCOT) ranked by EAO in the real-time challenge, and several recent methods: SPM [81], SiamMask [82], SiamDW_ST [13], and DiMP [83]. As shown in Table 4, our CLNet-RPN obtains the best score in terms of ACC. Compared with its baseline SiamRPN++, the proposed method achieves a significant performance improvement of 9.8% in terms of EAO, and also boosts the ACC and ROB scores. Our CLNet-BAN performs the second best in terms of EAO and ROB among all trackers, and improves its baseline, SiamBAN, by a large gain of 10.6% in terms of EAO. The last tracker, CLNet-FC, also boosts the EAO and ROB scores of its baseline, SiamFC-Pt. These results demonstrate that our approaches can effectively improve the short-term tracking ability of the base models.

VOT2020. VOT2020 re-defines the accuracy and robustness according to the anchor-based initialization. Here, we denote them as A and R. Firstly, basic trackers are evaluated under the real-time setting. As shown in Table 5, all of them obtain lower EAO scores than VOT2019, which indicates that new evaluation is more challenging. Then we run our trackers and achieve better EAO scores than all basic trackers. This demonstrates the effectiveness under the new evaluation. Our models obtain slight improvement compared with the VOT2019 results, since VOT2020 adopts segmentation masks for evaluation, while the baselines only produce bounding boxes leading to our CLNet cannot provide proper adjustments. Besides there is still a gap between our trackers and the champion AlphaRef [19] (0.486 EAO) since our basic trackers do not combine any mask refine strategy. We also provide the details of parameters and Multiply-accumulate Operations for all models. This indicates that our models only take a few additional space and computation costs while achieving promising improvement.

TABLE 2
Comparison on the LaSOT dataset [36]. The first and second best scores are highlighted in red and blue, respectively.

	MLT [65]	GradNet [67]	SiamDW [13]	C-RPN [11]	PrDiMP [78]	SiamRCNN [79]	Ocean [80]	SiamFC-Pt [6]	CLNet*-FC **Ours**	SiamRPN++ [12]	CLNet*-RPN **Ours**	SiamBAN [18]	CLNet*-BAN **Ours**
P (%) ↑	-	35.1	-	44.3	-	-	52.6	32.1	33.3	48.5	50.1	51.9	52.9
P_{norm} (%) ↑	-	-	-	54.2	-	72.2	-	41.2	42.4	56.5	58.3	59.7	62.3
AUC (%) ↑	34.5	36.5	38.4	45.5	59.8	64.8	52.6	33.2	33.3	49.3	50.2	51.4	52.6

TABLE 3
Comparison on the GOT10k dataset [20]. The first and second best scores are highlighted in red and blue, respectively.

	MDNet [14]	GOTURN [24]	MemTracker [45]	PrDiMP [78]	SiamRCNN [79]	SiamFC-Pt [6]	CLNet*-FC **Ours**	SiamRPN++ [12]	CLNet*-RPN **Ours**	SiamBAN [18]	CLNet*-BAN **Ours**
AO (%)	35.2	41.8	46.0	63.4	64.9	34.0	35.0	51.7	54.3	55.9	56.5
SR0.5 (%)	36.7	47.5	52.4	73.8	72.8	37.4	38.5	61.6	64.4	66.5	67.5

TABLE 4
Results of the real-time setting in VOT2019 [70]. The first and second best scores are highlighted in red and blue, respectively.

	SPM [81]	SiamMask [82]	SiamDW_ST [13]	DiMP [83]	SiamFCOT [70]	SiamMargin [70]	SiamFC-Pt [6]	CLNet-FC **Ours**	SiamRPN++ [12]	CLNet-RPN **Ours**	SiamBAN [18]	CLNet-BAN **Ours**
EAO ↑	0.275	0.287	0.299	0.321	0.350	0.366	0.180	0.192	0.285	0.313	0.322	0.356
ACC ↑	0.577	0.594	0.600	0.582	0.601	0.585	0.511	0.507	0.599	0.606	0.596	0.588
ROB ↓	0.507	0.461	0.467	0.371	0.386	0.321	0.968	0.883	0.482	0.461	0.401	0.341

TABLE 5
Results of the real-time setting in VOT2020 [19]. The first and second best scores are highlighted in red and blue, respectively. Pm and MAC are Parameters and Multiply-accumulate Operations, respectively.

	SiamFC-Pt [6]	CLNet-FC **Ours**	SiamRPN++ [12]	CLNet-RPN **Ours**	SiamBAN [18]	CLNet-BAN **Ours**
EAO ↑	0.172	0.177	0.242	0.261	0.263	0.266
A ↑	0.422	0.423	0.442	0.458	0.449	0.446
R ↑	0.479	0.499	0.678	0.69	0.714	0.722
Pm (M)	2.336	3.632	53.951	61.823	53.932	57.086
MACs (G)	3.190	3.201	59.521	60.009	59.509	60.004

4.6 Ablation Study

The analysis experiments are conducted on a combined dataset including NfS30 (30 FPS version) [68] and DTB70 [69] datasets. The new dataset includes 170 videos for thorough analysis. The baseline and variants of our approach are evaluated using AUC and precision (P) metrics [84].

Analysis of Key Components. To explore the influence of additional training iterations, we firstly retrain the base tracker (**BT**), SiamRPN++, by using our training method, and evaluate it on the combined dataset. As shown in Table 6, the retrained tracker (**RT**) slightly improve AUC (**BT**: 55.8% vs **RT**: 56.1%) and precision (**BT**: 69.9% vs **RT**: 71.4%). This indicates it will not provide significant gains by simplely using additional training iterations. Furthermore, to analyze the influence of our key components, they are added to our baseline (**BT**) one by one. We first use the adjustment network removing the latent encoder to tune the base model (**+AN**). The gains in terms of AUC (1.2%) and P (1.1%) scores demonstrate that the adjustment network can extract sequence-specific information to improve the base model. When adding the latent encoder (**+LE**), our CLNet obtains further improvement with a AUC gain of 1.9% and P gain of 1.5%. This indicates the advantages of our latent encoder in extract a compact and effective feature containing the sequence-specific information. Furthermore, the image-pairs are sampled from one sequence in each batch (**+OS**) to obtain a robust latent feature. This achieves a further improvement, with 0.2% and 1.3% gains in terms of AUC and P scores, respectively. Our diverse sample mining technique (**+DM**) increases AUC and P scores by another 0.9% and 0.6%. Finally, the conditional updating (**+CU**) strategy is applied for online tracking, and provide a AUC gain of 1.0% and P gain of 1.3%. In summary, compared with our baseline, the final version (**+CU**) achieves significant relative gains of 9.3% and 8.3% in terms of AUC and P scores. Besides, to investigate the impact on the regression branch, we can remove CLNets of the regression branch. We find that only adjusting the classification branch (**-RB**) obtain lower scores with 1.0% and 1.3% reduction in terms of AUC and P scores, compared with adjusting both branches (**+DM**). The results indicate that our CLNet can also provide benefits for the regression task, demonstrating the generalization.

Different Weight Augmentation. As mentioned before, our CLNet produces a deviation of the weights of the last layers (Eq. 9) and we use a simple additive augmentation to fuse it and the original weights (Eq. 10). Do other complex augmentations work for our model? To answer this question, we propose CBAM [85] (block attention) and FILM [86] (linear transformation) weight augmentations inspired by TACT [87], which is a tracking method using different feature augmentations for context embedding. Specifically, assume the dimension of original weights θ_1 is $m \times n$, *i.e.*, $\theta_1 \in \mathbb{R}^{m \times n}$, we

TABLE 6
Analysis of key components with AUC and precision (P) on combined NfS30 [68] and DTB70 [69] dataset. The base tracker (**BT**) is SiamRPN++ [12]. **RT** represents the retrained tracker. The components contain an adjustment network (**+AN**), latent encoder (**+LE**), one sequence for training (**+OS**), diverse sample mining (**+DM**), and conditional updating (**+CU**). **-RB** means omitting the regression branch.

	RT	BT	+AN	+LE	+OS	+DM	-RB	+CU
P (%)	71.4	69.9	71.0	72.5	73.8	74.4	73.1	75.7
AUC (%)	56.1	55.8	57.0	58.9	59.1	60.0	59.0	61.0

TABLE 7
Comparison of weight augmentation in terms of AUC and precision (P) scores on combined NfS30 [68] and DTB70 [69] dataset. The base tracker is SiamRPN++ [12]. CBAM [85] and FILM [86] represent a block attention and linear transformation weight augmentation methods, respectively.

	SiamRPN++	CLNet/CBAM	CLNet/FILM	CLNet-RPN
P (%)	69.9	72.2	73.4	74.4
AUC (%)	55.8	58.4	58.7	60.0

TABLE 8
AUC Scores of recent trackers on the LaSOT dataset [36].

	TransT [88]	TransT-M [89]	STARK [90]	E.T.Track [91]	LightTrack [92]	FEAR-L [93]	CLNet*-BAN **Ours**
AUC (%)	64.9	65.4	67.1	59.1	55.5	57.9	52.6

can generate two attention maps $\delta_m \in \mathbb{R}^{m\times 1}$ and $\delta_n \in \mathbb{R}^{1\times n}$ by adjusting the output of prediction subnetwork g_Δ. Then we obtain the adjusted weight by using the CBAM augmentation as follows $\theta_a = (\theta_1 \otimes \delta_m) \otimes \delta_n$, where $\otimes$ represents element-wise multiplication. Similarly, for FILM augmentation, we generate two coefficient maps $\gamma, \beta \in \mathbb{R}^{m\times n}$. Then we can obtain $\theta_a = (\theta_1 \otimes \gamma) + \beta$.

To focus on the impacts of different augmentations on original CLNet, we omit the CU strategy and denote the tracker based on SiamRPN++ [12] as CLNet-RPN. We retrain CLNets by replacing additive augmentation with CBAM and FILM, and denote them as CLNet/CBAM and CLNet/FILM. Then we evaluate these models on the combined dataset [69], [68]. As shown in Table 7, both CLNet/CBAM and CLNet/FILM obtain promising improvements than base tracker SiamRPN++ in terms of P and AUC, which indicates that our CLNet can utilize different weight augmentations. CLNet/FILM achieves better performance than CLNet/CBAM, since it uses more parameters for adjustment. However, CLNet-RPN with simple additive augmentation can still outperform them. The reason may be that CBAM and FILM are designed for feature augmentation, and they are not suitable for weight augmentation.

4.7 More Discussions

To follow up the recent hot topic of transformer based tracking, we compare three representative trackers: TransT [88], TransT-M [89], and STARK [90] on LaSOT. As shown in Table 8, there are gaps between transformer based models and our ResNet-based model (CLNet*-BAN), since former ones have stronger backbone. Besides, we also show the results of several efficient trackers: E.T.Track [91], LightTrack [92], and FEAR-L [93], which have real-time speeds on the edge-platforms. In future work, we will try to incorporate our CLNet into these recent models with more efficient backbones, to explore a more generalized and practical solution.

5 Conclusion

We have conducted an in-depth analysis to investigate Siamese-based trackers' performance degradation via visualization and tracking examples, and found that *missing decisive samples* during training is the critical factor. To alleviate it, we present a compact latent network (CLNet) to adjust Siamese-based models. Our CLNet is able to efficiently capture sequence-specific information from the initial frame. To further improve the training of adjusted networks, a new diverse sample mining approach is designed to enhance the discrimination ability. Extensive experiments on three representative trackers have clearly demonstrated the generalization ability and effectiveness of our CLNet.

References

[1] X. Dong, J. Shen, L. Shao, and F. Porikli, "CLNet: A Compact Latent Network for Fast Adjusting Siamese Trackers," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 378–395.
[2] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, p. 13, 2006.
[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst.* , 2012.
[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Proc. Neural Inf. Process. Syst.* , vol. 28, pp. 91–99, 2015.
[5] K. He, G. Gkioxari, P. Doll¨¢r, and R. B. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017.
[6] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-Convolutional Siamese Networks for Object Tracking," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2016, pp. 850–865.
[7] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High Performance Visual Tracking With Siamese Region Proposal Network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8971–8980.
[8] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proc. Eur. Conf. Comput. Vis.* , 2018, pp. 101–117.
[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[10] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, "End-to-end representation learning for correlation filter based tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2805–2813.
[11] H. Fan and H. Ling, "Siamese cascaded region proposal networks for real-time visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7952–7961.
[12] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "SiamRPN++: Evolution of Siamese Visual Tracking With Very Deep Networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4282–4291.
[13] Z. Zhang and H. Peng, "Deeper and wider siamese networks for real-time visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4591–4600.

[14] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4293–4302.

[15] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 472–488.

[16] S. Hong, T. You, S. Kwak, and B. Han, "Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 597–606.

[17] S. Khan, M. Hayat, S. W. Zamir, J. Shen, and L. Shao, "Striking the Right Balance With Uncertainty," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 103-112.

[18] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji, "Siamese Box Adaptive Network for Visual Tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6668–6677.

[19] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. Čehovin Zajc, M. Danelljan, A. Lukezic, O. Drbohlav, L. He, Y. Zhang, S. Yan, J. Yang, G. Fernandez, and et al., "The eighth visual object tracking vot2020 challenge results," 2020.

[20] L. Huang, X. Zhao, and K. Huang, "GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no.5, pp. 1562–1577, 2019.

[21] J. F. Henriques, C. Rui, P. Martins, and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol 37, no. 3, pp. 583–596, 2014.

[22] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6638–6646.

[23] X. Dong, J. Shen, D. Yu, W. Wang, J. Liu, and H. Huang, "Occlusion-Aware Real-Time Object Tracking," *IEEE Trans. Multimedia*, vol. 19, no. 4, pp. 763–771, Apr. 2017.

[24] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 749–765.

[25] X. Lu, C. Ma, B. Ni, X. Yang, I. Reid, and M.-H. Yang, "Deep regression tracking with shrinkage loss," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 353–369.

[26] B. Ma, J. Shen, Y. Liu, H. Hu, L. Shao, and X. Li, "Visual Tracking Using Strong Classifier and Structural Local Sparse Descriptors," *IEEE Trans. Multimedia*, vol. 10, no. 17, pp. 1818–1828, 2015.

[27] B. Ma, H. Hu, J. Shen, Y. Zhang, and F. Porikli, "Linearization to nonlinear learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4400–4407.

[28] G. E. Hinton and D. C. Plaut, "Using Fast Weights to Deblur Old Memories," in *Proc. Annual Conf. Cog. Sci. Society*, 1987, pp. 177–186.

[29] J. Schmidhuber, "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook," PhD Thesis, Technische Universität München, 1987.

[30] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual Tracking with Fully Convolutional Networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 3119–3127.

[31] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 3074–3082.

[32] R. Tao, E. Gavves, and A. W. Smeulders, "Siamese instance search for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1420–1429.

[33] J. Shen, X. Tang, X. Dong, and L. Shao, "Visual object tracking by hierarchical attention siamese network," *IEEE Trans. Cybernetics*, vol. 50, no. 7, pp. 3068–3080, 2019.

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll¨¢r, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[35] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke, "Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5296–5305.

[36] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, "Lasot: A high-quality benchmark for large-scale single object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5374–5383.

[37] L. Zhang, A. Gonzalez-Garcia, J. Van De Weijer, M. Danelljan, and F. S. Khan, "Synthetic data generation for end-to-end thermal infrared tracking," *IEEE Trans. Image Process.* , vol. 28, no. 4, pp. 1837–1850, 2018.

[38] A. He, C. Luo, X. Tian, and W. Zeng, "A twofold siamese network for real-time object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4834–4843.

[39] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank, "Learning attentions: residual attentional siamese network for high performance online visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4854–4863.

[40] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, and H. Lu, "Structured siamese network for real-time visual tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 351–366.

[41] X. Dong, J. Shen, D. Wu, K. Guo, X. Jin, and F. Porikli, "Quadruplet Network With One-Shot Learning for Fast Visual Object Tracking," *IEEE Trans. Image Process.* , vol. 28, no. 7, pp. 3516–3527, Jul. 2019.

[42] X. Dong and J. Shen, "Triplet Loss in Siamese Network for Object Tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 459–474.

[43] J. Shen, Y. Liu, X. Dong, X. Lu, F. S. Khan, and S. C. Hoi, "Distilled siamese networks for visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021, pp. 1–1.

[44] X. Wang, C. Li, B. Luo, and J. Tang, "Sint++: Robust visual tracking via adversarial positive instance generation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4864–4873.

[45] T. Yang and A. B. Chan, "Learning dynamic memory networks for object tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 152–167.

[46] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1763–1771.

[47] C. Huang, S. Lucey, and D. Ramanan, "Learning policies for adaptive tracking with deep feature cascades," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, p. 105–114.

[48] X. Dong, J. Shen, W. Wang, Y. Liu, L. Shao, and F. Porikli, "Hyperparameter Optimization for Tracking With Continuous Deep Q-Learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 518–527.

[49] X. Dong, J. Shen, W. Wang, L. Shao, H. Ling, and F. Porikli, "Dynamical Hyperparameter Optimization via Deep Reinforcement Learning in Tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43 no. 5, pp. 1515–1529, 2021.

[50] M. Kristan, J. Matas, A. Leonardis, T. Voj¨ª?, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. ?ehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2137-2155, 2016.

[51] J. Ba, G. E. Hinton, V. Mnih, J. Z. Leibo, and C. Ionescu, "Using Fast Weights to Attend to the Recent Past," in *Proc. Neural Inf. Process. Syst.*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, 2016, pp. 4331–4339.

[52] S. Thrun and L. Pratt, "Learning to learn: Introduction and overview," in *Learning to learn*, 1998, pp. 3–17.

[53] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent," in *Proc. Int. Conf. Art. Neural Net.*, 2001, pp. 87–94.

[54] M. Andrychowicz, M. Denil, S. G¨®mez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," *Proc. Neural Inf. Process. Syst.*, vol. 29, pp. 3981–3989, 2016.

[55] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. Int. Conf. Mach. Learn. deep learning workshop*, 2015.

[56] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching Networks for One Shot Learning," *Proc. Neural Inf. Process. Syst.*, vol. 29, pp. 3630–3638, 2016.

[57] J. Snell, K. Swersky, and R. Zemel, "Prototypical Networks for Few-shot Learning," *Proc. Neural Inf. Process. Syst.*, vol. 30, pp. 4077–4087, 2017.

[58] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1842–1850.

[59] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," *Proc. Int. Conf. Learn. Representations*, 2017.

[60] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[61] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," in *Proc. Neural Inf. Process. Syst.*, 2018, pp. 9537–9548.

[62] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," *Proc. Int. Conf. Learn. Representations*, 2019.

[63] H. Li, W. Dong, X. Mei, C. Ma, F. Huang, and B.-G. Hu, "Lgm-net: Learning to generate matching networks for few-shot learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3825–3834.

[64] E. Park and A. C. Berg, "Meta-tracker: Fast and robust online adaptation for visual object trackers," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 569–585.

[65] J. Choi, J. Kwon, and K. M. Lee, "Deep meta learning for real-time target-aware visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 911–920.

[66] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau, and M.-H. Yang, "Vital: Visual tracking via adversarial learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8990–8999.

[67] P. Li, B. Chen, W. Ouyang, D. Wang, X. Yang, and H. Lu, "Gradnet: Gradient-guided network for visual object tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6162–6171.

[68] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for speed: A benchmark for higher frame rate object tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1125–1134.

[69] S. Li and D.-Y. Yeung, "Visual object tracking for unmanned aerial vehicles: a benchmark and new motion models," in *Proc. Associ. Advance. Art. Intell.*, Feb. 2017, pp. 4140–4146.

[70] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. Čehovin Zajc, O. Drbohlav, A. Lukezic, A. Berg, A. Eldesokey, J. Kapyla, and G. Fernandez, "The seventh visual object tracking vot2019 challenge results," 2019.

[71] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4310–4318.

[72] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer, "Adaptive color attributes for real-time visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1090–1097.

[73] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5388–5396.

[74] Y. Li and J. Zhu, "A Scale Adaptive Kernel Correlation Filter Tracker with Feature Integration." in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 254–265.

[75] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang, "Hedged deep tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4303–4311.

[76] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. Lau, and M.-H. Yang, "Crest: Convolutional residual learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2555–2564.

[77] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: robust tracking via multiple experts using entropy minimization," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 188–203.

[78] M. Danelljan, L. V. Gool, and R. Timofte, "Probabilistic regression for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7183–7192.

[79] P. Voigtlaender, J. Luiten, P. H. Torr, and B. Leibe, "Siam r-cnn: Visual tracking by re-detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6578–6588.

[80] Z. Zhang, H. Peng, J. Fu, B. Li, and W. Hu, "Ocean: Object-aware anchor-free tracking," in *Proc. Eur. Conf. Comput. Vis.*. Springer, 2020, pp. 771–787.

[81] G. Wang, C. Luo, Z. Xiong, and W. Zeng, "Spm-tracker: Series-parallel matching for real-time visual object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3643–3652.

[82] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1328–1338.

[83] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Learning discriminative model prediction for tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6182–6191.

[84] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 2411–2418.

[85] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.

[86] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proc. Associ. Advance. Art. Intell.*, vol. 32, no. 1, 2018.

[87] J. Choi, J. Kwon, and K. M. Lee, "Visual tracking by tridentalign and context embedding," in *Proc. Asian. Conf. Comput. Vis.*, 2020.

[88] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8126–8135.

[89] X. Chen, B. Yan, J. Zhu, D. Wang, and H. Lu, "High-performance transformer tracking," *arXiv preprint arXiv:2203.13533*, 2022.

[90] B. Yan, H. Peng, J. Fu, D. Wang, and H. Lu, "Learning spatio-temporal transformer for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 10 448–10 457.

[91] P. Blatter, M. Kanakis, M. Danelljan, and L. Van Gool, "Efficient visual tracking with exemplar transformers," *arXiv preprint arXiv:2112.09686*, 2021.

[92] B. Yan, H. Peng, K. Wu, D. Wang, J. Fu, and H. Lu, "Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15 180–15 189.

[93] V. Borsuk, R. Vei, O. Kupyn, T. Martyniuk, I. Krashenyi, and J. Matas, "Fear: Fast, efficient, accurate and robust visual tracker," *arXiv preprint arXiv:2112.07957*, 2021.