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

Jin Gao[1,2] Weiming Hu[1,2] Yan Lu[3]
[1]NLPR, Institute of Automation, CAS [2]University of Chinese Academy of Sciences [3]Microsoft Research
{jin.gao, wmhu}@nlpr.ia.ac.cn yanlu@microsoft.com

Abstract

Online learning is crucial to robust visual object tracking as it can provide high discrimination power in the presence of background distractors. However, there are two contradictory factors affecting its successful deployment on the real visual tracking platform: the discrimination issue due to the challenges in vanilla gradient descent, which does not guarantee good convergence; the robustness issue due to over-fitting resulting from excessive update with limited memory size (the oldest samples are discarded).

Despite many dedicated techniques proposed to somehow treat those issues, in this paper we take a new way to strike a compromise between them based on the recursive least-squares estimation (LSE) algorithm. After connecting each fully-connected layer with LSE separately via normal equations, we further propose an improved mini-batch stochastic gradient descent algorithm for fully-connected network learning with memory retention in a recursive fashion. This characteristic can spontaneously reduce the risk of over-fitting resulting from catastrophic forgetting in excessive online learning. Meanwhile, it can effectively improve convergence though the cost function is computed over all the training samples that the algorithm has ever seen. We realize this recursive LSE-aided online learning technique in the state-of-the-art RT-MDNet tracker, and the consistent improvements on four challenging benchmarks prove its efficiency without additional offline training and too much tedious work on parameter adjusting.

1. Introduction

Online learning basically aims at updating the predictor for future data at each step when a continuous stream of data becomes available in a sequential order. Sometimes it needs dynamically adapting to new patterns in the stream when the data itself changes from time to time. Since it is just right for visual tracking, a number of interesting pioneering works (*e.g*. [31, 1, 12, 41, 6, 26, 17, 7, 4, 5]) that use variants of fundamental online learning methods for visual tracking are published during the past several years.

In contrast to the typical supervised learning tasks training static models with fixed volumes of data, *e.g*. image classification [34] and object detection [14], the online updated dynamic tracking models need to have 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, occlusion and background clutter). This ability is known as robustness. Nevertheless, a good tracker also need to strike a compromise between its robustness ability and instance-level discrimination power in case of background distractors, because these two requirements are sometimes contradictory and hard to be fulfilled at the same time [36].

By design, some earlier online trackers [6, 26, 17] face challenges in meeting those two demands. On one hand, since the standard gradient descent or its momentum-based stochastic twin [30] is applied in online updating the target classification model, it may cause a discrimination issue due to the convergence challenges stemming from the learning rate selection, model initialization, non-convex cost function, and so on [32]. Another key factor is the limited optimization iteration number allowed in the update stage for the sake of tracking speed. On the other hand, despite the benefits of improved instance-level discrimination power with frequent update, the excessive update with limited memory size (the oldest samples are discarded) may result in over-fitting to recent training samples, deteriorating the robustness.

Recently, there have been many dedicated techniques proposed to treat the above two issues more or less. For instance, the Conjugate Gradient method [33] is widely applied to efficiently solve normal equations derived from discriminative correlation filter (DCF) learning function [7, 4], or more generally the positive definite quadratic function [5], leading to improved convergence while online learning. Some other works [28, 21] concentrate on the offline training stage and exploit some one-shot/meta learning methods in this stage to provide a robust model initialization to account for the convergence challenges in online tracking stage. To relieve suffering from over-fitting to recent training samples, or even corrupted samples, some track-

2575-7075/20/$31.00 ©2020 IEEE
DOI 10.1109/CVPR42600.2020.00741

ers widely apply the moderately infrequent update [7, 4, 5], passive update [22], or long short-term complementary update [26, 17] settings in the update stage. In this paper, however, we take a new way to manage to handle both of them.

1.1. Motivation

Following [29], the parameters to be learned in an online learning algorithm $\mathcal{A}$ can be defined recursively as

$$\mathbf{W}_{n+1} = \mathcal{A}\left(\mathbf{W}_n, o_n\right) , \tag{1}$$

or a weaker notion with the memory of the past retained

$$\mathbf{W}_{n+1} = \mathcal{A}\left(\mathbf{W}_n, \{o_t\}_{t=1}^{n}\right) , \tag{2}$$

where $\mathbf{W}_1$ is the initialized weight parameters, and the observation o_t arrives sequentially in a continuous stream of data. It is well-known that the single instance stochastic gradient descent (SGD) can be used for online learning in the paradigm of Eq. (1) and meanwhile computationally cheap, and the model can adapt to changes if we use some art in choosing a good learning rate. Since the historical memory is not retained, directly applying this technique to the visual tracking problem will potentially be faced with the problem of being prone to tracking drift due to over-fitting to some non-accurate samples collected in the current observation o_n. As the number of arriving observations increases, the straightforward way of incorporating the memory of the target in visual tracking following Eq. (2) will encounter the significant computational overhead, because fine-tuning the parameters over the whole increasingly larger sample set gives rise to more mini-batch SGD iterations to achieve good convergence, which prohibits high tracking speed.

The aforementioned trackers adopting the long short-term complementary update strategy aim to strike a balance between the above two paradigms by setting the memory with constant length, while much earlier aspects of the observations are forgotten. As in RT-MDNet [17], two types of memory with different length (long-term and short-term) are maintained, and switching back and forth between them in fine-tuning depends on the tracking confidence of the tracker itself. However, there is still some memory missing and it may be difficult to know a priori what the best hyper-parameter of the long-term memory length is. The question is whether we have a more elegant path to both retain the memory and improve convergence while online learning.

1.2. Contributions

In this work, we provide a recursive solution based on the system of normal equations in solving the linear least-squares estimation (LSE) problem [13]. As our first contribution, we provide some derivations to connect each fully-connected layer with LSE separately via normal equations, which is a prerequisite for the consequent derivation of recursive LSE-aided online learning. Our second contribution is an improved mini-batch SGD algorithm for fully-connected network learning in a recursive fashion from the recursive LSE-aided online learning derivation with memory retention. This characteristic can spontaneously reduce the risk of over-fitting resulting from catastrophic forgetting in excessive online learning. Meanwhile, it can effectively improve convergence [13] though the cost function is computed over all the training samples that the algorithm has ever seen. It is noteworthy that the derivations can be easily transplanted to the case of cross-entropy loss function. As our final contribution, we realize this proposed recursive LSE-aided online learning technique in the state-of-the-art RT-MDNet tracker, and the consistent improvements on four challenging benchmarks, *i.e*. OTB-2015 [38], VOT2016/2017 [20, 19] and UAV123 [25], prove its efficiency without additional offline training and too much tedious hand engineered work on parameter adjusting.

2. Related work

Online tracking. Online learning has been an important part of visual tracking for about ten years since the classic IVT tracker [31] was first proposed in 2008. The subsequent research has been concentrated on online robust classifier construction, including MIL [1], Struck [12], MEEM [41], TGPR [9], DCF-based approaches [16, 6, 4], and deep classifiers [26, 17, 5], just to name a few. Despite their success, these approaches suffer from over-fitting to recent training samples, which may deteriorate the robustness.

In addition to the earlier mentioned different update settings during online learning, many approaches also rely on constructing heavy target classification models to improve robustness, leading to significant computational overhead while optimizing the cost functions. For instance, the DCF learning function always incorporates a spatial regularization term and is jointly optimized over several historical training samples [6, 7, 4]. To provide a meaningful 2D-location of the object in the target classification score map, ATOM [5] exploits a 2-layer fully convolutional network for learning based on the positive definite quadratic function derived through quadratic Gauss-Newton approximation. In contrast, the MDNet-style trackers [26, 17] learn the target-background binary classification models from a 3-layer fully-connected network based on the cross-entropy loss function. To overcome the convergence challenges in them, the Conjugate Gradient method [33] is widely applied in [7, 4, 5] to efficiently solve normal equations derived from the cost functions. Meta-learning [8] is also introduced to optimize the initial deep networks for a representation that can quickly adapt to a particular target in a test sequence using a small number of gradient steps and a small amount of training samples during online update [28].

Deep trackers. Recently, with the astonishing improvements in deep learning and CNN, deep features have been widely exploited in visual tracking. For instance, many DCF-based [24, 10, 7, 4] or Siamese-style [3, 37, 21, 42,

5] trackers adopt the off-the-shelf classification/detection CNN models or their offline fine-tuned versions only in feature extraction and set them fixed while tracking for efficiency. Note that some Siamese-style trackers [3, 21, 42] based on one-shot learning even completely abandon online learning to emphasize more on robustness improvement. Although they are not bothered by convergence challenges anymore, the instance-level discrimination issue due to background distractors still exits [42]. In contrast, some online deep trackers [26, 17, 28, 5] focus on learning deep target classification models with online fine-tuning, and the different strategies are designed for efficient optimization without heavy computational overhead. As the focus has been directed towards more powerful deep trackers, we align our work with this research line to set up a different view on visual online tracking.

Continual learning. The aim of continual learning is to achieve artificial general intelligence, which requires the ability of learning consecutive tasks without forgetting how to perform previously trained tasks. It has recently proved valuable in numerous supervised learning and reinforcement learning-based applications along with deep learning success, leading to an emerging field [18, 15, 39, 27]. Since our proposed recursive LSE-aided online learning method also aims at retaining past memory while learning different tracking models sequentially over time, we share the same purpose of tackling catastrophic forgetting incurred by plain SGD. However, these previous works only focus on the offline training phase to demonstrate the effectiveness of their memory retention methods in the classic image classification [18, 15, 39] or Atari 2600 games [18]. Our departure from them is that we are the first to improve on online learning procedure by incorporating memory retention and demonstrate its success in visual tracking.

3. Recursive LSE-aided online learning

In [35, 34], the multi-layer perceptron (MLP) with several fully-connected 1-D layers is applied to the last stage of deep learning architectures. The features present in the final 2-D feature maps are concatenated into one long input vector to the following MLP. By organizing the network parameters in matrices, the calculations in the networks can be quickly performed using fast linear algebra routines.

3.1. Some preliminaries

In the derivation that follows, we will describe the MLP network without considering the bias in each layer (*i.e*. set the `bias` option to `False` or add the bias unit $+1$ corresponding to the intercept term to the input of each layer) for the sake of simplicity. Specifically, we denote the calculations in the $(l+1)$-th layer as

$$\mathbf{z}^{l+1} = \mathbf{W}^l \mathbf{u}^l \,, \quad \mathbf{u}^{l+1} = f\left(\mathbf{z}^{l+1}\right) \,, \tag{3}$$

where the L-layered network weight matrix $\mathbf{W}^l = [w_{jk}^l]$, $1 \le l < L$ has each element associated with the connection between unit k in layer l and unit j in layer $l+1$, *i.e*.

$$z^{l+1,j} = \sum_k w_{jk}^l \cdot u^{l,k} \,. \tag{4}$$

The activation function $f(\cdot)$ is applied to vectors in an element-wise fashion with its output cast as the input of next layer or the final output of the whole network $h_{\mathbf{W}}(\mathbf{x})$, where $\mathbf{x}$ is the input of the whole MLP network and we can also denote $\mathbf{u}^1 = \mathbf{x}$ and $h_{\mathbf{W}}(\mathbf{x}) = \mathbf{u}^L$.

Since SGD is a commonly used strategy for training the networks, we hereby consider the feedforward pass for only one training example $(\mathbf{x}_i, \mathbf{d}_i)$, and calculate the desired partial derivatives for gradient descent in the backpropagation pass based on its individual prediction error. For a multi-class problem with c classes, we can consider the squared-error cost function defined as

$$\mathscr{L}(\mathbf{W}) = \frac{1}{2}\left\|\mathbf{d}_i - h_{\mathbf{W}}(\mathbf{x}_i)\right\|^2 + \frac{\lambda}{2}\sum_{l=1}^{L-1}\left\|\mathbf{W}^l\right\|^2 \tag{5}$$

$$= \frac{1}{2}\sum_{k=1}^{c}\left(d_i^k - u_i^{L,k}\right)^2 + \frac{\lambda}{2}\sum_{l=1}^{L-1}\left\|\mathbf{W}^l\right\|^2 \,, \tag{6}$$

where the target label $\mathbf{d}_i$ organized as a "one-of-c" code has its k-th element d_i^k positive (*e.g*. +1) if the pattern $\mathbf{x}_i$ belongs to class k, the rest of its entries will be zero if the sigmoid activation function is applied, and $u_i^{L,k}$ is similarly the value of the k-th output layer unit in response to $\mathbf{x}_i$. The L_2 regularization term of layer $l+1$, controlled by the weight decay λ, encourages its weights to be small in magnitude to improve the generalization performance of the network [40].

Mini-batch SGD (MBSGD) is a compromise between batch gradient descent (BGD) and the single instance SGD. Suppose there are b examples in one batch used for one update iteration of the MBSGD process, it is thus the following mean value of the partial derivatives over all the b examples that is used for updating $\mathbf{W}^l$ of each layer:

$$\Delta\mathbf{W}^l = \left[\frac{1}{2b}\sum_{i=1}^{b}\nabla_{\mathbf{W}^l}\left\|\mathbf{d}_i - h_{\mathbf{W}}(\mathbf{x}_i)\right\|^2\right] + \lambda\mathbf{W}^l \,, \tag{7}$$

$$\mathbf{W}^l \longleftarrow \mathbf{W}^l - \eta\Delta\mathbf{W}^l \,, \tag{8}$$

where the gradient matrix in Eq. (7) can be derived using some basic operations for derivatives of traces in the second-order case as follows:

$$\begin{aligned}&\nabla_{\mathbf{W}^l}\left\|\mathbf{d}_i - h_{\mathbf{W}}(\mathbf{x}_i)\right\|^2 \\ =&\nabla_{\mathbf{W}^l}\,\mathrm{Tr}\left[\left(\mathbf{d}_i - h_{\mathbf{W}}(\mathbf{x}_i)\right)\left(\mathbf{d}_i - h_{\mathbf{W}}(\mathbf{x}_i)\right)^\top\right] \,.\end{aligned} \tag{9}$$

3.2. SGD-related normal equations for solving LSE

In the squared-error cost function case of SGD-based MLP network learning, the intuition behind the backpropagation algorithm using Eq. (8) is strongly related to the

system of normal equations for solving the LSE problem. Naturally, we may derive this system of equations in its own independent way by formulating the gradient matrix $\nabla_{\mathbf{W}^l}\mathscr{L}$ in terms of the weights of $\mathbf{W}^l$ and then solving for $\mathbf{W}^l$ for which $\nabla_{\mathbf{W}^l}\mathscr{L}$ is zero. To this end, we make some simplifications again in the following way the cost function of Eq. (5) is constructed.

On the one hand we simplify the derivation of normal equations by ignoring the activation function in $h_{\mathbf{W}}(\mathbf{x}_i)$ leading to its linear version

$$h_{\mathbf{W}}(\mathbf{x}_i) = \prod_{l=1}^{L-1} \mathbf{W}^{L-l}\mathbf{x}_i \tag{10}$$

based on the following recurrence relation:

$$\mathbf{u}_i^{l+1} = \mathbf{W}^l\mathbf{u}_i^l \,, \tag{11}$$

$$\mathbf{u}_i^1 = \mathbf{x}_i \,. \tag{12}$$

Thus, if layer $l+1$ is chosen to derive the system of normal equations with the input $\mathbf{u}_i^l$ and other layers' weights are treated as constant, the squared-error cost function of Eq. (5) will degrade to

$$\mathscr{L}(\mathbf{W}^l) = \underbrace{\frac{1}{2}\left\|\mathbf{d}_i - \prod_{s=1}^{L-l}\mathbf{W}^{L-s}\mathbf{u}_i^l\right\|^2}_{Q_1} + \underbrace{\frac{\lambda}{2}\sum_{s=l}^{L-1}\|\mathbf{W}^s\|^2}_{Q_2} \,. \tag{13}$$

From the L_2 norm's sub-multiplicativity, it holds that

$$\left\|\prod_{s=1}^{L-l}\mathbf{W}^{L-s}\right\| \le \left\|\prod_{s=1}^{L-l-1}\mathbf{W}^{L-s}\right\| \cdot \left\|\mathbf{W}^l\right\| \,. \tag{14}$$

So, on the other hand we can use a new regularization term with different weight decay to substitute ($\longleftarrow$) the original term of Q_2 in Eq. (13) to facilitate the derivation without sacrificing consistency of network generalization performance improvement

$$\frac{\lambda}{2}\sum_{s=l}^{L-1}\|\mathbf{W}^s\|^2 \longleftarrow \frac{\widetilde{\lambda}}{2}\left\|\prod_{s=1}^{L-l}\mathbf{W}^{L-s}\right\|^2 \,. \tag{15}$$

Afterwards we could derive the gradient matrix $\nabla_{\mathbf{W}^l}\mathscr{L}$ of Eq. (13) by considering its two components separately. For *sum of error squares*,

$$\nabla_{\mathbf{W}^l}Q_1 = \left(\prod_{s=1}^{L-l-1}\mathbf{W}^{L-s}\right)^\top\left(\prod_{s=1}^{L-l}\mathbf{W}^{L-s}\mathbf{u}_i^l - \mathbf{d}_i\right)\left(\mathbf{u}_i^l\right)^\top \,. \tag{16}$$

For *regularizing term*,

$$\nabla_{\mathbf{W}^l}Q_2 = \widetilde{\lambda}\left(\prod_{s=1}^{L-l-1}\mathbf{W}^{L-s}\right)^\top\left(\prod_{s=1}^{L-l-1}\mathbf{W}^{L-s}\right)\mathbf{W}^l \,. \tag{17}$$

Isolating the term $\mathbf{W}^l$ corresponding to $s = L-l$ from the rest of the product of the network weight matrices, *i.e.* $\mathbf{W}_r = \prod_{s=1}^{L-l-1}\mathbf{W}^{L-s}$, in Eq. (16) and solving for $\mathbf{W}^l$ for which $\nabla_{\mathbf{W}^l}\mathscr{L}$ is zero, we may write

$$\mathbf{W}^l\left(\mathbf{u}_i^l\left(\mathbf{u}_i^l\right)^\top + \widetilde{\lambda}\mathbf{I}\right) = \left(\mathbf{W}_r^\top\mathbf{W}_r\right)^{-1}\mathbf{W}_r^\top\mathbf{d}_i\left(\mathbf{u}_i^l\right)^\top \,. \tag{18}$$

Recall that in the backpropagation pass, it is the derivative of the error with respect to a layer's total input that is propagated backwards from higher layers to lower layers, using the following recurrence relation (if the activation function is ignored):

$$\boldsymbol{\delta}_i^l = \left(\mathbf{W}^l\right)^\top\boldsymbol{\delta}_i^{l+1} \,, \tag{19}$$

where each entry of the "error term" $\boldsymbol{\delta}_i^l$ measures how much the corresponding node is "responsible" for any errors in the output, and

$$\boldsymbol{\delta}_i^L = \mathbf{u}_i^L - \mathbf{d}_i \,. \tag{20}$$

Then the right side of Eq. (18) can be written as a more intuitive term using some substitution operations, *i.e.*

$$\begin{aligned}&\left(\mathbf{W}_r^\top\mathbf{W}_r\right)^{-1}\mathbf{W}_r^\top\left(\mathbf{u}_i^L - \boldsymbol{\delta}_i^L\right)\left(\mathbf{u}_i^l\right)^\top \\ &= \left(\mathbf{W}_r^\top\mathbf{W}_r\right)^{-1}\left(\mathbf{W}_r^\top\mathbf{W}_r\mathbf{u}_i^{l+1} - \boldsymbol{\delta}_i^{l+1}\right)\left(\mathbf{u}_i^l\right)^\top \\ &= \left(\mathbf{u}_i^{l+1} - \left(\mathbf{W}_r^\top\mathbf{W}_r\right)^{-1}\boldsymbol{\delta}_i^{l+1}\right)\left(\mathbf{u}_i^l\right)^\top \,.\end{aligned} \tag{21}$$

As it can be seen, this new term includes the desired output $\mathbf{y}_i^{l+1}$ for layer $l+1$, *i.e.*

$$\mathbf{y}_i^{l+1} = \mathbf{u}_i^{l+1} - \left(\mathbf{W}_r^\top\mathbf{W}_r\right)^{-1}\boldsymbol{\delta}_i^{l+1} \,, \tag{22}$$

where $\mathbf{u}_i^{l+1}$ is the actual state of the $(l+1)$-th layer's output, and the last term is the "normalized" error term. Finally, the system of normal equations for layer $l+1$ is written as:

$$\mathbf{W}^l = \mathbf{y}_i^{l+1}\left(\mathbf{u}_i^l\right)^\top\left(\mathbf{u}_i^l\left(\mathbf{u}_i^l\right)^\top + \widetilde{\lambda}\mathbf{I}\right)^{-1} \,. \tag{23}$$

The above derivation concludes that each layer of MLP can be represented by the system of normal equations for solving the linear LSE problem.

3.3. Recursive formulation for memory retention

Following [13], we can expand the normal equations in Eq. (23) to include inputs collected from all the historical observations with variable length n over time, and introduce an exponential weighting factor, or forgetting factor, for different arriving time. By doing so, we are able to write the normal equations as follows in a straightforward manner

$$\mathbf{W}_n = \mathbf{Z}_n\boldsymbol{\Phi}_n^{-1} \,, \tag{24}$$

where $\mathbf{Z}_n$ and $\boldsymbol{\Phi}_n$ are defined respectively by Eqs. (25) and (26), and the superscripts of l for indicating different layers in Eq. (23) are dropped to simplify the notation. The time-average cross-correlation matrix $\mathbf{Z}_n$ between the desired output and the input is shown by the formula

$$\mathbf{Z}_n = \sum_{i=1}^{n}\beta^{n-i}\mathbf{y}_i\mathbf{u}_i^\top \,, \tag{25}$$

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

$$\mathbf{\Phi}_n = \sum_{i=1}^{n} \beta^{n-i} \mathbf{u}_i \mathbf{u}_i^\top + \widetilde{\lambda} \beta^n \mathbf{I} \,, \tag{26}$$

where $0 < \beta \leq 1$ is to measure the memory of the algorithm, the regularizing term is reformulated to make its effect forgotten with time for β less than unity, and we only consider one sample pair from each arriving observation for the sake of simplicity.

Isolating the terms corresponding to $i = n$ from the rest of the summations in Eqs. (25) and (26) yields the following recursions for updating $\mathbf{Z}_n$ and $\mathbf{\Phi}_n$ respectively:

$$\mathbf{Z}_n = \beta \mathbf{Z}_{n-1} + \mathbf{y}_n \mathbf{u}_n^\top \,, \tag{27}$$

$$\mathbf{\Phi}_n = \beta \mathbf{\Phi}_{n-1} + \mathbf{u}_n \mathbf{u}_n^\top \,. \tag{28}$$

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 [11]

$$\mathbf{\Phi}_n^{-1} = \beta^{-1} \mathbf{\Phi}_{n-1}^{-1} - \frac{\beta^{-2} \mathbf{\Phi}_{n-1}^{-1} \mathbf{u}_n \mathbf{u}_n^\top \mathbf{\Phi}_{n-1}^{-1}}{1 + \beta^{-1} \mathbf{u}_n^\top \mathbf{\Phi}_{n-1}^{-1} \mathbf{u}_n} \,. \tag{29}$$

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

$$\mathbf{k}_n = \frac{\beta^{-1} \mathbf{u}_n^\top \mathbf{P}_{n-1}}{1 + \beta^{-1} \mathbf{u}_n^\top \mathbf{P}_{n-1} \mathbf{u}_n} \,, \tag{30}$$

then Eq. (29) can be rewritten as

$$\mathbf{P}_n = \beta^{-1} \mathbf{P}_{n-1} - \beta^{-1} \mathbf{P}_{n-1} \mathbf{u}_n \mathbf{k}_n \,. \tag{31}$$

It is surprising to find that multiplying $\mathbf{u}_n^\top$ by each side of Eq. (31) yields the following simple expression

$$\mathbf{u}_n^\top \mathbf{P}_n = \beta^{-1} \mathbf{u}_n^\top \mathbf{P}_{n-1} - \beta^{-1} \mathbf{u}_n^\top \mathbf{P}_{n-1} \mathbf{u}_n \mathbf{k}_n = \mathbf{k}_n \,. \tag{32}$$

This facilitates the derivation for recursively updating $\mathbf{W}_n$ when we substitute Eqs. (27) and (31) into Eq. (24)

$$\begin{aligned} \mathbf{W}_n &= \beta \mathbf{Z}_{n-1} \mathbf{P}_n + \mathbf{y}_n \mathbf{u}_n^\top \mathbf{P}_n \\ &= \mathbf{Z}_{n-1} \mathbf{P}_{n-1} - \mathbf{Z}_{n-1} \mathbf{P}_{n-1} \mathbf{u}_n \mathbf{k}_n + \mathbf{y}_n \mathbf{k}_n \\ &= \mathbf{W}_{n-1} + \left(\mathbf{y}_n - \mathbf{W}_{n-1} \mathbf{u}_n \right) \mathbf{k}_n \,, \end{aligned} \tag{33}$$

where the expression inside the brackets on the right-hand side of the last line represents the estimation error based on the old least-squares estimate of the weights to be learned.

3.4. Improved MBSGD for online learning

The above recursive LSE solution inspires us to improve the MBSGD algorithm of Eqs. (7) and (8) for the online learning of each MLP layer in the paradigm of Eq. (2), while the historical memory is retained. To this end, we can substitute Eq. (30) into Eq. (33) to obtain

$$\mathbf{W}_n = \mathbf{W}_{n-1} - \frac{\left(\mathbf{W}_{n-1} \mathbf{u}_n - \mathbf{y}_n \right) \mathbf{u}_n^\top}{\beta + \mathbf{u}_n^\top \mathbf{P}_{n-1} \mathbf{u}_n} \mathbf{P}_{n-1} \,, \tag{34}$$

and consequently using the fact that the matrix product $\left(\mathbf{W}_{n-1} \mathbf{u}_n - \mathbf{y}_n \right) \mathbf{u}_n^\top$ represents the gradient $\Delta \mathbf{W}_n$ with respect to the weight parameters for the input $\mathbf{u}_n$, we can use the following new single instance SGD iteration to update $\mathbf{W}_n$ upon arrival of the input $\mathbf{u}_n$

$$\mathbf{W}_n \longleftarrow \mathbf{W}_{n-1} - \frac{\eta}{\beta + \mathbf{u}_n^\top \mathbf{P}_{n-1} \mathbf{u}_n} \Delta \mathbf{W}_n \mathbf{P}_{n-1} \,. \tag{35}$$

Compared to the original single instance SGD, Eq. (35) can not only adapt the learning rate by scalling the initial η up and down, but also modify the original gradient such that $\mathbf{W}_n$ solves both the current prediction task of $\mathbf{u}_n$ and all previous prediction tasks. In other words, the memory is retained in this updating process as shown in Eq. (24). This bears some similarities to continual learning literature on overcoming catastrophic forgetting [18, 15, 39, 27].

However, since the MBSGD algorithm is commonly used for training as in Eqs. (7) and (8), we can suppose there are b inputs $\{\mathbf{u}_{n,j}\}_{j=1}^{b}$ in one batch from the current arriving observation and the mean of them and their mean gradient are accordingly denoted as

$$\overline{\mathbf{u}}_n = \frac{1}{b} \sum_{j=1}^{b} \mathbf{u}_{n,j} \quad \text{and} \quad \overline{\Delta \mathbf{W}}_n = \frac{1}{b} \sum_{i=1}^{b} \Delta \mathbf{W}_{n,j} \,. \tag{36}$$

Hence, our improved MBSGD algorithm can be represented by the following iteration

$$\mathbf{W}_n \longleftarrow \mathbf{W}_{n-1} - \frac{\eta}{\beta + \overline{\mathbf{u}}_n^\top \mathbf{P}_{n-1} \overline{\mathbf{u}}_n} \overline{\Delta \mathbf{W}}_n \mathbf{P}_{n-1} \,, \tag{37}$$

where $\mathbf{P}_n$ is updated based on $\overline{\mathbf{u}}_n$ using Eqs. (30) and (31).

It is noteworthy that all the previous derivations can be easily transplanted to the case of cross-entropy loss except that $\mathbf{u}_i^L$ in Eq. (20) is replaced with layer L's softmax output.

4. Recursive LSE-aided online tracker

In this section, we show how our proposed recursive LSE-aided online learning technique can be realized in the state-of-the-art CNN-based tracker RT-MDNet [17], a tracking-by-detection based real-time tracker with online learning. Since we only concentrate on the online tracking part, 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.

Online learning in RT-MDNet. Being identical to MDNet, RT-MDNet only updates the fully-connected MLP layers (`fc4-6`) in an online manner while keeping convolutional feature extraction layers fixed, namely update stage. Before that, the MLP layers also need to be fine-tuned using the examples 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. Finally, all the fine-tuning and updating are based on the cross-entropy loss.

Recursive LSE-aided RT-MDNet. As the recursive LSE-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 recursive LSE-aided regular updates, while the setting of the occasional updates triggered by unreliable tracking is preserved intact, in that there may be failing cases for the model based on the regular updates and the model with the occasional updates customized (over-fitting) 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 recursive LSE-aided online learning, namely RLS-RTMDNet, are explained in Algorithm 1.

Algorithm 1: Recursive LSE-Aided Online Tracking Using RT-MDNet

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

Output: Estimated target states in the rest frames

Pre-process identical to RT-MDNet, *e.g.* draw positive sample set S_1^+ and negative sample set S_1^-

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 $\mathbf{W}^l \leftarrow \mathbf{W}_n^l$ using Eqs. (36) and (37)

  Update $\mathbf{P}_n^l$ using Eqs. (30) and (31)

Initialize the frame index set $\mathcal{T} \leftarrow \{1\}$

repeat

  Find the optimal target state like in RT-MDNet

  if `target score` > 0 **then**

    Draw S_t^+ and S_t^-, and set $\mathcal{T} \leftarrow \mathcal{T} \cup \{t\}$

    if $|\mathcal{T}| > \tau$ **then** $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\min_{v \in \mathcal{T}} v\}$

  if `target score` ≤ 0 **then**

    if $\{\mathbf{W}_{\text{bk}}^l\}_{l=4}^6 = \varnothing$ **then** Create backups $\{\mathbf{W}_{\text{bk}}^l\}_{l=4}^6 \leftarrow \{\mathbf{W}^l\}_{l=4}^6$

    Occasionally update $\{\mathbf{W}^l\}_{l=4}^6$ using $S_{v \in \mathcal{T}}^+ \cup S_{v \in \mathcal{T}}^-$ based on the original MBSGD

  else if t `mod` $10 = 0$ **then**

    if $\{\mathbf{W}_{\text{bk}}^l\}_{l=4}^6 \neq \varnothing$ **then** $\{\mathbf{W}^l\}_{l=4}^6 \leftarrow \{\mathbf{W}_{\text{bk}}^l\}_{l=4}^6$, and set $\{\mathbf{W}_{\text{bk}}^l\}_{l=4}^6 = \varnothing$

    Regularly update $\{\mathbf{W}^l\}_{l=4}^6$ using $S_{v \in \mathcal{T}}^+ \cup S_{v \in \mathcal{T}}^-$:

    for *the* n_{th} *improved MBSGD iteration* **do**

      Update $\mathbf{W}^l \leftarrow \mathbf{W}_n^l$ using Eqs. (36) and (37)

      Update $\mathbf{P}_n^l$ using Eqs. (30) and (31)

until *end of sequence*

5. Experiments

5.1. Experimental setup

We evaluate the proposed recursive LSE-aided online RT-MDNet tracking method thoroughly over OTB-2015 [38], VOT2016/2017 [20, 19] and UAV123 [25] by following rigorously the evaluation protocols.

OTB-2015. It 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 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. The OP score is the fraction of frames in a sequence where the inter-section-over-union overlap of the predicted and ground truth rectangles exceeds a given threshold.

VOT2016/2017. Different from OTB-2015, they 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

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

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.

UAV123. It compiles a large set of 123 video sequences (with more than $110K$ frames) captured by low-altitude UAVs, which is inherently different from the aforementioned OTB-2015 and VOT2016/2017 datasets. It uses the same evaluation protocol to OTB-2015.

Recursive LSE-aided online learning details. We here only show the hyper-parameters specific to our recursive LSE-aided online learning, and leave all the other hyper-parameters shared with the original RT-MDNet tracker unchanged[1]. For each of the fully-connected layers $\mathbf{W}^4$, $\mathbf{W}^5$ and $\mathbf{W}^6$, we set $\widetilde{\lambda}^l$ in Algorithm 1 and the parameter to mea-

[1]We refer the reader to [17] and our improved code and raw results available at `https://github.com/Amgao/RLS-RTMDNet`.

VOT		Staple [2]	SiamFC [3]	MEEM [41]	RT-MDNet [17]	simpleRT-MDNet	MetaSDNet [28]	CSRDCF++ [23]	ECOhc [4]	**RLS-RTMDNet**	CSRDCF [23]	C-COT [7]	ECO [4]
2016	EAO	0.295	0.277	-	0.302	0.304	0.314	-	0.322	0.339	0.338	0.331	0.374
	A	0.543	0.548	-	0.559	0.550	0.539	-	0.538	0.565	0.522	0.538	0.552
	$-\ln R_S$	0.378	0.382	-	0.305	0.294	0.261	-	0.303	0.259	0.238	0.238	0.200
2017	EAO	0.169	0.188	0.192	0.223	0.218	-	0.229	0.238	0.245	0.256	0.267	0.280
	A	0.530	0.503	0.463	0.514	0.508	-	0.453	0.496	0.518	0.491	0.494	0.484
	$-\ln R_S$	0.688	0.585	0.534	0.450	0.464	-	0.370	0.435	0.399	0.356	0.318	0.276
	$rtEAO$	0.170	0.182	0.072	0.162	0.154	-	0.212	0.177	0.166	0.099	0.058	0.078

Table 1: Comparison of our RLS-RTMDNet with its degraded version and some related competing algorithms on the VOT2016/2017 benchmarks; the results are reported as EAO, A, R_S ($S = 100$) and real-time EAO ($rtEAO$). 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.

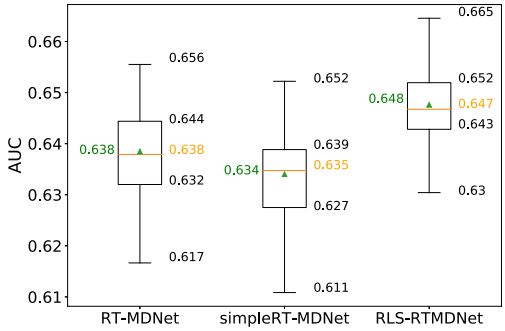
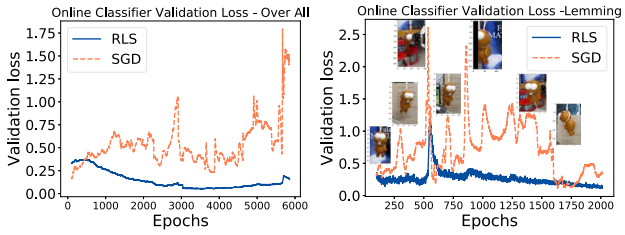


Figure 1: AUC scores of 50 trials for an ablation study comparison of our proposed RLS-RTMDNet with two baselines RT-MDNet and simpleRT-MDNet. For clarity, only the maximum, 1_{st} quartile (25%), median (orange, 50%), 3_{rd} quartile (75%), minimum and average (green) scores of each entry are shown.



(a) (b)

Figure 2: Validation loss plots based on some intermediate results to show how the over-fitting risk is reduced. Best viewed in color.

sure the memory in Eq. (26) to 1.0 for simplicity. Finally, the learning rates η for them in Eq. (37) in both initialization and update stages were fixed to 0.01, 0.01 and 0.1.

5.2. Experiment 1: Ablation study

Baseline setup. Since we selected RT-MDNet to demonstrate the effectiveness of our proposed online learning framework, we re-implemented its publicly available source code written in python as a baseline and followed all of its default settings. 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 sole memory maintained (short-term) 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 degradation of our tracker, namely simpleRT-MDNet.

We start by summarizing the ablation study with above entries in terms of AUC on the OTB-2015 benchmark in Figure 1. Due to the stochastic nature of RT-MDNet, we report the AUC scores of 50 trials for each entry. For clarity, we only show each entry's maximum, 1_{st} quartile (25%), median (50%), 3_{rd} quartile (75%), minimum and average scores. For example, the 1_{st} quartile (25%) score means that there are about 25% of the 50 trials achieving AUC scores larger than it. It is clearly shown that our online learning method improves all the six kinds of results over the baseline RT-MDNet, giving absolute gains around 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. This further demonstrates the effectiveness of our proposed recursive LSE-aided online learning method thanks to its characteristics of anti-overfitting with memory retention and the improved convergence when the cost function is computed over all historical training samples.

Moreover, we provide some intermediate results to further highlight how the recursive LSE-aided tracker's robustness is enhanced by reducing over-fitting resulting from catastrophic forgetting in SGD. The results can be intuitively presented as a validation loss plot w.r.t. each online training epoch as in Figure 2 by treating the collected samples in the first frame as validation set. The loss in Figure 2a is averaged over 50 trials of OTB-2015 and a large value means the method is prone to over-fitting and losing the memory of the first frame. Figure 2b specifically shows the case for the *Lemming* sequence, where the object's appearance and surroundings at some epochs significantly differ from its initial ones, leading to large loss for the SGD entry.

5.3. Experiment 2: State-of-the-art comparison

In contrast to the state-of-the-art comparison in the RT-MDNet paper, we replace the evaluation on TempleColor with the ones on more challenging VOT2016/VOT2017.

VOT2016/2017. 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.

We show the comparison results on VOT2016/2017 in

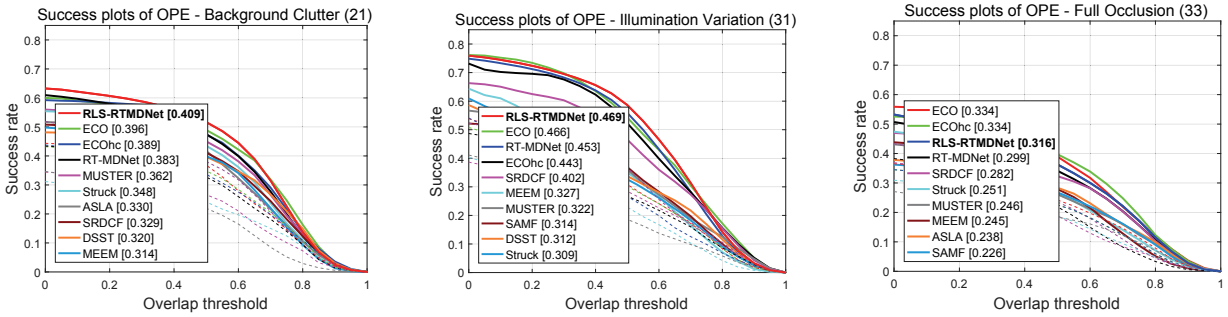


Figure 3: Success plots for UAV123 in terms of the attributes: Background Clutter (BC), Illumination Variation (IV) and Full Occlusion (FOC). The average over 50 runs is reported for MDNet-style trackers. The legends show the AUC scores. Best viewed in color.

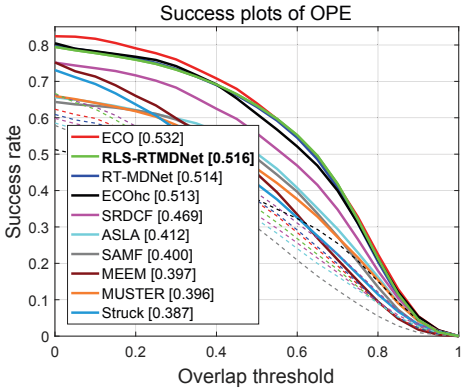


Figure 4: Success plots for the whole UAV123 dataset. We report the average over 50 runs for our tracker and the baseline RT-MDNet. The legends show the AUC scores. Best viewed in color.

Table 1. As similar to the analyses in ablation study, our new tracker always consistently improves upon its counterparts (*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, which uses sophisticated meta-training techniques to pre-train the MDNet tracker using more datasets than RT-MDNet. Note that our new online learning method doesn't deteriorate the real-time performance of RT-MDNet as shown in last line of Table 1.

In comparison with other top performers with online learning on VOT2016/2017, *e.g*. CSRDCF and ECO, which need to use their real-time counterparts (*i.e*. CSRDCF++ and ECOhc) to achieve top real-time performances, our improvement over the original RT-MDNet achieves a balance between the tracking accuracy and speed.

UAV123. We show the comparison results on the UAV123 dataset in Figure 4 and Table 2. We re-ran RT-MDNet along with our tracker over all 123 videos for 50 trails again due to their stochastic nature. Although we don not observe significant AUC improvement over RT-MDNet on this datatest due to its inherent difference from OTB-2015 and VOT2016/2017, we notice that the tracking accuracy is largely improved with an absolute gain of 1.0% in $OP_{0.75}$.

UAV123	MEEM [41]	SRDCF [6]	ECOhc [4]	RT-MDNet [17]	**RLS-RTMDNet**	ECO [4]
$OP_{0.50}$	0.446	0.557	0.610	0.630	0.633	0.640
$OP_{0.75}$	0.152	0.266	0.311	0.310	0.320	0.328
AUC	0.397	0.469	0.513	0.514	0.516	0.532

Table 2: Comparison on the UAV123 dataset in terms of AUC and mean OP given thresholds 0.50 ($OP_{0.50}$) and 0.75 ($OP_{0.75}$).

This can be attributed to our improved discrimination power based on the improved convergence when the cost function is computed over all historical training samples.

This viewpoint is further demonstrated by the improvements over the sequences with BC and IV attributes as shown in Figure 3. It is known that there is a high demand on a tracker's discrimination power in case of these two attributes. We also note that the memory retention in our formulation improves the ability of tackling the full occlusion challenge, giving an absolute AUC gain of 1.7% as shown in Figure 3. This is understandable because our memory retention improves robustness by reducing the risk of overfitting. Since we didn't rely on other sophisticated techniques for performance improvement, the AUC gains in case of other attributes in UAV123 are thus limited. Please refer to the supplementary material for more details.

6. Conclusions

We present a recursive least-squares estimator-aided online network learning algorithm that allows memory retention. We apply it to visual tracking and experimentally demonstrate its efficiency in both reducing the risk of overfitting without relying on the long-term memory and improving the convergence performance when the cost function is computed over all historical training samples. An interesting direction for future work is to integrate it into the meta-learning framework to gain further improvements.

Acknowledgments The authors would like to thank the anonymous reviewers for their valuable suggestions. This work is supported by National Key R&D Program of China (No. 2018AAA0102800, No. 2018AAA0102802, No. 2018AAA0102803), Natural Science Foundation of China (Grant No. 61972394) and MSRA Visiting Young Faculty Program (StarTrack).

References

[1] B. Babenko, M.H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1619–1632, Aug. 2011.

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

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

[4] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. ECO: Efficient convolution operators for tracking. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 6638–6646, 2017.

[5] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. ATOM: Accurate tracking by overlap maximization. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 4660–4669, 2019.

[6] 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.*, pages 4310–4318, 2015.

[7] 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.*, pages 472–488, 2016.

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

[9] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with Gaussian processes regression. In *Proc. Eur. Conf. Comput. Vis.*, pages 188–203, 2014.

[10] E. Gundogdu and A. A. Alatan. Good features to correlate for visual tracking. *IEEE Trans. on Image Process.*, 27(5):2526–2540, May 2018.

[11] W. H. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, Jun. 1989.

[12] S. Hare, A. Saffari, and P. H.S. Torr. Struck: Structured output tracking with kernels. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(10):2096–2109, Oct. 2016.

[13] S. S. Haykin. *Adaptive Filter Theory*. Pearson Education India, 2014.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 2961–2969, 2017.

[15] X. He and H. Jaeger. Overcoming catastrophic interference using conceptor-aided backpropagation. In *Proc. 6th Int. Conf. Learn. Represent.*, May 2018.

[16] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(3):583–596, Mar. 2015.

[17] I. Jung, J. Son, M. Baek, and B. Han. Real-time MDNet. In *Proc. Eur. Conf. Comput. Vis.*, pages 89–104, 2018.

[18] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Continual learning of context-dependent processing in neural networks. *Proc. Natl. Acad. Sci.*, 114(13):3521–3526, Mar. 2017.

[19] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, T. Vojíř, G. Häger, A. Lukežič, A. Eldesokey, and G. Fernández. The visual object tracking VOT2015 challenge results. In *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, pages 1949–1972, 2017.

[20] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, T. Vojíř, G. Häger, A. Lukěžič, and G. Fernández. The visual object tracking VOT2016 challenge results. In *Proc. Eur. Conf. Comput. Vis. Workshop*, pages 777–823, 2016.

[21] 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.*, pages 8971–8980, 2018.

[22] 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.*, pages 4904–4913, 2018.

[23] A. Lukežič, T. Vojíř, L. Čehovin Zajc, J. Matas, and M. Kristan. Discriminative correlation filter tracker with channel and spatial reliability. *Int. J. Comput. Vis.*, 126(7):671–688, July 2018.

[24] C. Ma, J.B. Huang, X. Yang, and M.H. Yang. Hierarchical convolutional features for visual tracking. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 3074–3082, 2015.

[25] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for UAV tracking. In *Proc. Eur. Conf. Comput. Vis.*, pages 445–461, 2016.

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

[27] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *CoRR*, 2018.

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

[29] T. A. Poggio, S. Voinea, and L. Rosasco. Online learning, stability, and stochastic gradient descent. *CoRR*, 2011.

[30] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Netw.*, 12(1):145–151, Jan. 1999.

[31] D. A. Ross, J. Lim, R.S. Lin, and M.H. Yang. Incremental learning for robust visual tracking. *Int. J. Comput. Vis.*, 77(1-3):125–141, May 2008.

[32] S. Ruder. An overview of gradient descent optimization algorithms. *CoRR*, 2016.

[33] J. R. Shewchuk. An introduction to the Conjugate Gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994.

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. 3rd Int. Conf. Learn. Represent.*, May 2015.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1–9, 2015.

[36] 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.*, pages 3643–3652, 2019.

[37] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu. DCFNet: Discriminant correlation filters network for visual tracking. *CoRR*, 2017.

[38] Y. Wu, J. Lim, and M.H. Yang. Object tracking benchmark. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(9):1834–1848, Sept. 2015.

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

[40] G. Zhang, C. Wang, B. Xu, and R. Grosse. Three mechanisms of weight decay regularization. In *Proc. 7th Int. Conf. Learn. Represent.*, May 2019.

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

[42] 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.*, pages 103–119, 2018.