IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Early Exiting-Enabled Siamese Tracking for Edge Intelligence Applications

Mohammad Fahd Hussein[1], and Serkan Özbay[1]

[1]Department of Electrical and Electronics Engineering, Gaziantep University, Gaziantep 27310, Turkey

Corresponding author: Mohammad Fahd Hussein (e-mail: fahd.hus@live.com).

ABSTRACT Visual object trackers based on deep neural networks have attained state-of-the-art performance in recent years. Despite the outstanding accuracy gained by deep layers, however, they also demand high computational cost and energy consumption in order to operate in real-time, making them inadequate for edge and latency-sensitive applications. In this paper, we propose an edge computing-friendly Siamese-based visual object tracker. This work concentrates on increasing the tracking speed by reducing computations through integration of side exit branches into the network, as well as skipping the multi-scale search for some frames. By employing exit branches, the tracker is capable of obtaining the result of easy samples from early layers once the criteria are satisfied. The network is trained offline to optimize a joint function that is composed of the weighted loss functions of all exit branches. During inference, the score map is derived from the network and determines the new object location, whereas multi-scale testing can identify scale updates which is only applied under specific conditions. Our proposed tracker deploys an adaptive scale search over two scales that runs at 247.5 FPS on GPU and 37.1 FPS on CPU providing a 2.5x faster rate of processing speed compared to SiamFC, with an acceptable amount of accuracy loss, especially when compared to the significant speed gain and gains in computational efficiency.

INDEX TERMS Deep Learning, early exit, edge computing, real-time tracking, Siamese network, visual tracking.

I. INTRODUCTION

Visual Object Tracking is one of the fundamental components in computer vision with a diverse range of applications including autonomous driving, surveillance, and robotics. Object tracking aims to locate a target efficiently in all subsequent frames despite any variations in conditions or through any disturbances. Moreover, generic object trackers are capable of tracking any object regardless of its class or its appearance in the training dataset.

As object tracking is an active research topic, many studies have proposed solutions to improve accuracy and robustness in order to solve common challenging problems related to object appearance such as deformation, occlusion, camera motion, and speed and scale variations [1]–[3]. With the remarkable success that has come through the application of deep neural networks to computer vision tasks, recent Siamese-based object tracking approaches [4]–[9] have introduced new state-of-the-art performance. A Siamese network [10] is a deep neural network whose architecture solves tracking problems through its ability to compare and identify similarities between two inputs: templates of the target object and images retrieved from subsequent frames. On the other hand, deep learning networks require more robust computational and memory resources, which also increases power demands.

With the increasing use of smart cities and internet of things (IoT) applications, processing speed and computational cost have become almost as essential as performance [11]. Cloud computing is one of the common solutions used to overcome hardware limitations [12]. In cloud computing, data are processed remotely on a processing device with sufficient computational resources that typically utilize powerful GPUs for visual tasks processing. However, the required communication between device and server carries various potential risks related to data security and privacy, network coverage, band width, and response time [13]. In many applications, particularly where real-time response is critical, latency or network loss can be detrimental to performance. It is for this reason that computation should be pushed to the edge near the sensor, a process called edge computing. In edge

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3119604, IEEE Access

computing, data are processed on a device locally with no need for any data transfer to the cloud server. This avoids any risks associated with cloud computing. However, edge computing requires greater attention to computational resources, power consumption, and storage [13].

Recently, more effort has been expended to boost the tracking speed and increase accuracy. Modern correlation filters, and Region Proposal Network (RPN) [14] based tracking approaches [5], [8], [9] have introduced impressive performance improvements with a response time beyond the real-time on GPU. Although recently proposed tracking methods operate at high framerates, the computational complexity remains extremely high for many low-power devices operating at the edge.

In this work, we attempt to propose a CPU-friendly generic object tracker based on deep Siamese networks that allow a viable trade-off between accuracy and speed. First, we train a Siamese Fully Convolutional (SiamFC) [7] based tracker with the modified AlexNet backbone. This is done after adding multiple exit branches to the network in order to terminate the inference early at the point when accuracy is determined to be adequate. At each exit point, the model compares the highest score from a heatmap with a corresponding threshold in order to decide when to exit. In order to retain the lowest computational cost, we use fixed thresholds for comparison. Second, we validate the accuracy at every exit point using OTB2013 dataset [15] and then tune the exiting thresholds, evaluate the joint model on OTB2015 [16] and VOT2018 [2] datasets, and compare the results with SiamFC (our baseline). Finally, we introduce a simple adaptive multi-scale test method which only uses a single scale unless the score goes below a pre-defined threshold. Skipping multiple scales tests whenever possible allows for extreme reduction in computational costs but can also introduce reductions in accuracy as well.

Our Early Exiting-Enabled Siamese tracker (SiamEE) can operate at an average speed 247.5 FPS on GPU and 37.1 FPS on CPU making it 150% faster than SiamFC.

The rest of this paper is organized as follows. Section II gives an overview of literature on related works. In Section III, we introduce a description of early exiting-enabled network architecture for Siamese object tracking describing our training and tracking approaches including our scale-search skipping method. Section IV provides the implementation details of our tracker. The results of experiments are presented in Section V. Finally, Section VI concludes the paper.

II. LITERATURE REVIEW

A. SIAMESE-BASED OBJECT TRACKING

Results of recent visual tracking challenges [1]–[3] have demonstrated improved performance with the wide use of Siamese-based trackers which are among the top trackers along with trackers based on discriminative correlation filters (DCF).

The recent state-of-the-art performance of Siamese-based trackers has attracted growing attention, making them an active research topic. Siamese-based trackers predict target location by checking feature similarities between object templates and search regions where both branches share the same parameters. Held et al. [17] proposed a regression-based generic object tracker GOTURN. The high speed of the GOTURN tracker is due to the direct bounding box predictions it can perform in a single feed-forward pass, which is achieved by adding fully connected layers after the two feature extraction branches. Gordon et al. [18] presented a recurrent neural network-based tracker Re3 to increase accuracy and efficiency. Bertinetto et al. [7] proved the advantage of using cross correlation to improve performance and speed with his introduction of the SiamFC tracker. In this tracker, features of the template image and search image are cross-correlated to obtain a heatmap. The highest heatmap score indicates the target location. The tracker predicts the target scale by scanning multiple scales of image, where the heatmap with the highest score points to the target scale. Then, CFNet [6] developed a modified network structure, adding a correlation filter into the template branch thus achieving high performance using a shallower network. In contrast to SiamFC, CFNet recomputes the template after each frame instead of relying on comparisons with the initial template. Although the SiamFC and CFNet run beyond real-time, they both use a multi-scale test method that requires higher computational demand that limits tracker speed.

SiamRPN [8] overcame this negative effect of multi-scale testing by introducing region proposal subnetwork (RPN) [14] after the features' extraction layers. The region proposal subnetwork consists of a foreground-background classification branch and a proposal regression branch. SiamRPN achieved state-of-the-art performance while operating at 160 FPS. The subsequent architectures proposed by Zhang et al. [9] and Li et al. [5] showed a significant performance improvement using deeper networks. This improvement however comes at the cost of computational power and speed.

Xu et al. [4] introduced SiamFC++ based on proposed guidelines to obtain a high-performance tracker while running over 160 FPS with an AlexNet backbone. Similar to SiamRPN trackers [8], SiamFC++ added classification and regression branches after the cross-correlated features but, moreover, it discarded the need of pre-defined anchor boxes. Also, SiamFC++ introduced the quality assessment branch following the guidelines in order to increase accuracy.

Huang et al. [19] introduced a high-speed adaptive tracker by implementing an early stopping method based on reinforcement learning. In addition to the early stopping approach, the authors added both pixel and HOG layers in order to process cheap features at high speed before proceeding to the deeper layers. Also, the scale estimation is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3119604, IEEE Access

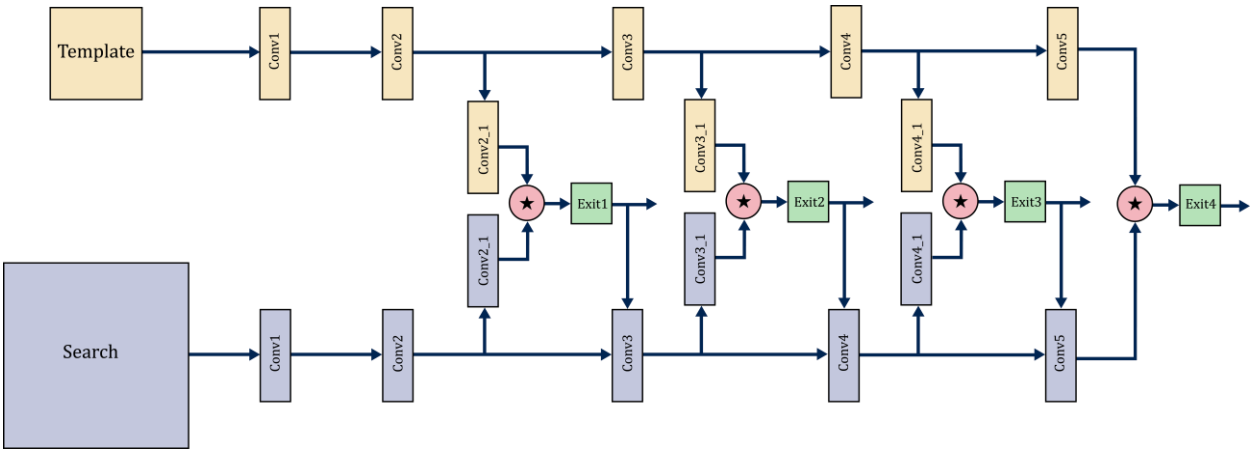

FIGURE 1. **The architecture of our SiamEE tracker which consists of two main branches and four exits.**

computed during the forward-pass using heatmaps, which overcomes the high computation threshold required for estimation using multiple scales tests. The tracker achieved 23 fps on a single CPU at a high level of performance.

B. DEEP NETWORK ACCELERATION

The high computational costs associated with the impressive success of deep neural networks have attracted more attention to model compression and acceleration techniques. These include such techniques as early exiting, knowledge distillation, pruning, and quantization [20].

Early exit approaches increase inference speed and thus save more energy by performing predictions of easier samples at earlier layers. After adding exit branches in specific locations of the deep network, the inference starts with earlier layers, and at each exit point the prediction result will be evaluated under defined criteria in order to decide whether the result is satisfying. After this evaluation, the inference either stops or proceeds to deeper layers. Using this architecture, BranchyNet [21] showed a remarkable increase in speed on both CPU and GPU, with accuracy comparable to many popular architectures. Later researches [22], [23] proposed learned policies for early exiting that automatically decide when to exit without using manually defined thresholds in order to maintain the accuracy.

III. THE PROPOSED SIAMESE TRACKER

In this section, we review the fully convolutional Siamese architecture as our proposed SiamEE tracker is based on SiamFC [7] which we modified to allow for the model to exit early when applicable. As shown in Fig. 1, we introduce three exit branches after each convolutional layer, beginning at the second layer and yielding four exit points including the main branch exit.

A. FULLY CONVOLUTIONAL SIAMESE NETWORK FOR OBJECT TRACKING

Comprising of two branches, Siamese networks can take a pair of images as inputs to predict an object location by comparing the similarities between a template patch and a search patch. The template image (denoted as z) represents the targeted object, an image which is generally taken from the first frame of the video and centered on the object, whereas the search image (denoted as x) is typically taken from subsequent frames which is larger than the template image and centered at the last object position. In both branches, the convolutional layers share the same parameters and are used to extract the features of the inputs. The feature representations are then cross-correlated to obtain the score map.

$$f_b(z,x) = \phi_b(z) \circledast \phi_b(x), b \in \{1,2,3,4\} \quad (1)$$

where f_b is the score map of branch b obtained by cross-correlating $\circledast$ the two feature representations ϕ_b of a template patch z and a search region x at branch b.

The highest value of the score map corresponds to the object location, which is normally close to the center as the displacement is generally small between two consecutive frames.

B. ARCHITECTURE

The main embedding backbone architecture matches the original version employed by SiamFC, which is a modified version of AlexNet. Inspired by BranchyNet [21], SiamEE introduces three additional exit branches to the network. As shown in Fig. 1, the exit branches are located after every convolutional layer except for the first layer. Experimentally, we found that the first layer has less ability to represent good features for tracking. Moreover, the score map obtained from the first layer has a larger size which needs to be down sampled to output size. The benefit of this early branch does not reasonably satisfy the demands of additional computational cost. Thus, we do not attach an early exit branch directly after the first layer.

Fig. 2 shows the number of parameters utilized by each exit branch and also compares them to the number of the baseline's parameters. The parameters of the overall network are increased by 12% more than the baseline's parameters due to the additional convolutional layers of exit branches.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3119604, IEEE Access

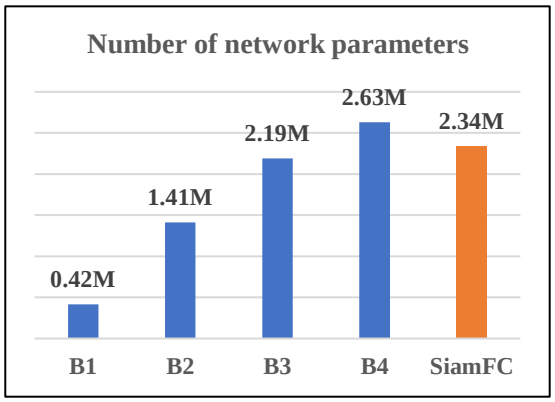

FIGURE 2. Comparison between number of the backbone parameters of each exit branch and the baseline.

C. TRAINING WITH EXIT BRANCHES

Using (1), we obtain four different response maps corresponding to the four exit branches. Since the objective is to minimize the logistic loss function of every exit score map, the network is trained offline to optimize a joint function:

$$\mathcal{L} = \sum_{b=1}^{4} w_b L_b(f_b, y) \quad (2)$$

This loss function $\mathcal{L}$ of the branchy network is the sum of the logistic loss of every branch output after weighting it by w_b. L_b is the logistic loss where y is a labels map with +1 values in the true object location and -1 otherwise. Since the training images are centered on the object, the values of the labels map are +1 if they exist within a specific radius of the center and -1 outside that region.

D. TRACKING ALGORITHM

Beginning with the closest exit branch to the input, the score map of the branch is obtained. Then the maximum score is compared to a pre-defined threshold γ_b. If this score is less than γ_b, then the score map is returned as a final output and the inference stops. Otherwise, the process continues all the way to the last point with examining the corresponding output at every exit branch successively. For a single sample, the output of each convolutional layer is computed only once and fed to next layers if the exit criteria were not satisfied.

Since our goal is to achieve reasonable accuracy without seeking extra computational overhead, we do not incorporate any dynamic rule for early exiting. Instead, we define fixed thresholds and tune them after the training is over. Moreover, these manually designated thresholds come in handy for adjusting the accuracy-speed trade-off.

During online tracking, the search image is compared to the initial template image, which is not updated in each frame, which means that the feature representations of the template image are computed only once for all exit branches and then used for all subsequent frames. This results in fast inference.

After attaining the score map, the object displacement is calculated by multiplying the network stride by the maximum score position with respect to the center, after applying cosine window to the response map in order to alleviate the effects of the boundary discontinuity, assuming that the displacement is small from frame to frame.

The exhaustive scale search strategy is another key source of expensive computations. Since the purpose of this study is to reduce computational cost while permitting a reasonable accuracy loss, we modified the scale search method by introducing a simple skipping rule. Skipping the scale search offers significant computational savings due to the fact that the scale difference can be negligible in frequent frames in many scenarios, particularly when running at high speed. As our model does not provide prediction confidence, we use the maximum value of the predicted score map instead. A greater score indicates a higher confidence. To decide whether we can skip the multi-scale search in the next frame, we observe this score value. If the score is decreasing, we then apply multi-scale search in the next frame. In this work, we only compared the score of the current frame with the previous score. However, there are other factors influencing the score, like the object appearance and occlusion. Using history vector holding of the most recent scores, instead of employing a single score for comparison, can better represent the score change.

IV. IMPLEMENTATION DETAILS

A. TRAINING PHASE

Our training procedure is similar to [7]. We use a modified AlexNet as a backbone. We then obtain its parameters by minimizing (2) with SGD, with a momentum of 0.9. The parameters are initialized by the improved Xavier method. The network is then trained for 50 epochs using batch size 8 with a learning rate decaying with a fixed factor at each epoch from 10^{-2} to 10^{-5}. The training set is ImageNet Video [24] which contains 4417 videos. The training set is curated to obtain template images with size of 127×127×3 and test images with size of 255×255×3 in which the target is centered.

We selected the weights w_b of (2) to be 0.3,0.4,0.5,1 for each branch respectively starting from the closest branch to the input. While calibrating our tracking problem, we discovered that granting a high weight to earlier exits adversely affects tracker accuracy.

After training, we then tune the exit thresholds γ_b using a simple method. First, we establish three special variants of the network, each of them comprised of a targeted branch, that we tune its threshold, and the last branch. In other words, the three formed variants only contain two exit branches: the first and last branches, the second and last branches, and, finally, the last two branches. Then we screen over γ_b by evaluating tracker performance under different values. The screening is performed using OTB2013 dataset. Lastly, we select the thresholds that satisfy our desired accuracy and speed. Although this method is time-consuming, it is only performed once after the training, so it does not affect the inference.

B. TEST PHASE

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3119604, IEEE Access

The output of our network is a score map with a size of 17×17. We upsample it to 272×272 and use the highest score to calculate the translation after penalizing large displacements by applying a cosine window on the score map.

To demonstrate the effects of the scale search method, we implemented four variants of our tracker: SiamEE (Siamese with Early Exiting) with exhaustive search over 3 scales {0.95, 1, 1.05}, SiamEE-3AS with adaptive search over 3 scales {0.95, 1, 1.05}, SiamEE-2AS with adaptive search over 2 scales {0.95, 1.05}, and No-EE-2AS which is early exiting-disabled with adaptive search over 2 scales {0.95, 1.05}. When searching over three scales, no-scaling is given a higher priority by multiplying the scores corresponding to the scaled object by 0.97. In adaptive scale search, no-scaling will be applied on the next frame if the current score is greater than the score threshold. This threshold is calculated by subtracting a small number $\beta = 0.3$ from the score of the previous frame. The early exiting approach is applied to the branchy trackers using the value 5 for all exit thresholds.

We evaluated our proposed tracker on a device with an Intel i7-10750H CPU, and NVIDIA GeForce GTX 1650 GPU.

V. EXPERIMENTS

We first performed our experiments on OTB2013 to compare the results at each exit branch independently. We then used a special variant SiamEE-2B which consists of the first and last exit branches only, so that we were able to study the effects of exit threshold γ_b. Finally, we tested our four variants on OTB2015 and VOT2018 benchmarks.

As the tracking speed is hardware-dependent, we also performed experiments on the baseline using the same resources. To achieve that, we re-implemented the tracking method of SiamFC-3s and used the originally submitted trained model. Additionally, we included the reported results of the baseline and another state-of-the-art tracker (SiamRPN) in our tables. Unless we declare that the result is reported, it refers to the results obtained by our implementation.

A. RESULTS ON OTB20 13

OTB2013 dataset [15] contains 50 videos for the evaluation of trackers. Fig. 3 shows the results of every exit branch individually in addition to SiamFC-3s (baseline). The precision plot (left) is based on the center location error, which is calculated as the average of the Euclidean distances between the center of the predicted bounding box and the center of the ground-truth of all frames of a sequence. The success plot (right) uses the average overlap of the predicted bounding box and the ground-truth which is drawn by generating threshold values from 0 to 1 then computing the percentage of overlap ratios exceed each threshold. We use precision scores as our accuracy metric, a score which is derived from predictions whose center location resides within a specific distance from the center of the ground-truth. OTB uses a distance threshold = 20 pixels for evaluation, so we report precision at point 20 as the precision score.

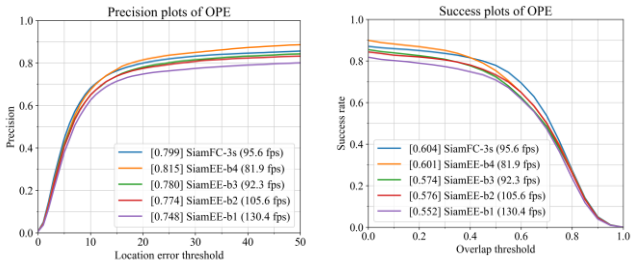

FIGURE 3. The results of the exit branches on OTB2013 compared with the baseline (SiamFC-3s). The precision plots are based on the center location error whereas the success plots represent the average overlap between the predicted bounding box and the ground-truth.

TABLE I
THE RESULTS OF SIAMEE-2B ON OTB2013 WITH DIFFERENT EXIT THRESHOLDS

γ	Success	Precision	Speed (FPS)
2	0.557	0.753	135.1
4	0.569	0.777	120.7
6	0.576	0.773	107.6
8	0.595	0.798	98.8

As shown in Fig. 3, the inference of earlier branches is performed with greater speed, but the accuracy is lower. The first exit branch is 59.2% faster than the last one, but its accuracy is about 8.2% lower. In comparison with the baseline, the last branch achieves the closest success and precision scores, but it operates at lower speeds due to the additional computations required for other exit branches.

Tracker performance using different exit threshold values is listed in Table I. Increasing the threshold encourages more samples to exit earlier which speeds up the inference at the cost of accuracy.

B. RESULTS ON OTB2015

We tested our tracker variants, SiamEE-3S, SiamEE-3AS, SiamEE-2AS, and No-EE-2AS, on the 100 videos of OTB2015 dataset [16] and compared the results with the original SiamFC tracker results. As shown in Table II, our proposed SiamEE-2AS tracker achieves the fastest inference running at more than 247 FPS, which is 160% faster than SiamFC, while our precision and success scores remain satisfying with 5.8% and 6.5% loss in each of them respectively. SiamEE-2AS also operates at real-time speed on CPU (37.1 FPS) making it CPU-friendly tracker.

The results of No-EE-2AS demonstrates that the skipping of the multi-scale search with no early exiting accelerated the tracking by 117.4% with only 2.3% loss in precision and 3% loss in success score. On the other hand, early exiting-enabled tracker with the exhaustive multi-scale search (SiamEE-3S) is able to run 20% faster than the baseline but with the cost of 3.9% loss in precision and 4.6% loss in success.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3119604, IEEE Access

FIGURE 4. Sample frames with the predicted bounding box of the tracked object are selected so that they have different difficulties. We have done this in order to illustrate the adaptive behavior of the early exiting method, which exits easier frames earlier. The shown images are picked from the KiteSurf sequence which are from left to right: frame 1 (the template image), frame 8 exited from branch 1, frame 25 exited from branch 2, frame 30 exited from branch 3, and frame 43 exited from the last branch. Sample is easier when the search region is more similar to the template image, because all sequence frames are compared to the template image. As shown, the object becomes harder to detect by proceeding towards the right image, so it exits at later branches.

TABLE II
THE RESULTS OF THE TRACKER VARIANTS ON OTB2015 WITH EXIT THRESHOLDS SET TO 5. WHERE BX% IS THE PERCENTAGE OF SAMPLES EXITED AT BRANCH X, AND SCALE SEARCHES% IS THE PERCENTAGE OF APPLYING MULTI-SCALE SEARCH

Tracker	Success	Precision	$FPS_{GPU/CPU}$	B1%	B2%	B3%	B4%	Scale Searches%
SiamEE-3S	0.557	0.746	114.2/18	58.41	8.11	11.65	21.82	100
SiamEE-3AS	0.544	0.729	236.1/34.4	55.19	7.78	11.39	25.64	26.23
SiamEE-2AS	0.546	0.731	247.5/37.1	55.87	7.84	11.16	25.14	26.47
No-EE-2AS	0.566	0.758	207/32.7	-	-	-	-	22.69
Baseline (SiamFC-3s)	0.584	0.776	95.2/14.9	-	-	-	-	100
Baseline (Reported)	0.582	0.771	86/-	-	-	-	-	100
SiamRPN (Reported)	0.637	0.851	160/-	-	-	-	-	-

TABLE III
THE RESULTS OF THE TRACKER VARIANTS ON VOT2018

Tracker	Accuracy ↑	Robustness ↓	EAO ↑	$FPS_{GPU/CPU}$
SiamEE-3S	0.486	0.754	0.171	99.64/17.18
SiamEE-3AS	0.474	0.754	0.165	177.99/30.88
SiamEE-2AS	0.482	0.768	0.167	185.26/31.65
No-EE-2AS	0.481	0.726	0.175	168.12/28.24
Baseline	0.473	0.642	0.199	89.25/14.89
Baseline (Reported)*	0.503	0.585	0.188	36.76/-
SiamRPN (Reported)	0.586	0.276	0.383	81.73/-

* SiamFC has two versions [6], [7] which differs in the network architecture. The second version of SiamFC is slower but has slightly higher accuracy. The baseline of our work is the first version of SiamFC whereas the recently reported results are based on the second version.

Additionally, the results show that more than 55% of frames are easy so that the first branch is capable of processing them. Sample frames with various difficulties are presented in Fig. 4. Object in a search region similar to the template region is easier to be located by a shallower network.

Table II also shows that our tracker needed to search over multiple scales only in less than 26.5% of the frames which extremely saves computations and thus boosts the inference. This result was due to the fact that the scale variation is low in most cases, and scale difference between consecutive frames when running at high speed is negligible as well.

C. RESULTS ON VOT2018

We evaluated our tracker variants on VOT2018 dataset [2] which includes 60 video sequences. The primary evaluation protocol of the short-term VOT2018 challenge is done through supervised evaluation in which the tracker is re-initialized whenever it fails. This tracking failure is detected when the overlap between the predicted bounding box and the ground-truth is zero. The tracker reset is performed after five frames from failure in order to reduce bias. The performance measures are: Accuracy (A) regarding the average overlap between successful prediction and the ground-truth, Robustness (R) depending on the percentage of failures, and Expected average overlap (EAO) with respect to both accuracy and robustness which is estimated by taking the average over all average overlaps of frames' segments. Segments are then extracted from sequences with respect to failures, then they are normalized to a specific length by either trimming or padding with zero. The results are reported in Table III.

As the tracker re-initializes the template after failure, the tracking speeds presented in Table III are lower than those in Table II due to the additional cost for re-computing the template.

Table III shows that our SiamEE-2AS has a close accuracy (1.9% better) compared to our re-implemented baseline, whereas the robustness loss is noticeable (about 19.6%).

VI. CONCLUSION

In this paper, we propose an approach that allowed us to significantly increase tracking speed while retaining reasonable accuracy. This was achieved by establishing a network architecture with multiple exits that enables the inference to stop at early stages, and applying a simple adaptive scale search method.

Our empirical evidence shows that plenty of frames are easy so that they can be detected at early stages alongside the negligible scale change between successive frames, which

allowed us to drastically reduce computation costs. We evaluated our tracker on OTB2013, OTB2015, and VOT2018 datasets and subsequently demonstrated that our approach can accelerate inference by 150% of our baseline with an accuracy loss less than 5%.

In this work, we concentrated on reducing computational costs as much as possible while permitting accuracy loss. However, future work should consider methods that maintain the robustness of the tracker. Our next research will be focused on exit thresholds which can be adaptive with careful attention to computational overhead. Additionally, the skipping method of the multi-scale search can utilize a history vector with a specific length, instead of relying on a single score for comparison. This history vector would hold the maximum values of the latest score maps, which could then be analyzed to predict skips more reliably.

REFERENCES

[1] M. Kristan *et al.*, "The Seventh Visual Object Tracking VOT2019 Challenge Results," *Proc. - 2019 Int. Conf. Comput. Vis. Work. ICCVW 2019*, pp. 2206–2241, 2019, doi: 10.1109/ICCVW.2019.00276.

[2] M. Kristan *et al.*, "The sixth visual object tracking VOT2018 challenge results," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11129 LNCS, no. 1, pp. 3–53, 2019, doi: 10.1007/978-3-030-11009-3_1.

[3] M. Kristan *et al.*, "The Eighth Visual Object Tracking VOT2020 Challenge Results," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12539 LNCS, pp. 547–601, 2020, doi: 10.1007/978-3-030-68238-5_39.

[4] Y. Xu, Z. Wang, Z. Li, Y. Yuan, and G. Yu, "SiamFC++: Towards Robust and Accurate Visual Tracking with Target Estimation Guidelines," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 07, pp. 12549–12556, 2020, doi: 10.1609/aaai.v34i07.6944.

[5] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "SIAMRPN++: Evolution of siamese visual tracking with very deep networks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 4277–4286, 2019, doi: 10.1109/CVPR.2019.00441.

[6] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-end representation learning for Correlation Filter based tracking," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 5000–5008, 2017, doi: 10.1109/CVPR.2017.531.

[7] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9914 LNCS, pp. 850–865, 2016, doi: 10.1007/978-3-319-48881-3_56.

[8] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High Performance Visual Tracking with Siamese Region Proposal Network," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 8971–8980, 2018, doi: 10.1109/CVPR.2018.00935.

[9] Z. Zhang and H. Peng, "Deeper and wider siamese networks for real-time visual tracking," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 4586–4595, 2019, doi: 10.1109/CVPR.2019.00472.

[10] G. Koch, "Siamese Neural Networks for One-shot Image Recognition," 2011.

[11] W. Yu *et al.*, "A Survey on the Edge Computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2017, doi: 10.1109/ACCESS.2017.2778504.

[12] T. Furuichi and K. Yamada, "Next generation of embedded system on cloud computing," *Procedia - Procedia Comput. Sci.*, vol. 35, pp. 1605–1614, 2014, doi: 10.1016/j.procs.2014.08.244.

[13] G. Plastiras, M. Terzi, C. Kyrkou, and T. Theocharidcs, "Edge Intelligence: Challenges and Opportunities of Near-Sensor Machine Learning Applications," *Proc. Int. Conf. Appl. Syst. Archit. Process.*, vol. 2018-July, 2018, doi: 10.1109/ASAP.2018.8445118.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[15] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: A benchmark," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2411–2418, 2013, doi: 10.1109/CVPR.2013.312.

[16] Y. Wu, J. Lim, and M. H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, 2015, doi: 10.1109/TPAMI.2014.2388226.

[17] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9905 LNCS, pp. 749–765, 2016, doi: 10.1007/978-3-319-46448-0_45.

[18] D. Gordon, A. Farhadi, and D. Fox, "Re 3 : Real-time recurrent regression networks for visual tracking of generic objects," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 788–795, 2018, doi: 10.1109/LRA.2018.2792152.

[19] C. Huang, S. Lucey, and D. Ramanan, "Learning Policies for Adaptive Tracking with Deep Feature Cascades," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, pp. 105–114, 2017, doi: 10.1109/ICCV.2017.21.

[20] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A Survey of Model Compression and Acceleration for Deep Neural Networks," pp. 1–10, 2017, [Online]. Available: http://arxiv.org/abs/1710.09282.

[21] S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," *Proc. - Int. Conf. Pattern Recognit.*, vol. 0, pp. 2464–2469, 2016, doi: 10.1109/ICPR.2016.7900006.

[22] X. Dai, X. Kong, and T. Guo, "EPNet: Learning to Exit with Flexible Multi-Branch Network," 2020, pp. 235–244, doi: 10.1145/3340531.3411973.

[23] X. Chen, H. Dai, Y. Li, X. Gao, and L. Song, "Learning to Stop While Learning to Predict," 2020, [Online]. Available: http://arxiv.org/abs/2006.05082.

[24] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015, doi: 10.1007/s11263-015-0816-y.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3119604, IEEE Access

MOHAMMAD FAHD HUSSEIN received the B.Sc. degree in electronic systems engineering from the faculty of Electrical and Electronic Engineering, Aleppo University, Aleppo, Syria, in 2013.

He is currently pursuing the M.Sc. degree in Electronic Circuits and Systems with the department of Electrical and Electronics Engineering, Gaziantep University, Gaziantep, Turkey. His research interests include embedded systems, edge intelligence, quantization, computer vision, and IoT.

SERKAN ÖZBAY received the B.Sc., M.Sc., and Ph.D. degrees in electrical and electronics engineering from Gaziantep University, Gaziantep, Turkey, in 2002, 2006, and 2015, respectively. From 2003 to 2016, he was a Lecturer with Gaziantep University, where he is currently an Assistant Professor. His research interests include computer vision, deep learning, machine learning with applications and IoT.