Recursive Least-Squares Estimator-Aided Online Learning for Visual Tracking

Jin Gao ⬡, Yan Lu ⬡, Xiaojuan Qi ⬡, Yutong Kou, Bing Li ⬡, Liang Li ⬡, Shan Yu ⬡, and Weiming Hu ⬡, *Senior Member, IEEE*

Abstract—Tracking visual objects from a single initial exemplar in the testing phase has been broadly cast as a one-/few-shot problem, i.e., one-shot learning for initial adaptation and few-shot learning for online adaptation. The recent few-shot online adaptation methods incorporate the prior knowledge from large amounts of annotated training data via complex meta-learning optimization in the offline phase. This helps the online deep trackers to achieve fast adaptation and reduce overfitting risk in tracking. In this paper, we propose a simple yet effective recursive least-squares estimator-aided online learning approach for few-shot online adaptation without requiring offline training. It allows an in-built memory retention mechanism for the model to remember the knowledge about the object seen before, and thus the seen data can be safely removed from training. This also bears certain similarities to the emerging continual learning field in preventing catastrophic forgetting. This mechanism enables us to unveil the power of modern online deep trackers without incurring too much extra computational cost. We evaluate our approach based on two networks in the online learning families for tracking, i.e., multi-layer perceptrons in RT-MDNet and convolutional neural networks in DiMP. The consistent improvements on several challenging tracking benchmarks demonstrate its effectiveness and efficiency.

Index Terms—Online learning, few-shot online adaptation, visual tracking, continual learning, recursive least-squares estimation

◆

1 INTRODUCTION

GIVEN an arbitrary detected or annotated object of interest in the initial video frame, visual object tracking aims at *recognizing* and *localizing* other instances of the same object in subsequent frames to facilitate understanding how it moves through this video sequence. Recently this paradigm of tracking visual objects from a single initial exemplar in the testing phase has been broadly cast as a one-/few-shot meta-learning problem [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22],

which pushes forward the development of deep trackers and achieves unprecedented performance improvements.

It is well known that Deep Neural Networks (DNNs) with large capacity are typically data-hungry. However, a single initial exemplar in the initial frame can only provide limited training set for sequence-specific adaptation of a deep tracker. In contrast to some non-meta-learning-based pioneering arts which only fine-tune several light-weight classification network layers [23], [24] or train them from scratch [25] to do the initial adaptation at the first frame for *recognizing*, many meta-learning-based deep trackers [2], [3], [5], [6], [7], [9], [10], [11], [15], [16], [17], [18], [20] are proposed to speed up this initial adaptation process and reduce the risk of overfitting. The major insight is to inject additional prior knowledge into the meta-learning-based one-shot adaptation process to achieve explicit model initialization θ_{init}^0 through solving massive amounts of small meta-learning tasks based on the single initial exemplar in the offline learning process.

It is also noteworthy that, in the visual tracking testing phase, it is possible to augment the initial limited training set to update the trackers online by exploiting the position predictions of the object in subsequent frames as additional samples. Despite the fact that introducing imperfect predictions into the learning process may have the effect of accumulating errors and leading to tracking drift, the conditions under which a visual tracking system setup is performed are somehow less extreme [26] in comparison with one-/few-shot classification. As the number of arriving samples increases, one may find an ideal and straightforward way to adapt the tracking models to new environments online and avoid overfitting, which is fine-tuning the model parameters over the whole increasingly larger sample set at each update step. However, in the circumstances of online deep trackers, it may encounter a significant computational overhead due

to the requirement of more gradient descent iterations to achieve good convergence, which prohibits high tracking speed. In practice, a handful of samples are maintained during the whole testing phase in a way similar to the *sliding-window* strategy with fixed-window size. Again, to speed up this online adaptation and reduce the risk of overfitting, some of the aforementioned meta-learning-based one-shot initial adaptation methods are extended [11], [17], [18], and some novel meta-learners are also carefully designed to offline-train updaters [4], [8], [12], [13], [14], [21], [22] for effective online adaptation, namely few-shot online adaptation.

Despite the success of both the meta-learning-based one-shot initial adaptation and few-shot online adaptation strategies in visual tracking to avoid overfitting and reduce memory or computation requirements based on fast domain adaptation, complex optimization is required in the offline training phase – especially for the latter adaptation strategy. Specifically, both of them incorporate the prior knowledge from large amounts of annotated training data for fast domain adaptation, e.g., the training splits of *ImageNet DET* and *VID* [27], *MS-COCO* [28], *GOT10k* [29], *TrackingNet* [30] and *LaSOT* [31]. In this paper, we take a different perspective than the above methods for addressing the few-shot online adaptation problem, and propose a simple yet effective online adaptation method based on recursive least-squares (RLS) estimation [32]. The motivation behind this is that we find it is feasible to sidestep the computational overhead of fine-tuning over the whole gradually enlarged sample set by exploring the RLS's ability to "not forget" the old knowledge about the object after discarding the old data. Our approach can be treated as a widely applicable and integrable module for existing deep trackers without incurring too much extra computational cost in the sense that it improves the plain gradient descent optimization over the *sliding-window* data for tracking whereas neither the time to perform optimizations nor the storage consumption significantly increase.

In other words, we propose to use RLS estimation to aid the online adaptation. Our aim is to retain past memory while updating tracking models sequentially based on a handful of samples maintained at each update step. This resembles the emerging continual learning literature [33], [34], [35], [36], which has recently proved valuable in numerous typical supervised learning and reinforcement learning-based tasks where some static models are offline trained based on continual learning with fixed volumes of data. The few-shot online adaptation is also closely related to generic online learning, which basically aims at updating the predictor for future data at each time index while a continuous stream of data become available in a sequential order. It sometimes need dynamically adapting to new patterns in a stream when the data itself change from time to time, which is just right for visual tracking as shown in Fig. 1. So our approach departs from the prior arts in that we are the first to improve on the online learning procedure by incorporating memory retention in the spirit of continual learning in humans without catastrophic forgetting.

The concrete implementation to demonstrate the effectiveness of our approach is based on the fact that the calculations in deep networks, no matter multi-layer perceptrons (MLPs) or convolutional neural networks (ConvNets), can be quickly performed using fast linear algebra routines by
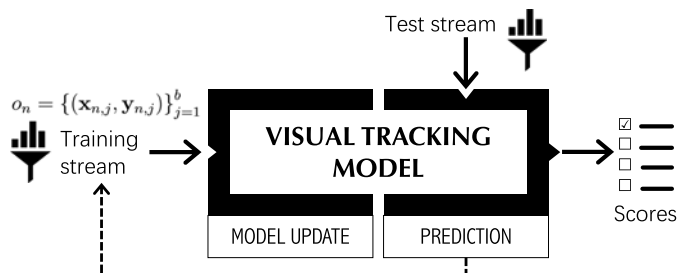

Fig. 1. Online learning in visual tracking. Here we denote the collected b training samples at time index n as $o_n = \{(\mathbf{x}_{n,j}, \mathbf{y}_{n,j})\}_{j=1}^b$.

organizing the network parameters in matrices. We study two networks in the online learning families for tracking: the MLPs-based as in RT-MDNet [24], and the ConvNets-based as in DiMP [10]. The reason for choosing these baselines is that they both use the *sliding-window* strategy for online learning classification models and *recognizing* objects in tracking, and the exploited optimization methods only enable the learning to converge to a *local* point with respect to the fixed-sized sample set at each update step. Instead, our improved optimization method in a recursive fashion can enable the learning to approximately converge to a *global* point with respect to all the historical training samples ever seen including the discarded old data. What's more, we do not rely on offline training to improve the tracking performance, and all the validations are based on the off-the-shelf models of the baselines in their papers.

In summary, through network online learning we investigate a continual-learning-inspired online learning approach with RLS estimator-aided online adaptation in tracking, which is orthogonal to the investigation of meta-learning approaches. We believe the observations in this paper will improve the understandings of network online learning for visual tracking, and we expect our simple-to-implement approach will further advance the few-shot online adaptation research. Our improved code and raw results are released at https://github.com/Amgao/RLS-OnlineTrack.

2 RELATED WORK

2.1 Online Learning in Visual Tracking

Online learning has been an important part of visual tracking for about ten years since the classic IVT tracker [37] was first proposed in 2008. The subsequent research has been concentrating on online robust classifier construction, including MIL [38], Struck [39], MEEM [40], TGPR [41], DCF-based approaches [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], deep classifiers [23], [24], [25], and some recent long-term trackers [19], [53], just to name a few. This is very different from the typical supervised learning tasks training with fixed volumes of data, e.g., image classification [54] and object detection [55].

In general, adapting to new data continually may give the online tracking models a tremendous advantage over the Siamese-style trackers [2], [3], [6], [7], [9], [15], [56] with online learning abandoned in discriminating background distractors. This instance-level discrimination power largely relies on the convergence rate of optimization algorithms, especially within limited optimization iterations allowed in the update stage for the sake of tracking speed. The

challenges may stem from the learning rate selection, model initialization, non-convex cost function, and so on [57]. Besides providing a good model initialization to account for the convergence challenges in the update stage as the aforementioned meta-learning-based one-shot initial adaptation methods [5], [10], [11], [16], [17], [18] have done, some novel LSTM model-based [4], [8], [21], [22], ConvNet model-based [14], and gradient-based [12], [13] meta-learners are proposed to efficiently adapt tracking models to appearance variations. Moreover, the Conjugate Gradient [58] methodology-based online optimizers [25], [47], [48] or Steepest Descent methodology-based ones [10] are also carefully designed to improve convergence while online learning.

In addition, despite the benefits of improved instance-level discrimination power with frequent update, the excessive update with limited memory size due to the discarding of old data may result in overfitting to recent training samples, deteriorating the robustness. Being robust is crucial for increasing the flexibility in tackling the different distortions of the object appearance over time (e.g., intrinsic object scale and pose variations, and variations caused by extrinsic illumination change, camera motion and occlusion). In other words, the plain online learning in visual tracking, e.g., the standard gradient descent (GD) or its momentum-based stochastic twin (SGD) [59] in the earlier online trackers [23], [24], [44], faces challenges in meeting both the instance-level discrimination and robustness demands at the same time, as they are sometimes contradictory [60]. Apart from the aforementioned meta-learning-based few-shot online adaptation methods, some online trackers widely apply the moderately infrequent and underfitted update [10], [16], [25], [47], [48], passive-aggressive update [49], [50], [51], [52], or long short-term complementary update [23], [24] settings in the update stage to relieve suffering from overfitting to recent training samples, or even corrupted samples.

These observations inspire the design of our proposed RLS estimator-aided online learning approach for visual tracking, which not only has instance-level discrimination power in case of background distractors as other online trackers does, but also can enhance the robustness ability thanks to the memory retention despite the discarding of old data.

2.2 One-/Few-Shot Learning in Visual Tracking

It is well known that humans can effortlessly learn novel concepts after only one or a few exposures by exploiting the knowledge primitives accrued in a lifetime. Inspired by this one-/few-shot learning ability of humans, there has been a recent resurgence of interest in two kinds of machine learning problems, i.e., (a) learning a meta-level knowledge across a large number of distinct training tasks to rapidly generalise to new concepts with small training data, which is termed as meta-learning or learning-to-learn (e.g., [3], [61], [62]), and (b) learning consecutive tasks to continually adapt to new data without forgetting how to perform previously trained tasks, namely continual learning (e.g., [33], [34], [35], [36]). Since the limited-data regime that characterises the visual tracking setup has attracted much attention recently, a promising research direction is the one set by applying one-/few-shot learning paradigm to visual tracking.

Siamese networks are first exploited to formulate the one-shot object recognition task as image matching via learning feature representations that preserve the class neighborhood structure and then inducing the l_1 component-wise distance metric to join the two Siamese twins for similarity computation [61]. This metric-learning-based meta-learning approach is further extended to be capable of predicting network parameters in [3] and two more distance metrics are evaluated, i.e., the inner-product and the euclidean distance. In [2], a Siamese architecture that is fully-convolutional with respect to the search image by obtaining the convolution parameters from the template image, namely SiamFC, is designed for dense and efficient matching-based visual tracking in a one-shot manner. This efficiency essentially benefits from encoding the inner-product similarity between the template image and each spatial location in the search image. Following the SiamFC paradigm, many Siamese network-based one-shot learning methods for initial adaptation [6], [7], [9], [10], [15], [16], [20] and few-shot learning methods for online adaptation [4], [8], [12], [13], [14], [22] in visual tracking are studied.

MAML [62], [63], a model-agnostic meta-learning approach, aims to help the network to learn a set of good initialization parameters that are suitable for fine-tuning to produce good generalization performance on a new task efficiently. Since it is compatible with different models trained with gradient descent without changing their architectures, some recent works [5], [11], [17], [18] formulate tracking in a MAML-based one-shot initial adaptation or few-shot online adaptation framework. They either improve the existing trackers [5], [18] or directly convert a modern object detector into a tracker [11], [17], leading to improvements in tracking speed and robustness, or resusability of the advancement in object detection.

Continual learning, an attractive route to artificial general intelligence, has been demonstrated to be capable of solving the classic image classification problem [33], [34], [35] or Atari 2600 games [33] through sequential learning. Besides overcoming catastrophic forgetting in the offline training phase like these earlier works, a preliminary version of this paper published in CVPR'20 [1] is the first to improve on the online learning procedure by incorporating memory retention and demonstrate its success in visual tracking. It has theoretically and experimentally demonstrated retaining past memory while updating tracking models sequentially based on RLS estimation can tackle catastrophic forgetting incurred by plain SGD in the online learning of MLP layers in RT-MDNet [24]. The present work goes further to investigate RLS's ability of a convolutional layer for online learning improvement in tracking. To this end, a recent state-of-the-art tracker DiMP [10] is exploited as another baseline, which enables us to demonstrate the effectiveness of our approach on the more challenging *VOT2018/2019* benchmarks [64], [65], the large-scale short-term *TrackingNet-test* [30] and *GOT10k-test* [29] tracking benchmarks, and the long-term *LaSOT-test* [31], *OxUvA-dev* [66] and *TLP* [67] benchmarks. The consistent improvements on several challenging benchmarks against the plain GD prove our online adaptation method's efficiency without additional offline training and too much tedious work on parameter adjusting. In the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.

org/10.1109/TPAMI.2022.3156977, we also provide additional technical details to explain, for instance, how to extend the derivation of normal equations for MLP in [1] to handle non-linear activation functions and how the derivation of normal equations for MLP in [1] can be easily transplanted to the case of cross-entropy loss.

3 RLS ESTIMATOR-AIDED ONLINE LEARNING WITH MEMORY RETENTION

Nowadays, online learning in visual tracking is increasingly related to online learning network parameters from a carefully designed and time-varying sampling set [10], [24]. By organizing the network parameters in matrices, the calculations in the networks can be quickly performed using fast linear algebra routines. In this section, a simple least-squares estimation (LSE) problem is first investigated in matrix form to facilitate the interpretation of our approach.

3.1 LSE in Online Learning

Consider the time-varying training set $\{X(n), Y(n)\}$ consisting of the observable input samples and desired responses within all of the time span until the time index n, the method of least squares can be used to estimate a set of weighting coefficients in $\mathbf{W}_n$ at the time index n in the online learning procedure, such that the following *cost function* is minimized

$$\mathscr{L}(\mathbf{W}_n) = \left\| \mathbf{\Lambda}(n)\big(Y(n) - X(n)\mathbf{W}_n^\top\big) \right\|^2 + \delta\beta^n \|\mathbf{W}_n\|^2, \quad (1)$$

where the left squared error term is defined based on the l_2 norm of the *residual* vectors, and the regularizing term, controlled by a positive constant δ, is included to stabilize the solution to the recursive formulation of this LSE problem [32], [68] as shown in Section 3.2. Here we denote $X(n)$, $Y(n)$ and $\mathbf{\Lambda}(n)$ in a *block* manner. The sampling set $\{X(n), Y(n)\}$ has growing dimensions in the number of data *blocks* in rows (growing-window)

$$X(n) = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} \in \mathbb{R}^{nb \times p}, \quad Y(n) = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} \in \mathbb{R}^{nb \times q}, \quad (2)$$

with $X_i \in \mathbb{R}^{b \times p}, Y_i \in \mathbb{R}^{b \times q}$ and $i = 1, 2, \dots, n$. Here b, p and q are to denote the *block* size and the dimensions of the input and output for each sample. The diagonal weighting matrix

$$\mathbf{\Lambda}(n) = diag\left(\sqrt{\beta}^{n-1}\mathbf{I}_b, \dots, \sqrt{\beta}\mathbf{I}_b, \mathbf{I}_b \right) \quad (3)$$

is to diminish the importance of those previous observations (rows) with $0 < \beta \leq 1$, which can measure the memory of the algorithm. The regularizing term is also reformulated to make its beneficial effect forgotten with time for β less than unity.

According to the method of LSE [32], the optimum value of $\widehat{\mathbf{W}}_n$ at the time index n, for which Eq. (1) attains its minimum value, is defined by the normal equations. Before that, we can rewrite the *cost function* in Eq. (1) as

$$\mathscr{L}(\mathbf{W}_n) = \sum_{i=1}^n \beta^{n-i}\left(\frac{1}{b}\left\|Y_i - X_i\mathbf{W}_n^\top\right\|^2\right) + \frac{\delta}{b}\beta^n\|\mathbf{W}_n\|^2. \quad (4)$$

Solving for $\mathbf{W}_n$ for which $\nabla_{\mathbf{W}_n}\mathscr{L}$ is zero, we may write the normal equations as follows in matrix form:

$$\widehat{\mathbf{W}}_n = \mathbf{Z}_n\mathbf{\Phi}_n^{-1}, \quad (5)$$

where $\mathbf{Z}_n$ and $\mathbf{\Phi}_n$ are defined respectively by Eqs. (6) and (7), i.e., the time-average cross-correlation matrix $\mathbf{Z}_n$ between the desired output and the input is shown by the formula

$$\mathbf{Z}_n = \sum_{i=1}^n \beta^{n-i}Y_i^\top X_i, \quad (6)$$

and the time-average cross-correlation matrix $\mathbf{\Phi}_n$ of the input including the regularizing term is defined by

$$\mathbf{\Phi}_n = \sum_{i=1}^n \beta^{n-i}X_i^\top X_i + \delta\beta^n\mathbf{I}. \quad (7)$$

In practice, SGD or its mini-batch version (MBSGD) is commonly used to optimize the parameters in $\mathbf{W}_n$ to obtain an approximate optimum value $\widehat{\mathbf{W}}_n^{appr}$ for problems similar to Eq. (1) or (4). The initialization of this optimization at the time index n is commonly set to $\widehat{\mathbf{W}}_{n-1}^{appr}$, which is the approximate optimum value at the time index $n-1$. This iterative optimization procedure for online learning can not only avoid the computationally expensive inverse operation in Eq. (5) for high-dimensional input data, but also facilitate the training of networks in deep learning by allowing processing the continuously and increasingly growing large dataset with limited GPU memory size. As the number of arriving *blocks* increases, however, fine-tuning the parameters over the whole increasingly larger training set $\{X(n), Y(n)\}$ will give rise to more gradient descent iterations to achieve good convergence, which leads to a significant computational overhead and thus prohibits high tracking speed. Moreover, the growing training set results in growing storage consumption.

In order to make a compromise between the optimization and tracking speed, traditional online learning in visual tracking commonly uses a carefully designed *sliding-window* strategy with fixed-window size to incorporate the new data segments and also remove the influence of the obsolete data. Similar to this traditional online learning paradigm, a strategy to solve Eq. (1) can be derived as follows. First, if we denote s as the number of *blocks* of the fixed-window size, then for $n \geq s$, the fixed-window data can be written from the growing-window data given in Eq. (2) by discarding the oldest $n - s$ *blocks* of data, i.e.,

$$X(n) = \begin{pmatrix} X_{n-s+1} \\ \vdots \\ X_n \end{pmatrix}, \quad Y(n) = \begin{pmatrix} Y_{n-s+1} \\ \vdots \\ Y_n \end{pmatrix}. \quad (8)$$

Second, the approximate optimum value $\widehat{\mathbf{W}}_n^{appr}$ can be obtained by minimizing Eq. (1) based on some optimization strategies, e.g., standard gradient descent or its stochastic twin SGD as in RT-MDNet [24], and Steepest Descent (SD) methodology as in DiMP [10]. The initialization at the time index n is also set to $\widehat{\mathbf{W}}_{n-1}^{appr}$. For instance, the standard batch gradient descent (BGD) can be used to optimize $\mathbf{W}_n$ based on the following iterations

$$\mathbf{W}_n^{iter+1} = \mathbf{W}_n^{iter} - \eta\big(\mathbf{W}_n^{iter}\mathbf{\Phi}_n - \mathbf{Z}_n\big) , \tag{9}$$

$$\mathbf{W}_n^0 = \widehat{\mathbf{W}}_{n-1}^{appr} . \tag{10}$$

Here η is a pre-defined learning rate, and $\mathbf{\Phi}_n$ and $\mathbf{Z}_n$ are obtained by summing over the fixed-window data.

Despite the success of this *sliding-window* strategy with fixed-window size, a limitation of historical memory loss prevents unveiling the power of online deep tracking in [10], [24], because this strategy will potentially be faced with the problem of being prone to overfitting to the recent fixed-window data especially when the size of the window is limited. This raises an interesting question that we discuss in the next subsection: Can we have a more elegant path to retain the memory while neither increasing the time to perform optimizations nor the storage consumption significantly? We hope our work and the presented online learning approach allow researchers to focus on the still unsolved critical challenges of online deep tracking.

3.2 Recursive Formulation With Memory Retention

This section investigates the relationship between $\widehat{\mathbf{W}}_n$ and $\widehat{\mathbf{W}}_{n-1}$ at the consecutive time indices and presents an interesting alternative to solving Eq. (1) in a recursive fashion [32]. In contrast to the aggressive online learning in the previous *sliding-window* strategy, the characteristic of this recursive online learning can spontaneously reduce the risk of overfitting thanks to the memory retention though the oldest *blocks* of data are discarded. In other words, the catastrophic forgetting of old knowledge about discriminating the discarded training samples during the independent optimization over the newly updated fixed-window sample set in Eqs. (9) and (10) is mitigated.

To simplify the derivation, we only consider one sample pair in each *block*, which means $b = 1$ in Eq. (1). Let $X_i = \mathbf{x}_i^\top$ and $Y_i = \mathbf{y}_i^\top$, the *cost function* of Eq. (1) will degrade to

$$\mathscr{L}(\mathbf{W}_n) = \sum_{i=1}^n \beta^{n-i}\big\|\mathbf{y}_i - \mathbf{W}_n\mathbf{x}_i\big\|^2 + \delta\beta^n\|\mathbf{W}_n\|^2 , \tag{11}$$

and the cross-correlation matrices in Eq. (5) are defined by

$$\mathbf{Z}_n = \sum_{i=1}^n \beta^{n-i}\mathbf{y}_i\mathbf{x}_i^\top \tag{12}$$

$$\mathbf{\Phi}_n = \sum_{i=1}^n \beta^{n-i}\mathbf{x}_i\mathbf{x}_i^\top + \delta\beta^n\mathbf{I} . \tag{13}$$

The normal equations in the case of $b = 1$ can be seen as an expansion of the single sample case without summing over all the historical observations:

$$\widehat{\mathbf{W}}_n = \mathbf{y}_n\mathbf{x}_n^\top\big(\mathbf{x}_n\mathbf{x}_n^\top + \delta\mathbf{I}\big)^{-1} . \tag{14}$$

Isolating the terms corresponding to $i = n$ from the rest of the summations in Eqs. (12) and (13) yields the following recursions for updating $\mathbf{Z}_n$ and $\mathbf{\Phi}_n$ respectively

$$\mathbf{Z}_n = \beta\mathbf{Z}_{n-1} + \mathbf{y}_n\mathbf{x}_n^\top , \tag{15}$$

$$\mathbf{\Phi}_n = \beta\mathbf{\Phi}_{n-1} + \mathbf{x}_n\mathbf{x}_n^\top . \tag{16}$$

With $\mathbf{\Phi}_n$ assumed to be non-singular and thus invertible, we may obtain the following recursive equation for the inverse of $\mathbf{\Phi}_n$ by applying the Sherman-Morris formula [69]

$$\mathbf{\Phi}_n^{-1} = \beta^{-1}\mathbf{\Phi}_{n-1}^{-1} - \frac{\beta^{-2}\mathbf{\Phi}_{n-1}^{-1}\mathbf{x}_n\mathbf{x}_n^\top\mathbf{\Phi}_{n-1}^{-1}}{1 + \beta^{-1}\mathbf{x}_n^\top\mathbf{\Phi}_{n-1}^{-1}\mathbf{x}_n} . \tag{17}$$

If we denote $\mathbf{P}_n = \mathbf{\Phi}_n^{-1}$ and let

$$\mathbf{k}_n = \frac{\beta^{-1}\mathbf{x}_n^\top\mathbf{P}_{n-1}}{1 + \beta^{-1}\mathbf{x}_n^\top\mathbf{P}_{n-1}\mathbf{x}_n} , \tag{18}$$

then Eq. (17) can be rewritten as

$$\mathbf{P}_n = \beta^{-1}\mathbf{P}_{n-1} - \beta^{-1}\mathbf{P}_{n-1}\mathbf{x}_n\mathbf{k}_n . \tag{19}$$

It is surprising to find that multiplying $\mathbf{x}_n^\top$ by each side of Eq. (19) yields the following simple expression

$$\mathbf{x}_n^\top\mathbf{P}_n = \beta^{-1}\mathbf{x}_n^\top\mathbf{P}_{n-1} - \beta^{-1}\mathbf{x}_n^\top\mathbf{P}_{n-1}\mathbf{x}_n\mathbf{k}_n = \mathbf{k}_n . \tag{20}$$

This facilitates the derivation for obtaining a recursive formula for the normal equations in the case of $b = 1$ when we substitute Eqs. (15) and (19) into Eq. (5)

$$\begin{aligned}\widehat{\mathbf{W}}_n &= \beta\mathbf{Z}_{n-1}\mathbf{P}_n + \mathbf{y}_n\mathbf{x}_n^\top\mathbf{P}_n \\ &= \mathbf{Z}_{n-1}\mathbf{P}_{n-1} - \mathbf{Z}_{n-1}\mathbf{P}_{n-1}\mathbf{x}_n\mathbf{k}_n + \mathbf{y}_n\mathbf{k}_n \\ &= \widehat{\mathbf{W}}_{n-1} + \big(\mathbf{y}_n - \widehat{\mathbf{W}}_{n-1}\mathbf{x}_n\big)\mathbf{k}_n \\ &= \widehat{\mathbf{W}}_{n-1} - \big(\widehat{\mathbf{W}}_{n-1}\mathbf{x}_n - \mathbf{y}_n\big)\mathbf{x}_n^\top\mathbf{P}_n ,\end{aligned} \tag{21}$$

where the expression inside the brackets on the right-hand side of the last line represents the *residual* based on the old least-squares estimate of the parameters to be learned.

Extending the above derivation to the case of $b > 1$ is not straightforward, because the equations of Eqs. (15) and (16) will be expanded to

$$\mathbf{Z}_n = \beta\mathbf{Z}_{n-1} + Y_n^\top X_n , \tag{22}$$

$$\mathbf{\Phi}_n = \beta\mathbf{\Phi}_{n-1} + X_n^\top X_n , \tag{23}$$

where

$$X_i = \begin{pmatrix}\mathbf{x}_{i,1}^\top \\ \vdots \\ \mathbf{x}_{i,b}^\top\end{pmatrix} \quad\text{and}\quad Y_i = \begin{pmatrix}\mathbf{y}_{i,1}^\top \\ \vdots \\ \mathbf{y}_{i,b}^\top\end{pmatrix} . \tag{24}$$

This obviously prohibits the derivation of Eq. (17). Alternatively, using the fact that the mean squared error inside the brackets of Eq. (4), which is based on the l_2 norm of the *residual* vectors for all of the sample pairs in the i_{th} *block*, serves as an upper bound of the error term for the virtual sample pair $\{\overline{\mathbf{x}}_i, \overline{\mathbf{y}}_i\}$, i.e.,

$$\big\|\overline{\mathbf{y}}_i - \mathbf{W}_n\overline{\mathbf{x}}_i\big\|^2 \leq \frac{1}{b}\big\|Y_i - X_i\mathbf{W}_n^\top\big\|^2 , \tag{25}$$

where

$$\overline{\mathbf{x}}_i = \frac{1}{b}\sum_{j=1}^b \mathbf{x}_{i,j} \quad\text{and}\quad \overline{\mathbf{y}}_i = \frac{1}{b}\sum_{j=1}^b \mathbf{y}_{i,j} , \tag{26}$$

we can instead approximately obtain the recursive formula for the normal equations in the case of $b > 1$ based on this virtual sample pair

$$\widehat{\mathbf{W}}_n = \widehat{\mathbf{W}}_{n-1} - \left(\widehat{\mathbf{W}}_{n-1}\overline{\mathbf{x}}_n - \overline{\mathbf{y}}_n\right)\overline{\mathbf{x}}_n^\top \mathbf{P}_n , \qquad (27)$$

where $\mathbf{P}_n$ is updated based on $\overline{\mathbf{x}}_n$ using Eqs. (18) and (19).

In practice, the exact optimum value $\widehat{\mathbf{W}}_{n-1}$ is hardly to be obtained and its approximate optimum value $\widehat{\mathbf{W}}_{n-1}^{appr}$ is commonly obtained from some optimization algorithm. However, we cannot directly use Eq. (27) to obtain $\widehat{\mathbf{W}}_n^{appr}$ as the relationship between $\widehat{\mathbf{W}}_n^{appr}$ and $\widehat{\mathbf{W}}_{n-1}^{appr}$ does not rigorously comply with Eq. (27). Inspired from the gradient descent paradigm, we can, for instance, model the changing between $\widehat{\mathbf{W}}_{n-1}^{appr}$ and $\widehat{\mathbf{W}}_n^{appr}$ by the following iterations

$$\mathbf{W}_n^{iter+1} = \mathbf{W}_n^{iter} - \eta\left(\mathbf{W}_n^{iter}\overline{\mathbf{x}}_n - \overline{\mathbf{y}}_n\right)\overline{\mathbf{x}}_n^\top \mathbf{P}_n , \qquad (28)$$

$$\mathbf{W}_n^0 = \widehat{\mathbf{W}}_{n-1}^{appr} , \qquad (29)$$

where η is another pre-defined learning rate. From the above equations, we see that the approximate optimum value $\widehat{\mathbf{W}}_n^{appr}$ is obtained only based on the most recently arrived data *block* n, though it is dedicated to satisfying the *cost function* in Eq. (4) and hence retaining the historical memory. Consequently using the fact that the matrix product $\left(\mathbf{W}_n^{iter}\overline{\mathbf{x}}_n - \overline{\mathbf{y}}_n\right)\overline{\mathbf{x}}_n^\top$ represents the gradient with respect to the weighting parameters $\mathbf{W}_n^{iter}$ for the virtual input $\overline{\mathbf{x}}_n$ of the n_{th} data *block* when considering its squared-error loss $\frac{1}{2}\left\|\overline{\mathbf{y}}_n - \mathbf{W}_n^{iter}\overline{\mathbf{x}}_n\right\|^2$ without the l_2 regularization term, we can cast the iterations in Eqs. (28) and (29) as an improved gradient descent algorithm for online learning with memory retention. Note that the initial condition of $\mathbf{P}_n$ is set to

$$\mathbf{P}_0 = \mathbf{\Phi}_0^{-1} = (\delta\mathbf{I})^{-1} = \mathbf{I}/\delta . \qquad (30)$$

Discussion. The traditional gradient descent algorithms following the *sliding-window* strategy for online learning compute the gradient at each optimization iteration based on the *cost function* similar to Eq. (1) for the whole batch of fixed-window data (BGD) or its mini-batch after random shuffle operation (MBSGD). An alternative strategy for improving these traditional algorithms to retain memory by performing minimal changes of their settings and implementations is to cast the whole batch (for BGD) or one mini-batch (for MBSGD) data as a data *block* at some time index in Eq. (4), and then improve the optimization iterations based on our proposed approach in Eqs. (28) and (29). Note that the gradient term based on the virtual sample pair in Eq. (28) can be approximately replaced with the real gradient at each optimization iteration computed from BGD or MBSGD.

4 RECURSIVE FORMULATION FOR MULTIPLE FULLY-CONNECTED LAYERS IN RT-MDNET

In the online deep tracker RT-MDNet [24] that is dedicated to target classification, the multi-layer perceptron (MLP) with several fully-connected 1-D layers is applied to the last stage of the deep learning architecture for classification. The features present in the final 2-D feature maps are concatenated into one long input vector to the following MLP. In this

section, we will show how our proposed RLS estimator-aided online learning approach for memory retention can be realized in this single-head real-time tracker to online learn those MLP layers.

Online Learning in RT-MDNet. Being identical to MDNet [23], RT-MDNet only updates the fully-connected MLP layers (fc4-6) $\left\{\mathbf{W}^l\right\}_{l=4}^6$ in an online manner while keeping the convolutional feature extraction layers (conv1-3) $\left\{\mathbf{W}^l\right\}_{l=1}^3$ fixed, namely update stage. Before that, the MLP layers also need to be fine-tuned using the samples from the initial frame to customize themselves to a new testing domain, namely initialization stage. There are two types of memory being maintained in the update stage to make compromises between robustness and adaptiveness: the long-term memory for regular updates with the samples collected for a long period of time; the recent short-term memory with the occasional updates triggered whenever the score of the estimated target is below a threshold, indicating the unreliable tracking.

During online tracking, the cross-entropy loss based on the softmax output of the single fully-connected binary classification layer $\mathbf{W}^6$ is used to fine-tune or update all the MLP layers and the optimization algorithm of MBSGD is exploited, which is a compromise between BGD and the single instance SGD. Specifically, suppose there are b samples in one mini-batch used for one optimization iteration of the MBSGD process, it is thus the following partial derivatives over the mean value of the Binary Cross Entropy (BCE) loss for all the b samples that is used for updating $\mathbf{W}^l$ of each layer

$$\Delta\mathbf{W}^l = \nabla_{\mathbf{W}^l}\left[\frac{1}{b}\sum_{j=1}^b BCE\left(\mathbf{x}_j, \mathbf{y}_j, \left\{\mathbf{W}^l\right\}_{l=4}^6\right)\right] + \lambda_r^l\mathbf{W}^l , \quad (31)$$

$$\mathbf{W}^l \longleftarrow \mathbf{W}^l - \eta_r^l\Delta\mathbf{W}^l , \qquad (32)$$

where $\mathbf{x}_j$ is the long input vector of the sample j in the current batch, $\mathbf{y}_j$ is the class (positive/negative) prediction vector, and λ_r^l and η_r^l are the weight decay and pre-defined learning rate respectively in the MBSGD process of RT-MDNet. The weight decay encourages the learned weights to be small in magnitude to improve the generalization performance of them [70].

RLS Estimator-Aided RT-MDNet. In our preliminary version [1], we have claimed and demonstrated that each layer of MLP in the case of squared-error loss function can be represented by the system of normal equations for RLS estimation by simply ignoring non-linear activation functions. In the present work, we add more explanations on how to extend the derivation of normal equations for MLP in [1] to generally handle other layers including non-linear activation functions and leave them in the supplementary material, available online, due to space constraints. We also provide more explanations on why all the derivations in the demonstration can be easily transplanted to the case of cross-entropy loss in this supplementary material, available online and refer the readers to [71] for more details. This means we can exploit our proposed RLS estimator-aided online learning approach to improve the updating of each MLP layer in RT-MDNet in order to retain memory. In specific, if we cast one mini-batch of samples used for one optimization iteration of

the MBSGD process as a data *block* at some time index n as in Section 3, then this optimization iteration can be carried out as follows

$$\mathbf{W}_n^l = \mathbf{W}_{n-1}^l - \widetilde{\eta}_r^l \Delta \mathbf{W}_n^l \mathbf{P}_n^l , \qquad (33)$$

$$\mathbf{P}_0^l = \mathbf{I}/\delta_r^l , \qquad (34)$$

for memory retention, where $\widetilde{\eta}_r^l$ is the new learning rate, δ_r^l is the positive constant for RLS estimation and the term $\mathbf{W}_{n-1}^l \left(\mathbf{I} - \widetilde{\eta}_r^l \lambda_r^l \mathbf{P}_n^l \right)$ serves as the weight decay for optimization. Note that Eq. (33) is only implemented once for improving the MBSGD process, in contrast to Eq. (28), as the same mini-batch of samples may appear several times for optimization at each update step. $\mathbf{P}_n^l$ for each layer is updated using Eqs. (18) and (19) based on each layer's virtual input

$$\overline{\mathbf{x}}_n^l = \frac{1}{b} \sum_{j=1}^{b} \mathbf{x}_j^l , \qquad (35)$$

where $\mathbf{x}_j^l$ is the input to layer l for the sample j.

Since we only concentrate on the online tracking part to improve RT-MDNet, we leave all the other parts of RT-MDNet almost unchanged including, for example, the improved RoIAlign technique, all their hyper-parameter counterparts and the off-the-shelf model trained with additional instance embedding loss in their paper. As the RLS estimator-aided online learning can overcome catastrophic forgetting of historical memory, we thus do not need maintaining the long-term memory to achieve robustness any more. That is to say, we still use the same recent short-term memory for RLS estimator-aided regular updates, while the setting of the occasional updates triggered by unreliable tracking is preserved intact, in that there may be failure cases for the model based on the regular updates and the model with the occasional updates customized (overfitting) to the recent short-term memory may work well with more discrimination. Note that the regularly updated model is fixed during the occasional updates. More details about the improved RT-MDNet with the RLS estimator-aided online learning, namely RLS-RTMDNet, are explained in Algorithm 1.

5 RECURSIVE FORMULATION FOR CONVOLUTIONAL LAYERS IN DiMP

In the recent state-of-the-art online tracker DiMP [10] that is dedicated to both target classification and location estimation, high level knowledge is incorporated to train the target location estimation component, following its pioneering work ATOM [25], through extensive offline learning so as to predict the overlap between the target object and an estimated bounding box; meanwhile a more powerful classification component is carefully designed to incorporate additional prior knowledge for fast initial adaptation through the offline meta-training process and finally online updated to guarantee high discriminative power in the presence of distractors. In specific, the target classification component has its target model constitute the filter weights of a convolutional layer and provides target classification scores as regression output without non-linear output activation. It also employs the block features from a ResNet-based backbone, which is pre-

trained on ImageNet [27] and fine-tuned on tracking datasets along with the target classification and location estimation components. In this section, we will show how our proposed RLS estimator-aided online learning approach for memory retention can be realized in this multi-head real-time tracker to online learn the convolutional layer.

Algorithm 1. RLS Estimator-Aided RT-MDNet

Input: Off-the-shelf deep RT-MDNet tracking model with multiple layers $\{\mathbf{W}^l\}_{l=1}^6$, initial $\mathbf{P}_0^l = \mathbf{I}/\delta_r^l$ for each fully-connected layer $(4 \leq l \leq 6)$, and test sequence with first frame annotated

Output: Estimated target states in the rest frames

1. Pre-process identical to RT-MDNet, e.g., draw positive sample set S_1^+ and negative sample set S_1^-
2. Fine-tune $\{\mathbf{W}^l\}_{l=4}^6$ using $S_1^+ \cup S_1^-$ in initialization stage:
   for the n_{th} improved MBSGD iteration **do**
       Update $\mathbf{P}_n^l$ based on the virtual input $\overline{\mathbf{x}}_n^l$ using Eqs. (18) and (19)
       Update $\mathbf{W}^l \leftarrow \mathbf{W}_n^l$ using Eqs. (31) and (33)
3. Initialize the frame index set $\mathcal{T} \leftarrow \{1\}$
   repeat
4.   Find the optimal target state like in RT-MDNet
       if *target score* > 0 **then**
5.   Draw S_t^+ and S_t^-, and set $\mathcal{T} \leftarrow \mathcal{T} \cup \{t\}$
6.   **if** $|\mathcal{T}| > \tau$ **then** $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\min_{\upsilon \in \mathcal{T}} \upsilon\}$
       if *target score* ≤ 0 **then**
7.   **if** $\{\mathbf{W}_{\text{bk}}^l\}_{l=4}^6 = \varnothing$ **then** create backups $\{\mathbf{W}_{\text{bk}}^l\}_{l=4}^6 \leftarrow \{\mathbf{W}^l\}_{l=4}^6$
8.   Occasionally update $\{\mathbf{W}^l\}_{l=4}^6$ using $S_{\upsilon \in \mathcal{T}}^+ \cup S_{\upsilon \in \mathcal{T}}^-$ based on the original MBSGD
       else if $t \bmod 10 = 0$ **then**
9.   **if** $\{\mathbf{W}_{\text{bk}}^l\}_{l=4}^6 \neq \varnothing$ **then** recover backups $\{\mathbf{W}^l\}_{l=4}^6 \leftarrow \{\mathbf{W}_{\text{bk}}^l\}_{l=4}^6$, and set $\{\mathbf{W}_{\text{bk}}^l\}_{l=4}^6 = \varnothing$
10.   Regularly update $\{\mathbf{W}^l\}_{l=4}^6$ using $S_{\upsilon \in \mathcal{T}}^+ \cup S_{\upsilon \in \mathcal{T}}^-$:
           for the n_{th} improved MBSGD iteration **do**
               Update $\mathbf{P}_n^l$ based on the virtual input $\overline{\mathbf{x}}_n^l$ using Eqs. (18) and (19)
               Update $\mathbf{W}^l \leftarrow \mathbf{W}_n^l$ using Eqs. (31) and (33)
   until *end of sequence*

Online Learning in DiMP. Besides the target model's filter weights, the target classification component also includes an initializer network for initial filter prediction and several free parameter predictors for setting the target mask, spatial weight, regularization factor and even regression target in the discriminative learning loss. In the first frame, the filter weights are initialized based on the initializer network and then fine-tuned based on the initial training samples from data augmentation, namely initialization stage. Subsequently, the extracted feature map annotated by a Gaussian centered at the estimated target location in every frame is added to the fixed-window sample set as a training sample. During online tracking, the convolutional layer with a single output channel is online optimized every 20_{th} frame, namely update stage.

Given the training set of samples, we can approximately formulate the online learning objective in DiMP as follows based on the l_2 norm of the *residuals*

$$\mathscr{L}(\mathbf{W}) = \sum_{j=1}^{N} \left\| \mathbf{\Gamma}_j \odot \left(\mathbf{Y}_j - \mathbf{W} * \mathbf{X}_j \right) \right\|^2 + \frac{\lambda_d}{2} \|\mathbf{W}\|^2 . \qquad (36)$$

Here, $\mathbf{X}_j$ is the extracted feature map of the sample j, $\mathbf{Y}_j$ is its annotated Gaussian confidence map, $\mathbf{W}$ represents the weights of the convolutional layer, $*$ denotes standard multi-channel convolution, λ_d is the weight decay parameter, N is the fixed-window size, $\odot$ denotes Hadamard product, and the impact of each training sample along with its spatial weight is controlled by the weighting factor $\mathbf{\Gamma}_j$.

In modern deep learning architectures, each convolutional layer is usually converted to a general matrix-matrix multiplication (GEMM) operation for fast execution on modern CPUs or GPUs. This is achieved by an unfolding and duplication of the input and an unrolling and concatenation of the kernels to produce the input matrix and kernel matrix. These transformations are realized based mainly on a function

$$f(\cdot) = img2col(\cdot)$$

in the modern deep learning libraries. Consequently, the l_2 norm term in Eq. (36) can be rewritten as

$$\left\| \mathbf{\Gamma}_j \odot \left(\mathbf{Y}_j - \mathbf{W} * \mathbf{X}_j \right) \right\|^2$$
$$= \left\| f^{\top}(\mathbf{\Gamma}_j) \odot \left(f^{\top}(\mathbf{Y}_j) - f^{\top}(\mathbf{W}) f(\mathbf{X}_j) \right) \right\|^2 . \quad (37)$$

If we denote the column number of $f(\mathbf{X}_j)$ as M, the k_{th} column in $f(\mathbf{X}_j)$ as $\mathbf{x}_{jk}$, the k_{th} element in $f(\mathbf{Y}_j)$ as y_{jk} and the k_{th} element in $f(\mathbf{\Gamma}_j)$ as γ_{jk}, then the learning objective of Eq. (36) can be rewritten as

$$\mathscr{L}(\mathbf{W}) = \sum_{j=1}^{N} \sum_{k=1}^{M} \left\| \sqrt{\gamma_{jk}} \left(y_{jk} - f^{\top}(\mathbf{W}) \mathbf{x}_{jk} \right) \right\|^2 + \frac{\lambda_d}{2} \|\mathbf{W}\|^2. \quad (38)$$

During online tracking, the BGD optimization or the SD methodology can be used to update $\mathbf{W}$ based on the above loss function. Specifically, in the case of BGD optimization, it is the following partial derivatives over the above squared-error loss for all the N samples that is used for updating $\mathbf{W}$ at each optimization iteration of the BGD process:

$$\Delta \mathbf{W} = \nabla_{\mathbf{W}} \mathscr{L}(\mathbf{W}) , \quad (39)$$

$$\mathbf{W} \longleftarrow \mathbf{W} - \eta_d \Delta \mathbf{W} , \quad (40)$$

where η_d is a pre-defined learning rate.

RLS Estimator-Aided DiMP. From Eq. (38) we can easily find that our proposed RLS estimator-aided online learning approach is also able to improve the updating of the convolutional layer $\mathbf{W}$ in DiMP in order to retain memory. This is achieved by applying BGD optimization method in the update stage of DiMP and then improving the updating process. In specific, if we cast all columns of $f(\mathbf{X}_j)$ for all the training samples in the N-sized sample set as a data *block* at some time index n as in Section 3, then the optimization of our improved BGD process can be carried out as follows

$$f\left(\mathbf{W}_n^{iter+1}\right) = f\left(\mathbf{W}_n^{iter}\right) - \widetilde{\eta}_d f\left(\Delta \mathbf{W}_n^{iter+1}\right) \mathbf{P}_n , \quad (41)$$

$$\mathbf{W}_n^0 = \widehat{\mathbf{W}}_{n-1}^{appr} , \quad (42)$$

$$\mathbf{P}_0 = \mathbf{I}/\delta_d , \quad (43)$$

where $\widehat{\mathbf{W}}_{n-1}^{appr}$ is the last approximate optimum value, $\widetilde{\eta}_d$ is the new learning rate, δ_d is the positive constant for RLS estimation, the term $f(\mathbf{W}_n^{iter})(\mathbf{I} - \widetilde{\eta}_d \lambda_d \mathbf{P}_n)$ serves as the weight decay for optimization, and $\mathbf{P}_n$ is updated using Eqs. (18) and (19) based on the virtual input, which is the mean value of the data *block* at the current time index n:

$$\bar{\mathbf{x}}_n = \frac{1}{\sqrt{NM}} \sum_{j=1}^{N} \sum_{k=1}^{M} \sqrt{\gamma_{jk}} \mathbf{x}_{jk}. \quad (44)$$

Since we only concentrate on the online tracking part to improve DiMP, we also leave all the other parts of DiMP almost unchanged including, for example, the default off-the-shelf model trained using the train splits of LaSOT, TrackingNet, GOT10k and MS-COCO, all their hyper-parameter counterparts and the settings of initialization stage to achieve $\widehat{\mathbf{W}}_0^{appr}$. It is noteworthy that the optimization strategy of the SD methodology is preserved intact along with all its parameter settings in the initialization stage. The reason is that the SD methodology exhibits superior convergence speed compared to the BGD optimization strategy as claimed in [10]. More details about the improved DiMP with the RLS estimator-aided online learning, namely RLS-DiMP, are explained in Algorithm 2.

Algorithm 2. RLS Estimator-Aided DiMP

Input: Off-the-shelf deep DiMP tracking model with one convolutional layer $\mathbf{W}$ for target classification, initial $\mathbf{P}_0 = \mathbf{I}/\delta_d$ for this layer $\mathbf{W}$, and test sequence with first frame annotated

Output: Estimated target states in the rest frames

1 Initialize the fixed-window sample set S with the data augmented samples extracted from the first frame

2 Predict a reasonable initial estimate of $\mathbf{W}$ and then fine-tune it using the SD methodology identical to DiMP in initialization stage, and set $\widehat{\mathbf{W}}_0^{appr} = \mathbf{W}$ in the following

repeat

3   Find the optimal target state like in DiMP

    if *update_flag* **then**

4   Update the sample set S along with its sample weights identical to DiMP

        if $t - 1 \bmod 20 = 0$ *or* *hard_negative* **then**

            for *the n_{th} improved BGD procedure* **do**

                Set $\mathbf{W}_n^0 = \widehat{\mathbf{W}}_{n-1}^{appr}$

                Update $\mathbf{P}_n$ based on the virtual input $\bar{\mathbf{x}}_n$ using Eqs. (18) and (19)

                repeat

                    Compute the current gradient $\Delta \mathbf{W}_n^{iter}$

                    Update $\mathbf{W}_n^{iter}$ based on Eq. (41)

                until *end of iteration*

                Update $\mathbf{W} \leftarrow \widehat{\mathbf{W}}_n^{appr}$

until *end of sequence*

---

6 EXPERIMENTS

6.1 Implementation Details and Baseline Setup

We here only show the hyper-parameters specific to our recursive LSE-aided online learning, and leave all the other hyper-parameters and the off-the-shelf tracking models

shared with the original RT-MDNet and DiMP trackers unchanged[1] unless otherwise specified.

In our previous implementation of RLS-RTMDNet in [1], δ_r^l in Algorithm 1 and the parameter to measure the memory in Eq. (13) for each of the fully-connected layers $\mathbf{W}^4$, $\mathbf{W}^5$ and $\mathbf{W}^6$ were simply set to 1.0; the learning rates $\widetilde{\eta}_r^l$ for them in Eq. (33) in both initialization and update stages were fixed to 0.01, 0.01 and 0.1. According to [32], [68], however, initializing $\mathbf{\Phi}_0$ with a "small" value for the most application scenarios will help the recursive LSE-aided online learning to achieve its optimal performance, which is insensitive to the exact "small" value used. So we experimentally found some new configurations for RLS-RTMDNet in this paper, and the experimental results show that roughly setting δ_r^l of each layer to $5e^{-4}$ and directly inheriting the learning rates $\widetilde{\eta}_r^l$ from the original RT-MDNet tracker give rise to a further performance improvement over our previous tracking results in [1], especially with a significant improvement on *UAV123* [72] . We re-implemented the public source code of the original RT-MDNet as a baseline and followed all of its default settings to demonstrate the effectiveness of our proposed online learning framework in the case of the MLP layers. Although we leave many settings of RT-MDNet unchanged in our RLS-RTMDNet tracker, there are still some differences in the online tracking part related to the solely maintained short-term memory and the weight backups, in addition to the improved MBSGD. So we prepared another baseline by only replacing the improved MBSGD with original MBSGD in Algorithm 1, leading to a degraded version, namely simpleRT-MDNet.

We also selected DiMP to demonstrate the effectiveness of our proposed online learning framework in the case of the convolutional layer. For the updating of $\mathbf{W}$ in Algorithm 2, we simply set δ_d to 0.1 and the parameter to measure the memory in Eq. (13) to 1.0. The feature dimensionality of $\mathbf{X}_j$ in Eq. (36) is 512, which makes the dimension size of $\mathbf{P}_n$ large enough for memory retention. Besides re-implementing the public source code of DiMP as one baseline, we add three more kinds of baselines for a more in-depth analysis. First, replacing only the improved BGD with vanilla BGD in Algorithm 2 leads to the baseline termed as BGD-DiMP, which can be used to directly validate the effectiveness of our approach. We experimentally found the optimal learning rates for BGD-DiMP while performing the BGD optimization with five recursions every 20_{th} frame, i.e., $3e^{-2}$ for short-term tracking benchmarks (the average sequence length does not exceed 600 frames), and $3e^{-3}$ for long-term ones. This learning rate setting is also applied in our RLS-DiMP for fair comparison. Second, increasing the SD optimizer recursion number from two to five for updating $\mathbf{W}$ every 20_{th} frame in the original DiMP leads to the baseline denoted by DiMP*, which can be used to study the overfitting issue caused by fast converging to a *local* point with respect to the fixed-sized sample set at each update step. Finally, considering that the exponential moving average (EMA) strategy is widely applied in a slow-moving

TABLE 1
Ablation Study Comparison of RLS-RTMDNet With Its Corresponding Baselines RT-MDNet and simpleRT-MDNet on the *OTB-2015* and *UAV123* Benchmarks

Trackers	OTB-2015		UAV123		Running Speed	GPU Memory Consumption
	Prec.	Succ.	Prec.	Succ.		
RT-MDNet	0.857	0.634	0.723	0.513	~32 *fps*	< 2400MiB
simpleRT-MDNet	0.854	0.630	0.722	0.509	~32 *fps*	< 2400MiB
RLS-RTMDNet	0.871	0.644	0.727	0.525	~31 *fps*	< 2400MiB

Prec. and Succ. Denote Precision Score at 20 Pixels and AUC of Success Plot Respectively. The Average Over 50 Runs Is Reported for These MDNet-Style Trackers.

fashion for striking a compromise between adapting quickly and maintaining robustness in visual tracking [42], [43], [44], [45], [46], [47], [49], [51], [73], [74], [75], [76], [77], or stabilizing the network representation with smoother changes in self-supervised learning [78], [79], [80], we can also use this strategy to combine the newly updated parameters with previous ones to save historical information and avoid overfitting. By setting different EMA coefficient for discounting older parameters in slow-moving, moderate-moving and fast-moving ways, we can achieve a new kind of baselines for comparison (see Sec. 6.3.1 for more details). Our RLS-DiMP and all the above baselines are implemented with a ResNet50 backbone [81].

Note that all the evaluations of our methods and the re-implemented baselines are conducted by running them several times due to the stochastic nature of RT-MDNet and DiMP, and performed on a single Nvidia RTX 3090 GPU accompanied by an Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz.

6.2 Experimental Analyses of RLS-RTMDNet

We evaluate the proposed RLS estimator-aided online RLS-RTMDNet thoroughly over *OTB-2015* [82], *UAV123* [72] and *VOT2016/2017* [83], [84] benchmarks by following rigorously the evaluation protocols. We choose these small-scale tracking benchmarks for this evaluation due to the reason that the off-the-shelf RT-MDNet tracking model is trained without additionally relying on the *MS-COCO* dataset [28] or recently proposed large-scale tracking datasets, which makes it hard to achieve good results on the more challenging large-scale tracking benchmarks and thus do meaningful analyses.

6.2.1 Ablation Study

We start by summarizing the ablation study comparison of our proposed RLS-RTMDNet with its corresponding baselines on the *OTB-2015* and *UAV123* benchmarks in Table 1. *OTB-2015* includes 100 objects to be tracked in 98 challenging sequences, and provides a no-reset One-Pass Evaluation (OPE) criterion by running test trackers until the end of a sequence. In its success plot for quantifying performance, the success rate (SR) refers to the mean overlap precision (OP) score over all sequences and is plotted against a uniform range of some thresholds between 0 and 1. An area-under-the-curve (AUC) metric can also be computed to rank the trackers. The OP score is the fraction of frames in a
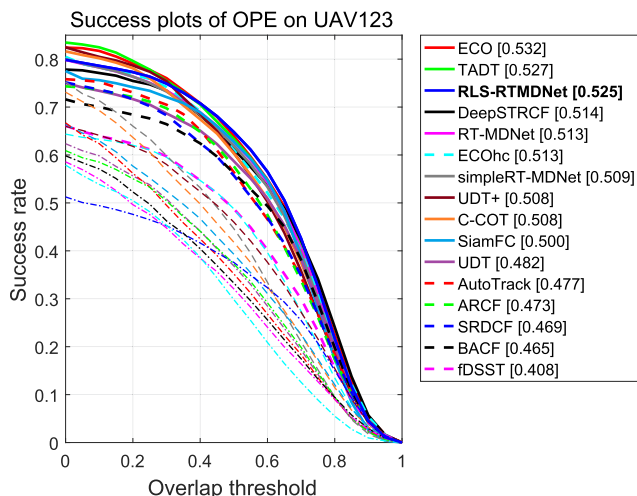
Fig. 2. Success plots for the whole *UAV123* benchmark. We report the average over 50 runs for our tracker RLS-RTMDNet and its corresponding baselines. The legends show the AUC scores. Best viewed in color.

TABLE 2
Comparison With Some Top-Performing Deep Trackers on UAV123 in Terms of Mean OP Given Thresholds 0.50 ($OP_{0.50}$) and 0.75 ($OP_{0.75}$)

UAV123	RT-MDNet	DeepSTRCF	**RLS-RTMDNet**	TADT	ECO
$SR_{0.5}$	0.627	0.612	0.648	0.634	0.640
$SR_{0.75}$	0.308	0.342	0.326	0.307	0.328

ResNet50 backbone. Results for our degraded version simpleRT-MDNet are also included.

VOT creates a platform for organizing tracking challenges every year since 2013 by establishing datasets, evaluation measures and toolkits. *VOT2017* departs from *VOT2016* in two aspects: 10 least challenging sequences in *VOT2016* is replaced with new ones; and a new experiment for evaluating real-time performance is introduced in *VOT2017*. Different from *UAV123*, all these *VOT* challenges apply a reset-based methodology in the toolkit. Whenever a failure is detected, the tracker is re-initialized five frames after the failure. Thus, two weakly correlated performance measures can be used: the accuracy (A) measures the average overlap between the predicted bounding box and the ground truth computed over the successfully tracked frames; the robustness is estimated by considering the reliability (R_S) which shows the probability that the tracker will still successfully track the object up to S frames since the last failure and is computed using the failure rate measure (R_{fr}, the average number of failures) as follows

$$R_S = \exp\left(-S\frac{R_{\mathrm{fr}}}{N_{\mathrm{frames}}}\right),$$

where N_{frames} is the average length of the sequences. A more principled expected average overlap (EAO) measure is also proposed to measure the expected no-reset average overlap (AO) of a tracker run on a short-term sequence, although it is computed from the reset-based methodology.

We first show the comparison on *UAV123* in Fig. 2 and Table 2. All the entries in this comparison can be roughly categorized into DCF-based, Siamese-style and MDNet-style approaches. In specific, the DCF-based trackers ECO [48], DeepSTRCF [50] and C-COT [47] all use pre-trained deep models to extract powerful features for more robust and accurate tracking. SiamFC [2] is the first proposed Siamese-style tracker which can track object very efficiently with online learning abandoned. This is achieved by exploiting end-to-end training along with the advent of deep learning for training tracking models. TADT [86] further improves the Siamese-style tracking framework by introducing a target-aware feature module to guide the generation of target-active and scale-sensitive features. UDT [77] is proposed to explore a feasible unsupervised training approach for deep tracking. AutoTrack [52] and ARCF [46] are both DCF-based trackers and dedicated to the real-world low-cost UAV tracking scenario for more efficient tracking, which has been validated in a real-world UAV localization system. As can be seen, the classical DCF-based tracker ECO with deep features and the Siamese-style tracker TADT can both surpass RT-MDNet and simpleRT-MDNet by large margins, while our approach can unveil the power of RT-MDNet to achieve

sequence where the inter-section-over-union overlap of the predicted and ground-truth rectangles exceeds a given threshold. In its precision plot, the precision encodes the proportion of frames where the euclidean distance between the predicted and ground-truth centroids is smaller than a given threshold. *UAV123* compiles a larger set of 123 videos (with more than $110K$ frames) captured by low-altitude UAVs, which is inherently different from *OTB-2015* despite that it also uses the same evaluation protocol to *OTB-2015*.

It is clearly and statistically shown in Table 1 that our online learning approach can unveil the power of RT-MDNet without incurring too much extra computational and memory cost. Specifically, without bells and whistles, our RLS-RTMDNet improves all the precision (at 20 pixels) and AUC scores over the baseline RT-MDNet on both the *OTB-2015* and *UAV123* benchmarks, giving absolute AUC gains no less than 1.0%. Our degraded version simpleRT-MDNet, however, exhibits a little inferior performance to RT-MDNet due to the reason that the update is based solely upon the recent short-term memory. It is noteworthy that our improvement of precision score on *UAV123* is much less than on *OTB-2015*, which can be attributed to the reason that the size of the ground-truth bounding boxes for most of the objects in *UAV123* is much too small so that the tracking deviation from object area does not punish the precision score at 20 pixels too much. This is also why the metric of normalized precision is proposed in [30].

6.2.2 Comparison With Others

We perform the state-of-the-art comparison on *UAV123* and *VOT2016/VOT2017* by comparing our proposed RLS-RTMDNet to the state-of-the-art CPU-based trackers, i.e., MEEM [40], SRDCF [44], Staple [74], CSRDCF with its real-time version CSRDCF++ [85], fDSST [43], BACF [45], ARCF [46], AutoTrack [52], ECOhc [48], and some deep trackers trained without additionally relying on the *MS-COCO* dataset or recently proposed large-scale tracking datasets, i.e., SiamFC [2], C-COT [47], RT-MDNet [24], Meta-SDNet [5], DeepSTRCF [50], UDT with its improved version UDT+ [77], TADT [86], ECO [48], LADCF-R50 [49] with a

TABLE 3
Comparison of Our RLS-RTMDNet With Its Corresponding Baselines and Some Related Competing Trackers on the *VOT2016/2017* Benchmarks; the Results are Reported as EAO, A, R_S ($S = 100$) and Real-Time EAO ($rtEAO$)

	VOT	Staple	SiamFC	MEEM	RT-MDNet	simpleRT-MDNet	MetaSDNet	CSRDCF++	ECOhc	**RLS-RTMDNet**	CSRDCF	C-COT	ECO	LADCF-R50
2016	$EAO \uparrow$	0.295	0.277	-	0.302	0.304	0.314	-	0.322	0.336	0.338	0.331	0.374	-
	$A \uparrow$	0.547	0.550	-	0.559	0.550	0.539	-	0.542	0.573	0.524	0.541	0.555	-
	$-\ln R_S \downarrow$	0.378	0.382	-	0.305	0.294	0.261	-	0.303	0.255	0.238	0.238	0.200	-
2017	$EAO \uparrow$	0.169	0.188	0.192	0.223	0.218	-	0.229	0.238	0.241	0.256	0.267	0.280	0.389
	$A \uparrow$	0.530	0.503	0.463	0.514	0.508	-	0.453	0.496	0.516	0.491	0.494	0.484	0.503
	$-\ln R_S \downarrow$	0.688	0.585	0.534	0.450	0.464	-	0.370	0.435	0.392	0.356	0.318	0.276	0.159
	$rtEAO \uparrow$	0.170	0.182	0.072	0.162	0.154	-	0.212	0.177	0.150	0.099	0.058	0.078	0.066

For All These Metrics Except rtEAO, the Stochastic Trackers are Run 15 Times on Each Sequence to Reduce the Variance of Their Results. Best Viewed in Color.

comparable performance with both of them and even the best $SR_{0.5}$ score.

The comparison on *VOT2016/2017* is shown in Table 3. As similar to the analyses in ablation study, our RLS-RTMDNet always consistently improves upon its baselines (i.e., RT-MDNet and simpleRT-MDNet) by achieving significant gains on *VOT2016/2017*, and our degraded version simpleRT-MDNet achieves similar or a little worse performance than RT-MDNet due to the lack of long-term memory. What's more, our performance gains are larger than MetaSDNet [5], which uses sophisticated meta-training techniques to train the MDNet tracker using more datasets than RT-MDNet.

6.3 Experimental Analyses of RLS-DiMP

We evaluate the proposed RLS estimator-aided online RLS-DiMP tracker thoroughly over the large-scale short-term *TrackingNet-test* [30], *GOT10k-test* [29] and *VOT2018/2019* [64], [65] tracking benchmarks, and long-term *LaSOT-test* [31], *OxUvA-dev* [66] and *TLP* [67] benchmarks by following rigorously the evaluation protocols. As the off-the-shelf DiMP tracking model is fully trained with the training splits of the *TrackingNet, LaSOT, GOT10k* and *MS-COCO* [28] datasets, we believe this evaluation on more challenging and larger benchmarks can clearly and statistically show the effectiveness of our online learning approach by unveiling the power of the state-of-the-art DiMP tracker.

6.3.1 Ablation Study

We start by summarizing the ablation study comparison of our proposed RLS-DiMP with its corresponding baselines

TABLE 4
Ablation Study Comparison of RLS-DiMP With its Corresponding Baselines on the Long-Term *LaSOT-test* Benchmark

Trackers	LaSOT-test			Running Speed	GPU Memory Consumption
	Prec.	*Norm. Prec.*	*Succ.*		
DiMP	$0.567^{\dagger}$	$0.650^{\dagger}$	$0.569^{\dagger}$	~41 *fps*	~2211 MiB
DiMP*	0.556	0.636	0.557	~40.6 *fps*	
BGD-DiMP	0.575	0.673	0.574	~41 *fps*	~2211 MiB
BGD-DiMP^{fm}	0.574	0.673	0.574		
BGD-DiMP^{mm}	0.576	0.675	0.575		
BGD-DiMP^{sm}	0.558	0.659	0.561		
RLS-DiMP	0.578	0.676	0.577	~40 *fps*	~2339 MiB

Prec., Norm. Prec. and Succ. Denote Precision Score at 20 Pixels, Normalized Precision Score and AUC of Success Plot Respectively. The Average Over 20 Runs is Reported for These DiMP-Style Trackers Except That We Report the Publicly Available 5-Run Results (Indicated by †) for DiMP.

on the long-term *LaSOT-test* benchmark in Table 4 and Fig. 3. *LaSOT* has much longer video sequences with average duration of 84 seconds (2500 frames in average) and each sequence of it comprises at least 1000 frames, which is dedicated to the long-term tracking principle. It is split into training and testing subsets, and the testing subset consists of 280 sequences with $690K$ frames. It also uses the same evaluation protocol to *OTB-2015*. However, due to the fact that the precision metric is sensitive to the resolution of the images and the size of the bounding boxes, a metric of normalized precision [30] over the size of the ground-truth bounding box can be exploited and the trackers can be then ranked using the AUC for normalized precision between 0 and 0.5.

As claimed in Section 6.1, we add the baselines using the EMA strategy for saving historical information in addition to BGD-DiMP and DiMP*. Due to the reason that our memory retention is applied to improve the vanilla BGD in BGD-DiMP and BGD-DiMP performs better than the original DiMP on *LaSOT-test*, we hence apply the EMA strategy on the baseline BGD-DiMP to construct new baselines. In specific, let α denote the EMA coefficient, and then the approximate optimum value $\widehat{\mathbf{W}}_n^{appr}$ after the vanilla BGD-based optimization at each update stage of BGD-DiMP can be further updated as follows

$$\widehat{\mathbf{W}}_n^{appr} \longleftarrow (1 - \alpha)\widehat{\mathbf{W}}_{n-1}^{appr} + \alpha\widehat{\mathbf{W}}_n^{appr} ,$$

where $\alpha \in [0, 1]$ and represents the degree of weighting decrease. A higher α discounts older parameters faster.

The commonly used slow-moving average strategy as mentioned in Sec. 6.1 always sets α to be less than 0.05 so that the online tracking model's parameters or the network representation in self-supervised learning evolve smoothly. However, applying this kind of EMA strategy on BGD-DiMP makes it prone to lower adaptive capacity while saving historical information, leading to decrease of performance on *LaSOT-test*. It is also impractical to do an exhaustive grid search to obtain the optimum value of α. So we roughly set three baselines as follows: BGD-DiMP^{sm} for discounting older parameters in a slow-moving way with α set to 0.01, BGD-DiMP^{mm} in a moderate-moving way with α set to 0.5, and BGD-DiMP^{fm} in a fast-moving way with α set to 0.99. This practice can also be observed in LADCF [49] that the EMA coefficient is set to 0.95 and 0.13 for hand-crafted and deep LADCF respectively. It is noteworthy that the baseline BGD-DiMP can be seen as a special case of applying the EMA strategy by setting α to 1.0.
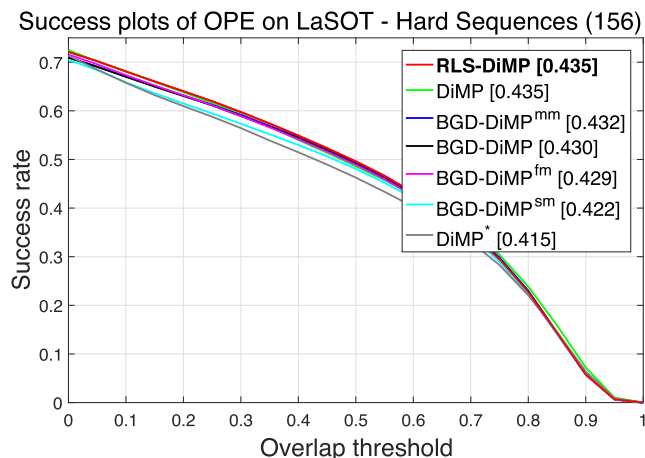
Fig. 3. Success plots for the hard sequences of the *LaSOT-test* dataset. We report the average over 20 runs for our RLS-DiMP and the re-implemented baselines, except that the publicly available 5-run results are used for DiMP. The legends show the AUC scores. Best viewed in color.

It is clearly and statistically shown in Table 4 that our online learning approach can also unveil the power of the state-of-the-art DiMP tracker without incurring too much extra computational and memory cost. As *LaSOT-test* is dedicated to the long-term tracking principle, increasing the SD optimizer recursion number from two to five in DiMP* may aggravate the overfitting issue caused by fast converging to a *local* point with respect to the fixed-sized sample set, and harm the overall performance on *LaSOT-test*. By experimentally setting an appropriate small learning rate for BGD-DiMP, we can reduce overfitting and achieve superior results compared to the original DiMP tracker. Thanks to the retention of historical information by exploiting the EMA strategy, the baseline BGD-DiMP^{mm} with the coefficient α roughly set to 0.5 can further enhance this superiority. We believe we can use some art to choose an optimum value of α for this EMA strategy to make a comprise between adapting quickly and maintaining robustness, and thus improve BGD-DiMP by a larger margin. However, our proposed online learning approach has been demonstrated theoretically to be able to retain the memory in a more elegant way, and thus reduce the risk of overfitting as well as maintain adaptive capacity. That means our RLS-DiMP can improve the performance on *LaSOT-test* without requiring fine-grained searching for hyper-parameters by hand.

Since some sequences in *LaSOT-test* are less challenging for the DiMP-style trackers to achieve good performance, we selected some difficult ones for an in-depth analysis as shown in Fig. 3, which also makes the abovementioned phenomena more clearly understood. As each sequence was evaluated with several runs by each entry in Table 4, we can define the sequence for which the AUC scores of different runs by our RLS-DiMP are all above 0.75 as easy sequence and remove it from our evaluation to achieve the success plots on the remaining hard sequences of *LaSOT-test*, leading to 156 hard sequences in total. From Fig. 3 we can see that, our RLS-DiMP improves the baseline BGD-DiMP with an absolute AUC gain of 0.5% on these hard sequences, which is computed based on 20 runs of the test sequences due to the stochastic nature of DiMP-style trackers. By comparing the performance gains between Table 4 and Fig. 3,

we can also find that introducing memory retention mainly helps to improve the tracking performance on the more challenging sequences, which is preferable as we aim to handle the challenging factors in tracking. It is noteworthy that only concentrating on saving historical information without maintaining adaptive capacity also harm the overall performance on *LaSOT-test*.

6.3.2 Comparison With Others

We perform the state-of-the-art comparison on *TrackingNet-test*, *GOT10k-test*, *LaSOT-test*, *OxUvA-dev*, *TLP* and *VOT2018/2019* by comparing our proposed RLS-DiMP to some short-term trackers, i.e., SiamFC [2], MDNet [23], ECO [48], DaSiamRPN [7], ATOM [25], GradNet [13], SiamMask [87], UpdateNet-DaSiamRPN [14], SiamRPN++ [9], SiamFC++-GoogLeNet [15], D3S [88], FCOS-MAML [17], SiamBAN [56], ROAM++ [21], PrDiMP [16], and long-term trackers, i.e., SPLT [89], GlobalTrack [20], Siam R-CNN [90], LTMU [19]. Most of these competitors are trained additionally relying on the *MS-COCO* dataset or recently proposed large-scale tracking datasets. Results for the re-implemented baselines are also included.

As most researchers have designed tracking methods tailored to the short-term scenario, which is poorly representative of practitioners' needs, *TLP* and *OxUvA* are recently proposed aiming to address this disparity. The former compiles 50 HD videos from real world scenarios, encompassing a duration of over 400 minutes (676K frames), while the latter comprises 366 sequences spanning 14 hours of video. *OxUvA* is also split into *dev* and *test* sets of 200 and 166 tracks respectively, where the trackers can be evaluated through an online server. What's more, *OxUvA* contains an average of 2.2 absent labels per track and at least one labelled disappearance in 52% of the tracks, which enables to evaluate trackers in terms of *TPR* (fraction of present objects that are reported present and correctly located) and *TNR* (fraction of absent objects that are reported absent). A problem with *OxUvA* is that it does not provide dense annotations in consecutive frames, i.e., each video in *OxUvA* is annotated every 30 frames, ignoring rich temporal context between consecutive frames for developing a tracker. So the aforementioned long-term *LaSOT* benchmark with high-quality dense annotations is proposed to bridge this gap, leading to the largest high-quality dense tracking benchmark which also provides a training subset with 2.8 million boxes for developing a tracker.

Before *LaSOT*, *TrackingNet* is the first large-scale tracking benchmark by simultaneously providing the training subset for developing a tracker and the test subset for evaluating different trackers. An online server is also provided for this evaluation. However, *TrackingNet* is manually labeled at 1 *fps* while its all other annotations are automatically generated using correlation filter-based tracking. The most recently proposed *GOT10k* benchmark offers a much wider coverage of object classes and first introduces the one-shot protocol to avoid evaluation bias towards seen familiar object classes, i.e., the training and test classes are zero-overlapped, which can be used to evaluate the generalization of different trackers. This is very different from *LaSOT* and *TrackingNet*. Besides *SR*, *GOT10k* also exploits *AO* to denote

TABLE 5
Comparison of Our RLS-DiMP With its Corresponding Baselines and Some State-of-the-Art Competing Trackers
on *TrackingNet-test*, *GOT10k-test*, *LaSOT-test*, *OxUvA-dev*, *TLP*; the Results of Our RLS-DiMP
and its Corresponding Baselines are Reported Based on 20 Runs on Each Sequence

Trackers	Year	*TrackingNet-test* $(511, 226K)$		*GOT10k-test* $(420, 56K)$			*LaSOT-test* $(280, 685K)$		*OxUvA-dev* $(200, 547K)$		*TLP* $(50, 676K)$	
		Succ.	*Prec.*	*AO*	$SR_{0.5}$	$SR_{0.75}$	*Succ.*	*Prec.*	*TPR*	*TNR*	*Succ.*	*Prec.*
Short-term Trackers												
SiamFC	2016	0.571	0.663	0.348	0.353	0.098	0.336	0.339	0.391	0.0	0.237	0.280
MDNet	2016	0.606	0.565	0.299	0.303	0.099	0.397	0.373	0.472	0.0	0.372	0.381
ECO	2017	0.554	0.492	0.316	0.309	0.111	0.324	0.301	-	-	0.205	0.211
ATOM	2019	0.703	0.648	0.556	0.634	0.402	0.515	0.505	-	-	-	-
UpdateNet-DaSiamRPN	2019	0.677	0.625	-	-	-	0.475	-	-	-	-	-
SiamRPN++	2019	0.733	0.694	0.517	0.616	0.325	0.496	0.491	-	-	-	-
SiamFC++-GoogLeNet	2020	0.754	0.705	0.595	0.695	0.479	0.544	0.547	-	-	-	-
D3S	2020	0.728	0.664	0.597	0.676	0.462	-	-	-	-	-	-
FCOS-MAML	2020	0.757	0.725	-	-	-	0.523	0.531	-	-	-	-
SiamBAN	2020	-	-	-	-	-	0.514	0.521	-	-	-	-
ROAM++	2020	0.670	0.623	0.465	0.532	0.236	0.447	0.445	-	-	-	-
PrDiMP	2020	0.758	0.704	0.634	0.738	0.543	0.598	0.608	-	-	-	-
Long-term Trackers												
SPLT	2019	-	-	-	-	-	0.426	0.396	0.498	0.776	0.416	0.403
GlobalTrack	2020	0.704	0.656	-	-	-	0.521	0.527	0.574	0.633	0.520	0.556
Siam R-CNN	2020	0.812	0.800	0.649	0.728	0.597	0.648	-	0.701	0.745	-	-
LTMU	2020	-	-	-	-	-	0.572	0.572	0.749	0.754	0.571	0.608
Our Implementations												
DiMP	2019	0.740	0.689	0.609	0.709	0.491	0.569†	0.567†	0.727	0.0	0.502	0.513
DiMP*	2019	0.742	0.690	0.608	0.709	0.490	0.557	0.556	0.714	0.0	0.497	0.507
BGD-DiMP	Ours	0.741	0.689	0.606	0.713	0.495	0.574	0.575	0.731	0.0	0.508	0.506
RLS-DiMP	Ours	0.740	0.688	0.611	0.713	0.492	0.577	0.578	0.735	0.0	0.519	0.518

Prec. and Succ. Denote Precision Score at 20 Pixels and AUC of Success Plot Respectively. (·, ·) Denotes the Number of Video Sequences and Frames for Evaluation in the Corresponding Dataset.
† *We report the publicly available 5-run results for DiMP.*

the average of overlaps between all ground-truth and estimated bounding boxes, which is proved to be equivalent to the *AUC* metric.

We first show the comparison on these large-scale tracking benchmarks in Table 5. Since *GOT10k* requires the trackers to be trained on its training set with no extra training data used and the DiMP tracking model following this protocol is not public available, we thus re-trained the DiMP tracking model for the evaluation on *GOT10k*. Our implementation results of RLS-DiMP and its corresponding baselines are reported based on 20 runs on each sequence. It clearly shows that our online learning approach can effectively unveil the power of DiMP and surpass the baseline BGD-DiMP by achieving better performance on all the long-term *LaSOT-test*, *OxUvA-dev* and *TLP* datasets. This is desirable as our framework is designed for memory retention and overfitting alleviation. What's more, the original DiMP tracker based on the SD optimizer performs worse on all these long-term datasets, especially with a significant performance decrease for DiMP*. This is consistent with the analysis in Sec. 6.3.1 that DiMP* may aggravate the overfitting issue and harm the overall performance in long-term tracking.

However, short-term tracking may prefer overfitting more than our memory retention, which makes DiMP* perform best on *TrackingNet-test* in our implementations. This phenomenon can also be attributed to the reason that the object classes between training and test sets in *TrackingNet* are fully overlapped with close distribution, and the DiMP

tracking model trained with *TrackingNet-train* used may lead to biased evaluation results towards familiar objects on *TrackingNet-test*. This is further validated by the evaluation on *GOT10k-test*, which shows that our online learning approach can make the DiMP tracking model generalize to the unseen object classes better than the other SD and vanilla BGD optimization methods in the one-shot tracker evaluation protocol.

Table 5 also shows that the novel ROAM++ [21] and UpdateNet-DaSiamRPN [14] trackers with corresponding LSTM-based and ConvNet model-based meta-learners exhaustively offline trained achieve substantially inferior results compared to our RLS-DiMP because of their exploited modest baselines. Our approach, however, can effectively unveil the power of the top-performing baseline tracker DiMP without incurring too much extra computational cost. In spite of the dominant performance on *TrackingNet-test*, *GOT10k-test* and *LaSOT-test*, Siam R-CNN [90] still generalizes not well on *OxUvA-dev* compared with LTMU [19] due to the missing of online learning.

We also report the comparison results on *VOT2018/2019* short-term challenges in Table 6. Since the *VOT2017* dataset had not saturated by the time the *VOT2018*'s challenges were held, the dataset was used unchanged in the *VOT2018* short-term challenge. In the *VOT2019* short-term challenge, it had been decided to refresh the public dataset by replacing 12 least difficult sequences with more challenging ones. As can be seen in Table 6, our RLS-DiMP surpasses all its corresponding baselines on both *VOT2018* and *VOT2019* by

TABLE 6
Comparison of Our RLS-DiMP With its Corresponding Baselines and Some State-of-the-Art Competing Trackers on *VOT2018/2019*; the Results are Reported as EAO, A, R_S ($S = 100$) and Real-Time EAO ($rtEAO$)

	VOT	GradNet	DaSiamRPN	ROAM++	SiamMask	UpdateNet (-DaSiamRPN)	MAML (-FCOS)	SiamRPN++	SiamFC++ (-GoogLeNet)	BGD-DiMP	DiMP	DiMP*	**RLS-DiMP**	PrDiMP
2018	$EAO \uparrow$	0.247	0.326	0.380	0.384	0.393	0.392	0.415	0.426	0.415	0.423	0.425	0.443	0.442
	$A \uparrow$	0.507	0.570	0.543	0.609	-	0.635	0.600	0.587	0.598	0.602	0.604	0.602	0.618
	$-\ln R_S \downarrow$	0.375	0.337	0.195	0.281	-	0.220	0.234	0.183	0.163	0.170	0.164	0.138	0.165
	$rtEAO \uparrow$	-	-	-	0.384	-	-	0.415	-	0.393	0.418	0.450	0.422	-
2019	$EAO \uparrow$	-	-	0.281	0.287	-	0.295	0.285	-	0.324	0.357	0.360	0.371	-
	$A \uparrow$	-	-	0.561	0.594	-	0.637	0.599	-	0.599	0.588	0.583	0.594	-
	$-\ln R_S \downarrow$	-	-	0.438	0.461	-	0.421	0.482	-	0.323	0.284	0.280	0.265	-
	$rtEAO \uparrow$	-	-	0.110	0.287	-	-	0.285	-	0.328	0.353	0.340	0.354	-

For All These Metrics Except $rtEAO$, the Stochastic Trackers are Run 15 Times on Each Sequence to Reduce the Variance of Their Results. Best Viewed in Color.

significantly reducing the number of tracking failures thanks to the proposed memory retention mechanism.

6.4 Complexity Analyses

In Tables 1 and 4, we also show some detailed comparisons regarding the efficiency of our proposed approach by reporting the running speed and GPU memory consumption for both our proposed methods and the re-implemented baselines. It can be seen that our approach enables us to unveil the power of RT-MDNet and DiMP trackers without incurring too much extra computational cost.

From Section 3.2 we can find that there are some matrix multiplication operations in our approach, which leads to the runtime of $\mathcal{O}(p^2)$ with respect to the dimension p of the input $\mathbf{x}$. However, all these operations are executed by the GPU along with the other neural network operations so that the final tracking speed is only slightly affected. In order to achieve memory retention, our approach requires the trackers to maintain the matrix $\mathbf{P}_n$ in the whole tracking phase and also update it with above matrix multiplication operations, which results in the $\mathcal{O}(p^2)$ memory requirement on the GPU. This factor has little impact on the GPU memory consumption for our RLS-RTMDNet while affecting RLS-DiMP too much due to the $img2col$ operation and its subsequent high-dimensional input for online learning ($p = 4 \times 4 \times 512$). We thus use the half-float precision for maintaining and updating $\mathbf{P}_n$, which reduces the RLS's GPU memory usage significantly. Finally, we can use some code optimization techniques to further improve the efficiency, though it is outside the scope of this work.

7 CONCLUSION

We present a recursive least-squares estimator-aided network online learning approach that allows in-built memory retention mechanism for improving few-shot online adaptation in tracking. We apply it to two networks in the online learning families for tracking: the MLPs-based as in RT-MDNet and the ConvNets-based as in DiMP. The experiments demonstrate its efficiency in reducing the risk of overfitting through memory retention while maintaining adaptive capacity. This can be attributed to the reason that this recursive learning approach enables the optimization to approximately converge to a *global* point with respect to all the historical training samples ever seen including the discarded old data. An interesting direction for future work is to integrate it into the meta-learning-based offline training framework to gain further improvements.

REFERENCES

[1] J. Gao, W. Hu, and Y. Lu, "Recursive least-squares estimator-aided online learning for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7384–7393.

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. Eur. Conf. Comput. Vis. Workshop*, 2016, pp. 850–865.

[3] L. Bertinetto, J. F. Henriques, J. Valmadre, P. H. Torr, and A. Vedaldi, "Learning feed-forward one-shot learners," in *Proc. 30th Conf. Adv. Neural Info. Process. Syst.*, 2016, pp. 523–531.

[4] T. Yang and A. B. Chan, "Recurrent filter learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, 2017, pp. 2010–2019.

[5] E. Park and A. C. Berg, "Meta-tracker: Fast and robust online adaptation for visual object trackers," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 587–604.

[6] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8971–8980.

[7] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 103–119.

[8] T. Yang and A. B. Chan, "Learning dynamic memory networks for object tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 153–169.

[9] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "SiamRPN++: Evolution of siamese visual tracking with very deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4277–4286.

[10] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Learning discriminative model prediction for tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6181–6190.

[11] L. Huang, X. Zhao, and K. Huang, "Bridging the gap between detection and tracking: A unified approach," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3998–4008.

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

[13] P. Li, B. Chen, W. Ouyang, D. Wang, X. Yang, and H. Lu, "GradNet: Gradient-guided network for visual object tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6161–6170.

[14] L. Zhang, A. Gonzalez-Garcia, J. V. D. Weijer, M. Danelljan, and F. S. Khan, "Learning the model update for siamese trackers," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 4009–4018.

[15] Y. Xu, Z. Wang, Z. Li, Y. Yuan, and G. Yu, "SiamFC++: Towards robust and accurate visual tracking with target estimation guidelines," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 12549–12556.

[16] M. Danelljan, L. V. Gool, and R. Timofte, "Probabilistic regression for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7181–7190.

[17] G. Wang, C. Luo, X. Sun, Z. Xiong, and W. Zeng, "Tracking by instance detection: A meta-learning approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6287–6296.

[18] I. Jung, K. You, H. Noh, M. Cho, and B. Han, "Real-time object tracking via meta-learning: Efficient model adaptation and one-shot channel pruning," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 11205–11212.

[19] K. Dai, Y. Zhang, D. Wang, J. Li, H. Lu, and X. Yang, "High-performance long-term tracking with meta-updater," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6297–6306.

[20] L. Huang, X. Zhao, and K. Huang, "GlobalTrack: A simple and strong baseline for long-term tracking," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 11037–11044.

[21] T. Yang, P. Xu, R. Hu, H. Chai, and A. B. Chan, "ROAM: Recurrently optimizing tracking model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6717–6726.

[22] T. Yang and A. B. Chan, "Visual tracking via dynamic memory networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 360–374, Jan. 2021.

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

[24] I. Jung, J. Son, M. Baek, and B. Han, "Real-time MDNet," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 89–104.

[25] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ATOM: Accurate tracking by overlap maximization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4655–4664.

[26] L. Bertinetto, "Learning (to learn) from few examples," Ph.D. dissertation, Dept Eng. Sci., Univ. Oxford, Oxford, U.K., 2019.

[27] O. Russakovsky *et al.*, "Image net large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[28] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[29] L. Huang, X. Zhao, and K. Huang, "GOT-10k: A large high-diversity benchmark for generic object tracking in the wild," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 5, pp. 1562–1577, May 2021.

[30] M. MÄller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem, "TrackingNet: A large-scale dataset and benchmark for object tracking in the wild," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 310–327.

[31] H. Fan *et al.*, "LaSOT: A high-quality benchmark for large-scale single object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5369–5378.

[32] S. S. Haykin, *Adaptive Filter Theory*, London, U.K.: Pearson, 2014.

[33] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[34] X. He and H. Jaeger, "Overcoming catastrophic interference using conceptor-aided backpropagation," in *Proc. Int. Conf. Learn. Representations*, 2018.

[35] G. Zeng, Y. Chen, B. Cui, and S. Yu, "Continual learning of context-dependent processing in neural networks," *Nat. Mach. Intell.*, vol. 1, pp. 364–372, Aug. 2019.

[36] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review, *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1802.07569

[37] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, no. 1–3, pp. 125–141, May 2008.

[38] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1619–1632, Aug. 2011.

[39] S. Hare, A. Saffari, and P. H. Torr, "Struck: Structured output tracking with kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2096–2109, Oct. 2016.

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

[41] J. Gao, Q. Wang, J. Xing, H. Ling, W. Hu, and S. Maybank, "Tracking-by-fusion via Gaussian process regression extended to transfer learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 939–955, Apr. 2020.

[42] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.

[43] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Discriminative scale space tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 8, pp. 1561–1575, Aug. 2017.

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

[45] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1144–1152.

[46] Z. Huang, C. Fu, Y. Li, F. Lin, and P. Lu, "Learning aberrance repressed correlation filters for real-time UAV tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 2891–2900.

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

[48] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6631–6639.

[49] T. Xu, Z.-H. Fen, X.-J. Wu, and J. Kittler, "Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual object tracking," *IEEE Trans. Image Process.*, vol. 28, no. 11, pp. 5596–5609, Nov. 2019.

[50] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang, "Learning spatial-temporal regularized correlation filters for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4904–4913.

[51] T. Xu, Z.-H. Fen, X.-J. Wu, and J. Kittler, "Joint group feature selection and discriminative filter learning for robust visual object tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7949–7959.

[52] Y. Li, C. Fu, F. Ding, Z. Huang, and G. Lu, "AutoTrack: Towards high-performance visual tracking for UAV with automatic spatio-temporal regularization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11920–11929.

[53] T. Yang, B. Li, and M. Q.-H. Meng, "Robust object tracking with reacquisition ability using online learned detector," *IEEE Trans. Cybern.*, vol. 44, no. 11, pp. 2134–2142, Nov. 2014.

[54] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.

[55] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[56] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji, "Siamese box adaptive network for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6667–6676.

[57] S. Ruder, An overview of gradient descent optimization algorithms, *CoRR*, 2016. [Online]. Available: https://arxiv.org/abs/1609.04747

[58] J. R. Shewchuk, "An introduction to the Conjugate Gradient method without the agonizing pain," Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-94-125, Aug. 1994.

[59] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, Jan. 1999.

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

[61] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. 32nd Int. Conf. Mach. Learn. Deep Learn. Workshop*, 2015.

[62] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[63] A. Antoniou, H. Edwards, and A. Storkey, "How to train your MAML," in *Proc. 7th Int. Conf. Learn. Representations*, 2019, pp. 21–49.

[64] M. Kristan *et al.*, "The sixth visual object tracking VOT2018 challenge results," in *Proc. Eur. Conf. Comput. Vis. Workshop*, 2018, pp. 3–53.

[65] M. Kristan *et al.*, "The seventh visual object tracking VOT2019 challenge results," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, 2019, pp. 2206–2241.

[66] J. Valmadre *et al.*, "Long-term tracking in the wild: A benchmark," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 692–707.

[67] A. Moudgil and V. Gandhi, "Long-term visual object tracking benchmark," in *Proc. Asi. Conf. Comput. Vis.*, 2019, pp. 629–645.

[68] G. V. Moustakides, "Study of the transient phase of the forgetting factor RLS," *IEEE Trans. Signal Process.*, vol. 45, no. 10, pp. 2468–2476, Oct. 1997.

[69] W. H. Hager, "Updating the inverse of a matrix," *SIAM Rev.*, vol. 31, no. 2, pp. 221–239, Jun. 1989.

[70] G. Zhang, C. Wang, B. Xu, and R. Grosse, "Three mechanisms of weight decay regularization," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.

[71] K. Saitoh, *Deep Learning From the Basics*, Birmingham, U.K.: Packt, 2021.

[72] M. MÃller, N. Smith, and B. Ghanem, "A benchmark and simulator for UAV tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 445–461.

[73] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 2544–2550.

[74] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary learners for real-time tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1401–1409.

[75] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, "End-to-end representation learning for Correlation Filter based tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2805–2813.

[76] Y. Li, C. Fu, F. Ding, Z. Huang, and J. Pan, "Augmented memory for correlation filters in real-time UAV tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot Syst.*, 2020, pp. 1559–1566.

[77] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, and H. Li, "Unsupervised deep tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1308–1317.

[78] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proc. Int. Conf. Neural Info. Process. Syst.*, 2017, pp. 1195–1204.

[79] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9729–9738.

[80] J.-B. Grill *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," in *Proc. Int. Conf. Neural Info. Process. Syst.*, 2020, pp. 9729–9738.

[81] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[82] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[83] M. Kristan *et al.*, "The visual object tracking VOT2016 challenge results," in *Proc. Eur. Conf. Comput. Vis. Workshop*, 2016, pp. 777–823.

[84] M. Kristan *et al.*, "The visual object tracking VOT2017 challenge results," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, 2017, pp. 1949–1972.

[85] A. Lukežič, T. Vojíř, L. Č. Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter tracker with channel and spatial reliability," *Int. J. Comput. Vis.*, vol. 126, no. 7, pp. 671–688, Jul. 2018.

[86] X. Li, C. Ma, B. Wu, Z. He, and M.-H. Yang, "Target-aware deep tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1369–1378.

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

[88] A. Lukežič, J. Matas, and M. Kristan, "D3S – A discriminative single shot segmentation tracker," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7133–7142.

[89] B. Yan, H. Zhao, D. Wang, H. Lu, and X. Yang, "Skimming-Perusal, Tracking: A framework for real-time and robust long-term tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 2385–2393.

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

Jin Gao received the BS degree from the Beihang University, Beijing, China, in 2010, and the PhD degree from the University of Chinese Academy of Sciences (UCAS), in 2015. Now he is an associate professor with the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA). His research interests include object tracking, autonomous vehicles, and service robots.
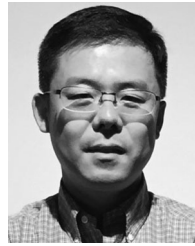
Yan Lu received the PhD degree in computer science from the Harbin Institute of Technology, China, in 2004. He joined with Microsoft Research Asia in 2004, where he is now the partner research manager with Media Computing Group, leading the development of core technologies in the fields of real-time communication, computer vision, video analytics, audio enhancement, virtualization, and mobile-cloud computing. From 2001 to 2004, he was the team lead of video coding group with the JDL Lab, Institute of Computing Technology, China. From 1999 to 2000, he was with the City University of Hong Kong as a research assistant. His research interests include image and video coding, computer vision, audio and speech, multimedia system, networking, and remote computing. He has published more than 100 + papers and holds more than 30+ granted U.S. patents in the field of multimedia and computer vision.

Xiaojuan Qi received the BEng degree in electronic science and technology from Shanghai Jiao Tong University (SJTU), in 2014, and the PhD degree in computer science and engineering from the Chinese University of Hong Kong in 2018. She is currently an assistant professor with the University of Hong Kong.

Yutong Kou received the BS degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 2021. He is currently working toward the master's degree with the Institute of Automation, Chinese Academy of Sciences (CASIA), University of Chinese Academy of Science (UCAS). His research interests include both theory and applications of object tracking.

Bing Li received the PhD degree from the Department of Computer Science and Engineering, Beijing Jiaotong University, Beijing, China, in 2009. From 2009 to 2011, he worked as a postdoctoral research fellow with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing. He is currently a professor with CASIA. His current research interests include computer vision, color constancy, visual saliency detection, multi-instance learning, and data mining.

Liang Li received the PhD degree from the Fourth Military Medical University (FMMU), in 2017. Now he is a visiting scholar at Institute of Automation, Chinese Academy of Sciences, and supported by the Brain Research Center, Beijing Institute of Basic Medical Sciences. His research interests include biologically inspired computing and computer vision.

Shan Yu received the BS and PhD degrees in biology from the University of Science and Technology of China, Hefei, China, in 2000 and 2005, respectively. From 2005 to 2014, he conducted postdoctoral research with the Max-Planck Institute of Brain Research, Germany (2005–2008), and the National Institute of Mental Health, USA (2008–2014). In 2014, he was with the Institute of Automation, Chinese Academy of Sciences (CASIA), as a recipient of the "One Hundred Talents" program of CAS. He is a professor with the Brainnetome Center and National Laboratory of Pattern Recognition (NLPR), CASIA. Since 2018, he has been the deputy director with the NLPR. He has authored and coauthored more than 30 peer-reviewed papers in neuroscience and other interdisciplinary fields with leading international journals such as the *Nature Machine Intelligence*, the *Journal of Neuroscience, eLife*, etc. His current research interests include neuronal information processing, brain-inspired computing, and artificial intelligence.

Weiming Hu (Senior Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering, Zhejiang University, Zhejiang, China. Since 1998, he has been with the Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, where he is currently a professor. He has published more than 200 papers on peer reviewed international conferences and journals. His current research interests include visual motion analysis and recognition of harmful Internet multimedia.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.