CLNet: A Compact Latent Network for Fast Adjusting Siamese Trackers

Xingping Dong[1 [0000−0003−1613−9288]], Jianbing Shen[1⋆ [0000−0003−1883−2086]], Ling Shao[1,2 [0000−0002−8264−6117]], and Fatih Porikli[3 [0000−0002−1520−4466]]

[1] Inception Institute of Artificial Intelligence, Abu Dhabi, UAE.
[2] Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE.
[3] Australian National University, Australia.

Abstract. In this paper, we provide a deep analysis for Siamese-based trackers and find that the one core reason for their failure on challenging cases can be attributed to the problem of *decisive samples missing* during offline training. Furthermore, we notice that the samples given in the first frame can be viewed as the decisive samples for the sequence since they contain rich sequence-specific information. To make full use of these sequence-specific samples, we propose a compact latent network to quickly adjust the tracking model to adapt to new scenes. A statistic-based compact latent feature is proposed to efficiently capture the sequence-specific information for the fast adjustment. In addition, we design a new training approach based on a diverse sample mining strategy to further improve the discrimination ability of our compact latent network. To evaluate the effectiveness of our method, we apply it to adjust a recent state-of-the-art tracker, SiamRPN++. Extensive experimental results on five recent benchmarks demonstrate that the adjusted tracker achieves promising improvement in terms of tracking accuracy, with almost the same speed. The code and models are available at `https://github.com/xingpingdong/CLNet-tracking`.

Keywords: Siamese tracker, latent feature, sequence-specific, sample mining, fast adjustment

1 Introduction

Recently, Siamese-based trackers [3, 31, 30] have attracted significant attention in the tracking community, since they successfully incorporated data-driven deep learning with real-time visual tracking, and achieved impressive tracking accuracy. However, these trackers are still unable to address more challenging situations, such as similar distractors or huge deformation as shown in Fig. 1. To analyze the core reason for these failures, we simplify the Siamese model to a linear binary-classifier and transfer the tracking task to a classification problem. We find that the issues can be attributed to the problem of *decisive samples missing*, i.e. some key samples, such as the ones in the above challenging cases,

⋆ Corresponding author: Jianbing Shen (shenjianbingcg@gmail.com)

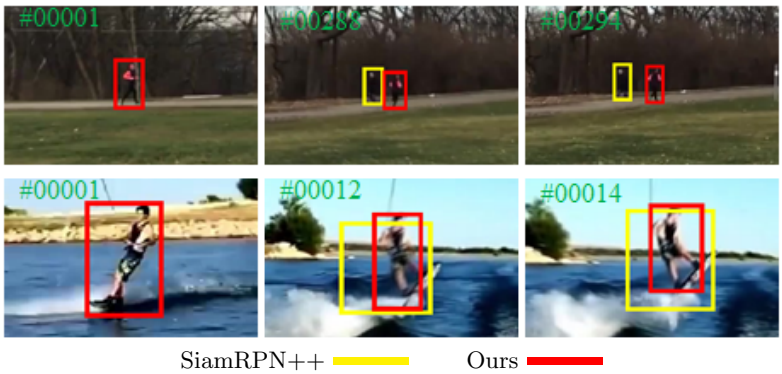

Fig. 1. Sample results of our compact latent network incorporated into the state-of-the-art tracker, SiamRPN++ [30]. By only adjusting the base model (SiamRPN++) with the sequence-specific samples from the first frame, we significantly improve its discrimination ability under the challenging cases, such as similar distractors (top) or huge deformation (bottom).

are rare or unseen during offline training. This results in the trained model having poor discrimination ability for these challenging samples. This is a normal problem for most data-driven models since they cannot capture all data for training. In contrast, in the tracking task, annotated bounding boxes are given in the first frame of each sequence, which can provide sequence-specific samples to improve the discrimination ability of the model, since these samples are similar to the ones in the other frames of the same sequence. In addition, most Siamese-based methods ignore the rich context information, only applying the annotations to extract templates. Thus, the ignored context information can be used to adjust these Siamese-based models and improve the discrimination ability for all samples in the sequence. As shown in Fig. 1, the model, adjusted by the sequence-specific supervision information in the first frame, achieves significant improvement compared with the original model.

Making full use of the sequence-specific information to adjust the offline trained model is not a trivial task, especially for real-time tracking, because of the limitation of computational load. A simple solution is to directly retrain the Siamese-based model using an optimization method, such as stochastic gradient descent (SGD) [42], ride regression [6], or Lagrange multipliers [24]. However, these methods are often time-consuming and not practicable for tracking.

To meet real-time requirements and extract effective information from sequence-specific samples, we propose a Compact Latent Network (CLNet) to adjust recent Siamese-based trackers. Our CLNet contains an feature-adjusting sub-network, latent encoder, and prediction sub-network. The first provides the adjusting feature for each sequence-specific sample, using three 1×1 convolutional layers for efficiency. The core module, the latent encoder, produces 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. Then, the latent feature is fed into the last sub-network (a three-layer perceptron), to predict the adjusting parameters.

It is worth mentioning that our latent feature is compact, with only a few thousand parameters, and more robust than the normal features, since statistics-based features contain uncertainty information in the distribution, which is beneficial for better classification performance [26]. In addition, we propose a new training method based on diverse sample mining to further enhance the training performance. In contrast to the training method of Siamese models, we sample image-pairs from a sequence for each batch to effectively extract the sequence-specific information for the latent feature. We also incorporate diverse sample mining to benefit the adjustment network training. These diverse samples improve the discrimination ability of the adjustment network enabling to address similar distractors and significant appearance changes.

To evaluate the effectiveness of the proposed approach, we take the state-of-the-art tracker, SiamRPN++ [30], as our basic model. To maintain the generalization ability of this model, we use the proposed CLNet to adjust the last layers in the classification and regression branches. For fast adjustment, we only run CLNet once in the first frame and omit it in the following frames. A thorough evaluation on five popular benchmarks (seen in §4) clearly demonstrates the advantage of our algorithm.

2 Related Work

Although many techniques have been successfully applied to visual object tracking, such as the correlation filter [21, 5, 7], regression model [20, 37], and sparse coding [40, 39], here, we only focus on the recent Siamese network based trackers [3, 31], which are mainstream in the tracking community, and meta-learning approaches [22, 50], which are related to our latent feature.

2.1 Siamese Network Based Trackers

The pioneering work, SiamFC [3], provided a new paradigm based on the Siamese network to fully exploit the increasing number of labeled video datasets. After this work, several researchers tried to further mine the potentiality of the offline tracking model by designing different Siamese architectures [19, 60, 67, 12], using the powerful training loss [8], learning efficient Siamese networks [36], and so on [63, 64, 51]. Some focus on improving the performance of the online updating method through various strategies, such as incorporating a correlation filter [58], learning a dynamic network [18], utilizing deep reinforcement learning [25, 10, 11]. Among these Siamese-based trackers, the recent SiamRPN [31], which introduces the region proposal network after the Siamese network, achieves very high speed and impressive tracking accuracy on the popular VOT benchmark [29]. The following works [69, 14, 30, 68] also obtain state-of-the-art performance on this benchmark. For example, Zhu *et al.* [69] proposed a distractor-aware data

augmentation approach and an incremental learning method to enhance the offline model and online tracking mechanism, respectively. In consequence, more attention is paid to improving the generalization of the base network by using more discriminative network structures, such as cascaded RPN [14], deeper networks [30], and wider networks [68]. These state-of-the-art trackers prefer to provide a more powerful offline model that can use more training data. However, they do not fully exploit the supervised information in the first frame

2.2 Meta-learning

Meta-learning can be interpolated as the inception of *fast weights* [22, 2], or *learning-to-learn* [50, 57, 23, 1], and is usually applied to few-shot classification, specifically for the image recognition task. There are three major categories of recent meta-learning approaches, including 1) metric-based methods [27, 59, 55], which focus on learning powerful similarity metrics to discriminate samples from the same class; 2) memory-based methods [49, 44], which explore the storage of critical training examples with effective memory architectures or encoding fast adaptation methods, and 3) optimization-based methods [15, 16], which search for adaptive parameters to quickly adjust the model for new tasks.

Although meta-learning has been widely applied for few-shot classification [15, 48, 32], there exist few related works in the community of visual object tracking [43, 4]. Park *et al.* [43] proposed an optimization-based meta-learner, analogous to MAML [15], to learn an adaptive step for gradient-based online updating. This method has been applied to two trackers based on online training [42, 56] for acceleration, by reducing the number of training iterations. However, it cannot be directly used for offline training models, like Siamese-based trackers. Recently, a meta-learner network [4] and gradient-guided network [33] were proposed to update the Siamese network based tracker online, by utilizing the gradients to extract target-specific information. In contrast, our approach explores a new direction to capture the sequence-specific information, by using a statistic-based latent feature. Besides, we only adjust the basic model in the first frame, rather than using online updating. Thus, the time cost for our adjustment is almost negligible.

3 Siamese Tracker with Compact Latent Network

SiamRPN++ [30], a recent representative Siamese tracker, is adopted as our basic tracker. Thus, we briefly introduce the framework of this tracker in §3.1. Then, we conduct a deep analysis into the issue of *decisive samples missing* in Siamese-based trackers (see §3.2). To overcome this issue, we propose an efficient Compact Latent Network (CLNet) in §3.3, to adjust the basic model, and provide a diverse sample mining strategy to boost training performance in §3.4. Fig. 2 shows the framework of our method.

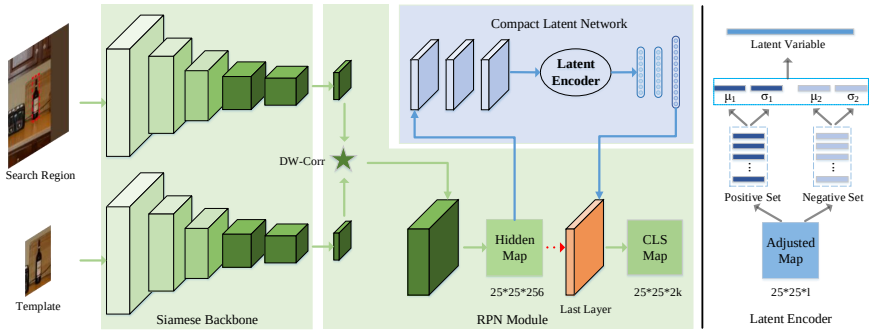

Fig. 2. The framework of Compact Latent Network (CLNet) for adjusting SiamRPN++ [30]. For clarity, we only show the classification branch. Given an Annotated Bounding Box (AB-Box) in the first frame, the proposed CLNet predicts the weights in the last layer of SiamRPN++. At this time, the hidden map is fed into CLNet, not the last layer (i.e. the red arrow is removed). In the following frames, we remove the CLNet to reduce computational requirements and use the adjusted model for tracking. The latent encoder module is shown on the right, where μ and σ present the mean and standard deviation of a sample set. The positive and negative sets are produced using the AB-Box.

3.1 Revisiting SiamRPN for Tracking

In the early Siamese network based tracking algorithm, SiamFC [3], the tracking task is formulated as a nearest neighbor searching problem in a semantic embedding space. Given a target patch provided in the first frame of a sequence, the goal of SiamFC is to seek the most similar patch from the search region in the subsequent frames. To extract representative semantic features, SiamFC [3] build a fully-convolutional Siamese network ϕ, which contains a template branch for representing the target patch $\mathbf{z}$, and an instance branch for representing the search region $\mathbf{x}$. Then the cross-correlation (convolution) operation is applied on these semantic features to produce the final similarity map $\mathbf{S}$, which is formulated as $\mathbf{S}(\mathbf{z},\mathbf{x}) = \phi(\mathbf{z}) * \phi(\mathbf{x}) + b$, where b is the offset of the similarity value, and $*$ represents the cross-correlation operation. $\mathbf{S} \in \mathbb{R}^{w \times h}$ measures the similarity between the template and instances in the search region, where w and h are the width and height of this map. Then the peak of $\mathbf{S}$ corresponds to the target location in the search region.

To avoid multi-scale estimation, SimRPN [31] extends SiamFC by incorporating an additional Region Proposal Network (RPN). The outputs of SiamRPN include one classification branch and one regression branch to regress the target bounding box (b-box) for both position and scale estimation. A multi-anchors technique [46] is further used for better performance. Combined with the Siamese network, SiamRPN produces $w \times h \times k$ anchors, where k is the anchor number for each searching position on instances feature map $\phi(\mathbf{x})$. For each anchor, the regression and classification branches obtain a proposal (b-box) and corresponding

score, respectively. The corresponding outputs are formulated as follows,

$$\mathbf{A}^{cls} = con_{cls}^{fea}(\phi(\mathbf{x})) * con_{cls}^{ker}(\phi(\mathbf{z})), \mathbf{A}^{loc} = con_{loc}^{fea}(\phi(\mathbf{x})) * con_{loc}^{ker}(\phi(\mathbf{z})), \quad (1)$$

where *con* indicates the couvolutional network obtaining new feature maps. Then the final target b-box is the regression output on the anchor with the highest classification score. Furthermore, Li *et al.* [30] proposed SiamRPN++ to utilize the asymmetrical depth-wise cross correlation to reduce the number of parameters in the RPN block, providing efficient computation and solving the problem caused by the differences between the classification and regression branches. The new RPN block is formulated as follows,

$$\mathbf{A}^{cls} = h^{cls}\left(\alpha_{cls}^{fea}(\phi(\mathbf{x})) \star \alpha_{cls}^{ker}(\phi(\mathbf{z}))\right), \mathbf{A}^{loc} = h^{loc}\left(\alpha_{loc}^{fea}(\phi(\mathbf{x})) \star \alpha_{loc}^{ker}(\phi(\mathbf{z}))\right), \quad (2)$$

where α is an adjust (1×1 convolutional) layer, $\star$ is the depth-wise cross correlation, and h^s denotes the head block for predicting the classification and regression map.

3.2 Analysis for Siamese-based Training Method

Siamese-based trackers take advantage of large numbers of image pairs to train a network from labeled video datasets (e.g., VID [47]) which can provide richer training samples. These numerous training samples can effectively enhance the generalization of tracking models for most testing sequences. However, this training strategy does not utilize sequence-specific information, provided by the annotation in the first frame of each testing sequence. Here, we introduce a simple binary-classification case to intuitively illustrate the underlying problem caused by using the general training method. In fact, SiamFC [3] can also be viewed as a binary-classification problem, where the template is the classifier discriminating the instance patch in the search region. Thus, the following analysis is suitable for most methods based on SiamFC [3] or SiamRPN [31].

Without loss of generality, we can assume that there are various positive and negative samples that come from different videos during the training phase, which is consistent with the Siamese-based training method. As shown in Fig. 3, we use a group of positive samples (the big blue ellipse) as well as negative samples (the big green ellipse) for training. The optimal decision hyperplane should be located in the middle of the two groups of samples to maintain the largest margin between the boundary samples and itself. Then, we can assume w^1 is the ideal decision hyperplane based on these two groups of training data in Fig. 3. During the testing phase, the positive and negative samples have no overlap with the training samples (i.e., we apply the learned tracker on unseen videos), and the number of testing samples is usually far less than that of the training phase. Thus, we use small ellipses to represent the testing samples and assume the challenging samples lie near the decision hyperplane. As shown in Fig. 3, some challenging samples may pass through the decision hyperplane into the wrong region, which means the trained classifier is invalid.

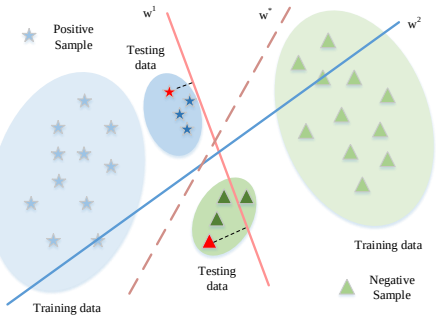

Fig. 3. Illustration of issue on binary-classification with the general training method. w^1 and w^2 are the ideal decision hyperplanes based on training and testing sets, respectively. w^* is the decision boundary considering both two sets.

In the general classification task, an error will occur if any negative samples score is larger than some positive sample. In contrast, a tracking error occurs only when the classification score of one negative sample is larger than all positive samples scores. However, the tracking task still suffers from the issue of an unsuitable decision hyperplane (like w^1). For example, in Fig. 3, we can assume that the testing data points are sampled from a search region. The red triangle is the point with the largest distance from the decision boundary w^1 among all negative samples, which means it has the highest positive classification score. Similarly, the red star has the highest score among all positive samples. Obviously, the score of the red triangle is larger than that of the red star; thus, the red triangle is regarded as the target, resulting in tracking faults.

In this paper, we focus on alleviating this issue and find a reasonable decision hyperplane for the tracking task. We find that the target ground-truth b-box, given in the first frame of a sequence, can provide rich supervision information to enhance the model discrimination ability for this sequence which helps to find a reasonable decision hyperplane. However, most Siamese-based trackers only use this b-box to extract the template feature and ignore the discriminative context information in the first frame. Effectively utilizing the supervision information is vital for the tracking model (classifier) learning. One simple solution is to finetune the tracking model with the positive and negative samples (testing data in Fig. 3) in the first frame. This is time-consuming since the model finetuning may converge after numerous iterations with normal gradient descent optimization [42]. Furthermore, the limited samples generated in the first frame easily lead to over-fitting. As shown in Fig. 3, if we only consider the testing data, the optimal decision hyperplane is w^2. However, this is not suitable for some training samples, and provides wrong labels for them. In fact, the ideal decision hyperplane for both training and testing data is w^*, which is not easily found. In the next section, we try to find an ideal tracking model which is suitable for both training data (large-scale labeled videos) and testing data (samples in the first frame), by using a compact latent network.

3.3 Compact Latent Network

To demonstrate the effectiveness of our compact latent network, we select the recent state-of-the-art Siamese-based model, SiamRPN++ [30], as our base tracker. To maintain the generality of the original tracking model, we fix all layers except the last one in the classification and regression branches. Here, we only show how to incorporate our compact latent network into the classification branches, and a similar process is also applied to the regression branches. For simplicity, we omit the notation *cls* in the following description. Firstly, we reformulate the base model to better explain the new proposed module. As mentioned in Sec. 3.1, the last head block in SiamRPN++ contains two convolutional layers. We decompose the head block into two components, i.e. $h = h_1(h_0)$. Denoting all layers except the last one in the classification branch as a function f, the classification map in Eq. (2) can be reformulated as follows:

$$\mathbf{A} = h_1(f(\mathbf{x}, \mathbf{z}); \theta_1), \tag{3}$$

where $f(\mathbf{x}, \mathbf{z}) = h_0\left(\alpha^{fea}(\phi(\mathbf{x})) \star \alpha^{ker}(\phi(\mathbf{z}))\right)$. In fact, $f(\mathbf{x}, \mathbf{z}) \in \mathbb{R}^{w \times h \times c}$ is viewed as a $w \times h$ feature map $\mathbf{M}$ with c channels, and θ_1 is the parameter in h_1.

To better exploit the feature map $\mathbf{M}$, we provide a compact feature representation by utilizing the statistical information inside it. Firstly, this feature map $\mathbf{M}$ can be regarded as a set including the hidden vectors corresponding to the template-instance pair, i.e., $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_{wh}\}$. Furthermore, according to the classification labels, this set can be split into two groups, the positive set $\mathcal{P} = \{\mathbf{m}_1^+, \mathbf{m}_2^+, \cdots, \mathbf{m}_{n^+}^+\}$ and negative set $\mathcal{N} = \{\mathbf{m}_1^-, \mathbf{m}_2^-, \cdots, \mathbf{m}_{n^-}^-\}$. Given these two sets, we capture the statistical information by computing the mean μ and standard deviation σ as follows,

$$\mu^\rho = \frac{1}{n^\rho} \sum\nolimits_{i=1}^{n^\rho} g_a(\mathbf{m}_i^\rho), \sigma^\rho = \sqrt{\frac{1}{n^\rho} \sum\nolimits_{i=1}^{n^\rho} (g_a(\mathbf{m}_i^\rho) - \mu^\rho)^2}, \tag{4}$$

where $\mu \in \mathbb{R}^l$, $\sigma \in \mathbb{R}+^l$, $\rho \in \{+, -\}$, and g_a is a feature-adjusting sub-network containing three 1×1 convolutional layers, which is used to reduce the channel number of the latent vector from c to l. Here, we set $l \leq 2c$ to avoid a too high computational burden. Then, the final latent compact feature $\mathbf{c}$ for the original $\mathbf{m}$ is constructed by concatenation, i.e., $\mathbf{c} = concat(\mu^+, \sigma^+, \mu^-, \sigma^-)$.

In practice, the proposed statistic-based feature is more robust than the original features. We give an intuitive example in Fig. 4 to demonstrate the powerful representative ability of the statistic-based feature. Assume the positive (blue stars) and negative (green triangles) samples are respectively drawn from two Gaussian distributions (the blue and green ellipses). As shown in Fig. 4, we can easily find the 'ideal' decision hyperplane w^n according to the rule of the largest margin. However, the true decision hyperplane should be w^g, since the true sample boundaries are the edges of the two ellipses. The incorrect decision hyperplane is caused by *decisive samples missing*, i.e. the key samples (red star and triangle) are not drawn. Usually, the critical samples are difficult to capture, but we can provide the mean and standard deviation of each sample set to reshape the Gaussian distribution. Then the distributions can be used to find the

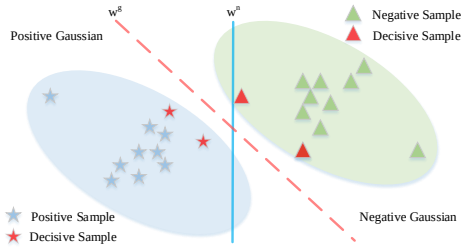

Fig. 4. Comparison of binary-classification based on normal samples and Gaussian distributions. w^n and w^g are the corresponding ideal decision hyperplanes.

ideal decision hyperplane and achieve better classification performance, by utilizing some uncertainty-based optimization methods [26]. Notice that a Gaussian distribution only depends on its mean and standard deviation, which indicates that these contain enough information to represent the distribution. Thus, we use the mean and standard deviation as the compact feature to alleviate the issue of *decisive samples missing*, by utilizing the underlying distribution.

The proposed compact representation brings two advantages: firstly, it has fewer parameters than the original feature. Notice that l is kept in the same order of magnitude as c. Thus, in practice, the parameter number $4 \times l$ of the compact feature $\mathbf{c}$ is less than the original one $w \times h \times c$. Secondly, the compact feature $\mathbf{c}$ is number-free, in other words, we can use a different number of positive and negative samples to construct this feature. This is very suitable for SiamRPN-based tracking models, since the number of positive or negative samples changes with the different input image-pairs [31].

After obtaining the latent compact feature $\mathbf{c}$, a multi-layer perceptron is used as the prediction sub-network to produce a deviation for the weights of the last head layer:

$$\Delta\theta_1 = g_\Delta(\mathbf{c}), \tag{5}$$

where g_Δ is the weight deviation predictor containing three fully connected layers. These deviations are added into the corresponding weights to adjust the model for different sequences, which is formulated as:

$$\theta_a = \theta_1 + \Delta\theta_1. \tag{6}$$

Finally, the adjusted weight θ_a is used to produce the classification map. For the regression branch, the computation procedure for weight adjustment is similar except that we use the positive sample set to build the compact latent feature, since only the positive samples are used for training. In summary, the adjusted classification map $\mathbf{A}_a^{cls}$ and regression map $\mathbf{A}_a^{loc}$ are formulated as follow:

$$\mathbf{A}_a^{cls} = h_1^{cls}(f^{cls}(\mathbf{x}, \mathbf{z}); \theta_a^{cls}), \mathbf{A}_a^{loc} = h_1^{loc}(f^{loc}(\mathbf{x}, \mathbf{z}); \theta_a^{loc}). \tag{7}$$

These maps are applied to predict the tracked object, similar to SiamRPN++ [30].

3.4 Training with Diverse Sample Mining

To train the proposed compact latent networks, we fix all parameters in the original model, and use the same softmax loss L^{cls} and smooth-L1 loss L^{loc} in SiamRPN++ [30] for the adjusted classification and regression maps, respectively. The final loss is formulated as follows,

$$L = L^{cls}(\mathbf{A}_a^{cls}, y^{cls}) + \lambda L^{loc}(\mathbf{A}_a^{loc}, y^{loc}), \tag{8}$$

where y^{cls} and y^{loc} are the classification and regression ground-truth, respectively, and λ is the trade-off parameter between the two losses, which is set as 1.2 in our experiments. In SiamRPN++, the image-pairs are sampled from different sequences in one training batch to achieve a more discriminative feature, which is not suitable for training our adjustment network. This is because the goal of our adjustment network is to predict the adjusted weights for each specific sequence rather than the whole videos. During training, we need to capture the important information inside a specific sequence not the general information across the sequences. Besides, our statistic-based latent feature also requires the local statistical information in a sequence not the global statistical information across sequences. Thus, in each training batch, we randomly sample several image-pairs from the same sequence and feed them into the network.

To enhance the training performance, we propose a diverse sample mining strategy, by ranking the classification scores of unused negative samples. SiamRPN++ [30] adopts the IoU between the anchor and ground-truth b-box to select the positive and negative samples. The negative ones are the anchors with $IoU < 0.3$, and the positive samples are those with $IoU > 0.6$. Only at most 16 positive samples and a total of 64 samples are used for one training pair. The unused negative samples may contain some diverse samples, which are helpful for enhancing the model discrimination ability. To mine these diverse samples, we first compute the original score map $\mathbf{A}^{cls}$. Then, we sort the unused negative samples in descending order according to the positive scores in $\mathbf{A}^{cls}$. The first 16 negative samples are regarded as the diverse samples and appended into the training samples. Thus, a total of 80 samples are used for one training pair. The newly proposed diverse samples, near the decision hyperplane of the original model, are more likely to be the decisive samples, such as the ones in Fig. 4. As the analysis in Sec. 3.3, these samples are helpful to find the ideal decision hyperplane and build a more robust compact latent feature.

4 Experimental Results

Our approach is implemented in Python 3 using Pytorch 0.4.0 and evaluated on a single Nvidia RTX 2080Ti GPU. We compare our tracker and the base tracker with several representative trackers on NFS [17], DTB70 [34], UAV123 [41], VOT2019 [28], and LaSOT [13] benchmarks. The tracking speed of our method is about 45.6 fps, closing to the base tracker SiamRPN++ [30] (46.9 fps). We evaluate the base model with its official code in our machine for fair comparison.

4.1 Implementation Details

Architecture. Our Compact Latent Network (CLNet) has two sub-networks g_a and g_δ. In our experiments, g_a contains three convolutional blocks constructed by a 1×1 convolution, batch-norm, and ReLU layer. For the classification branch, the output channel number of each convolutional layer is set as 256, and that of the regression branch is set as 512. This ensures the compact latent features are the same size in both two branches. Sub-network g_δ contains three fully connected layers, the first two layers of which are followed by a ReLU layer and the last of which is followed a *Tanh* layer. For the classification branch, the output numbers of the three layers are [128, 128, $n_{\theta_1^{cls}}$], where $n_{\theta_1^{cls}}$ is the parameter number of θ_1^{cls} to be adjusted. Similarly, the numbers in the regression branch are set as [128, 128, $n_{\theta_1^{loc}}$]. Here, we adopt the SiamRPN++ [30] with three layer-wise features aggregation as the base model. Since the output feature maps in these three layers have the same size, we can use the adjustment network with the same structure for each feature map.
Training. We follow the training protocol of SiamRPN++ [30], using the same training datasets (COCO [35], DET [47], VID [47], and YouTubeBB [45]). Synchronized stochastic gradient descent is used for training over four GPUs with a total of 64 pairs per minibatch (16 pairs per GPU). We train over 20 epochs, each of which contains 60,000 training pairs. We use a step learning rate from 0.001 to 0.005 for the first five epochs as a warmup training. In the last 15 epochs, the learning rate is exponentially decayed from 0.005 to 0.0005. The other hyper-parameters are the same as SiamRPN++.

4.2 Comparison with Other Trackers

NFS30 Dataset. Need for Speed (NFS) [17] includes 100 challenging sequences with fast-moving objects. We evaluate the proposed method on the 30 FPS version (NFS30) by comparing the trackers in [17]. As shown in Fig. 5, our tracker achieves a precision and AUC 21.8% and 24.2% higher than MDNet [42], which is the best tracker reported in the original paper. Compared with the baseline SiamRPN++, we also achieve the significant relative gains of 8.2% in terms of both precision and AUC.
DTB70 Dataset. Our method is also evaluated on the Drone Tracking Benchmark [34], which includes 70 videos captured by drone cameras. We report the precision and AUC plots in Fig. 6 in comparison to the recent SiamFC-v2 [58], and other trackers reported in DTB70. Specifically, our tracker significantly outperforms the second-best tracker, SiamRPN++, by large gains of 5.5% and 6.8% in terms of precision and AUC. We attribute the performance promotion to the proposed compact latent network.
UAV123 Dataset. UAV123 [41] includes 123 low altitude aerial sequences captured from an aerial viewpoint, with an average sequence length of 915 frames. Besides the recent methods in UAV123, we add SiamFC-v2 [58] into the comparison. Fig. 7 shows the precision and AUC plots for the top ten trackers. Among the compared methods, SiamRPN++ obtains the best precision and AUC scores

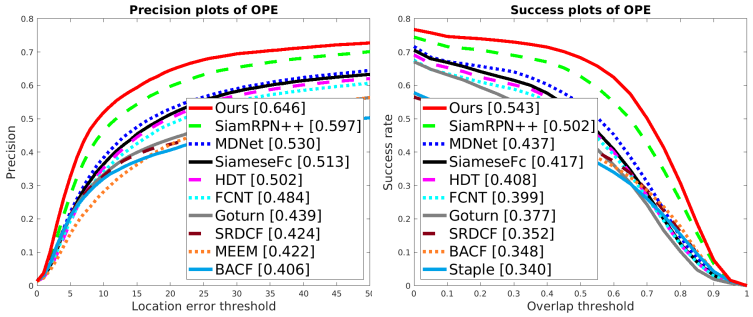

Fig. 5. Precision and success plots on the NFS30 dataset [17].

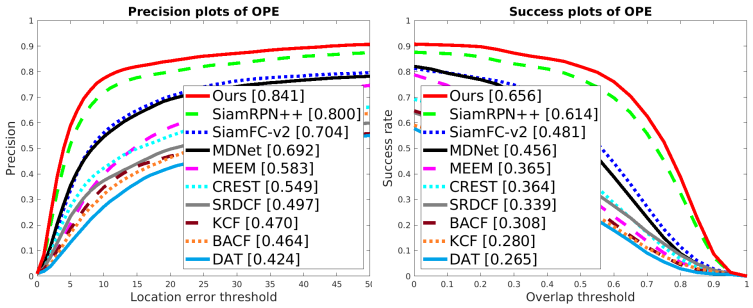

Fig. 6. Precision and success plots on the DTB70 dataset [34].

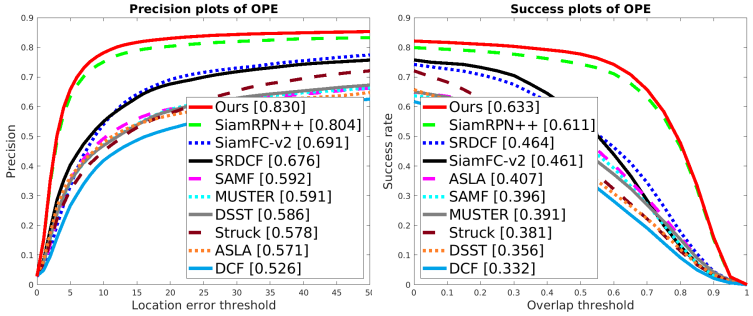

Fig. 7. Precision and success plots on the UAV123 dataset [41].

of 0.804 and 0.611, respectively. However, our approach outperforms this with a precision score of 0.830 and an AUC score of 0.633, which further demonstrates the effectiveness of our method.

VOT2019 Dataset. We also evaluate our CLNet on the real-time challenge of the VOT2019 Dataset [28], which contains 60 public sequences with different challenging factors. This dataset performs re-initialization of a tracker when it

	SiamCRF_RT	SiamMask	ARTCS	SiamDW_ST	DCFST	DiMP	SiamFCOT	SiamMargin	SiamRPN++	**Ours**
EAO ↑	0.262	0.287	0.287	0.299	0.317	0.321	0.350	0.366	0.285	0.313
ACC ↑	0.549	0.594	0.602	0.600	0.585	0.582	0.601	0.585	0.599	0.606
ROB ↓	0.346	0.461	0.482	0.467	0.376	0.371	0.386	0.321	0.482	0.461

Table 1. Evaluation of the real-time challenge in VOT2019 [28] by EAO, ACC, and ROB. The best scores are highlighted in red.

	StrSiam [67]	SiamFC [3]	VITAL [56]	MDNet [42]	MLT [4]	GradNet [33]	SiamDW [68]	C-RPN [14]	SiamRPN++ [30]	**Ours**
P (%)	34	34.1	37.2	37.4	-	35.1	-	44.3	48.5	49.4
P_{norm} (%)	44.3	44.9	48.4	48.1	-	-	-	54.2	56.5	57.4
AUC (%)	35.6	35.8	41.2	41.3	34.5	36.5	38.4	45.5	49.3	49.9

Table 2. Comparison on the LaSOT dataset [13] by the success (AUC), precision (P), and normalized precision (P_{norm}). The best scores are highlighted in red.

fails to track the target, to order to measure its short-term tracking performance. According to the evaluation protocol of VOT2019, we adopt the Expected Average Overlap (EAO), Accuracy (ACC) and Robustness (ROB) to compare different trackers, including the top nine algorithms ranked by EAO in the real-time challenge. As shown in Table 1, our approach obtains the best score in terms of ACC. Compared with our 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. This demonstrates our approach can effectively improve the short-term tracking ability of the base model.

LaSOT Dataset. The recent LaSOT [13] provides high-quality manual annotations for a large-scale dataset, which includes a total 1,400 sequences with an average sequence length of 2,512 frames. To further validate the proposed approach on a larger and longer dataset, we conduct experiments on the testing set of LaSOT with 280 sequences, comparing the recent MLT [4], GradNet [33], SiamDW [68], and C-RPN [14], as well as the trackers from LaSOT. Following [13], three metrics are used for evaluation, including the success (AUC), precision (P), and normalized precision (P_{norm}). As shown in Table 2, our tracker achieves the best results in terms of all three metrics, which clearly demonstrates the generalization ability of our approach for the large-scale dataset.

4.3 Ablation Study

Extensive experiments are performed to analyze the main components of the proposed compact latent network. We conduct these experiments on a combined dataset including the whole DTB70 [34], NFS30 (30 FPS version) [17] and UAV123 [41] datasets. This new combined dataset contains 293 diverse videos for thorough analysis. The variants of our method and the baseline are evaluated using the precision (P) and AUC metrics [65].

We retrain our base tracker (**BT**) SiamRPN++ initialized with its original parameters using our training method and test it on the combined dataset.

	RT	BT	+AN	+LE	+OS	+DM
P (%)	74.3	73.3	73.8	75.5	76.6	77.2
AUC (%)	57.4	57.7	58.2	60.1	60.2	61.1

Table 3. Comparison of key components in terms of precision (P) and AUC scores on the combined DTB70 [34], NFS30 [17], and UAV123 [41] dataset. SiamRPN++ [30] is the base tracker (**BT**). **RT** means the retrained base tracker. The critical components include an adjustment network (**+AN**), latent encoder (**+LE**), one sequence for training (**+OS**), and diverse sample mining (**+DM**).

The results show that the retrained tracker (**RT**) improves the precision (**BT**: 73.3% vs **RT**: 74.3%) but reduces the AUC (**BT**: 57.6% vs **RT**: 57.4%), which indicates that the additional training on the original model does not provide significant gains. To analyze the impact of key components in the proposed framework, we add them to the base tracker (**BT**) one by one. All results are shown in Table 3. We first train a adjustment network without the latent encoder to adjust the base model (**+MN**). The performance gains in terms of P (0.5%) and AUC (0.5%) scores indicate that the base model benefits from the sequence-specific information, extracted by the adjustment network. By adding the latent encoder (**+LE**), CLNet achieves a major improvement with a P gain of 1.7% and AUC gain of 1.9%. This demonstrates the advantages of the latent encoder in providing a compact and effective feature for the sequence-specific information. Furthermore, we sample the image-pairs from one sequence in each training batch (**+OS**), to provide a more robust latent feature. This yields a further improvement, with 1.1% and 0.1% gains in terms of P and AUC scores, respectively. Finally, the diverse sample mining technique (**+DM**) improves the P and AUC score by another 0.6% and 0.9%. In summary, compared with our baseline, the final version (**+DM**) achieves significant relative gains of 5.3% and 5.9% in terms of the P and AUC scores.

5 Conclusion

This paper provided an in-depth analysis into the performance degradation of Siamese-based trackers and found that an important factor is *decisive samples missing* during training. To alleviate this problem, we proposed a Compact Latent Network (CLNet), which can efficiently extract the sequence-specific information in the first frame, to adjust the Siamese-based model. To further facilitate the adjustment network training, we proposed a new training strategy based on diverse sample mining to enhance the discrimination ability of CLNet. Since we only carry out one forward computation for the adjustment in each sequence, the time cost is almost negligible. Extensive evaluations on SiamRPN++ have clearly demonstrated the effectiveness of the proposed CLNet. In the future, we will apply our method to other vision tasks, such as multi-object tracking [52, 66], segmentation [9, 38], deblurring [54, 53], and saliency detection [61, 62].

References

1. Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M.W., Pfau, D., Schaul, T., Shillingford, B., De Freitas, N.: Learning to learn by gradient descent by gradient descent. In: NeurIPS (2016)
2. Ba, J., Hinton, G.E., Mnih, V., Leibo, J.Z., Ionescu, C.: Using fast weights to attend to the recent past. In: NeurIPS (2016)
3. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: ECCV Workshop (2016)
4. Choi, J., Kwon, J., Lee, K.M.: Deep meta learning for real-time target-aware visual tracking. ICCV (2019)
5. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M., et al.: Eco: Efficient convolution operators for tracking. In: CVPR (2017)
6. Danelljan, M., Robinson, A., Shahbaz Khan, F., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: ECCV (2016)
7. Dong, X., Shen, J., Yu, D., Wang, W., Liu, J., Huang, H.: Occlusion-aware real-time object tracking. IEEE TMM (2017)
8. Dong, X., Shen, J.: Triplet loss in siamese network for object tracking. In: ECCV (2018)
9. Dong, X., Shen, J., Shao, L., Van Gool, L.: Sub-markov random walk for image segmentation. IEEE TIP (2015)
10. Dong, X., Shen, J., Wang, W., Liu, Y., Shao, L., Porikli, F.: Hyperparameter optimization for tracking with continuous deep q-learning. In: CVPR (2018)
11. Dong, X., Shen, J., Wang, W., Shao, L., Ling, H., Porikli, F.: Dynamical hyperparameter optimization via deep reinforcement learning in tracking. IEEE TPAMI (2019)
12. Dong, X., Shen, J., Wu, D., Guo, K., Jin, X., Porikli, F.: Quadruplet Network With One-Shot Learning for Fast Visual Object Tracking. IEEE TIP (2019)
13. Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., Ling, H.: Lasot: A high-quality benchmark for large-scale single object tracking. In: CVPR (2019)
14. Fan, H., Ling, H.: Siamese cascaded region proposal networks for real-time visual tracking. In: CVPR (2019)
15. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML (2017)
16. Finn, C., Xu, K., Levine, S.: Probabilistic model-agnostic meta-learning. In: NeurIPS (2018)
17. Galoogahi, H.K., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: ICCV (2017)
18. Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., Wang, S.: Learning dynamic siamese network for visual object tracking. In: ICCV (2017)
19. He, A., Luo, C., Tian, X., Zeng, W.: A twofold siamese network for real-time object tracking. In: CVPR (2018)
20. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 fps with deep regression networks. In: ECCV (2016)
21. Henriques, J.F., Rui, C., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE TPAMI (2015)
22. Hinton, G.E., Plaut, D.C.: Using fast weights to deblur old memories. In: CCSS (1987)

23. Hochreiter, S., Younger, A.S., Conwell, P.R.: Learning to learn using gradient descent. In: ICANN. Springer (2001)
24. Hong, S., You, T., Kwak, S., Han, B.: Online tracking by learning discriminative saliency map with convolutional neural network. In: ICML (2015)
25. Huang, C., Lucey, S., Ramanan, D.: Learning policies for adaptive tracking with deep feature cascades. In: ICCV (2017)
26. Khan, S., Hayat, M., Zamir, S.W., Shen, J., Shao, L.: Striking the right balance with uncertainty. In: CVPR (2019)
27. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop (2015)
28. Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Pflugfelder, R., Kamarainen, J.K., Čehovin Zajc, L., Drbohlav, O., Lukezic, A., Berg, A., Eldesokey, A., Kapyla, J., Fernandez, G.: The seventh visual object tracking vot2019 challenge results (2019)
29. Kristan, M., Matas, J., Leonardis, A., Vojíř, T., Pflugfelder, R., Fernandez, G., Nebehay, G., Porikli, F., Čehovin, L.: A novel performance evaluation methodology for single-target trackers. IEEE TPAMI (2016)
30. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: Siamrpn++: Evolution of siamese visual tracking with very deep networks. In: CVPR (2019)
31. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: CVPR (2018)
32. Li, H., Dong, W., Mei, X., Ma, C., Huang, F., Hu, B.G.: Lgm-net: Learning to generate matching networks for few-shot learning. ICML (2019)
33. Li, P., Chen, B., Ouyang, W., Wang, D., Yang, X., Lu, H.: Gradnet: Gradient-guided network for visual object tracking. In: ICCV (2019)
34. Li, S., Yeung, D.Y.: Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models. In: AAAI (2017)
35. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
36. Liu, Y., Dong, X., Lu, X., Khan, F.S., Shen, J., Hoi, S.: Teacher-Students Knowledge Distillation for Siamese Trackers. arXiv (2019)
37. Lu, X., Ma, C., Ni, B., Yang, X., Reid, I., Yang, M.H.: Deep regression tracking with shrinkage loss. In: ECCV (2018)
38. Lu, X., Wang, W., Shen, J., Tai, Y.W., Crandall, D.J., Hoi, S.C.: Learning video object segmentation from unlabeled videos. In: CVPR (2020)
39. Ma, B., Hu, H., Shen, J., Zhang, Y., Porikli, F.: Linearization to nonlinear learning for visual tracking. In: ICCV (2015)
40. Ma, B., Shen, J., Liu, Y., Hu, H., Shao, L., Li, X.: Visual tracking using strong classifier and structural local sparse descriptors. IEEE TMM (2015)
41. Mueller, M., Smith, N., Ghanem, B.: A benchmark and simulator for UAV tracking. In: ECCV (2016)
42. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR (2016)
43. Park, E., Berg, A.C.: Meta-tracker: Fast and robust online adaptation for visual object trackers. ECCV (2018)
44. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. ICLR (2017)
45. Real, E., Shlens, J., Mazzocchi, S., Pan, X., Vanhoucke, V.: Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In: CVPR (2017)

46. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NeurIPS (2015)
47. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV (2015)
48. Rusu, A.A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., Hadsell, R.: Meta-learning with latent embedding optimization. ICLR (2019)
49. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: Meta-learning with memory-augmented neural networks. In: ICML (2016)
50. Schmidhuber, J.: Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook. Ph.D. thesis, Technische Universität München (1987)
51. Shen, J., Tang, X., Dong, X., Shao, L.: Visual Object Tracking by Hierarchical Attention Siamese Network. IEEE TCYB (2020)
52. Shen, J., Yu, D., Deng, L., Dong, X.: Fast online tracking with detection refinement. IEEE TITS (2017)
53. Shen, Z., Lai, W.S., Xu, T., Kautz, J., Yang, M.H.: Exploiting semantics for face image deblurring. IJCV (2020)
54. Shen, Z., Wang, W., Lu, X., Shen, J., Ling, H., Xu, T., Shao, L.: Human-aware motion deblurring. In: ICCV (2019)
55. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: NeurIPS (2017)
56. Song, Y., Ma, C., Wu, X., Gong, L., Bao, L., Zuo, W., Shen, C., Lau, R., Yang, M.H.: Vital: Visual tracking via adversarial learning. CVPR (2018)
57. Thrun, S., Pratt, L.: Learning to learn: Introduction and overview. In: Learning to learn, pp. 3–17. Springer (1998)
58. Valmadre, J., Bertinetto, L., Henriques, J.F., Vedaldi, A., Torr, P.H.: End-to-end representation learning for correlation filter based tracking. In: CVPR (2017)
59. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: NeurIPS (2016)
60. Wang, Q., Teng, Z., Xing, J., Gao, J., Hu, W., Maybank, S.: Learning attentions: residual attentional siamese network for high performance online visual tracking. In: CVPR (2018)
61. Wang, W., Shen, J., Dong, X., Borji, A.: Salient object detection driven by fixation prediction. In: CVPR (2018)
62. Wang, W., Shen, J., Dong, X., Borji, A., Yang, R.: Inferring salient objects from human fixations. IEEE TPAMI (2019)
63. Wang, X., Li, C., Luo, B., Tang, J.: Sint++: Robust visual tracking via adversarial positive instance generation. In: CVPR (2018)
64. Yang, T., Chan, A.B.: Learning Dynamic Memory Networks for Object Tracking. In: ECCV (2018)
65. Yi, W., Jongwoo, L., Yang, M.H.: Object tracking benchmark. IEEE TPAMI (2015)
66. Yin, J., Wang, W., Meng, Q., Yang, R., Shen, J.: A unified object motion and affinity model for online multi-object tracking. In: CVPR (2020)
67. Zhang, Y., Wang, L., Qi, J., Wang, D., Feng, M., Lu, H.: Structured siamese network for real-time visual tracking. In: ECCV (2018)
68. Zhang, Z., Peng, H.: Deeper and wider siamese networks for real-time visual tracking. In: CVPR (2019)
69. Zhu, Z., Wang, Q., Li, B., Wu, W., Yan, J., Hu, W.: Distractor-aware siamese networks for visual object tracking. In: ECCV (2018)