Global Filter Pruning with Self-Attention for Real-Time UAV Tracking

Mengyuan Liu¹
mengyuaner1122@qq.com
Yuelong Wang¹
wang_yuelong@139.com
Qiangyu Sun²
centuryqsun@163.com
Shuiwang Li³¹,³
lishuiwang0721@163.com

¹ Guilin University of Technology, China
² Hubei Enshi College, China
³ Guangxi Key Laboratory of Embedded Technology and Intelligent System, China

Abstract

Unmanned aerial vehicle (UAV)-based tracking has wide perspective applications, however, due to the limitations of computing resources, battery capacity, and maximum load of UAVs, efficiency seems more thorny and imperative as an issue than precision in UAV tracking, which explains why discriminative correlation filters (DCF)- instead of deep learning (DL)- based trackers are usually preferred in this field, as the former are known for high efficiency, whereas, the latter hardly achieve real-time tracking on a single CPU. However, without deep representation learning the precision of DCF-based trackers is extremely limited. This paper dedicates to boost the efficiency of DL-based trackers for UAV tracking by presenting a global filter pruning and proposes a self-attention module, which seeks to learn backbone features that highlight meaningful visual inter-dependencies, in order to combat the precision drop, and, importantly, to avoid the arduous process of determining layer-wise pruning ratios in the original ranked-based filter pruning method. Remarkably, self-attention is utilized here to guide the training without introducing any extra computational burden into the inference phase. Extensive experiments on four UAV benchmarks show that the proposed tracker strikes a remarkable balance between efficiency and precision and achieves state-of-the-art performance in UAV tracking.

1 Introduction

Unmanned aerial vehicle (UAV)-based tracking has wide perspective applications in navigation, aviation, transportation, agriculture, public security and many other fields, and is developing rapidly recently [5, 28, 30, 46]. Undoubtedly, their wide and mature applications rely heavily on the tracking precision and efficiency. However, UAV tracking faces more severe challenges compared with visual tracking in general scenes [11, 28, 30, 31]. On the one hand, great challenges are posed to the precision of the UAV tracking algorithms by e.g. extreme viewing angle, motion blur, scale changes, and severe occlusion; on the other
hand, huge challenges are raised to their efficiency by e.g. limited computing resources, battery capacity limitations, low power consumption requirements, and UAV’s maximum load [26, 31, 32]. Nevertheless, at the current technical level, efficiency seems more thorny and imperative in UAV tracking, which explains why discriminative correlation filters (DCF)-instead of deep learning (DL)-based trackers are usually preferred in the UAV tracking community, since the former are known for high efficiency, whereas, the latter, especially state-of-the-art ones, hardly achieve real-time tracking on a single CPU despite their relatively higher precision, hindering their deployment on UAVs to a great extent. Although their efficiency is more favourable, without the great power of deep representation learning the precision of DCF-based trackers is so extremely limited that they hardly remain robustness under very challenging conditions. Very recently, in [5] an efficient and effective deep tracker for UAV tracking was proposed, uses a lightweight backbone for efficiency and a hierarchical feature transformer to combine features from shallow and deep layers for robust representation learning. Although it has obtained a good balance between efficiency and precision, and demonstrated remarkable performance in UAV tracking, this tracker is not yet real-time on a single CPU. But importantly, it suggests that better balance between efficiency and precision may be more easily achieved by effective and lightweight DL-based trackers than fiddling with DCF-based ones, which motivates us to explore implementing real-time yet effective DL-based trackers with model compression techniques.

Model compression aims to reduce the cost of large models by representing the model in a more efficient format with minimal impact on its performance, which are usually used to deploy deep networks in resource-constrained and low-power edge devices [8]. However, the selection of DL tracker and compression method makes a huge difference to our purpose of implementing real-time yet effective DL-based trackers. Considering DL-based trackers, SiamFC [8] is a very efficient DL-based tracker, based upon which SiamFC++ [47] demonstrates state-of-the-art performance in both precision and speed with the proposed regression branch and center-ness branch, which is chosen as the baseline DL-based tracker for compression. Regarding model compression methods, the rank-based filter pruning method proposed in [33] is straightforward yet efficient in training as no cumbersome retraining is required. But there is a shortcoming: layer-wise pruning ratios are difficult and time-consuming to decide. Adopting a global pruning ratio is a simple solution to this problem, which, however, may compromise the precision to a great extent. In view of recent advances in Natural Language Processing (NLP) and vision tasks (such as the Transformer architecture [37], BERT [13] and ViT [14]) are largely attributed to the attention mechanism, and, particularly, self-attention has shown great power in image classification and object detection tasks thanks to its capacity to learn meaningful visual inter-dependencies, in this paper, we use the self-attention mechanism to mitigate accuracy loss when pruning SiamFC++ [47] with a global pruning ratio. The proposed self-attention module seeks to learn backbone features that highlight meaningful visual inter-dependencies through guiding the finetuning process. We name the proposed method ’PS-SiamFC++’, since our tracker is obtained by applying pruning with self-attention on SiamFC++. Our contributions can be summarized as follows:

• Our work provides a fresh perspective to improve efficiency and precision of UAV tracking by developing DL-based trackers with filter pruning method, which has not been well explored before.

• We proposed a method of global filter pruning with self-attention for real-time UAV tracking, with which the proposed PS-SiamaFC++ can globally compress the baseline
SiamFC++ to about 60% of its original model size, meanwhile maintaining and even significantly boosting precision simultaneously.

- We evaluate our PS-SiamFC++ on four public UAV benchmarks, i.e., UAV123@10fps [35], DTB70 [29], UAVDT [15] and VisDrone2018 [45]. Experimental results show that the proposed PS-SiamFC++ achieves state-of-the-art performance.

2 Related Works

2.1 Visual Tracking

Modern trackers are classified as DCF-based trackers or DL-based trackers. Using DCF for visual tracking starts with the minimum output sum of squared error (MOSSE) filter [4]. Afterwards, great progresses have been witnessed [11, 17, 20, 21, 24, 25, 27, 51]. DCF-based trackers usually adopt handcrafted features and can be calculated in the Fourier domain, which leads to competitive performance with high efficiencies. Since efficiency is a critical aspect in UAV tracking, DCF-based trackers, therefore, dominate the UAV tracking community currently. However, because of the restricted representation capability of handcrafted features, DCF-based trackers frequently fail to retain robustness in complex situations. Deep learning for visual tracking has shown to be quite effective in recent years, dramatically improving tracking precision and robustness. SiamFC [1] employed the Siamese network to quantify the similarity between the target and search pictures, making it one of the first attempts to consider visual tracking as a generic similarity-learning issue. Many DL-based trackers using Siamese topologies have been presented since then. Recently, SiamRPN++ [23] and SiamBAN [7] use deeper architectures to further improve tracking precision. However, their tracking efficiency has dropped significantly. In contrast, SiamFC++ [47] is a simple yet powerful framework as it has a lightweight backbone and a quality assessment branch that is effective for enhancing performance. Unfortunately, despite its excellent GPU speed, its CPU speed appears to be too slow to fulfill strict real-time requirements (i.e., with a speed of $\gg 30$ FPS). In this work, we attempt to increase the efficiency of SiamFC++ while keeping as much precision as possible for UAV tracking.

2.2 Filter Pruning

Pruning is a common technique for compressing neural networks, which are classified as weight pruning and filter pruning. The former usually removes neurons or weights, but its acceleration on general-purpose hardware is hard to achieve. [8]. While filter pruning removes the entire filters or channels, it is much easier to achieve considerable speed-up [33]. The pruning ratio determines how many weights to eliminate, and it is generally settled in one of two methods. The first is a specified global ratio or a series of layer-wise ratios. The second option is to alter the pruning ratio indirectly, for example by employing a regularization-based pruning approach. However, the second method necessitates considerable technical modifications to attain specific ratios [40]. The pruning criterion determines which weights should be pruned. For filter pruning, Frobenius norm or sparsity of the filter response, and the scaling factor of the Batch Normalization layer are commonly used criteria [41]. Last but not least, to specify how the sparsity of the network changes from zero to the target number, i.e., pruning schedule, there are two typical choices [33, 44]: (1) a single step (one-shot), then finetune, (2) progressive pruning and training are interleaved. Although the
progressive approach is better to the one-shot approach since it provides for more time for training, the latter is more efficient and can alleviate the effort of developing complex training strategies. By and large, filter pruning so far remains an open problem. Recently, Lin et al. [33] proposed an effective and efficient filter pruning approach. They scheduled the pruning in a one-shot manner, using the rank of the feature map in each layer as the pruning criterion, which simplifies the process of pruning to a great extent. However, this approach requires a laborious and time-consuming process to determine the layer-wise pruning ratios. We propose to use a global pruning ratio to get around this issue. Furthermore, to prevent a potential precise decrease, we utilize self-attention to guide the finetuning process seeking to learn backbone features that highlight meaningful visual inter-dependencies.

2.3 Self-Attention in Vision

Attention mechanism is an attempt to mimic the human brain action of selectively concentrating on a few relevant things, while ignoring others in deep neural networks [37]. As a special case, self-attention at the outset is the primary workhorse in NLP as it is an effective and computationally efficient mechanism for capturing global interactions between words in a sentence. But self-attention has properties, such as content-based interactions, ability to capture long-range dependencies, flexibility to handle multiple types of data and etc, that make it a good fit for vision tasks as well [38]. For instance, Wang et al. [43] presented non-local operations for capturing long-range dependencies for video understanding, Fu et al. [16] proposed DANet for semantic segmentation, Zhang et al. [50] demonstrated the effectiveness of the self-attention in image generation, and Zhao et al. [53] explored two forms of self-attention for image recognition. The usefulness of self-attention in many NLP and computer vision tasks has already gotten the extensive identification. Although self-attention has spawned the rise of so many recent breakthroughs in NLP and computer vision, including the Transformer architecture [37], BERT [13] and ViT [14], they come at a cost considering the computing and memory overheads involved. In view of this and our goal of real-time DL-based trackers, in this paper we avoid using self-attention in inference, but instead exploit it to guide our tracker in the training phase, which enables us to boosts tracking precision without introducing additional computation overhead in the inference phase.

3 Proposed Method

3.1 PS-SiamFC++ Overview

The overview of the proposed PS-SiamFC++ is shown in Fig. 1. It consists of a backbone, a neck, a head network, and a self-attention module. The target patch Z and the search patch X are the inputs for the template branch and the search branch, respectively. The shared backbone network of the two branches is denoted by \( \phi(\cdot) \). The cross-correlation operation is conducted to the output backbone features of the two branches before they are passed to subsequent classification and regression tasks. The features produced by the cross-correlation operation are formulated by:

\[
f_l(Z,X) = \psi_l(\phi(Z)) \ast \psi_l(\phi(X)), \psi_l \in \{\psi_{cls}, \psi_{reg}\},
\]

where \( \psi_{cls}(\cdot) \) and \( \psi_{reg}(\cdot) \) denote the layer that is specifically designed for the tasks of classification and regression, respectively. \( \ast \) represents the cross-correlation operation. The classification branch predicts the category for each location, and its output is denoted by
PS-SiamFC++ inherits the pipeline of SiamFC++ with the difference that the filters considered less important are pruned and a self-attention module is incorporated to guide the finetuning process. Let’s first describe the rank-based filter pruning. We denote the \( i \)-th \((1 \leq i \leq K)\) convolutional layer \( C^i \) of SiamFC++ by a set of 3-D filters \( W^i_C = \{w^i_{1}, w^i_{2}, \ldots, w^i_{m}\} \in \mathbb{R}^{n_i \times n_{i-1} \times k_i \times k_i} \), where \( n_i \) is the number of filters in \( C^i \), \( k_i \) denotes the kernel size, and the \( j \)-th filter is \( w^i_{j} \in \mathbb{R}^{n_{i-1} \times k_i \times k_i} \). The filters’ output feature maps are denoted by \( O^i_C = \{o^i_{1}, o^i_{2}, \ldots, o^i_{m}\} \in \mathbb{R}^{n_g \times h_i \times w_i} \), where \( o^i_{j} \in \mathbb{R}^{g \times h_i \times w_i} \) is associated with \( w^i_{j} \), \( g \) is the number of input images, \( h_i \) and \( w_i \) denote the height and width of the feature maps, respectively.

The rank-based filter pruning aims to minimize the following objective function:

\[
\min_{\delta_{i,j}} \sum_{i=1}^{K} \sum_{j=1}^{n_i} \delta_{i,j} \mathcal{E}_{I \sim P(I)} [\mathcal{R}(o^i_j(I))], \quad s.t \sum_{j=1}^{n_i} \delta_{i,j} = n'_p, \tag{2}
\]

where \( I \) follows the \( P(I) \) distribution representing an input image, \( n'_p \) represents the number of filters to be pruned in \( C^i \). \( \delta_{i,j} \in \{0, 1\} \) indicates whether or not the filter is pruned, \( \delta_{i,j} = 1 \) if it is, otherwise \( \delta_{i,j} = 0 \). \( \mathcal{R}(\cdot) \) calculates a feature map’s rank as a measure of how rich its information is. The expectation of the rank generated by a single filter is empirically proved to be robust to the input images \([\mathbb{E}]\), by which Eq. (2) is approximated by

\[
\min_{\delta_{i,j}} \sum_{i=1}^{K} \sum_{j=1}^{n_i} \delta_{i,j} \sum_{t=1}^{g} \mathcal{R}(o^i_j(I_t)), \quad s.t \sum_{j=1}^{n_i} \delta_{i,j} = n'_p, \tag{3}
\]

where \( t \) indexes the input images. Eq. (3) is readily minimized by pruning \( npi \) filters that have the lowest average rank of feature maps.

After pruning the less important filters, the compressed network will be finetuned to optimize the parameters for the compressed model. To make the finetuning more productive, we
utilize the self-attention mechanism to draw dependencies between spatial features, seeking to enhance significant parts while diminishing less informative parts of the features output by the pruned backbone for our tracking task. How the self-attention module plays a part is illustrated in Fig. 1. The backbone output in the template branch, denoted by $f_Z$, will be fed into a multi-head self-attention module to generate an enhanced feature representation $f_Z^*$, which is used in turn to supervise $f_Z$ with the mean squared error (MSE) loss $L_{mse}$. The self-attention module consists of a Multi-Head Self-Attention layer (refer to supplementary material for its concrete structure). Intuitively, the attention mechanism describes a weighted average of (sequence) elements with the weights dynamically computed based on an input query and elements’ keys. In our implementation, $f_Z^*$ is encoded in a pixel-wise manner, i.e., the spatial coordinates of $f_Z$ index the tokens, and the query, key and value are initially the same, for simplicity. Note that the output of the self-attention module is used for finetuning only, the module plays no part in the inference phase.

We now formulate the losses for finetuning the PS-SiamFC++. Let $(x_0, y_0)$ and $(x_1, y_1)$ denote the ground truth bounding box’s left-top and right-bottom coordinates, and $(x, y)$ denote the corresponding location of point $(i, j)$, then the regression target $\hat{x}_{i,j} = \{(\hat{x}_{i,j}), \hat{y}_{i,j}, \hat{w}_{i,j}, \hat{h}_{i,j}\}_{k=0}^3$ of $O_{h\times w\times 4}^{reg}(i, j, :)$ is

$$\hat{x}_{i,j}^0 = \hat{x} = x - x_0, \hat{y}_{i,j}^0 = \hat{y} = y - y_0, \hat{w}_{i,j}^0 = \hat{w} = x_1 - x, \hat{h}_{i,j}^0 = \hat{h} = y_1 - y.$$  (4)

The differences between $O_{h\times w\times 4}^{reg}(i, j, :)$ and the regression target is penalized by the loss

$$L_{reg} = \frac{1}{\sum_{i,j} \mathbb{I}(\hat{t}_{i,j})} \sum_{i,j} \mathbb{I}(\hat{t}_{i,j}) L_{IOU}(O_{h\times w\times 4}^{reg}(i, j, :), \hat{t}_{i,j}),$$  (5)

where $L_{IOU}$ is the IOU loss as defined in [33], $\mathbb{I}(\cdot)$ is the indicator function defined as follow

$$\mathbb{I}(\hat{t}_{i,j}) = \begin{cases} 1 & \text{if } \hat{t}_{i,j}^k > 0, k = 0, 2, 3 \\ 0 & \text{otherwise.} \end{cases}$$  (6)

Denote $O_{h\times w\times 1}^{cen}(i, j)$, i.e., the centerness score at $(i, j)$, by $c(i, j)$ as follows,

$$c(i, j) = \mathbb{I}(\hat{t}_{i,j}) \sqrt{\frac{\min(\hat{x}, \hat{w})}{\max(\hat{x}, \hat{w})}} \times \frac{\min(\hat{y}, \hat{h})}{\max(\hat{y}, \hat{h})}.$$  (7)

The centerness loss is defined by

$$L_{cen} = \frac{-1}{\sum_{i,j} \mathbb{I}(\hat{t}_{i,j}) \mathbb{I}(\hat{t}_{i,j}) = 1} \sum_{i,j} c(i, j) \log(O_{h\times w\times 1}^{cen}(i, j)) + (1 - c(i, j)) \log(1 - O_{h\times w\times 1}^{cen}(i, j)).$$  (8)

Finally, the overall loss for finetuning PS-SiamFC++ is:

$$L = L_{cls} + \lambda_1 L_{reg} + \lambda_2 L_{cen} + \lambda_3 L_{mse}(f_Z, f_Z^*),$$  (9)

where $L_{cls}$ is the cross-entropy loss for classification, $\lambda_1, \lambda_2$, and $\lambda_3$ are predefined constants.

### 3.3 Pruning Schedule

The pipeline of pruning is: First, calculate the average rank of the feature map of any filter in each layer to obtain the rank sets $\{R_i\}_{i=1}^K = \{\{r_{i_1}, r_{i_2}, \ldots, r_{i_{n_i}}\}\}_{i=1}^K$. Second, each set $R_i$ is sorted in decreasing order, resulting in $\tilde{R}_i = \{r_{j_1}, r_{j_2}, \ldots, r_{j_{s_{n_i}}}\}$, where $s_{j_i}$ denotes the index of the $j$-th top value in $R_i$. Third, perform filter pruning with a predefined global pruning ratio $\rho$, after which $R_i$ turns to $\hat{R}_i = \{r_{s_{j_1}}, r_{s_{j_2}}, \ldots, r_{s_{j_{s_{n_i}}}}\}$, $\hat{n}_i = n_i - \rho n_i$ and incorporate the self-attention module to obtain the PS-SiamFC++ model. Finally, PS-SiamFC++ is finetuned after the remained filters are initialized with the original weights in the trained SiamFC++. 
4 Experiments

4.1 Experiment Settings

Our experiments are conducted on four challenging UAV benchmarks, i.e., UAV123@10fps [35], DTB70 [29], UAVDT [15] and VisDrone2018 [45]. All evaluation experiments are conducted on a PC (with i9-10850K processor (3.6GHz), 16GB RAM, and an NVIDIA TitanX GPU) and on a tiny mini PC, i.e., Intel NUC (with an i5-1135G7 processor, 16GB RAM). Our PS-SiamFC++ is set with a global pruning ratio of 0.4. Other settings for training and inference follow SiamFC++ [47]. Code is available on: https://github.com/PS-SiamFCpp/PS-SiamFCpp.

4.2 Comparison with DCF-based Trackers

Ten state-of-the-art trackers with handcrafted features are used for comparison, including KCF [21], fDSST [8], Staple-CA [36], BACF [1], ECO-HC [10], MCCT-H [42], STRCF [24], ARCF-HC [22], AutoTrack [30], and RACF [26]. Fig. 2 demonstrates the overall performance of PS-SiamFC++ with the competing trackers on the four benchmarks. PS-SiamFC++ outperformed all other trackers by a significant margin on three benchmarks, namely UAV123@10fps, DTB70, and UAVDT. On the three benchmarks, in terms of precision and area under curve (AUC), PS-SiamFC++ exceeds the second tracker RACF by (4.9%, 7.5%), (7.4%, 10.1%), and (3.2%, 7.5%), respectively. Although PS-SiamFC++ is inferior to the first tracker RACF in precision (by 1.8%), we get the best AUC when combined with RACF for VisDrone2018. It’s worth noting that the settings of are dataset specific but ours are not. We evaluate the average FPS over the four benchmarks on the CPU of the PC and the NUC, respectively. Table 1 shows the average precision and FPS produced by different trackers. PS-SiamFC++ is the best real-time tracker (speed of >30FPS) on the PC and the NUC and it beats all rival trackers in terms of precision. Specifically, PS-SiamFC++ achieves 79.1% in precision at a frame rate of 71.3 FPS and 62.4 FPS on the PC and the NUC, respectively. Qualitative comparison can be found in the supplementary material.

---

1 Unless otherwise specified, the precision metric in our experiment refers to distance precision at 20 pixels.
Table 1: Comparison of average precision and speed (FPS) between PS-SiamFC++ and hand-crafted based trackers on the four benchmarks. The reported FPSs are evaluated on a single CPU. Red, blue and green respectively show the first, second and third places.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>PS-SiamFC++</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCFT-TPAMI 15</td>
<td>53.3</td>
<td>57.7</td>
</tr>
<tr>
<td>DSSS-IVPVR 16</td>
<td>60.4</td>
<td>59.4</td>
</tr>
<tr>
<td>Single-CA ICCV 17</td>
<td>64.2</td>
<td>50.1</td>
</tr>
<tr>
<td>RADM-ECVCVPR 17</td>
<td>65.3</td>
<td>77.8</td>
</tr>
<tr>
<td>ECO-ECVCVPR 17</td>
<td>68.8</td>
<td>26.3</td>
</tr>
<tr>
<td>MCCT-IVPVR 18</td>
<td>66.6</td>
<td>31.4</td>
</tr>
<tr>
<td>STRCF-CVPR 18</td>
<td>67.4</td>
<td>13.4</td>
</tr>
<tr>
<td>ARCF-ECVCVPR 19</td>
<td>71.9</td>
<td>32.2</td>
</tr>
<tr>
<td>AutoTrack-CVPR 20</td>
<td>73.3</td>
<td>61.8</td>
</tr>
<tr>
<td>RACF-3DV 21</td>
<td>75.7</td>
<td>37.5</td>
</tr>
<tr>
<td>Precision</td>
<td>66.5</td>
<td>82.6</td>
</tr>
<tr>
<td>FPS (PC)</td>
<td>7.2</td>
<td>42.1</td>
</tr>
<tr>
<td>FPS (NUC)</td>
<td>66.7</td>
<td>29.4</td>
</tr>
</tbody>
</table>

In Fig. 3, we compare our method’s qualitative tracking results to four top CPU-based trackers. As can be seen, when the objects are subjected to substantial deformations (i.e., BMX5), pose changes (i.e., truck1 and S0309), and partial occlusion (i.e., uav000029400000 s), the four CPU-based trackers eventually fail to accurately track the objects. However, thanks to the strong deep representation learning, our PS-SiamFC++ performs better and is visually more pleasing. It implies that building more efficient DL-based trackers, for example, by model compression, might be more beneficial in improving UAV tracking precision.

4.3 Comparison with DL-based Trackers

Ten state-of-the-art DL-based trackers are also compared with the proposed PS-SiamFC++, including PrDiMP18 [12], SiamR-CNN [39], D3S [62], KYS [48], SiamGAT [62], LightTrack [62], TransT [8], HiFT [8], SOAT [62], and AutoMatch [62]. Table 2 displays the FPSs and precisions of the trackers on UA VDT. As is shown, although PS-SiamFC++ (pruning ratio ρ = 0.4) is not as accurate as TransT, SOAT, AutoMatch and LightTrank, the gap is not more than 2.1%. Moreover, PS-SiamFC++ has a GPU performance about 7 times that of the first tracker TransT and 10 times that of the second tracker SOAT. PS-SiamFC++, in particular, achieves a precision of 80.5 percent and a GPU performance of 291.9 FPS. It achieves a fantastic balance between precision and efficiency when compared to the first tracker TransT, which achieves 82.6 percent precision and 42.1 FPS GPU speed (i.e., speed).

4.4 Ablation Study

Effect of pruning with self-attention: We integrate the proposed method into two baseline trackers, i.e., SiamCAR [13] and SiamFC++, and evaluate their performance with and without the proposed components (i.e., filter pruning and self-attention) on the four benchmarks. Table 3 shows the precisions and speeds evaluated on the PC. As can be seen, when applying filter pruning with a global pruning ratio of 0.4, resulting in P-SiamCAR and P-SiamFC++, the model size of both baselines are reduced to 60.0% of the original size, i.e., 5.1M and 5.8M, respectively. Their speeds are thus increase significantly. Specifically, from 40.7 FPS to 79.4 FPS and from 36.5 FPS to 71.3 FPS, respectively. However, all their precisions decrease, except the ones of SiamFC++ on UAVDT and VisDrone2018. This suggests that filter pruning, especially with a global pruning ratio, can either improve or decrease the precision of the model to be compressed, which depends on both the model itself and the dataset for evaluation. Remarkably, when the self-attention component is integrated into
Figure 3: Qualitative evaluation on 4 sequences from UA V123@10fps, DTB70, UA VDT and VisDrone2018 (i.e. truck1, BMX5, S0309 and uav0000294_00000_s), respectively. The results of different methods have been shown with different colors.

Table 3: Comparison of the proposed PS-SiamFC++ tracker with two baseline trackers in terms of model size (Parameters), precision (DP), and tracking speed (FPS) on the PC CPU.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Parameters</th>
<th>Pruning</th>
<th>Self-attention</th>
<th>UA V123@10fps</th>
<th>DTB70</th>
<th>UA VDT</th>
<th>VisDrone2018</th>
<th>Avg. Precision</th>
<th>Avg. FPS (CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiamCAR [18]</td>
<td>8.5M</td>
<td></td>
<td></td>
<td>73.7</td>
<td>76.6</td>
<td>76.1</td>
<td>80.3</td>
<td>76.7</td>
<td>40.7</td>
</tr>
<tr>
<td>P-SiamCAR</td>
<td>5.1M</td>
<td>✓</td>
<td></td>
<td>70.9</td>
<td>68.2</td>
<td>72.7</td>
<td>74.6</td>
<td>71.6</td>
<td>79.4</td>
</tr>
<tr>
<td>PS-SiamCAR</td>
<td>5.1M</td>
<td>✓</td>
<td>✓</td>
<td>71.0</td>
<td>73.2</td>
<td>77.1</td>
<td>75.8</td>
<td>74.1</td>
<td>79.3</td>
</tr>
<tr>
<td>SiamFC++ [47]</td>
<td>9.7M</td>
<td></td>
<td></td>
<td>72.8</td>
<td>80.5</td>
<td>76.2</td>
<td>72.5</td>
<td>75.5</td>
<td>36.5</td>
</tr>
<tr>
<td>P-SiamFC++</td>
<td>5.8M</td>
<td>✓</td>
<td></td>
<td>71.9</td>
<td>79.5</td>
<td>78.8</td>
<td>79.3</td>
<td>77.4</td>
<td>71.3</td>
</tr>
<tr>
<td>PS-SiamFC++</td>
<td>5.8M</td>
<td>✓</td>
<td>✓</td>
<td>74.3</td>
<td>79.9</td>
<td>80.5</td>
<td>81.6</td>
<td>79.1</td>
<td>71.1</td>
</tr>
</tbody>
</table>

P-SiamCAR and P-SiamFC++, resulting in PS-SiamCAR and PS-SiamFC++, all the precisions of both compressed models are improved. For example, the precisions of P-SiamCAR on DTB70 and UAVDT are raised by 5.0% and 4.4% and the precisions of P-SiamFC++ on UAV123@10fps and VisDrone2018 are improved by 2.4% and 2.3%, respectively. In average, PS-SiamCAR and PS-SiamFC++ achieve an improvement of 2.5% and 1.7% in precision over P-SiamCAR and P-SiamFC++, respectively. Note that speeds of the models with and without the self-attention component are very close since the self-attention impacts the training only but not the inference of the model. These results justify the effectiveness of the proposed method, which can be attributed to the effectiveness of filter pruning for improving efficiency and the effectiveness of self-attention in highlighting relevant visual inter-dependencies thus providing more effective feature representations for tracking.

Impact of the global pruning ratio: To see how the global pruning ratio affects the final precision, S-SiamFC++ was trained and evaluated with different global pruning ratios. For further comparison, it was also trained and evaluated without the proposed self-attention module (i.e., P-SiamFC++). The global ratio $\rho$ ranges from 0.1 to 0.8. Note that the higher the ratio, the more filters will be removed. Table 4 shows the precisions of PS-SiamFC++
Table 4: Illustration of how the precision on DTB70 of PS-SiamFC++ varies with the global pruning ratio, with or without the self-attention module. The precisions that have been improved by the self-attention component are marked in bold.

<table>
<thead>
<tr>
<th>ρ</th>
<th>UAV123@10fps w/o</th>
<th>UAV123@10fps w/</th>
<th>DTB70 w/o</th>
<th>DTB70 w/</th>
<th>UAVDT w/o</th>
<th>UAVDT w/</th>
<th>VisDrone2018 w/o</th>
<th>VisDrone2018 w/</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>70.8</td>
<td>72.2</td>
<td>79.6</td>
<td>79.4</td>
<td>81.4</td>
<td>81.3</td>
<td>79.6</td>
<td>83.1</td>
</tr>
<tr>
<td>0.2</td>
<td>71.6</td>
<td>72.3</td>
<td>80.0</td>
<td>80.1</td>
<td>76.9</td>
<td>77.2</td>
<td>80.2</td>
<td>77.7</td>
</tr>
<tr>
<td>0.3</td>
<td>71.3</td>
<td>72.4</td>
<td>81.0</td>
<td>81.5</td>
<td>83.9</td>
<td>80.3</td>
<td>75.6</td>
<td>79.3</td>
</tr>
<tr>
<td>0.4</td>
<td>71.9</td>
<td>74.3</td>
<td>79.5</td>
<td>79.9</td>
<td>78.8</td>
<td>80.5</td>
<td>79.3</td>
<td>81.6</td>
</tr>
<tr>
<td>0.5</td>
<td>71.1</td>
<td>69.2</td>
<td>77.6</td>
<td>79.2</td>
<td>77.5</td>
<td>78.0</td>
<td>72.7</td>
<td>76.4</td>
</tr>
<tr>
<td>0.6</td>
<td>68.7</td>
<td>69.2</td>
<td>78.6</td>
<td>77.1</td>
<td>76.6</td>
<td>78.3</td>
<td>75.2</td>
<td>76.9</td>
</tr>
<tr>
<td>0.7</td>
<td>69.1</td>
<td>66.0</td>
<td>77.9</td>
<td>74.9</td>
<td>77.8</td>
<td>80.7</td>
<td>76.7</td>
<td>79.4</td>
</tr>
<tr>
<td>0.8</td>
<td>65.2</td>
<td>66.0</td>
<td>76.4</td>
<td>70.1</td>
<td>74.6</td>
<td>77.0</td>
<td>71.8</td>
<td>74.9</td>
</tr>
</tbody>
</table>

with and without the self-attention module with respect to the global pruning ratio. As can be seen, the highest precisions are primarily reached at ρ less than 0.5, which is consistent with our expectation for filter pruning techniques. Remarkably, incorporating the proposed self-attention improves most precisions. On UAV123@10fps and UAVDT, for example, six out of eight precisions are improved, and on VisDrone2018 seven out of eight precisions is raised. This allows us to maintain highly favorable precisions with larger pruning ratios, particularly when the global pruning ratio is set to the default value of 0.4, where the increases in precision are 2.4%, 0.4%, 1.7%, and 2.3%, on UAV123@10fps, DTB70, UAVDT, and VisDrone2018, respectively. This demonstrates the remarkable balance between precision and efficiency achieved by our method, justifying the its effectiveness for real-time UAV tracking.

5 Conclusion

In this work, we present a method of global filter pruning with self-attention for real-time UAV tracking and achieve state-of-the-art performance on four public UAV tracking benchmarks. When using the proposed method to improve UAV tracking efficiency, experimental results reveal that the proposed method is quite effective at maintaining and even improving precision. Surprisingly, the proposed PS-SiamFC++ not only outperforms the baseline SiamFC++ in terms of efficiency (PS-SiamFC++ can run at and more than 62 FPS on a single CPU of a mini PC, i.e., Intel NUC), but it also outperforms the baseline in terms of precision on UAVDT and VisDrone2018, well combating the adverse effects of filter pruning.

6 acknowledgement

Thanks to the support by Guangxi Science and Technology Base and Talent Special Project (No. Guike AD22035127)

References


[28] Shuiwang Li, Yuting Liu, Qijun Zhao, and Ziliang Feng. Learning residue-aware correlation filters and refining scale for real-time uav tracking. Pattern Recognition, 2022.


