Towards Sequence-Level Training for Visual Tracking

Minji Kim[1*] Seungkwan Lee[3,4*] Jungseul Ok[3] Bohyung Han[1,2] Minsu Cho[3]

[1]ECE & [2]IPAI, Seoul National University
[3]Pohang University of Science and Technology (POSTECH)
[4]Deeping Source Inc.
https://github.com/byminji/SLTtrack

Abstract. Despite the extensive adoption of machine learning on the task of visual object tracking, recent learning-based approaches have largely overlooked the fact that visual tracking is a sequence-level task in its nature; they rely heavily on frame-level training, which inevitably induces inconsistency between training and testing in terms of both data distributions and task objectives. This work introduces a sequence-level training strategy for visual tracking based on reinforcement learning and discusses how a sequence-level design of data sampling, learning objectives, and data augmentation can improve the accuracy and robustness of tracking algorithms. Our experiments on standard benchmarks including LaSOT, TrackingNet, and GOT-10k demonstrate that four representative tracking models, SiamRPN++, SiamAttn, TransT, and TrDiMP, consistently improve by incorporating the proposed methods in training without modifying architectures.

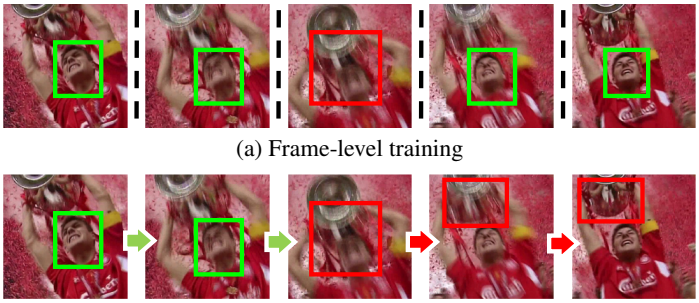
Keywords: visual tracking, sequence-level training, reinforcement learning

1 Introduction

Visual object tracking aims to estimate the spatial extent, *e.g.*, a bounding box, of a target object over a sequence of video frames [34, 30, 23]. This task has been drawing significant attention due to its wide range of applications including visual surveillance, robotics, and autonomous driving [14, 12]. Unlike standard recognition tasks such as image classification and object detection, the class of the target object is unknown and only its bounding box at the initial frame is given for testing. Despite the long history of its study [30], object tracking in the wild still remains challenging due to appearance variation, occlusion, interference, distracting clutter, etc. To tackle the issues, recent methods increasingly rely on robust feature representations that are learned by deep neural networks with convolutions [25, 19, 2, 1, 21, 43] and attention [39, 32, 4].

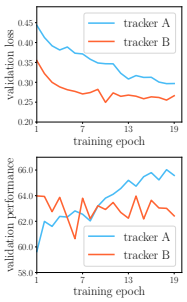
Although learning to track has been widely adopted in the research community, it is largely overlooked that visual tracking is essentially a *sequence-level* task; the estimated target state in the current frame is affected by the history of target states in the previous frames and also influences tracking results in the subsequent frames. For example, recent state-of-the-art methods [25, 2, 5, 8, 20, 39, 32, 43, 38] rely heavily on *frame-level* training, which encourages the trackers to better localize target objects in each frame through supervised learning. While it greatly improves the tracker by

* These authors contributed equally to this work.



(a) Frame-level training

(b) Sequence-level self-testing



(c) Loss *vs.* Perf.

Fig. 1: Pitfall of frame-level training for visual tracking. Training a tracker to better localize a target in each of individual frames of (a) does not necessarily improve actual tracking in the sequence of (b). Green/red boxes indicate success/failure in localization. Due to the issue, inconsistency between the loss and the performance is often observed during training as shown in (c) where trackers, A and B, being frame-level trained are evaluated while the training loss (top) and the tracking performance of average overlap (bottom) are measured. After 10 epochs, A outperforms B in spite of higher losses.

learning robust features for tracking, disregarding the sequential dependency across frames can lead to unexpected tracking failures. Let us assume a tracker that is trained with a typical frame-level training scheme using a set of annotated training videos; random pairs of a target template and a search frame are sampled from a video and the tracker is trained to best localize the target on the search frame independently for each pair. As shown in Fig. 1, now consider one of the training videos that contains a *hard* frame, where the tracker fails to localize the target object. Although the tracker is trained to perform almost perfectly on frame-level localization except for the hard frame (Fig. 1a), its sequence-level performance may turn out to be poor as it loses the target from the hard frame in actual tracking on the sequence (Fig. 1b).

This pitfall of frame-level training mainly stems from inconsistency between training and testing in terms of both *data distributions* and *task objectives*. First, the tracker observes data samples, *i.e.*, tracking situations, that significantly deviate from a real data distribution. That is, while in actual tracking the search window at each frame is determined based on the estimation at the previous frame, in the frame-level training it is not; the search window is typically sampled by adding a random transformation to the ground-truth bounding box. Second, the tracker learns with an objective, *i.e.*, a reward system, that is largely different from actual tracking. The tracking performance in testing puts significant importance on retaining localization accuracy over a sequence, whereas it is only immediate localization quality that matters in the frame-level training. The mismatch of the objectives between training and testing often leads to unexpected results as shown in Fig. 1c; two trackers, A and B, being trained with the same network architecture and the same frame-level objective, are tested on the GOT-10k validation

split where the loss and the performance are measured[1]. While constantly yielding a higher loss, tracker A achieves better performance than tracker B after 10 epochs. Such inconsistency should be rectified for more robust tracking but has hardly been explored so far in the tracking community.

This work investigates the sequence-level training for visual object tracking and analyzes how the performance of a tracking algorithm improves by resolving the aforementioned inconsistency issues. Without adding any architectural components, we train a tracker end-to-end by simulating sequence-level tracking scenarios with a properly matching reward system in the framework of reinforcement learning (RL). Specifically, our tracker observes a sequence of frames sampled from an actual tracking trajectory and optimizes the objective based on the test-time metric such as the average overlap [34, 16]. Our training strategy not only resolves the inconsistency in data distributions but also addresses the discrepancy in task objectives by teaching the tracker how its decision in the current frame affects future ones. Furthermore, this approach enables us to extend data augmentation to a temporal domain; on top of commonly-used data augmentation strategies in the spatial domain [23], we can simulate temporally-varying tracking scenarios for training, corresponding to videos with diverse object/camera motions (Sec. 3). Note that this new type of augmentation has not been available under the frame-level training of previous trackers. To sum up, the proposed sequence-level training allows a tracker to learn a robust strategy for realistic tracking scenarios by leveraging the simulated tracking samples, (*i.e.*, *sequence-level sampling*), the long-term objective (*i.e.*, *sequence-level objective*), and the data augmentation in the temporal domain (*i.e.*, *sequence-level augmentation*).

Our contributions are summarized as follows.

- We analyze the inherent drawbacks of frame-level training adopted in recent trackers, which motivate sequence-level training (SLT) for robust visual object tracking.
- We introduce an SLT strategy for visual tracking in an RL framework and propose an effective toolset of data sampling, training objectives, and data augmentation.
- We demonstrate the effectiveness of SLT using four recent trackers, SiamRPN++ [20], SiamAttn [39], TransT [4], and TrDiMP [32], and achieve competitive performance on the standard benchmarks, LaSOT [10], TrackingNet [24], and GOT-10k [16].
- We provide in-depth analyses of SLT by studying the effects of the sequence-level data sampling and the corresponding objective as well as the sequence-level data augmentation diversifying tracking episodes in a temporal domain.

2 Related Work

2.1 Visual Object Tracking (VOT)

There has been a large body of active research on VOT [30, 23, 18], which still remains one of the major topics in computer vision. Recent methods for VOT have greatly improved tracking performance based on deep learning with large-scale datasets [26,

[1] Both trackers A and B adopt SiamRPN++ as their network architectures, but the backbone of tracker A is frozen in the early training stages.

29, 22, 24, 16, 10]. Currently, state-of-the-art trackers are represented by two families of trackers: Siamese [1, 21, 20, 39, 4] and DiMP [7, 2, 8, 32]. The Siamese trackers [1, 21, 20] rely heavily on an effective template-matching mechanism that is trained offline using large-scale datasets. While being fast and accurate in a short term, they tend to be vulnerable to long-term tracking due to the lack of online adaptability. Recent variants mitigate this limitation by updating template features during tracking [44, 45] or using an attention mechanism to diversify the feature representation [39]. DiMP trackers [2, 8, 32] learn the online model predictor for target center regression and combine it with bounding box regression. Their model predictor, which is an iterative optimization-based neural module, is trained offline with a meta-learning-based objective and is used to build an online target model during tracking. Recently, Transformer-based architectures are adopted in both Siamese [4] and DiMP [32] trackers to enhance target feature representations.

Note that all of these state-of-the-art trackers are trained with frame-level objectives; it forces the model to take an instantaneous greedy decision at each frame, ignoring that the tracking errors accumulate over the sequence. In contrast, we study how to improve tracking by considering temporal dependency in the sequence of frames.

2.2 Reinforcement Learning for VOT

Our sequence-level training scheme is built on reinforcement learning (RL), which provides a natural framework for sequential decision-making on the problem with interactive temporal dependency. RL is not new in visual tracking and there exist several RL-based trackers [9, 17, 15, 27, 40, 3, 42, 41]. Most of them [9, 27, 17, 15] aim to assist tracking by learning an additional RL agent while considering both a given tracker and its input sequence as an environment. HP-Siam [9] uses RL to optimize hyper-parameters of the tracker such as scale step, penalty, and window weight. DRL-IS [27] and P-Track [17] learn a policy to decide the state transition of the tracker. EAST [15] uses RL to speed up tracking by learning to stop feed-forwarding frames through layers.

Only a few RL-based methods [40, 3, 41] learn the tracker itself as an RL agent, which performs actual tracking, *e.g.*, estimating a target bounding box. ADNet [40] formulates tracking as a discrete box adjustment problem, where at each time step the agent observes a current frame and a previous localization box and then decides discrete actions for adjusting the box. In a similar manner, ACT [3] learns to predict box transformation parameters in a continuous space of actions while PACNet [41] jointly learns both box estimation and state transition of the tracker. Their methods, however, require a specific form of output for training trackers in RL, *e.g.*, the output of box transformation parameters [3, 41] and pre-defined actions for box adjustment [40], and hardly exploit the advantage of RL training in a generic perspective. In contrast, we introduce a generic training strategy using RL and analyze its advantages over its frame-level counterpart, which is prevalent in recent state-of-the-art trackers. We advocate sequence-level training *per se* as the integral role of RL in tracking, showing that recent learnable trackers greatly benefit from RL-based training without additional components.

Regarding our effort to address the training-testing inconsistency, the most related work is [28], which tackles a similar problem in image captioning. It proposes self-critical sequence training (SCST), a form of the well-known REINFORCE [33] algo-

rithm, to train image captioning models directly on NLP metrics. We build our sequence-level training scheme on SCST and adapt it for visual object tracking.

3 Our Approach

3.1 Sequence-Level Training (SLT)

Given a video $\mathbf{v} = (v_0, ..., v_T)$ of $T+1$ frames and the ground-truth bounding box g_0 of the target object in the initial frame v_0, a tracker sequentially predicts a bounding box l_t of the target in each frame v_t, $t = 1, 2, ..., T$. The tracker, parameterized by θ, is modeled as a function π_θ that takes observation o_t and predicts l_t, *i.e.*, $l_t = \pi_\theta(o_t)$, where o_t is the information available at time t including the video frames $(v_0, ..., v_t)$, the initial target bounding box g_0, and the previously estimated target states $(l_1, ..., l_{t-1})$. In online tracking, most trackers estimate the current state based only on the previous prediction l_{t-1} and the observation of the current frame v_t, *e.g.*, searching for the optimal local window in v_t around l_{t-1}. The objective of a tracking algorithm is to maximize the *sequence-level* performance $r(\mathbf{l})$, where $\mathbf{l} = (l_1, ..., l_T)$ includes all the frames in a video and r is an evaluation metric, *e.g.*, average overlap ratio [34, 16].

Due to the sequential structure of the tracking process, its current decision l_t naturally affects the future ones $l_{t+1}, ..., l_T$. In the frame-level training, which is the de facto standard for recent methods [20, 39, 4, 2, 32, 37], however, trackers do not simulate sequential target state estimation procedure in training. In other words, given each frame v_t, trackers are trained to localize the target object bounding box g_t from its random perturbation $\rho(g_t)$ instead of l_{t-1}, where ρ is a random perturbation function. Such a frame-level approximation, *i.e.*, $l_{t-1} \approx \rho(g_t)$, requires additional hyper-parameters for ρ and, more importantly, introduces inconsistency of data distributions between training and testing; the trackers have no opportunity to learn how the previous decision affects the current one in a real tracking scenario.

To overcome the limitation of frame-level training and capture the temporal dependency between decisions, we build our sequence-level training scheme based on RL. We simulate the tracker (agent) on a sequence of video frames, and directly optimize with respect to the test-time evaluation metric:

$$L(\theta) := -\mathbb{E}_{\mathbf{l}\sim\pi_\theta}[r(\mathbf{l})] \,. \tag{1}$$

Note that this *sequence-level objective* directly optimizes the real objective of tracking, and thus is a more natural way to train trackers than frame-level counterparts. This objective relieves the aforementioned issues in frame-level training; the tracker observes the real data distributions via *sequence-level sampling*, which draws samples from actual tracking trajectories and facilitates learning temporal dependency in tracking.

To directly optimize the task objective in (1), we employ the REINFORCE algorithm [33]. According to the algorithm, the expected gradient is computed as follows:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{\mathbf{l}\sim\pi_\theta}[r(\mathbf{l})\nabla_\theta \log p_\theta(\mathbf{l})]. \tag{2}$$

In practice, the expected gradient is approximated by using a single Monte Carlo sample $\mathbf{l} = (l_1, ..., l_T)$ of sequential decisions from π_θ. For each training episode, the gradient

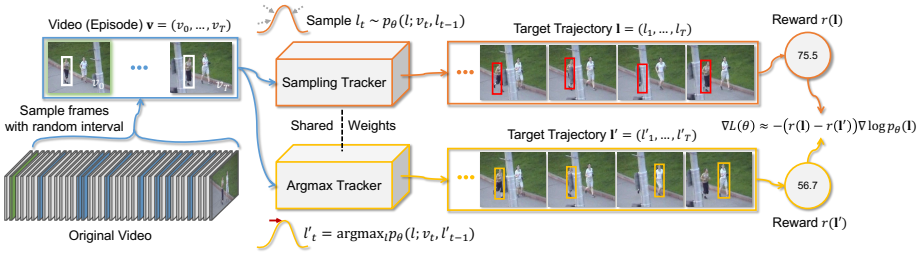


Fig. 2: Illustration of our sequence-level training framework. In training time, a video (episode), which is sampled from the original video with random intervals, is tracked twice by the sampling tracker and the argmax tracker. In this example, when a target person is fully occluded for a while, the argmax tracker mistakenly localizes the other person as the target due to its highest score in the occluded scene. In contrast, the sampling tracker stays nearby the previously estimated location (because of the random sampling) and successfully re-tracks the target object. In such a case, the reward becomes positive so that the sampled action is encouraged. In the opposite case, the reward becomes negative so that the sampled action is discouraged.

is given by

$$\nabla_\theta L(\theta) \approx -r(\mathbf{l}) \nabla_\theta \log p_\theta(\mathbf{l}). \tag{3}$$

To reduce the variance of gradient estimation, we adopt the self-critical sequence training (SCST) [28], which exploits the test-mode performance of the current model as a baseline for the reward. To be specific, we adopt two trackers sharing network parameters: a sampling tracker and an argmax tracker. During training, a video (episode) is played twice independently by both trackers. Given the probability distribution of actions, the sampling tracker decides the target bounding box stochastically while the argmax tracker selects the most confident one. If the agent obtains a higher reward from the sampling mode than the argmax mode, the resulting reward becomes positive to encourage the sampled actions. Otherwise, the agent receives a negative reward to suppress the sampled actions. In this way, we employ the SCST algorithm to train the tracker using the following gradient:

$$\nabla_\theta L(\theta) \approx -(r(\mathbf{l}) - r(\mathbf{l}')) \nabla_\theta \log p_\theta(\mathbf{l}), \tag{4}$$

where $r(\mathbf{l})$ and $r(\mathbf{l}')$ are rewards obtained from the current model by the sampling mode and the argmax mode during training, respectively. Such a REINFORCE-based training scheme is a certain realization of SLT and may be further improved by other RL-based algorithms. We illustrate the proposed sequence-level training pipeline in Fig. 2, and provide the pseudo-code in Algorithm 1.

3.2 Integration into Tracking Algorithms

We now present how to integrate the proposed SLT scheme into existing trackers. To demonstrate the effect of SLT, we adopt four representative trackers as our baselines:

Algorithm 1 Sequence-Level Training

procedure SEQUENCE-LEVEL TRAINING
Input: A tracker parametrized by θ, training dataset Γ
  while not converged **do**
    Sample a video $\mathbf{v} = (v_0, ..., v_T)$ and ground-truth $\mathbf{g} = (g_0, ..., g_T)$ from Γ
    Initialize the tracker by using $\{v_0,\ g_0\}$
    $l_0 = g_0$ ▷ Initial target location for the sampling tracker
    $l'_0 = g_0$ ▷ Initial target location for the argmax tracker
    for $t \in 1, ..., T$ **do**
      $l_t =$ sample l from $p_\theta(l; v_t, l_{t-1})$
      $l'_t = \arg\max_l p_\theta(l; v_t, l'_{t-1})$
    end for
    $r =$ evaluate$(\{l_1, ..., l_T\}, \{g_1, ..., g_T\})$
    $r' =$ evaluate$(\{l'_1, ..., l'_T\}, \{g_1, ..., g_T\})$
    $L = -(r - r') \sum_{t=1}^{t=T} \log p_\theta(l_t; v_t, l_{t-1})$ ▷ (4)
    $\theta = \theta - \alpha \nabla_\theta L$ ▷ Update model parameters
  end while
end procedure

SiamRPN++ [20], SiamAttn [39], TransT [4], and TrDiMP [32]. In the following, we briefly describe each tracker and explain how it is trained.

SLT-SiamRPN++. SiamRPN++ [20] is a representative Siamese tracker based on the region proposal network (RPN) [11]. This method tracks the target object by repeatedly matching between feature embeddings of a template image and a search image [1]. Specifically, the tracker outputs confidence scores and box coordinates of N anchor boxes and performs greedy selection to choose the most confident one, where its box coordinates are selected as the estimation of the target location in the current frame. Since the current prediction for the target state, *i.e.*, position and size, determines the search area in the next frame, this decision not only influences the current frame but also will potentially affect predictions in the future. Thus, we reinforce the box selection procedure of SiamRPN++ with the proposed sequence-level training strategy to teach the tracker temporal dependencies.

Since our training method assumes that the target localization of the tracker is a stochastic action, we convert the greedy anchor selection of SiamRPN++ to become stochastic. Let $\mathbf{x} = (x_1, ..., x_N) \in \mathbb{R}^N$ denote the output anchor scores of SiamRPN++, where $N = a \times H \times W$ denotes the number of candidates in a score map with size of $H \times W$ and a anchor types. We define a categorical distribution $p(n)$ as follows:

$$p(n) = \frac{\exp(\sigma^{-1}(x_n))}{\sum_{m=1}^{N} \exp(\sigma^{-1}(x_m))}, \tag{5}$$

where σ^{-1} indicates the logit (inverse sigmoid) function. Since x_n is a normalized score whose value is between 0 and 1, we first apply the logit function before applying the softmax function. In training time, the tracker samples an anchor box from p for

the current target localization. In test time, it selects the most confident anchor box deterministically as in the original SiamRPN++. For each training episode, the loss in the classification branch is given by

$$L = -(r(\mathbf{l}) - r(\mathbf{l}')) \sum_{t=1}^{T} \log p(n_t), \tag{6}$$

where $(r(\mathbf{l}) - r(\mathbf{l}'))$ is the self-critical reward (Sec. 3.1) and n_t is the sampled anchor box at frame t. The overall sequence-level training loss of SiamRPN++ is defined by combining the loss L for the classification branch and the ℓ_1 loss for the bounding-box regression branch [20], which is given by

$$L_{\text{siamrpn++}} = L + L_{\text{bbox}}. \tag{7}$$

SLT-SiamAttn. SiamAttn [39] is an extension of SiamRPN++ with an additional bounding box refinement module and a mask prediction module, along with attention modules for enhancing the feature representation. Similar to SiamRPN++, SiamAttn takes greedy anchor selection to choose the best target candidate among N anchor boxes from RPN. Once the box is chosen, SiamAttn additionally refines the bounding box using a deformable RoI pooling operation [6]. We thus use the same loss L for the classification branch. The overall sequence-level training loss of SiamAttn is to enhance the capability of target classification with SLT and increase the accuracy of localization modules with the help of sequence-level data sampling:

$$L_{\text{siamattn}} = L + \lambda_1 L_{\text{bbox}} + \lambda_2 L_{\text{refine-bbox}} + \lambda_3 L_{\text{mask}}, \tag{8}$$

where each loss term except for L follows [39]. The weight parameters are set as $\lambda_1 = 0.5$, $\lambda_2 = 0.5$, and $\lambda_3 = 0.2$.

SLT-TransT. TransT [4] adopts Transformer-like feature fusion networks into Siamese architecture and localizes the target object by computing attention between template vectors and search vectors. Unlike SiamRPN++ and SiamAttn, TransT has neither anchor points nor anchor boxes, and its prediction heads directly make N classification results and N normalized box estimations from fusion vectors corresponding to each position of the feature map, where $N = H \times W$ denotes the size of the feature map. We also take a categorical distribution $p(n)$ of equation (5) for N candidate vectors, and use the same loss L for the classification branch. The overall sequence-level training loss for TransT is:

$$L_{\text{transt}} = L + \lambda_4 L_{\text{bbox-L1}} + \lambda_5 L_{\text{bbox-GIoU}}, \tag{9}$$

where the box regression losses follow [4] and $\lambda_4 = 0.33$, $\lambda_5 = 0.13$ in our implementation.

SLT-TrDiMP. TrDiMP [32] is one of the state-of-the-art DiMP tracker using Transformer architectures. Its tracking procedure consists of two steps: target center prediction and bounding box regression. Given template samples generated from the initial frame, the model predictor generates a discriminative CNN kernel to convolve with the feature embedding from a search image for target response generation. The most confident location of the score map becomes the target center prediction, and starting from the randomly drawn candidate boxes around the location, the final bounding box estimation l_t is obtained from the IoU-Net-based box optimization [7, 8].

For sequence-level training of TrDiMP, we convert the target center prediction into stochastic action by simply taking softmax on its score prediction. Let $\mathbf{y} = (y_1, ..., y_N) \in \mathbb{R}^N$ denotes the center prediction score of TrDiMP, where $N = H \times W$ means a number of candidates in a score map with a spatial size of $H \times W$. Now we define a categorical distribution $p(n)$ as follows:

$$p(n) = \frac{\exp(y_n)}{\sum_{m=1}^{N} \exp(y_m)}. \tag{10}$$

The loss for the center prediction module is same with L, and the overall sequence-level training loss for TrDiMP is:

$$L_{\text{trdimp}} = L + \lambda_6 L_{\text{iou-net}}, \tag{11}$$

where the loss term for IoU-Net follows [32] and λ_6 is set to 0.0025.

3.3 Sequence-Level Data Augmentation

Learning visual tracking in a sequence level naturally motivates sequence-level data augmentation that is conceptually incompatible with frame-level training. To improve the data quality and avoid the over-fitting problem of the networks, data augmentation strategies in a spatial domain such as geometric transformation, color perturbations, and blur, are widely adopted for convolutional trackers [23]. Conventional frame-level training, however, treats the training sequence merely as a group of independent images that need to be sampled and cropped, hardly considering the relationship between each image, thereby missing the potential effect of data augmentation towards the temporal domain. In contrast, our sequence-level training effectively benefits from exploring diverse changes in the temporal axis that can enrich the tracking scenarios, as well as the conventional data augmentation strategies in the spatial axis.

Among many possible ways of sequence-level augmentation, here we focus on a simple frame-interval augmentation, *i.e.*, subsampling the training videos with different frame intervals. In our sequence-level augmentation setting, some episodes are sampled from the original video with random intervals as shown on the left side of Fig. 2. This scheme simulates dynamic visual differences along time steps and teaches the tracker to adapt to situations in which objects and/or cameras move faster. Diversifying the tracking scenarios in terms of frame rates makes the tracker improve or at least maintain the performance in general test videos. Experimental results in Sec. 4 show how our augmentation strategy affects the tracking performance. We believe that more advanced sequence-level augmentation strategies, *e.g.*, temporal motion blur, may help sequence-level training further in general.

4 Experiments

This section presents the effectiveness of the proposed sequencel-level training using four baseline trackers, SiamRPN++ [20], SiamAttn [39], TransT [4], and TrDiMP [32], on three standard benchmarks, LaSOT [10], TrackingNet [24], and GOT-10k [16].

4.1 Implementation Details

As widely adopted in deep RL [13, 28, 40], we pre-train the trackers with supervised learning, which is done with frame-level training in our case, to stabilize and speed up the subsequent SLT. For each training iteration, k tracking episodes are randomly sampled from training datasets, where k is set to 8 for SiamRPN++, TransT, and TrDiMP and 12 for SiamAttn, respectively. Each episode is composed of T video frames and a single template frame and T is a hyper-parameter for training. For frame-interval augmentation, the interval is randomly chosen every time sampling the video frames and its maximum is set to 7 for SiamRPN++ and SiamAttn, and 10 for TransT and TrDiMP, respectively. We use the average overlap (AO) score for the reward function r in Eq. 6.

Many recent trackers have post-processing strategies based on geometric priors [20, 39, 2, 32, 4]. For example, SiamRPN++ has the cosine-window penalty and the shape penalty, which prevent drastic updates in target bounding box estimation. These penalties are typically not applied during frame-level training, which also brings inconsistency between training and testing. However, our sequence-level training also resolves this inconsistency problem. Note that x_n in Eq. 5 is the anchor score after post-processing.

The argmax and sampling trackers in our SLT framework share all weights for training, and our tracker behaves like the argmax tracker during inference. Thus, the additional memory cost is marginal in training, and the test-time efficiency of the original tracker is not affected by SLT. Our algorithm is implemented in Python using PyTorch with NVIDIA RTX A6000 2GPUs.

4.2 Training Dataset

For a fair comparison, we aligned the pre-training datasets for the baseline with the fine-tuning datasets for SLT as similar as possible. 1) For SiamRPN++, we adopt LaSOT, TrackingNet, and GOT-10k for both pre-training and fine-tuning. 2) Following the original paper, SiamAttn is trained on LaSOT, TrackingNet, COCO [22], and YouTube-VOS [35]. Since COCO is an image dataset, a data augmentation scheme such as shift, scale, and blur is adopted to extend the image to compose an episode. Note that the data augmentation strategy except for frame-interval augmentation is used only for COCO. 3) Finally, TransT and TrDiMP are both pre-trained using LaSOT, TrackingNet, GOT-10k, and COCO, as same as the original papers, and then fine-tuned on three video datasets, LaSOT, TrackingNet, and GOT-10k.

4.3 Evaluation

We compare the performance of SLT with four baseline trackers. Note that for a fair comparison, we strictly maintain the same test-time hyper-parameters for each method

Table 1: Performance of sequence-level training on LaSOT, TrackingNet, and GOT-10k.

Method		LaSOT		TrackingNet			GOT-10k		
		AUC (Δ)	P_{Norm}	AUC (Δ)	P_{Norm}	P	AO (Δ)	$SR_{0.5}$	$SR_{0.75}$
SiamRPN++	Base	51.0	60.3	68.2	78.3	68.9	49.5	58.0	30.5
	+SLT	58.4 (+7.4)	66.6	75.8 (+7.6)	81.0	71.3	62.1 (+12.6)	74.9	49.0
SiamAttn	Base	54.8	63.5	74.3	80.9	70.6	53.4	61.8	36.4
	+SLT	57.4 (+2.6)	66.2	76.9 (+2.6)	82.3	72.6	62.5 (+9.1)	75.4	50.2
TrDiMP	Base	63.3	72.3	78.1	83.3	73.1	67.1	77.4	58.5
	+SLT	64.4 (+1.1)	73.5	78.1 (+0.0)	83.1	73.1	67.5 (+0.4)	78.8	58.7
TransT	Base	64.2	73.7	81.1	86.8	80.1	66.2	75.5	58.7
	+SLT	66.8 (+2.6)	75.5	82.8 (+1.7)	87.5	81.4	67.5 (+1.3)	76.5	60.3

for all datasets. Only TrDiMP uses a different hyper-parameter setting for evaluation in LaSOT following the baseline paper [32].

LaSOT [10] is a recently published dataset that consists of 1,400 videos with more than 3.5M frames in total. This benchmark is widely used to measure the long-term capability of trackers. The average video length of LaSOT is more than 2,500 frames, and each sequence comprises various challenging attributes. The one-pass evaluation (OPE) protocol is used to measure the normalized precision (P_{Norm}) and the area under curve (AUC) of the success plot. Table 1 shows that the proposed method consistently improves all baseline trackers.

TrackingNet [24] is a large-scale dataset that provides 30K videos in training split and 511 videos in test split. We evaluate our trackers on the test split of TrackingNet through the evaluation server. Table 1 shows that our SLT improves the AUC score by 7.6%p, 2.6%p, and 1.7%p for SiamRPN++, SiamAttn, and TransT, respectively. It is noteworthy that the simple convolutional tracker SiamRPN++ shows competitive performance with SiamAttn with the power of SLT.

GOT-10k [16] is a large-scale dataset that contains 10k sequences for training and 180 videos for testing. For evaluation metrics, the average overlap (AO) and the success rate (SR) at overlap thresholds 0.5 and 0.75 are adopted. Following the evaluation protocol of GOT-10k, we retrain our models using only the GOT-10k train split and submit the tracking results to the evaluation server. Since the GOT-10k benchmark does not provide mask annotations, SLT-SiamAttn is trained without the mask branch. Table 1 shows that our SLT successfully improves all the baseline trackers in all evaluation metrics. Baseline models are reproduced using only the GOT-10k train split.

Comparison with SOTA trackers. We compare the performance of the proposed SLT family with the other state-of-the-art trackers on LaSOT and TrackingNet as shown in Table 2 and 3. When compared to the recently proposed RL-based tracker PACNet [41], all four SLT trackers are showing superior performance by a large margin. SLT-TransT, which is our best model, achieves state-of-the-art performance in both benchmarks.

Table 2: Comparison with the state-of-the-art trackers on LaSOT.

	PACNet [41]	Ocean [43]	DiMP50 [2]	PrDiMP50 [8]	TransT [4]	STARK-ST50 [37]	STARK-ST101 [37]	**SLT-SiamRPN++**	**SLT-SiamAttn**	**SLT-TrDiMP**	**SLT-TransT**
AUC (%)	55.3	56.0	56.9	59.8	64.2	66.4	**67.1**	58.4	57.4	64.4	66.8
P_{Norm} (%)	62.8	65.1	64.3	68.0	73.7	76.3	**77.0**	66.6	66.2	73.5	75.5

Table 3: Comparison with the state-of-the-art trackers on TrackingNet.

	DiMP50 [2]	SiamFC++ [36]	MAML [31]	PrDiMP50 [8]	TransT [4]	STARK-ST50 [37]	STARK-ST101 [37]	**SLT-SiamRPN++**	**SLT-SiamAttn**	**SLT-TrDiMP**	**SLT-TransT**
AUC (%)	74.0	75.4	75.7	75.8	81.1	81.3	82.0	75.8	76.9	78.1	**82.8**
P_{Norm} (%)	80.1	80.0	82.2	81.6	86.8	86.1	86.9	81.0	82.3	83.1	**87.5**

Table 4: Comparison with the state-of-the-art trackers on GOT-10k. 'Add. data' denotes that trackers are trained using additional training datasets other than GOT-10k.

	Add. data	SiamFC++ [36]	DiMP50 [2]	Ocean [43]	PrDiMP50 [8]	TransT [4]	TrDiMP [32]	STARK-ST50 [37]	**SLT-SiamRPN++**	**SLT-SiamAttn**	**SLT-TrDiMP**	**SLT-TransT**
AO (%)		59.5	61.1	61.1	63.4	66.2	67.1	**68.0**	62.1	62.5	67.5	67.5
$SR_{0.5}$ (%)	-	69.5	71.7	72.1	73.8	75.5	77.4	77.7	74.9	75.4	**78.8**	76.5
$SR_{0.75}$ (%)		47.9	49.2	47.3	54.3	58.7	58.5	**62.3**	49.0	50.2	58.7	60.3
AO (%)	✓	-	60.4	-	65.2	71.9	68.6	71.5	56.9	62.8	69.0	**72.5**

Note that STARK-ST101 [37] uses deeper backbone (ResNet101) than TransT, which use ResNet50 backbone. SLT-TransT thus needs to be compared with STARK-ST50 for fairness. Table 4 also shows that both SLT-TrDiMP and SLT-TransT achieve comparable performance with state-of-the-art trackers on GOT-10k.

4.4 Analysis

We also analyze the effects of SLT using SiamRPN++ as the base tracker. The experimental analyses in this subsection are done on the *validation* split of GOT-10k and the *test* splits of LaSOT and TrackingNet.

Sequence-level training components. The benefit of SLT comes from sequence-level sampling, sequence-level objective, and sequence-level data augmentation. We validate the components of SLT by measuring the accuracy gains on the three benchmarks (Table 5) and performing an attribute-based analysis on LaSOT (Fig. 3).

Sequence-level sampling (SS). To measure the net effect of SS, we train a tracker with SS but use the frame-level objective. As shown at '+SS' in Table 5, by learning more accurate input data distribution, the tracker with SS outperforms the baseline with frame-level sampling by 4.1%p, 5.3%p, and 3.8%p, respectively, on the three benchmarks. SS allows the tracker to observe realistic appearance variations of target objects during tracking, making itself more robust to variations of aspect ratio, scale, rotation, and illumination (ARC, SC, R, IV) as seen in Fig. 3.

Sequence-level objective (SO). As shown at '+SS+SO' in Table 5, SO additionally improves the performance by 2.2%p, 1.5%p, and 3.6%p on three benchmarks, respectively. SO enables the tracker to reflect accumulated localization errors, preventing it from losing the target in challenging situations such as full occlusion, background clutters, and motion blur (FO, BC, MB) as seen in Fig. 3.

Table 5: Effect of sequence-level training components.

Benchmark	SiamRPN++			
	Baseline	+SS (Δ)	+SS+SO (Δ)	+SS+SO+SA (Δ)
LaSOT (AUC)	51.0	55.1 (+4.1)	57.3 (+6.3)	58.4 (+7.4)
TrackingNet (AUC)	68.2	73.5 (+5.3)	75.0 (+6.8)	75.8 (+7.6)
GOT-10k (AO)	66.4	70.2 (+3.8)	73.8 (+7.4)	74.3 (+7.9)

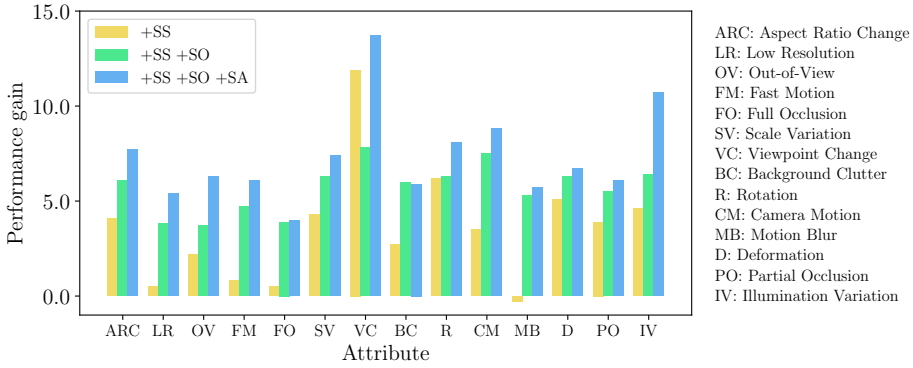


Fig. 3: Benefits of sequence-level training components to individual attributes on the LaSOT dataset. The baseline tracker is SiamRPN++, and the y-axis is performance (AUC) gain compared with the baseline tracker.

Sequence-level augmentation (SA). As shown at '+SS+SO+SA' in Table 5, SA further improves the performance by 1.1%p, 0.8%p, and 0.5%p, respectively, resulting in the significant gain of SLT in total as 7.4%p, 7.6%p, and 7.9%p. The effectiveness of SA is also evident from the improvement in the overall attributes in Fig. 3.

To show that the frame-interval augmentation strategy of SA also potentially helps in adapting to videos with diverse frame rates, we set up tracking scenarios with lower frame rates (*i.e.*, faster motion). In the evaluation protocol, we track objects every ith frame only, skipping all the other frames; when the interval is 1, the evaluation protocol is the same as the original benchmark. Table 6 shows that the frame-interval augmentation strategy not only improves the performance in normal videos, but also makes the tracker more robust to videos with lower frame rates.

Length of training episodes. In training time, we randomly sample training episodes of pre-defined sequence length T. Learning temporal dependency may be affected by the length of training sequences. We thus experiment with varying T while fixing the sampled frame interval to 1. The best result is obtained with $T = 24$, as can be seen in Table 7. When $T = 1$, the performance does not improve over the pre-trained tracker.

Frame-level pre-training. In our experiments, we perform SLT from a model pre-trained by frame-level training (FLT). We analyze the effect of warm-up FLT in Table 8,

Table 6: Effect of sequence-level augmentation (SA) in terms of video frame interval with the low frame rate protocol. The frame interval is denoted by i.

Method	SA	GOT-10k (AO)			LaSOT (AUC)			
		$i=1$	$i=2$	$i=3$	$i=1$	$i=2$	$i=3$	$i=4$
SiamRPN++	-	66.4	63.1	60.8	51.0	50.0	50.2	48.8
SLT-SiamRPN++	-	73.8	67.9	65.5	57.3	55.1	54.1	52.6
SLT-SiamRPN++	✓	74.3	70.8	67.8	58.4	56.9	56.2	54.6

Table 7: Effect of training sequence length.

Benchmark	Training sequence length (T)					
	1	4	8	16	24	32
GOT-10k (AO)	65.8	69.6	70.1	73.0	**73.8**	73.4

Table 8: Effect of frame-level pre-training. The zero (0) epoch stands for random initialization.

Method	Frame-level pre-training (epoch)				
	0	1	5	10	20
SiamRPN++	-	62.0	64.2	64.9	66.4
SLT-SiamRPN++	60.3	68.3	70.6	72.1	74.3

showing that it improves tracking performance indeed, and FLT with only a few epochs is sufficient for the warm-up. We also observed that the gain from SLT easily disappears by another few epochs of FLT (*i.e.*, FLT $\rightarrow$ SLT $\rightarrow$ FLT). In particular, the AO score of SiamRPN++ on the GOT-10k validation split reverted from 74.3% to 66.2% after 5 epochs of FLT. This indicates that SLT grants a unique gain, which FLT cannot provide.

For additional analysis and qualitative results, see the supplementary material.

5 Conclusion

We have proposed a novel sequence-level training strategy for visual object tracking to resolve the training-testing inconsistency problem of existing trackers. Unlike existing methods, it trains a tracker by actually tracking on a video and directly optimizing a tracking performance metric, boosting the generalization performance without modifying the model architecture. Experiments on SiamRPN++, SiamAttn, TransT, and TrDiMP trackers show that sequence-level sampling, objective, and augmentation are all effective in learning visual tracking.

Acknowledgments. This work was supported by Samsung Advanced Institute of Technology (Neural Processing Research Center), the NRF grants (No. 2021M3E5D2A01023887, No. 2022R1A2C3012210) and the IITP grants (No. 2021-0-01343, No. 2022-0-00959) funded by the Korea government (MSIT).

References

1. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: ECCV (2016) 1, 4, 7
2. Bhat, G., Danelljan, M., Gool, L.V., Timofte, R.: Learning discriminative model prediction for tracking. In: ICCV (2019) 1, 4, 5, 10, 12
3. Chen, B., Wang, D., Li, P., Wang, S., Lu, H.: Real-time 'actor-critic' tracking. In: ECCV (2018) 4
4. Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., Lu, H.: Transformer tracking. In: CVPR (2021) 1, 3, 4, 5, 7, 8, 10, 12
5. Chen, Z., Zhong, B., Li, G., Zhang, S., Ji, R.: Siamese box adaptive network for visual tracking. In: CVPR (2020) 1
6. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: ICCV (2017) 8
7. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: Atom: Accurate tracking by overlap maximization. In: CVPR (2019) 4, 9
8. Danelljan, M., Gool, L.V., Timofte, R.: Probabilistic regression for visual tracking. In: CVPR (2020) 1, 4, 9, 12
9. Dong, X., Shen, J., Wang, W., Liu, Y., Shao, L., Porikli, F.: Hyperparameter optimization for tracking with continuous deep q-learning. In: CVPR (2018) 4
10. 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) 3, 4, 10, 11
11. Girshick, R.: Fast R-CNN. In: ICCV (2015) 7
12. Henschel, R., Zou, Y., Rosenhahn, B.: Multiple people tracking using body and joint detections. In: CVPR (2019) 1
13. Hester, T., et al.: Deep q-learning from demonstrations. In: AAAI (2018) 10
14. Hu, H.N., Cai, Q.Z., Wang, D., Lin, J., Sun, M., Krahenbuhl, P., Darrell, T., Yu, F.: Joint monocular 3d vehicle detection and tracking. In: ICCV (2019) 1
15. Huang, C., Lucey, S., Ramanan, D.: Learning policies for adaptive tracking with deep feature cascades. In: ICCV (2017) 4
16. Huang, L., Zhao, X., Huang, K.: Got-10k: A large high-diversity benchmark for generic object tracking in the wild. TPAMI (2019) 3, 4, 5, 10, 11
17. III, J.S.S., Ramanan, D.: Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In: ICCV (2017) 4
18. Javed, S., Danelljan, M., Khan, F.S., Khan, M.H., Felsberg, M., Matas, J.: Visual object tracking with discriminative filters and siamese networks: A survey and outlook. arXiv preprint arXiv:2112.02838 (2021) 3
19. Jung, I., Son, J., Baek, M., Han, B.: Real-time MDNet. In: ECCV (2018) 1
20. Li, B., Wu, W., Wang, Q., Zhang, F., Junliang Xing, J.Y.: Siamrpn++: Evolution of siamese visual tracking with very deep networks. In: CVPR (2019) 1, 3, 4, 5, 7, 8, 10
21. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: CVPR (2018) 1, 4
22. Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft COCO: Common objects in context. In: ECCV (2014) 4, 10
23. Marvasti-Zadeh, S.M., Cheng, L., Ghanei-Yakhdan, H., Kasaei, S.: Deep learning for visual tracking: A comprehensive survey. IEEE Transactions on Intelligent Transportation Systems (2021) 1, 3, 9

24. Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., Ghanem, B.: TrackingNet: a large-scale dataset and benchmark for object tracking in the wild. In: ECCV (2018) 3, 4, 10, 11
25. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR (2016) 1
26. 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) 4
27. Ren, L., Yuan, X., Lu, J., Yang, M., Zhou, J.: Deep reinforcement learning with iterative shift for visual tracking. In: ECCV (2018) 4
28. Rennie, S.J., Marcheret, E., Mroueh, Y., Ross, J., Goel, V.: Self-critical sequence training for image captioning. In: CVPR (2017) 4, 6, 10
29. 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) 4
30. Smeulders, A.W., Chu, D.M., Cucchiara, R., Calderara, S., Dehghan, A., Shah, M.: Visual tracking: An experimental survey. TPAMI (2013) 1, 3
31. Wang, G., Luo, C., Sun, X., Xiong, Z., Zeng, W.: Tracking by instance detection: A meta-learning approach. In: CVPR (2020) 12
32. Wang, N., Zhou, W., Wang, J., Li, H.: Transformer meets tracker: Exploiting temporal context for robust visual tracking. In: CVPR (2021) 1, 3, 4, 5, 7, 9, 10, 11, 12
33. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning (1992) 4, 5
34. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: CVPR (2013) 1, 3, 5
35. Xu, N., Yang, L., Fan, Y., Yang, J., Yue, D., Liang, Y., Price, B., Cohen, S., Huang, T.: Youtube-vos: Sequence-to-sequence video object segmentation. In: ECCV (2018) 10
36. Xu, Y., Wang, Z., Li, Z., Yuan, Y., Yu, G.: Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In: CVPR (2020) 12
37. Yan, B., Peng, H., Fu, J., Wang, D., Lu, H.: Learning spatio-temporal transformer for visual tracking. In: ICCV (2021) 5, 12
38. Yan, B., Zhang, X., Wang, D., Lu, H., Yang, X.: Alpha-refine: Boosting tracking performance by precise bounding box estimation. In: CVPR (2021) 1
39. Yu, Y., Xiong, Y., Huang, W., Scott, M.R.: Deformable siamese attention networks for visual object tracking. In: CVPR (2020) 1, 3, 4, 5, 7, 8, 10
40. Yun, S., Choi, J., Yoo, Y., Yun, K., Young Choi, J.: Action-decision networks for visual tracking with deep reinforcement learning. In: CVPR (2017) 4, 10
41. Zhang, D., Zheng, Z., Jia, R., Li, M.: Visual tracking via hierarchical deep reinforcement learning. In: AAAI (2021) 4, 11, 12
42. Zhang, W., Song, R., Li, Y., et al.: Online decision based visual tracking via reinforcement learning. In: NIPS (2020) 4
43. Zhang, Z., Peng, H., Fu, J., Li, B., Hu, W.: Ocean: Object-aware anchor-free tracking. In: ECCV (2020) 1, 12
44. Zhang, Z., Gonzalez-Garcia, A., van de Weijer, J., Danelljan, M., Khan, F.S.: Learning the model update for siamese trackers. In: ICCV (2019) 4
45. Zhu, Z., Wang, Q., Li, B., Wu, W., Yan, J., Hu, W.: Distractor-aware siamese networks for visual object tracking. In: ECCV (2018) 4