IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 22, 2019, accepted July 7, 2019, date of publication August 2, 2019, date of current version August 15, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2932733

Efficient Convolution Neural Networks for Object Tracking Using Separable Convolution and Filter Pruning

YUANHONG MAO, ZHANZHUANG HE, ZHONG MA, XUEHAN TANG, AND ZHUPING WANG
Xi'an Microelectronics Technology Institute, Xi'an 710065, China

Corresponding author: Zhong Ma (mazhong@mail.com)

This work was supported by the National Natural Science Foundation of China under Grant 61702413.

ABSTRACT Object tracking based on deep learning is a hot topic in computer vision with many applications. Due to high computation and memory costs, it is difficult to deploy convolutional neural networks (CNNs) for object tracking on embedded systems with limited hardware resources. This paper uses the Siamese network to construct the backbone of our tracker. The convolution layers used to extract features often have the highest costs, so more improvements should be focused on them to make the tracking more efficient. In this paper, the standard convolution is optimized by the separable convolution, which mainly includes a depthwise convolution and a pointwise convolution. To further reduce the calculation, filters in the depthwise convolution layer are pruned with filters variance. As there are different weight distributions in convolution layers, the filter pruning is guided by a hyper-parameter designed. With the improvements, the number of parameters is decreased to 13% of the original network and the computation is reduced to 23%. On the NVIDIA Jetson TX2, the tracking speed increased to 3.65 times on the CPU and 2.08 times on the GPU, without significant degradation of tracking performance in VOT benchmark.

INDEX TERMS Object tracking, deep learning, separable convolution, filter pruning.

I. INTRODUCTION

Visual object tracking tasks predict the object region in the subsequent frames when its size and position are given in the first video frame. Visual object tracking is a hot field in computer vision. It has a wide range of applications, such as video surveillance, human-computer interaction, unmanned vehicle driving and so on. Object tracking is a challenging task because the object itself and the tracking scenes often are very complex and change frequently.

Visual object tracking methods can be classified into two categories: methods based on generative models and methods based on discriminant models. The methods based on generative models suggest that tracking is a process of object matching, which locates the region with the best matching or the smallest reconstruction error, i.e. the largest posterior probability region, as the object region prediction [1]–[3]. The methods based on discriminant models consider tracking as a binary classification. By classifying the object boundary in the background, object and background are identified. Since they can distinguish background and object more significantly, the discriminant models are more robust and thus have gradually dominated the object tracking research field. The methods based on discriminant models mainly include correlation filtering algorithm and deep learning algorithm in recent years. The correlation filtering algorithm finds the maximum response region by comparing the signals similarity, then determines the object location, such as MOSSE [4], KCF [5], C-COT [6], ECO [7].

Before the emergence of deep learning, the discriminant model mainly used hand-crafted features. There are two deficiencies in this way. Firstly, with the increasing samples in tracking data sets, the hand-crafted features in the early tracking algorithm are too simple to satisfy the sample diversity. Secondly, in traditional tracking algorithms, feature extraction and classification are two independent and unrelated parts, which can't be optimized end-to-end. Therefore, the classification results can't be used to evaluate the performance of the feature extraction.

With the widespread application of deep learning in computer vision, it has achieved excellent performance in such fields as classification, detection, semantics segmentation and

The associate editor coordinating the review of this manuscript and approving it for publication was Huazhu Fu.

so on. Many algorithms use CNN to extract features or train the tracking network end-to-end directly. As CNNs have significant advantages, the traditional tracking methods, such as C-COT and ECO, also use deep learning to extract features. For example, ECO algorithm based on deep learning improves accuracy by 16% and robustness by 33% compared with that based on hand-crafted features (HOG and Color Names).

Although CNN has excellent performance in computer vision, it has millions of parameters and billions of calculations. Tracking algorithms based on the deep neural network are all very slow due to the tremendous amount of computation. So, the greatest challenge of tracking algorithm based on deep neural network is to reduce its computation. In real scenarios, deep learning-based tracking algorithms often run on terminals with limited storage and computation. Optimizing these algorithms is necessary.

Convolution layers in deep neural networks usually extract the features of the object region and each video frame. This process incurs most of the parameters and calculations in tracking networks. Therefore, we should make convolution layers more efficient.

In this paper, we propose an approach to significantly reduce the computation of the tracking algorithm based on a deep neural network. The standard convolution is optimized by a separable convolution in the object tracking network. There may be redundant filters in convolution. To further reduce the network computation, a filter pruning scheme is proposed based on the analysis of filters variance.

The contributions of this paper are as follows:

- The separable convolution is used to optimize the traditional convolution. With this improvement, the number of parameters is decreased to 17% of the original network and the computation is decreased to 28%.
- Filters are pruned in depthwise convolution layer. The parameters and calculations in the network are further reduced. The parameter size is decreased to 13% and the computation is reduced to 23%.
- We propose a hyperparameter α, which can help us prune the unimportant filters. We sort the filters according to their variance in each convolutional layer. The cumulative variance of pruned filters should be less than the total variance multiplied with α, instead of a fixed percentage.

Although our approach implemented is based on the SiameseFC tracking network, we mainly focus on accelerating the convolution layers for feature extraction. In theory, the separable convolution and filter pruning in this paper can also be applied to other deep tracking networks where real-time performance is critical.

The remainder of this paper is organized as follows. In section II, we introduce related works. In section III, we describe how our approach makes the tracking network more efficient in details. Section IV shows the experiment results. Section V summarizes the paper.

II. RELATED WORK

A. OBJECT TRACKING METHOD BASED ON DEEP LEARNING

Deep learning-based tracking algorithms have developed rapidly since 2013. In recent years, they have gradually surpassed the traditional methods in accuracy and robustness. With deep learning, the tracking network can be trained end-to-end and can extract the features from labeled data sets. Now visual object tracking algorithms based on deep learning have become a mainstream of tracking algorithms.

MDNet[8] is a multi-domain learning framework based on CNN, which separates domain-independent information from domain-specific information to obtain effective feature representation. It uses offline training and bounding-box regression for object region prediction. Its large computation costs have severely affected its application. Faster algorithms have been proposed, such as GOTURN [9], however, GOTURN only runs on a high-performance GPU at the expense of accuracy. The fully convolution Siamese networks for object tracking (SiameseFC) [10] is a typical network that applies Siamese CNN with end-to-end training. Based on SiameseFC network, many algorithms, such as CFNet [11], PTAV [12], Siam RPN [13], SA-Siam [14], FPSN-MOT [15] have been derived and achieved good performance.

B. ACCELERATING CNN METHODS

Although the deep learning method improves the accuracy and robustness in tracking, it is hard to be deployed on mobile devices. At present, there are many methods to reduce the number of parameters and accelerate the calculation. Among them, the convolution decomposition is an effective method. Following these ideas, several more compact network architectures have emerged in the industry. SqueezeNet [16] constructs the fire module in the network to reduce parameters. Xception[17] network effectively compresses the feature map channels by 1×1 convolution. MobileNet [18] decompose a standard layer into a depthwise convolution and a channel convolution. To further reduce calculation, ShuffleNet [19] remove the channel convolutions by channels shuffling to integrate information among channels.

In addition, pruning can be used to simplify networks for trained models. Pruning includes structured pruning and unstructured pruning, according to the pruning granularity. Unstructured pruning mainly prunes the single weight or the single vector in filters. Although unstructured pruning can effectively reduce the parameters and produce a sparse network model, it does not directly reduce calculation. Accelerating the neural network with unstructured pruning must be supported by special libraries. Structured pruning deletes filters or feature maps according to a specific criterion, which directly reduces the filters or feature maps related to matrix multiplication. It does not require special libraries.

There have been many discussions on the criteria of filter pruning or feature map pruning. Li *et al.* [20] evaluate the absolute weights of filters in a network, concluding that the

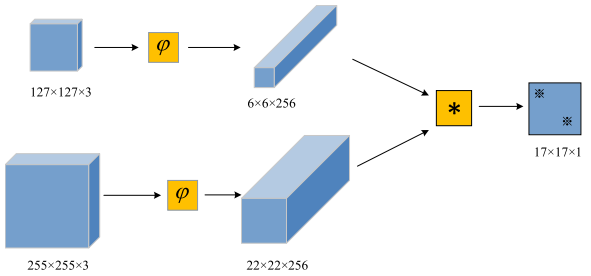

FIGURE 1. Outline of our network architecture.

filter with smaller values should be pruned first. This method only evaluates the absolute value of the weights in the filter. In fact, filters with a large fixed absolute value still maybe can't extract important information. Han *et al.* [21] use iterative validation to determine which weights have more impact on the loss function. Weights with smaller impact should be pruned first. Molchanov *et al.* [22] use the first order Taylor expansion to evaluate the importance of filters in iterative training. Hu *et al.* [23] propose the Average Percentage of Zeros (APoZ) in the activation layer to determine which feature map is significant. The pruning methods [21]–[23] require many iterations in the training process to determine the importance of a filter or feature map, which is very time-consuming.

III. OUR APPROACH

In this section, we first illustrate the basic structure of our tracking algorithm. Then, we describe our method to make the tracker more efficient. The standard convolutions in the two CNN branch are optimized with the separable convolution and the network is composed of new convolutions trained from scratch. To further reduce computation, filters in the depthwise convolution layer are pruned in trained models. After filter pruning, retraining is used to ensure network performance.

A. OUR NETWORK ARCHITECTURE

We construct the Siamese network as network infrastructure. Siamese network is a very typical network in deep learning. It uses end-to-end training in neural networks, rather than merely extracting features by CNN. Many tracking algorithms based on deep learning derived from it.

As shown in Fig. 1, the Siamese network uses two CNN branches to extract features respectively. The object region $(127 \times 127 \times 3)$ in the first frame is input into a CNN branch to extract the object feature maps $(6 \times 6 \times 256)$. The subsequent video frames $(255 \times 255 \times 3)$ are fed into another CNN branch to get the frame feature maps $(22 \times 22 \times 256)$. We locate the object position by comparing the similarity between the objects and the candidate region in the subsequent frames. The yellow block marked with the symbol "φ" represents the convolution neural network used to extract features. The yellow block marked with the symbol "*" represents features cross-correlation.

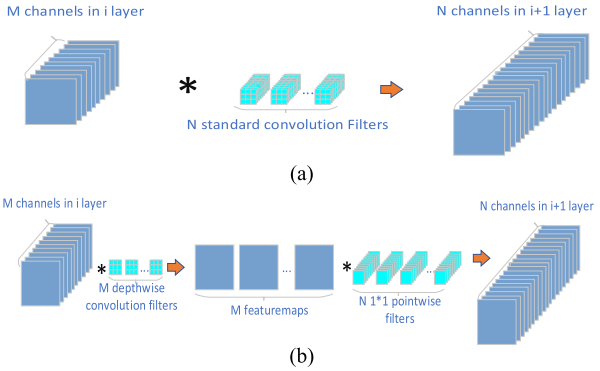

FIGURE 2. (a) The process of standard convolution. (b) The process of separable convolution.

The two CNN branches share the same network weights. Because the two convolution branches account for most of the calculations in the network, we mainly focus on the CNN branches improvement.

B. EFFICIENT NEURAL NETWORK BASED ON SEPARABLE CONVOLUTION

The separable convolution substitutes the standard convolution in the original Siamese network, except for the first convolution layer.

The first convolution layer in the network is not changed to avoid loss of the receptive field. A standard convolution becomes a block structure of depthwise Conv + BN + ReLU + 1×1 pointwise Conv + BN + ReLU. Fig. 2 illustrates the standard convolution and separable convolution. The BN and ReLU layers are not shown in Fig. 2. The 1×1 pointwise convolution means the filter size is 1×1 in the convolution.

As shown in Fig. 2, a separable convolution block mainly includes a depthwise convolution and a 1×1 convolution. Separable convolution ensures that features can be extracted in both spatial and channel levels.

The dimension of traditional convolution filters is $H \times W \times C$ (Height × Width × Channel). It is necessary to ensure that the area features are extracted from a two-dimensional plane ($H \times W$ size). On the other hand, other crucial features also should be obtained from the channels. The Separable convolution also has two stages: depthwise convolution and pointwise convolution. As shown in Fig. 2(b), the depthwise convolution focuses on the area information. The pointwise convolution is mainly concerned with channel information. To some extent, we can consider separable convolution as a convolution matrix decomposition.

In depthwise convolution, the channel number of input data is consistent with the number of filters. For example, the first input channel is convoluted only with the first filter; the second input channel is convoluted only with the second filter, and so on. The results of each channel convolution are independent and not cumulative. Since depthwise convolution confines feature information to a single channel,

TABLE 1. Comparison of the Siamese network structures using separable convolution.

Original Network architecture of a single branch in the Siamese network			
Layer	Kernel Size	Chan×map	Stride
Conv1	11×11	96×3	2
Pool1	3×3		2
Conv2	5×5	256×96	1
Pool2	3×3		2
Con3	3×3	384×256	1
Conv4	3×3	384×384	1
Conv5	3×3	256×384	1

Proposed Network architecture of a single branch in the Siamese network			
Layer	Kernel Size	Chan×map	Stride
Conv1	11×11	96×3	2
Pool1	3×3		2
Con2_1	5×5	96×1	1
Con2_2	1×1	256×96	1
Pool2	3×3		2
Con3_1	3×3	256×1	1
Con3_2	1×1	384×256	1
Con4_1	3×3	384×1	1
Con4_2	1×1	384×384	1
Con5_1	3×3	384×1	1
Con5_2	1×1	256×384	1

the 1×1 pointwise convolution layer is necessary to integrate the feature maps of each channel.

For standard convolution, the computation complexity can be estimated as

$$D_K \times \mathrm{D_K} \times \mathrm{M} \times N \times \mathrm{D_F} \times \mathrm{D_F} \tag{1}$$

where $\mathrm{D_K}$ is the filter size in current convolution layer. $\mathrm{D_F}$ is the size of the input feature map. M indicates the number of input channels. N indicates the number of output channels. The computation complexity in separable convolution block is formulated as

$$D_K \times \mathrm{D_K} \times \mathrm{M} \times \mathrm{D_F} \times \mathrm{D_F} + \mathrm{M} \times \mathrm{N} \times D_F \times D_F \tag{2}$$

The production of $D_K \times \mathrm{D_K} \times \mathrm{M} \times \mathrm{D_F} \times \mathrm{D_F}$ is the computation complexity of depthwise convolutions which focus on the spatial information. The production of $\mathrm{M} \times \mathrm{N} \times D_F \times D_F$ is the computation complexity of 1×1 pointwise convolution, which integrates multichannel information across the M channels. A standard convolution is decomposed into a spatial convolution and a channel convolution respectively. It is obvious that the computation of a separable convolution block is much less than that of standard convolution. The ratio is shown as

$$\frac{D_K \times \mathrm{D_K} \times \mathrm{M} \times \mathrm{D_F} \times \mathrm{D_F} + \mathrm{M} \times \mathrm{N} \times D_F \times D_F}{D_K \times \mathrm{D_K} \times \mathrm{M} \times N \times \mathrm{D_F} \times \mathrm{D_F}} = \frac{1}{N} + \frac{1}{D_K^2} \tag{3}$$

For example, the size $\mathrm{D_K}$ is 3 in our third convolution layer. The number of parameters is 1/9 of that before optimization. 1/N is usually small and can be neglected.

With this method, the new architecture of a single branch in Siamese networks is summarized in Table 1. The description of the original network is on the left. Our network is on the right. The word ''Chan'' represents the channel number of the output feature map. The word ''map'' represents the channel number of the input feature map. On the right, the Con∗_1 layers are depthwise convolutions. The number of the input channel is equal to the number of filters in depthwise convolution. The symbol ''∗'' represents the number from 2 to 5. The Con∗_2 layers are 1×1 pointwise convolutions. The number of 1×1 filter channel is equal to the input channel number in pointwise convolution. In the proposed network architecture, the number of parameters is decreased to 17% of the original network and the computation is reduced to 28%.

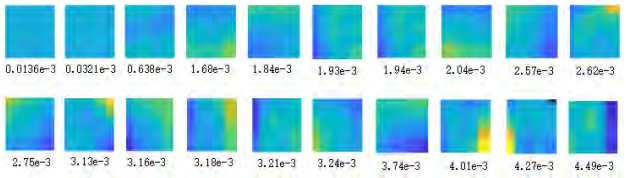

FIGURE 3. Visualization of twenty filters selected randomly in Con2_1 layer. Each patch represents a filter. The number below each patch is the variance of the weights of that filter.

C. FILTER PRUNING BY VARIANCE IN NETWORK

To further reduce the computation of the tracking algorithm, we statistically analyzed the filters. It is found that the internal weights in some trained filters may have little difference. We suggest these filters could contribute little to features extraction. Thus, these filters can be pruned to reduce network size and computation further.

To give an intuition of this idea, we visualized some filters in Fig. 3. Twenty filters were randomly selected from the Con2_1 layer and visualized after normalization. Each patch represents a filter. The filters are sorted according to their variance. From Fig. 3, we find that the first and second filters do not have a significant pattern. They are not able to extract discriminative features from the image. They will contribute little to the final object tracking.

The variance of these weights can evaluate their importance. Fig. 3 also shows that the weights variance without a significant pattern is very small, near to zero. In theory, the larger the filter variance, the more important the filter is. Thus, the filter with smaller variance should be pruned first.

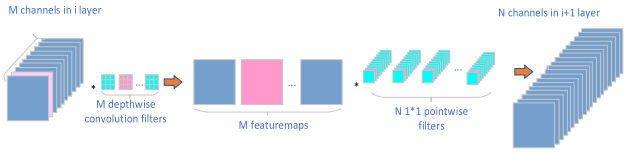

FIGURE 4. The filters pruning in depthwise convolution.

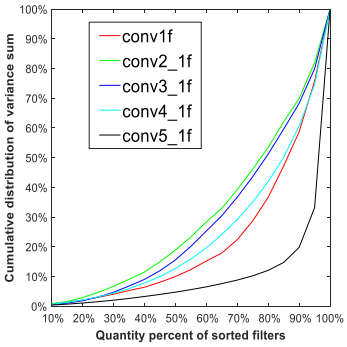

FIGURE 5. The variance distribution of filter among different layers.

In the separable convolution block, the filter is pruned in the depthwise convolution, which is a $D_K \times D_K \times 1 \times M$ dimension tensor. The third dimension of the tensor is one. It is more convenient to calculate the variance in a single channel rather than many channels.

Fig. 4 illustrates the filter pruning process. Pruning the pink filter with small variance in the depthwise convolution, the input feature map for the depthwise convolution will be removed accordingly. Moreover, in the subsequent 1×1 pointwise convolution, the number of channels in the 1×1 pointwise filters will be reduced due to the reduction of input feature maps.

The hyper-parameter in this method is the pruning ratio, which means how many filters should be pruned. As the variance distribution for each convolution layer is different, it is an inadvisable strategy to prune the whole network with a fixed pruning ratio. To better illustrate this fact, we show the variance distribution of filter weights after normalization in each layer in Fig. 5.

We sort the filters in ascending order by variance in each layer. The horizontal axis of Fig. 5 indicates the quantity percentage of sorted filters in each layer. The vertical axis of Fig. 5 indicates the cumulative distribution of the variance sum in each layer. The different color lines represent different convolution layers. Several phenomena are shown in Fig. 5.

Firstly, the filters variances vary significantly in a layer. There is no linear relation between the filters number and their variances sum. The top 40% filters, which are sorted in variance ascending order, only account for about 10% of the total variance in every layer. It shows that many filters have small weights variance.

Secondly, there are differences in the filter variance distribution among different layers. For con2_1 layer, the top 40% filters account for 10% of its total variances. However, for the con5_1 layer, the top 50% filters account for 10% of its total variances. It is obvious that the con2_1 layer has more discriminative filters than con5_1 layer has.

The variance distribution of each filter layer is different, so it is not a good choice to use a fixed pruning ratio. We introduce a hyper-parameter α to control the pruning ratio. Suppose that there are n filters in a convolution layer, we need to prune m filters. We rank n filters in ascending order of variance. The smallest m filters should be pruned firstly. In (4), we set a constraint that the sum of m filters variance is not larger than $\left(\sum_{i=1}^{n} var_i\right) \times \alpha$.

$$\sum_{i=1}^{m} var_i \leq \left(\sum_{i=1}^{n} var_i\right) \times \alpha$$
$$0 \leq \alpha \leq 1 \tag{4}$$

$\sum_{i=1}^{n} var_i$ is the sum of all filter variances in this layer. With the hyper-parameter α, more filters can be pruned in more redundant convolution layer and fewer filters can be pruned in less redundant convolution layer.

Note: Because the variance of many filters in each layer is small, many filters should be pruned, even if the hyper-parameter α is small. For example, as illustrated in Fig. 5, if the hyper-parameter α is equal to 0.10, 40% filters are pruned in the con2_1 layer and 50% filters are pruned incon 5_1 layer.

IV. EXPERIMENT

To verify our approach, we implemented several sets of contrasting tracking experiments. Our approach can greatly reduce the computational complexity in the network without significant degradation of tracking performance.

A. EXPERIMENT SETUP

In the training process, we used the ImageNet Video data set as the training set, which contains 4000 videos. The ImageNet Video data set has not only more samples, but also more diversity and scenes. The similarity loss between the prediction and ground truth is optimized by a standard SGD algorithm in training. The parameters are initialized with Gauss distribution, scaled according to the improved Xavier method [24]. We implement 50 training epochs. Each epoch contains 50,000 samples. Negative samples are chosen from different videos, accounting for 25% of the total samples. The minibatch size is 8. The learning rate decreases evenly from 10^{-2} to 10^{-4} according to the number of epochs.

To ensure experimental conditions consistent with the original Siamese network, we also use the 2015 version of Visual Object Tracking (VOT) benchmark in our experiment. VOT data set is a tracking platform for single target. Since 2013, VOT has become a main platform of single target tracking. VOT dataset not only has large-scale samples but also has a comprehensive evaluation for occlusion, deformation, illumination, low resolution, motion blurring and other factors in video tracking.

All experiments were performed on a computer equipped with Nvidia TITAN XP GPU. Our tracker is trained with MatConvNet [25] in MATLAB. Since TensorFlow supports separable convolution better, we import the trained weights into TensorFlow for testing.

B. EVALUATION METRICS

We use the following metrics to evaluate the tracking performance:

Accuracy – measured with the average intersection-over-union (IoU) between the predicted bounding box and the ground truth in a single video sequence. It takes the value between zero and one. One means complete overlap and zero means no intersection at all.

Failure rate – an average ratio of failed frames in each video. With the same failed frames, the shorter the video is, the higher the failure rate is. The failure rate is used to measure tracking robustness. The failure is deemed to have occurred when IoU between the estimated bounding box and the ground truth becomes zero. If the tracking fails, the tracker will be reinitialized after five frames in VOT. The failure of the tracking may be due to illumination or occlusion. In such scenarios, if the tracker is initialized immediately, it may increase the failure probability.

Overall performance – evaluated using Expected Average Overlap (EAO) which accounts for both accuracy and robustness. EAO [26] is used to measure the expected average IoU of the predicted region and the ground truth without re-initialization. Since EAO comprehensively measures the accuracy and robustness of trackers, it is the most important indicator for evaluating trackers.

Million Paras – the number of parameters of a CNN branch. Two CNN branches share the same parameters.

Million Mult-Adds – the total amount of calculations of two CNN branches. As the input images size is different in two branches, calculations of the two branches are different.

C. EXPERIMENT RESULT

To validate our approach, we have implemented the following experiments on a variety of network architectures.

- Network based on standard convolution. We download the network model from the GitHub website https://github.com/bertinetto/siamese-fc.It is a baseline experiment.
- Network based on the separable convolution, where the separable convolution only replaces standard convolution. It illustrates the contribution of separable convolution to network complexity reduction.
- Network based on separable convolution and filter pruning. In addition to replacing standard convolution with separable convolution, the filter is pruned according to hyper-parameter $\alpha = 0.01$. It shows the contribution of filter pruning to network complexity reduction. Experiments show that the hyper-parameter α is 0.01, which can trade off the performance and computation cost. Fig. 8 illustrates its tracking snapshots on VOT benchmark.
- Network based on separable convolution and 10% filter pruning at random in each convolution layer. The number of its filters pruned is the same as that of the network with hyper-parameter α equals 0.01. It demonstrates the effectiveness of our pruning approach.

TABLE 2. Our tracker compared with baseline on VOT benchmark.

Network	Accuracy	Failure	EAO	Million Paras	Million Mult-Adds
base net	0.54	2.05	0.2472	2.334	3318.13
separable conv net	0.53	1.76	0.2435	0.417	935.91
separable conv net with $\alpha = 0.01$	0.55	2.34	0.2466	0.317	790.06
separable conv net with random pruning	0.48	3.49	0.1446	0.350	817.82

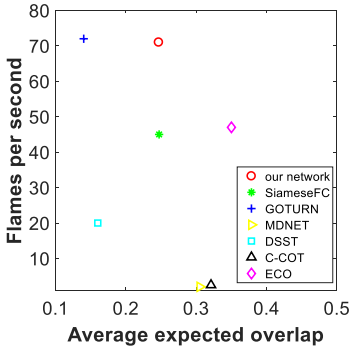

FIGURE 6. Our tracker compared with mainstream tracking network based on deep learning with Nvidia TITAN XP GPU on VOT 2015 benchmark.

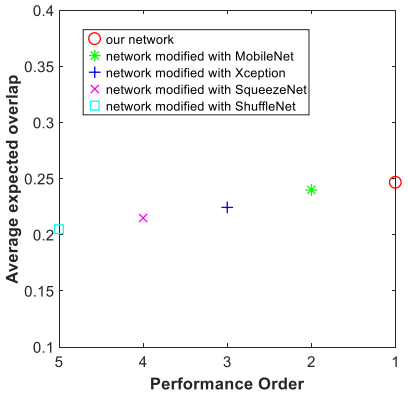

FIGURE 7. Our tracker compared with state-of-the-art accelerating methods on VOT 2015 benchmark.

After the filters pruning, the remaining weights in the network are used as the initial values for retraining. We remove the filter and related feature maps to be pruned directly in MATLAB to avoid their forward and backward calculation.

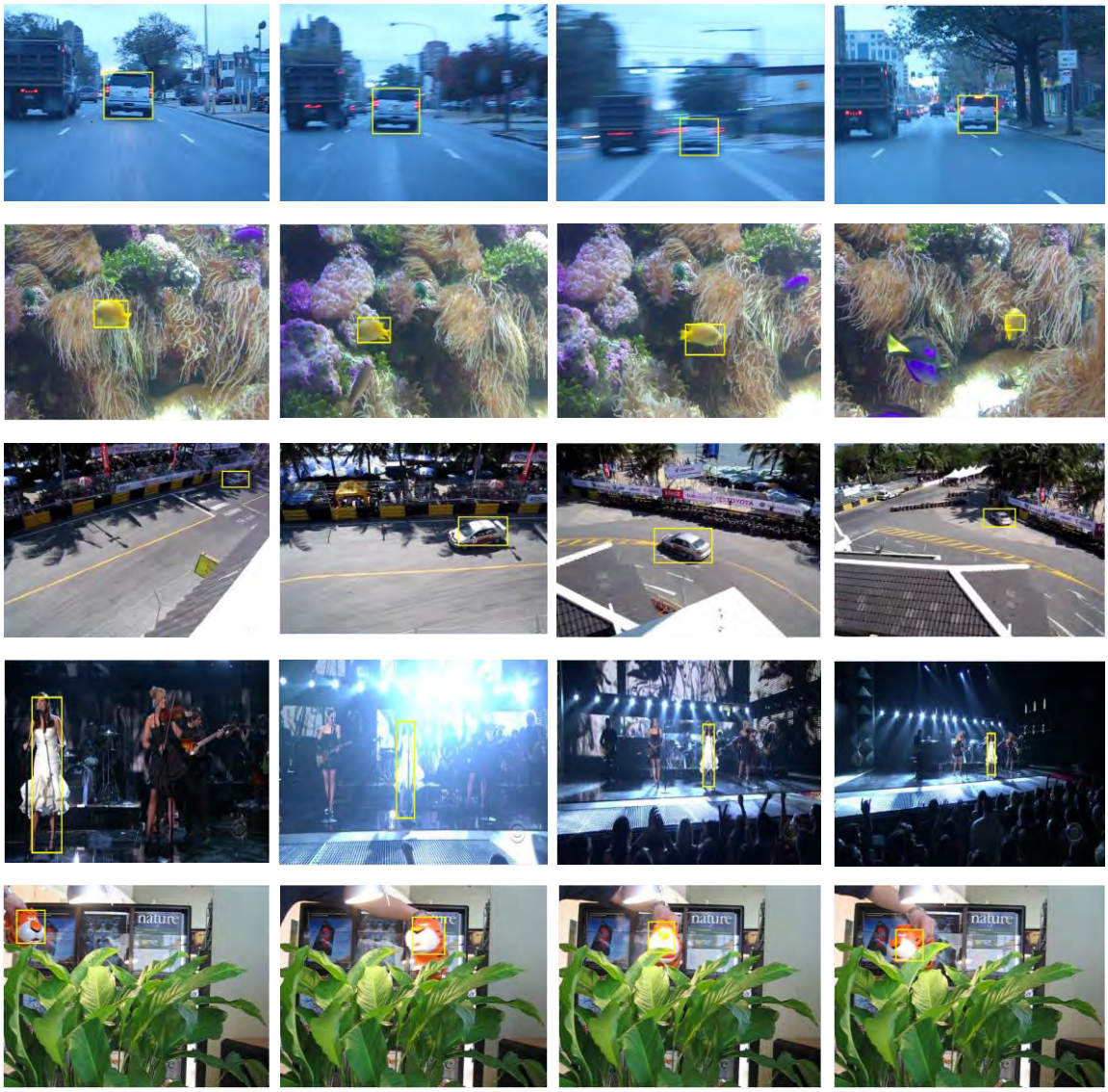

FIGURE 8. Snapshots of tracker described in Section 4.3, which based on separable convolution and filter pruning hyper-parameter α is 0.01. All sequences come from the VOT2015 benchmark: Car1, fish3, racing, singer1 and tiger. Four tracking snapshots are evenly sampled from the video sequences, which the trackers are not re-initialized. The scenarios include camera blur (row 1), motion change (rows 2 and 3), poor illumination (row 4), occlusion (row 5) and scale change (row 3 and 4). Our tracker withstands the extreme scenes test. It achieves good performance without tracker reinitialization.

As shown in Table 2, the third results achieve a good trade-off between tracking performance and network efficiency. The number of network parameters decreases to 13% and the amount of computation decreases to 23%, without significant degradation in the overall performance. The fourth experiment shows that the random filter pruning leads to poor performance.

To demonstrate our approach in speed, we compared our approach with several state-of-the-art tracking networks. We import the trained network weights into TensorFlow platform. Compared with the SiameseFC baseline, our tracker speed is much improved. As shown in Fig. 6, our tracker speed almost reaches the GOTURN, which is a very fast tracker based on deep learning. However, in terms of accuracy, our tracker's performance is twice that of GOTURN. Our network has more advantages over other methods in speed.

In addition, we demonstrate our approach on NVIDIA Jetson TX2. NVIDIA Jetson TX2 is an embedded system-on-module (SoM) with dual-core NVIDIA Denver2 + quad-core ARM Cortex-A57, 8GB 128-bit LPDDR4 and integrated 256-core Pascal GPU. It is ideal for intelligent edge devices like robots, drones, smart cameras, and portable devices. The experiment is implemented with Tensorflow platform. In the experiment, the parameters size and the amount of calculation are the same as on TITAN XP GPU, so the EAO has not changed. We set the "MAX-N" mode in NVIDIA Jetson TX2. As shown in Table 3, the excellent acceleration

TABLE 3. Our tracker comparison with NVIDIA Jetson TX2 on VOT benchmark.

Network	EAO	Million Paras	Million Mult-Adds	FPS (cpu)	FPS (gpu)
base net	0.2472	2.334	3318.13	0.98	6.59
separable conv net	0.2435	0.417	935.91	3.21	12.4
separable conv net with $\alpha = 0.01$	0.2466	0.317	790.06	3.58	13.72

TABLE 4. Performance different hyper-parameter.

Hyper-para α	Accuracy	Failure	EAO	Million Paras	Million Mult-Adds
0.01	0.55	2.34	0.2466	0.317	790.06
0.02	0.54	2.54	0.2203	0.254	707.15
0.05	0.53	2.45	0.2169	0.190	617.56
0.10	0.53	3.14	0.2010	0.116	494.49
0.20	0.52	3.13	0.1942	0.061	360.86

performance has been achieved on the Jetson TX2. For mobile devices with limited resources, our improvements lead to greater acceleration ratios.

Compared with the benchmark, the computation in our network has decreased by four times. The CPU's acceleration ratio is nearly four times. However, the speed has not increased by four times on GPUs. Qin *et al.* [27] believe the exchange frequency in memory is higher than the traditional convolution because of the depthwise convolution kernel low reuse rate. Besides, the matrix operation in depthwise convolution is very small, so it is not easily fully parallelized.

D. DIFFERENT HYPER-PARAMETER α

More experimental details can be referred to in Table 4. We retrain the network to restores performance after pruning. The larger the hyper-parameter α, the greater the potential loss on the performance. We use experimental validation to determine the hyperparameter α at present. We find that the network can guarantee its performance when hyperparameter α equals 0.01.

Therefore, in practical application, it is necessary to make a trade-off between performance and efficiency. If the speed is a priority, such as on mobile devices, which are sensitive to network size and computational complexity, the larger α can be chosen with more performance degradation.

E. DIFFERENT NETWORK ACCELERATION METHODS COMPARISON

We have compared our approach with state-of-the-art accelerating methods, such as SqueezeNet, Xception, ShuffleNet, MobileNet. We use a series of simplified convolution structures to transform the traditional CNN branches, which is responsible for feature extraction. Fig. 7 shows the performance of the networks using various acceleration method.

The tracking based on Siamese CNN is a correlation comparison between the object feature map and each frame feature map. We use the maximum response area on the feature maps to determine the object position in the video frame. Therefore, it is better to use the same size filters in convolution. In the SqueezeNet, Xception, and ShuffleNet, we use different size kernel among channels in the same convolution layer to get more scale information. It will result in the object position change between different channels in a convolution. This positional offset has no significant impact on the classification network. However, in the SiameseFC network architecture, it is possible to cause inaccurate object position.

V. CONCLUSION

With the application of deep learning in object tracking, the millions of parameters and huge computation in CNN are a challenge for tracking performance. In this paper, the standard CNN network is improved by separable convolution and filters pruning. With our approach, the number of parameters is decreased to 13% of the original Siamese network and the computation is reduced to 23%. On the NVIDIA Jetson TX2, the tracking speed increased to 3.65 times on the CPU and 2.08 times on the GPU, without significant degradation of tracking performance in VOT benchmark. As this paper mainly optimizes CNN of feature extraction, our approach also can be applied to other deep learning-based tracking algorithms. Deep learning-based tracking algorithms can be more convenient to deploy on devices with limited resources, such as mobile terminals.

REFERENCES

[1] L. Sevilla-Lara and E. Learned-Miller, "Distribution fields for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 1910–1917.

[2] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 125–141, 2008.

[3] S. Chan, X. Zhou, and S. Chen, "Robust adaptive fusion tracking based on complex cells and keypoints," *IEEE Access.*, vol. 5, pp. 20985–21001, 2017.

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

[5] 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.

[6] 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.*, vol. 9909, 2016, pp. 472–488.

[7] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6638–6646.

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

[9] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 749–765.

[10] 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.*, vol. 2016, pp. 850–865.
[11] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-end representation learning for correlation filter based tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2805–2813.
[12] H. Fan and H. Ling, "Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5486–5494.
[13] 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. (CVPR)*, Jun. 2018, pp. 8971–8980.
[14] A. He, C. Luo, X. Tian, and W. Zeng, "A twofold siamese network for real-time object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4834–4843.
[15] S. Lee and E. Kim, "Multiple object tracking via feature pyramid siamese networks," *IEEE Access*, vol. 7, pp. 8181–8194, 2018. doi: 10.1109/ACCESS.2018.2889442.
[16] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," 2017*arXiv:1602.07360*. [Online]. Available: https://arxiv.org/abs/1602.07360
[17] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2017, pp. 1800–1807.
[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: https://arxiv.org/abs/1704.04861
[19] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 6848–6856.
[20] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*. [Online]. Available: https://arxiv.org/abs/1608.08710
[21] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.
[22] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2016, *arXiv:1611.06440*. [Online]. Available: https://arxiv.org/abs/1611.06440
[23] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network Trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016, *arXiv:1607.03250*. [Online]. Available: https://arxiv.org/abs/1607.03250
[24] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[25] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2015, pp. 689–692.
[26] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, R. Pflugfelder, A. Gupta, A. Bibi, A. Lukezic, A. Garcia-Martin, A. Saffari, A. Petrosino, and A. S. Montero, "The visual object tracking VOT2015 challenge results," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop (ICCVW)*, Jul. 2016, pp. 564–586.
[27] Z. Qin, Z. Zhang, D. Li, Y. Zhang, and Y. Peng, "Diagonalwise refactorization: An efficient training method for depthwise convolutions," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2018, pp. 1–8.

YUANHONG MAO received the B.S. degree in computer science from the Xi'an Jiaotong University, in 2004, and the M.S. degree in computer science from Xidian University, Xi'an, China, in 2009. He is currently pursuing the Ph.D. degree in computer science with the Xi'an Microelectronics Technology Institute, Xi'an.

His research interests include computer vision, machine learning, and artificial intelligence.

ZHANZHUANG HE received the Ph.D. degree in computer science from the Xi'an Microelectronics Technology Institute, Xi'an, China, in 2006, where he is currently a Professor and a Doctoral Supervisor. He is also an Adjunct Professor of the Xi'an University of Technology and Xi'an Technological University. His research interests include embedded system architecture, computer operating systems, and computer control.

ZHONG MA received the Ph.D. degree in aeronautical and astronautical science and technology from Northwestern Polytechnical University, China, in 2015. He currently holds a postdoctoral position with the Xi'an Microelectronics Technology Institute, Xi'an, China. His current research interests include computer vision, machine learning, and artificial intelligence.

XUEHAN TANG is currently a Professor and Master Supervisor with the Xi'an Microelectronics Technology Institute. Her research interests include embedded computer architecture and computer operating systems.

ZHUPING WANG received the M.S. degree from the Xi'an Microelectronics Technology Institute, in 1989. He is currently a Professor and Master Supervisor with the Xi'an Microelectronics Technology Institute. His research interests include high performance computing and redundant systems.