UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Mobile Object Tracking with Siamese Neural Network

Author:
Vasyl BORSUK

Supervisor:
Orest KUPYN

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

I, Vasyl BORSUK, declare that this thesis titled, "Mobile Object Tracking with Siamese Neural Network" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Mobile Object Tracking with Siamese Neural Network

by Vasyl BORSUK

Abstract

Visual object tracking is one of the most fundamental research topics in computer vision that aims to obtain the target object's location in a video sequence given the object's initial state in the first video frame. The recent advance of deep neural networks, specifically Siamese networks, has led to significant progress in visual object tracking. Despite being accurate and achieving high results on academic benchmarks, current state-of-the-art approaches are compute-intensive and have a large memory footprint that cannot satisfy the strict performance requirements of real-world applications. This work focuses on designing a novel lightweight framework for resource-efficient and accurate visual object tracking. Additionally, we introduce a new tracker efficiency benchmark and protocol where efficiency is defined in terms of both energy consumption and execution speed on edge devices.

Acknowledgements

I would like to express gratitude to Orest Kupyn for supervising this research project and providing valuable feedback. Many thanks to my colleague Roman Vei with whom we worked on the related research and who has made an invaluable contribution to this work. I am incredibly grateful to Ampersand Foundation and Alexander Kutovoy for granting the scholarship and supporting my studies. I want to thank Ukrainian Catholic University and the Faculty of Applied Sciences for organizing the Master's Program in Data Science. All experiments were run using computational resources generously provided by PiñataFarms. My deepest gratitude goes to my family for their constant motivation to finish the work. I express my sincere thanks to my friends and comrades for helping me out during the last six years of study and for being an inspiration. I am extremely grateful to the Armed Forces of Ukraine for providing security to complete this work. Without all of you, this work would never be finished.

Contents

List of Figures

List of Tables

List of Abbreviations

ANE	**A**pple **N**eural **E**ngine
AO	**A**verage **O**verlap
EAO	**E**expected **A**verage **O**verlap
FLOPS	**F**loating-**P**oint **O**perations **P**er **S**econd
FPS	**F**rames **P**er **S**econd
GOT-10k	**G**eneric **O**bject **T**racking benchmark
LaSOT	**La**rge-scale **S**ingle **O**bject **T**racking benchmark
NFS	**N**eed **F**or **S**peed benchmark
VOT	**V**isual **O**bject **T**racking challenge

List of Symbols

I_T	static template image
I_S	search image
I_d	dynamic template image
I_N	negative image
F_T	static template feature map
F_S	search feature map
F_d	dynamic template feature map
F_N	negative feature map
e_T	static template embedding
e_S	search embedding
e_d	dynamic template embedding
e_N	negative embedding
w	dual-template interpolation learnable parameter

Dedicated to the Armed Forces of Ukraine

Chapter 1

Introduction

Visual object tracking is one of fundamental research topics in computer vision. It aims to estimate the state of an arbitrary target in a video sequence, given only its location in the first frame. Visual object tracking has many applications such as autonomous driving [Gao et al., 2020], surveillance [Xing, Ai, and Lao, 2010], augmented reality [Zhang and Vela, 2015], and robotics [Robin and Lacroix, 2016]. However, building a general system for tracking an arbitrary object in the wild using only information about the location of the object at the first frame is non-trivial and has many edge cases [Wu, Lim, and Yang, 2013], such as occlusions, deformations, lighting changes, background cluttering, reappearance, etc. Figure 1.1 illustrates several visual object tracking samples. The leftmost image is the first frame of a video and red box annotates the object to track. The consecutive three images are 50th, 100th and 200th video frames respectively where tracking results are visualized with a yellow bounding box.

The recent adoption of deep neural networks, specifically Siamese networks [Koch et al., 2015], has led to significant progress in visual object tracking [Bertinetto et al., 2016a; Li et al., 2018; Xu et al., 2020; Li et al., 2019a; Zhu et al., 2018; Zhang and Peng, 2019; Zhang et al., 2020a]. One of the main advantages of Siamese trackers is the possibility of end-to-end offline learning. In contrast, methods incorporating online learning [Danelljan et al., 2019; Bhat et al., 2019; Nam and Han, 2016] increase computational complexity to an unacceptable extent for real-world scenarios [Marvasti-Zadeh et al., 2021].

Current state-of-the-art approaches for visual object tracking achieve high results

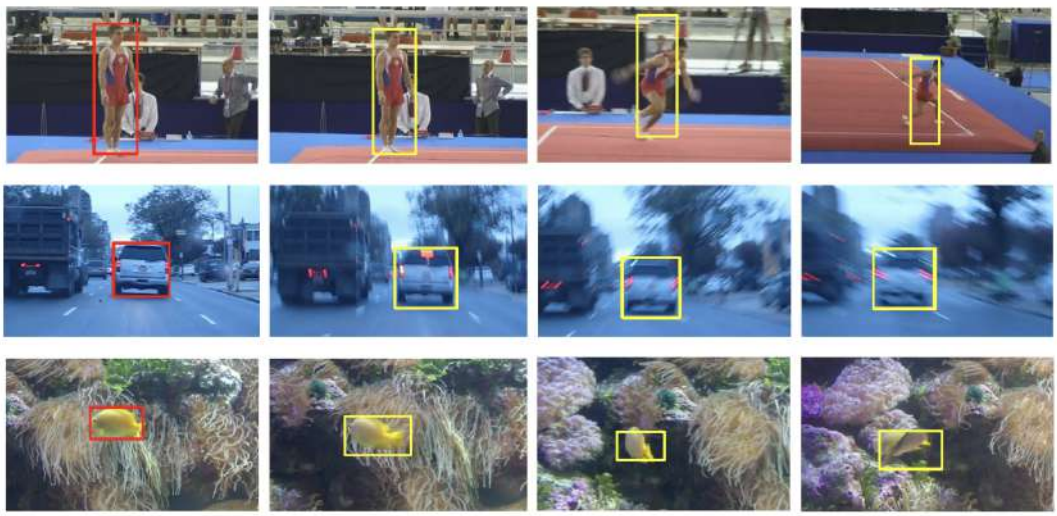


FIGURE 1.1: Snapshot of a simple visual object tracking [Bertinetto et al., 2016a].

on several benchmarks [Kristan et al., 2016; Kristan et al., 2020] at the cost of heavy computational load. Top-tier visual trackers like SiamRPN++ [Li et al., 2019a] and Ocean [Zhang et al., 2020a] exploit complex feature extraction and cross-correlation modules, resulting in 54M parameters and 49 GFLOPs, and 26M parameters and 20 GFLOPs, respectively. Recently, STARK [Yan et al., 2021b] introduced a transformer-based encoder-decoder architecture for visual tracking with 23.3M parameters and 10.5 GFLOPs.

The large memory footprint of modern visual object trackers cannot satisfy the strict performance requirements of real-world applications. Employing a mobile-friendly backbone into the Siamese tracker architecture does not lead to a significant boost in the inference time, as most memory and time-consuming operations are in the decoder or bounding box prediction modules. Therefore, designing a lightweight visual object tracking algorithm, efficient across a wide range of hardware, remains a challenging problem. Moreover, it is essential to incorporate temporal information into the algorithm to make a tracker robust to pose, lighting, and other object appearance changes. This usually assumes adding either dedicated branches to the model [Yan et al., 2021b], or online learning modules [Bhat et al., 2019]. Either approach results in extra FLOPs that negatively impact the run-time performance.

In this paper, we design a tracking algorithm with a high computational efficiency on mobile devices while still surpassing or achieving comparable accuracy to the state-of-the-art deep learning methods. Moreover, we compare the proposed method with SOTA trackers in terms of energy efficiency and inference speed on mobile devices.

Chapter 2

Related Work

2.1 Visual Object Tracking.

Conventional tracking benchmarks such as annual VOT challenges [Kristan et al., 2016] and the Online Tracking Benchmark [Wu, Lim, and Yang, 2013] have historically been dominated by hand-crafted features-based solutions [Vojír, Noskova, and Matas, 2013; Henriques et al., 2014; Bertinetto et al., 2016b]. These algorithms are prone to failure and are usually not applicable to a real-world scenario due to poor generalization capabilities. With the rise of deep learning, they lost popularity constituting only 14% of VOT-ST2020 [Kristan et al., 2020] participant models. New algorithms like SINT [Tao, Gavves, and Smeulders, 2016] and SiamFC [Bertinetto et al., 2016a] replaced those hand-crafted trackers. They learned the matching function in a deep learning fashion and used it without any adapting to track the target based upon the original observation from the first frame, the latter adopting a fully convolutional architecture relative to the search image.

The recent prosperity of deep neural networks led to a significant progress in visual object tracking. Lately, short-term visual object tracking task [Kristan et al., 2020] was mostly addressed using either discriminatory correlation filters [Danelljan et al., 2017; Bhat et al., 2019; Danelljan et al., 2019; Zheng et al., 2020; Xu et al., 2019; Chen et al., 2020a] or Siamese neural networks [Zhang et al., 2020a; Li et al., 2018; Li et al., 2019a; Zhu et al., 2018; Xu et al., 2020; Zhang and Peng, 2019; Held, Thrun, and Savarese, 2016], as well as both combined [Ma et al., 2020; Zhang et al., 2020b; Yan et al., 2021a]. One of the main advantages of Siamese trackers is the ability for end-to-end offline learning and its efficiency. Modern Siamese networks show high generalisation results and can be optimized for real-time inference on GPUs. In contrast, trackers that use deep off-the-shelf features [Danelljan et al., 2017; Danelljan et al., 2016; Danelljan et al., 2015] are prone to overfitting to the initial targets and show limited performance due to inconsistencies in the objectives. Furthermore, methods incorporating online learning [Danelljan et al., 2019; Bhat et al., 2019; Nam and Han, 2016] increase computational complexity to an extent that is unacceptable for real-world scenarios [Marvasti-Zadeh et al., 2021], but they show better target classification and result in a more stable tracking.

2.2 Siamese trackers.

Trackers based upon Siamese correlation networks [Koch et al., 2015] perform tracking based on offline learning of a matching function. This function acts as a similarity metric between the features of the template image and the cropped region of the candidate search area. Siamese network consists of an encoder subnetwork

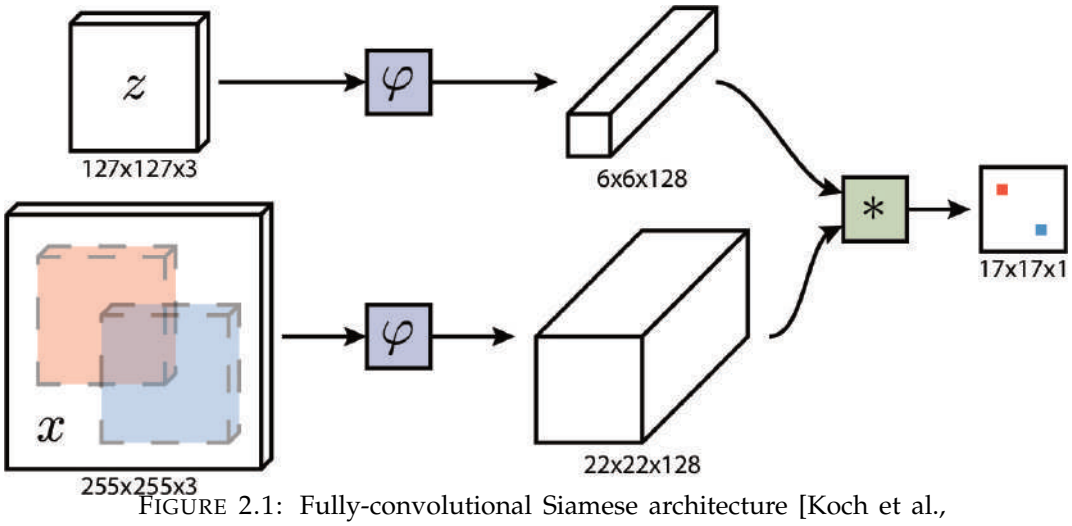


FIGURE 2.1: Fully-convolutional Siamese architecture [Koch et al., 2015]

which encodes both search and template images and is followed by a feature combinations module, which is usually based on feature cross-correlation Figure 2.1. Siamese trackers initially became popular due to their impressive trade-off between accuracy and efficiency [Tao, Gavves, and Smeulders, 2016; Bertinetto et al., 2016a; Wang et al., 2018; Li et al., 2018; Zhu et al., 2018].

SiamRPN [Li et al., 2018] proposed a classification- and regression- objectives for the region proposal subnetwork [Ren et al., 2016] of the tracker model - the former for the foreground-background estimation and the latter for proposal refinement based upon predefined anchor boxes. Its modifications [Zhu et al., 2018; Li et al., 2019a; Fan and Ling, 2019], to name a few, formed a family of the methods inspired by this anchor-based approach: DaSiamRPN [Zhu et al., 2018] introduced a distractor-aware module, taking care of the negative pairs during training to improve the discriminability of the model; SiamRPN++ [Li et al., 2019a] adopted a new sampling strategy to guarantee translation invariance; C-RPN [Fan and Ling, 2019] proposed the cascade-like training leading to more accurate target localization, also making the classifiers of RPNs more discriminative in distinguishing the difficult distractors.

One of the state-of-the-art methods, Ocean [Zhang et al., 2020a], incorporates FCOS [Tian et al., 2019] anchor-free object detection paradigm for tracking, directly regressing the distance from the point in the classification map to the corners of the bounding box. In addition, they introduce a feature alignment module containing 2D spatial transformation that aligns the feature sampling locations with the regions of candidate objects.

Recently, transformer has shown their great potential in vision tasks like image classification [n.d.(b)], object detection [Carion et al., 2020], multiple object tracking [Sun et al., 2021; Meinhardt et al., 2021], etc. Inspired by DETR [Carion et al., 2020] object detection transformer, STARK [Yan et al., 2021b], introduces transformer-based encoder and decoder in a Siamese fashion: flattened and concatenated search and template feature maps serve as an input to the transformer network. STARK also presents a dynamic template update module to encode both spatial and temporal information efficiently. The temporal component of this method allows network to capture object appearance change.

Siamese trackers have impressive trade-off between accuracy and efficiency, allowing Siamese tracker to run realtime on modern GPU devices. Yet, those trackers

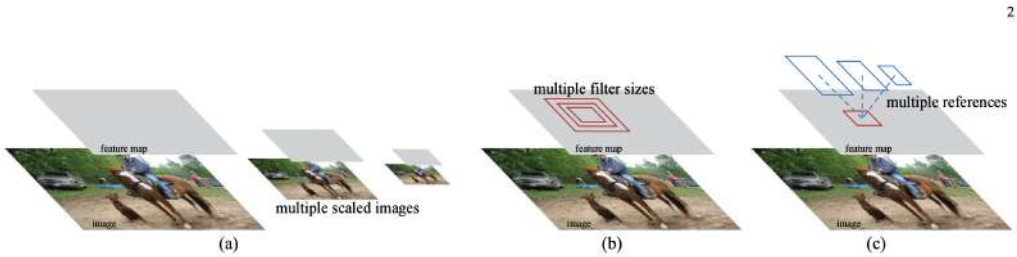


FIGURE 2.2: Anchors at different scales and sizes.

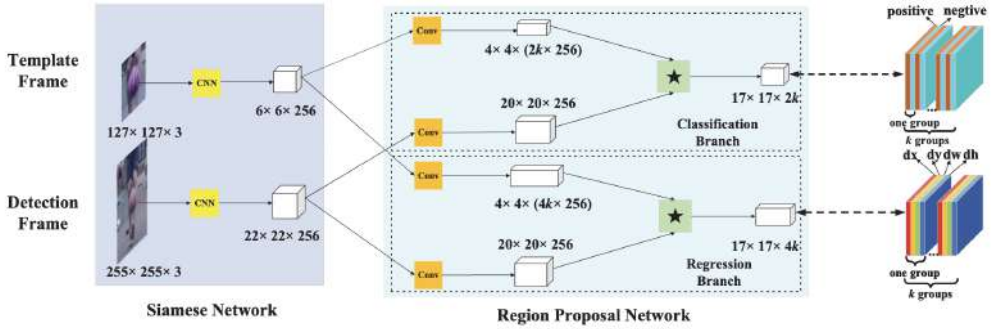


FIGURE 2.3: Framework of Siam-RPN [Li et al., 2018].

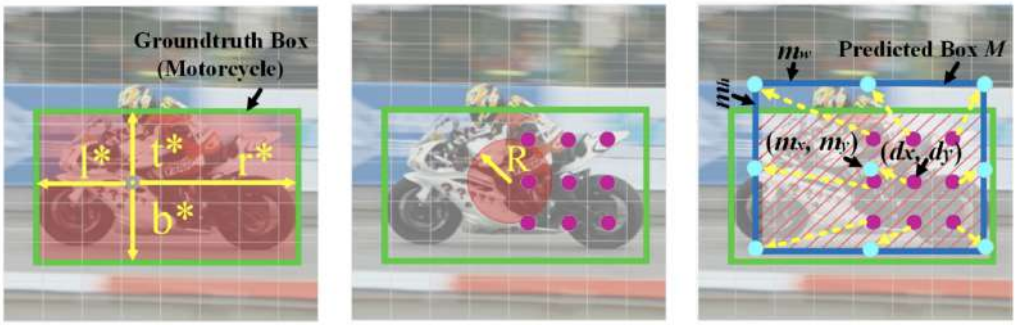
still have a large memory footprint and require a lot of computations. As a result, this type of visual object tracking algorithms is difficult to adapt for some real-world problems of tracking on devices with limited computational resources, like mobile phones. LightTrack [Yan et al., 2021c] made a huge step in designing a lightweight network for tracking on mobile. They used NAS [Pham et al., 2018; Chen et al., 2019] with FLOPS minimization optimization target to design efficient model architecture for resource constrained devices. But FLOPS does not always reflect the actual latency [Wu et al., 2019]

2.3 Anchor-Based Object Trackers

The anchor-based networks play an important role in the object detection task. Anchors were first introduced by Faster-RCNN [Ren et al., 2016], where a set of anchors at multiple scales and aspect ratios is created and served as a reference points (Figure 2.2). The detection algorithm leverages anchor boxes by assigning each object to a single or multiple anchors using classification and localization model predictions.

A number of Siamese trackers have adopted this concept, starting with SiamRPN [Li et al., 2018], followed by modifications such as SiamRPN++ [Li et al., 2019a], C-RPN [Fan and Ling, 2019], and DaSiamRPN [Zhu et al., 2018]. The first two works, introduce a region-proposal network [Ren et al., 2016] after the siamese network and perform joint classification and regression for tracking, while the latter one introduces a distractor-aware module to improve a disriminative power of the model. In these methods, not only the hand-crafted anchors should be generated over the image densely enough to ensure sufficiently high IoU, but also sizes and aspect ratios of the anchor boxes are the additional hyperparameters that need to be carefully tuned. This decrease the robustness and generalization of the RPN-based trackers.

Figure 2.3 shows the framework of Siam-RPN. First, siamese network extracts features from the search and template images. It is followed by a Region Proposal



(a) Regression (b) Regular-region Classification (c) Object-aware Classification

FIGURE 2.4: Ocean [Zhang et al., 2020a] anchor-free regression, regular-region classification and object-aware classification network estimation targets

Network which performs features cross-correlation and has two predictions branches: one for classification and one for regression of bounding box offset and size. Each of k final predictions correspond to one anchor box.

2.4 Anchor-Free Object Trackers

Meanwhile, inspired by point-based object detectors, such as ExtremeNet [Zhou, Zhuo, and Krahenbuhl, 2019], CornerNet [Law and Deng, 2018], CenterNet [Duan et al., 2019], CenterNet-OP [Zhou, Wang, and Krähenbühl, 2019], FCOS [Tian et al., 2019], and FSAF [Zhu, He, and Savvides, 2019], anchor-free object tracking methods started gaining more popularity.

Recent anchor-free trackers include Ocean [Zhang et al., 2020a], SiamCAR [Guo et al., 2020], SiamBAN [Chen et al., 2020b], SiamFC++ [Xu et al., 2020], SiamCorners [Yang et al., 2021], Zhang *et al.* [Zhang and Zhang, 2020], AFSN [Peng et al., 2020]. SiamFC++ [Xu et al., 2020] were the first to introduce this paradigm to the tracking task. They decompose the tracking task into classification (per-pixel objectiveness) and regression (bounding boxes based on deep features in a per-pixel fashion). [Yang et al., 2021] predicts corners of the bounding box via the introduced corner pooling layer, which can predict multiple corners for a tracking target; [Zhang and Zhang, 2020] builds upon FCOS [Tian et al., 2019] anchor-free pipeline and predicts distances from the pixel to the sides of its bounding box; AFSN [Peng et al., 2020] propose three branches, classifying fore- and background, predicting offset to eliminate deviation, and predicting the size of an object with a scale parameter.

Ocean [Zhang et al., 2020a] brings novelty in introducing different sampling strategy to the classification branch, as opposed to using all the pixels as training samples; it also presents object-aware features along with regular-region features to adapt to scale change of the object. All these anchor-free trackers disengage the tracking pipeline from hyperparameter-tuning and priors upon data aspect ratio/scale distribution.

2.5 Online Template Update

Exploitation of both spatial and temporal information from a frame sequence is one of core problems in visual object tracking. Existing trackers can be divided into spatial and spatial-temporal classes. Most of trackers [Li et al., 2018; Zhang and Peng,

2019; Li et al., 2018; Zhang et al., 2020a] utilize only spatial relationship while ignoring global information from the whole video. It makes them prone to failure on cases with occlusions, object deformation, lighting changes, etc. In contrast, spatio-temporal trackers additionally exploit temporal information to improve trackers' performance on such cases.

Spatio-temporal trackers can be devided into gradient-based and gradient-free classes. MD-Net [Nam and Han, 2016] is one the first gradient-based trackers which updates domain-specific layers of the neural network with gradient descent during inference. Later works [Danelljan et al., 2019; Li et al., 2019b; Wang et al., 2020] adopt more advanced optimization techniques or meta-learning update strategies. However, performing backpropagation on most of edge devices is not efficient and its inference time is too high for real-world applications. By contrast, gradient-free methods [Yang and Chan, 2018; Zhang et al., 2019; Zhang et al., 2020a; Yan et al., 2021b] exploit an extra branch in the network to update the template during inference. Although being effective, these methods still require a lot of extra computations to perform template update on each frame.

Chapter 3

The Proposed Method

We design our tracker in a single, unified model composed of a feature extraction network, feature fusion blocks, and task-specific subnetworks for bounding box regression and classification. Given a static template image, I_T, a search image crop, I_S, and a dynamic template image, I_d, the feature extraction network yields the feature maps over these inputs. The template feature representation is then computed as a linear interpolation between static and dynamic template image features. Next, it is fused with the search image features in the pixel-wise fusion blocks. Finally, the resulting tensors are passed to the classification subnetwork that predicts the feature map of the object presence probabilities, and the regression subnetwork that estimates the distances from each pixel within the target bounding box to the four sides of the ground truth bounding box. Every stage is described in detail further on, and the overview of the proposed network architecture is illustrated in Figure 3.1.

3.1 Feature Extraction Network

Efficient tracking pipeline requires a flexible, lightweight, and accurate feature extraction network. Moreover, the outputs of such backbone network should have high enough spatial resolution to have optimal feature capability of object localization [Li et al., 2019a] while not increasing the computations for the consecutive layers. Most of the current Siamese trackers [Zhang et al., 2020a; Li et al., 2019a] increase the spatial resolution of the last feature map, which significantly degrades the performance of successive layers. We suggest to keep the original spatial resolution of the feature extraction network to significantly reduce the computational cost of both backbone and prediction heads, as shown in table Table 3.1.

We use the first four stages of the neural network pretrained on the ImageNet [Deng et al., 2009] as a feature extraction module. The proposed tracker will adopt

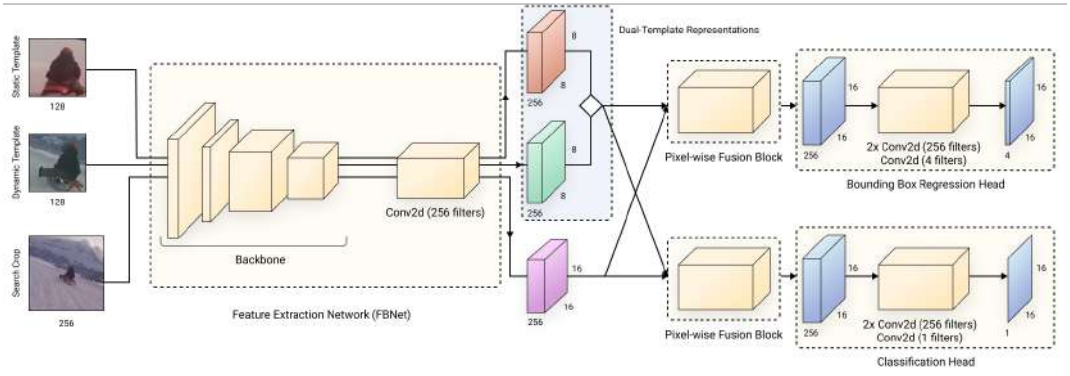


FIGURE 3.1: The proposed network architecture.

Model architecture	Backbone GigaFLOPs	Prediction heads GigaFLOPs
Our tracker	0.318	0.160
Our tracker ↑	0.840	0.746
OceanNet	4.106	1.178
OceanNet ↑ (original)	14.137	11.843

TABLE 3.1: GigaFLOPs, per frame, of our tracker and OceanNet [Zhang et al., 2020a] architectures; ↑ indicates the increased spatial resolutions of the backbone.

FBNet [Wu et al., 2019] family of models as a backbone. Unlike previous works [Yan et al., 2021c] that design backbone by optimizing FLOPs which do not reflect actual inference time on mobile, FBNet is designed via differentiable neural architecture search that directly optimizes model inference on mobile devices. The smallest FBNet achieves 73% top-1 accuracy on ImageNet and 2.9 ms latency (345 frames per second) on a Samsung S8.

The output of the backbone network is a feature map of stride *16* for both template and search images. To map the depth of the output feature map to a constant smaller number of channels, we use an *AdjustLayer* which is a combination of Convolutional and Batch Normalization [Ioffe and Szegedy, 2015] layers.

We show in Table 3.1 and Section 6.6 that upscaling the spatial resolution has a negligible effect on accuracy while increasing FLOPs significantly. Table 3.1 and Figure 6.1 demonstrate that even a lightweight encoder does not improve the model efficiency of modern trackers due to the complex correlation operations and prediction heads. Thus, we further design a lightweight and accurate decoder part of the network.

3.2 Feature Fusion Block

The cross-correlation module is the core operation to combine template and search image features. Most existing Siamese trackers use either simple cross-correlation operation [Bertinetto et al., 2016a; Xu et al., 2020; Li et al., 2018] or more lightweight depth-wise cross-correlation [Li et al., 2019a]. They are further passed to consecutive networks, such as the bounding box regressor. Recently, Alpha-Refine [Yan et al., 2021a] avoided correlation window blurring effect by adopting the pixel-wise correlation as it ensures that each correlation map encodes information of a local region of the target. Extending this idea, we introduce a pixel-wise fusion block which enhances the similarity information obtained via pixel-wise correlation with position and appearance information extracted from the search image (see Table 6.2).

We pass the search image feature map through one 3x3 Conv-BN-ReLU block, and calculate the point-wise cross-correlation between these features and template image features. Then, we concatenate the computed correlation feature map with the search image features and pass the result through one 1x1 Conv-BN-ReLU block to aggregate them. With this approach, learned features are more discriminative and can efficiently encode object position and appearance by combining both visual and similarity features. The overall architecture of a pixel-wise fusion block is visualized in Figure 3.2.

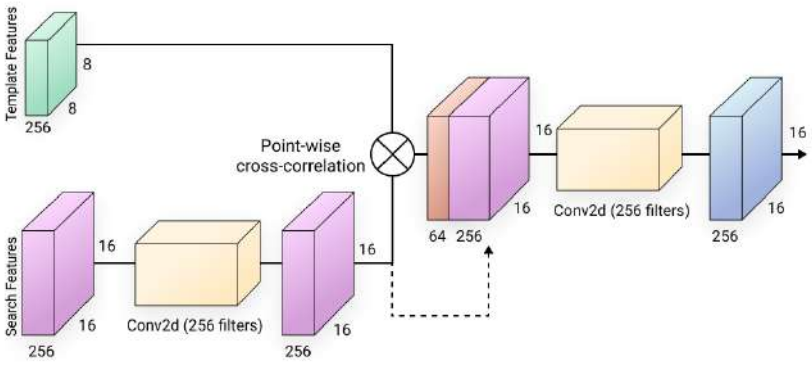


FIGURE 3.2: The pixel-wise fusion block.

Regular cross-correlation cannot be efficiently executed by most mobile neural network inference engines such as CoreML [*Core ML* n.d.] due to unsupported convolutional operation with dynamic weights from the template features. Thus, we reformulated the pixel-wise cross-correlation operation as a matrix multiplication operation to get the best inference speed on mobile devices. Given input image features Φ_S and template image features Φ_T flattened along the spatial dimensions to shapes $C \times WH$ and $C \times wh$ respectively, we compute pixel-wise cross-correlation features Φ_{corr} as:

$$\Phi_{corr} = \Phi_T^\top \Phi_S \tag{3.1}$$

The resulting Φ_{corr} will be a tensor of shape $wh \times WH$ that is further resized to a regular size $wh \times W \times H$, where wh is the number of output channels.

3.3 Classification and Bounding Box Regression Heads

We use a similar setup to Ocean [Zhang et al., 2020a] for classification and bounding box regression. The core idea of a bounding box regression head is to estimate the distance from each pixel within the target object's bounding box to the ground truth bounding box sides [Zhang et al., 2020a; Tian et al., 2019]. Each pixel in the regression feature map is considered as the regression sample if its coordinates fall into the groundtruth bounding box. Such bounding box regression takes into account all of the pixels in the ground truth box during training, so it can accurately predict the magnitude of target objects even when only a tiny portion of the scene is designated as foreground. The bounding box regression network is a stack of two simple 3x3 Conv-BN-ReLU blocks. We use just two such blocks instead of four proposed in Ocean [Zhang et al., 2020a] to reduce computational complexity.

The classification head employs the same structure as a bounding box regression head. The only difference is that we use one filter instead of four in the last Convolutional block. This head predicts a score map, where each pixel represents a confidence score of object appearance in the corresponding region of the search crop.

3.4 Online Update Module

We further equip the proposed algorithm with an online update module. The online branch in the tracker has proven [Danelljan et al., 2019; Zhang et al., 2019] to

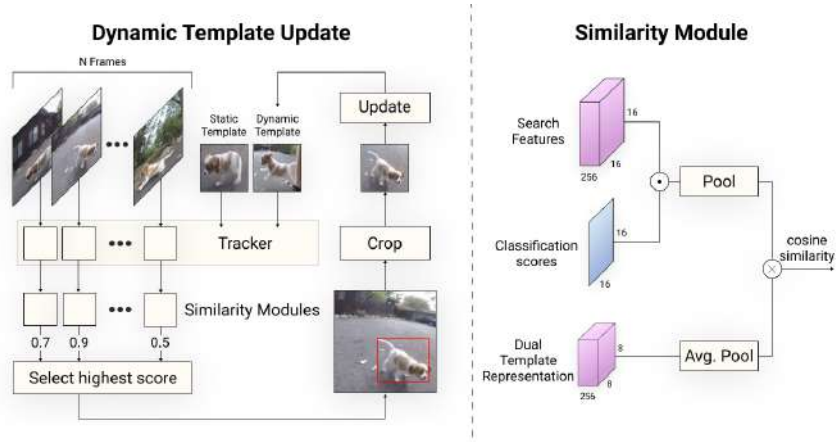


FIGURE 3.3: Dynamic Template update.

increase trackers' robustness to appearance change of the target object during tracking. Current gradient-free online modules [Zhang et al., 2020a; Yan et al., 2021b] require a lot of additional computations to dynamically compute template features update from the search image on each video frame. Therefore, we propose to design a lightweight online update module.

The general scheme of the Dynamic Template Update algorithm is shown in Figure 3.3. In addition to the main static template I_T and search image I_S, we randomly sample a dynamic template image, I_d, from a video sequence during model training to capture the object under various appearances. We pass I_d through the feature extraction network, and the resulting feature map, F_d, is linearly interpolated with the main template feature map F_T via a learnable parameter w:

$$F'_T = (1 - w)F_T + wF_d \tag{3.2}$$

We further pass F'_T and F_S to the Similarity Module that computes cosine similarity between the dual-template and search image embeddings. The search image embedding e_S is obtained via the Weighted Average Pooling (WAP) [Shin et al., 2019] of F_S by the classification confidence scores; the dual-template embedding e_T is computed as an Average Pooling [Lee, Gallagher, and Tu, 2016] of F'_T.

During inference, for every N frames we choose the search image with the highest cosine similarity with the dual-template representation, and update the dynamic template with the predicted bounding box at this frame. In addition, for every training pair we sample a negative crop I_N from a frame that does not contain the target object. We pass it through the feature extraction network, and extract the negative crop embedding e_N similarly to the search image, via WAP. We then compute Triplet Loss [Hoffer and Ailon, 2015] with the embeddings e_T, e_S, e_N extracted from F'_T, F_S and F_N, respectively. This training scheme does provide a signal for the dynamic template scoring while also biasing the model to prefer more general representations.

Unlike STARK [Yan et al., 2021b], which incorporates additional temporal information by introducing *a separate score prediction head* to determine whether to update the dynamic template, we present a *parameter-free* similarity module as a template update rule, optimized with the rest of the network. Moreover, STARK concatenates the dynamic and static template features, increasing the size of a tensor passed to the encoder-decoder transformer resulting in more computations. Our dual-template representation interpolates between the static and dynamic template features with *a*

single learnable parameter, not increasing the template tensor size.

In Chapter 6, we demonstrate the efficiency of our method on a large variety of academic benchmarks and challenging cases. The dual-template representation module allows the model to efficiently encode the temporal information as well as the object appearance and scale change. The increase of model parameters and FLOPs is small and even negligible, making it almost a cost-free temporal module.

3.5 Overall Loss Function

Training a Siamese tracking model requires a multi-component objective function to simultaneously optimize classification and regression tasks. As shown in previous approaches [Zhang et al., 2020a; Yan et al., 2021b], IoU loss [Rezatofighi et al., 2019] and classification loss are used to efficiently train the regression and classification networks jointly. The regression loss term is computed as:

$$L_{reg} = 1 - \sum_{i} IoU(t_{reg}, p_{reg}), \tag{3.3}$$

where t_{reg} denotes the target bounding box, p_{reg} denotes the predicted bounding box, and i indexes the training samples.

For classification loss term, we use Focal Loss [Lin et al., 2017]:

$$L_c = -(1 - p_c)^{\gamma} \log(p_c), \tag{3.4}$$

where p_c is the classification score map computed by the classification prediction head.

In addition, we supplement those training objectives with **triplet loss**, which enables performing the online branch of the network. The triplet loss term is computed from template (e_T), search (e_S), and negative crop (e_N) feature maps:

$$L_t = \max\left\{d(e_T, e_S) - d(e_T, e_N) + \text{margin}, 0)\right\}, \tag{3.5}$$

where margin encourages that dissimilar pairs will be distant from any similar pairs by at least a certain margin. We set margin to default value of 1. The distance measurement in this loss term is

$$d(x_i, y_i) = \|x_i - y_i\|_2. \tag{3.6}$$

The overall loss function is a linear combination of the three components:

$$L = \lambda_1 * L_t + \lambda_2 * L_{reg} + \lambda_3 * L_c. \tag{3.7}$$

where λ_1, λ_2 and λ_3 denote the weights for each loss component and are equal to 2, 1 and 1, respectively.

Chapter 4

Evaluation

4.1 Evaluation Metrics

We use the following metrics to evaluate the tracker:

- **Accuracy** is defined as the average overlap between the target predictions and the ground truth calculated from the frames before the tracker fails on that subsequence. Given a subsequence starting from an anchor a of sequence s, the accuracy $A_{s,a}$ is defined as

$$A_{s,a} = \frac{1}{N^F_{s,a}} \sum_{i=1:N^F_{s,a}} \Omega_{s,a}(i)$$

  where $N^F_{s,a}$ is the number of frames before the tracker failed and $\Omega_{s,a}(i)$ is the overlap between the predicted and ground truth bounding boxes at i-th frame.

- **Robustness** measures the extent of the sub-sequence before the tracking failure and is defined as

$$R_{s,a} = N^F_{s,a} / N_{s,a}$$

  where $N_{s,a}$ is the total number of frames in the subsequence.

- **Expected Average Overlap (EAO)** combines Accuracy and Robustness into a single performance score. The expected average overlap curve between predicted and ground truth bounding boxes is calculated and averaged over an interval of typical short-term sequence lengths into the EAO measure. The value of the EAO curve $\hat{\Phi}_i$ at sequence length i is defined as

$$\hat{\Phi}_i = \frac{1}{|\mathcal{S}(i)|} \sum_{s,a \in \mathcal{S}(i)} \Phi_{s,a}(i)$$

  where $\Phi_{s,a}(i)$ is the average overlap calculated between the first and i-th frame starting at anchor a of sequence s. The EAO measure it than calculated by averaging the EAO curve from N_{lo} to N_{hi}

$$EAO = \frac{1}{N_{\text{hi}} - N_{\text{lo}}} \sum_{i=N_{\text{lo}}:N_{\text{hi}}} \hat{\Phi}_i$$

- **Average Overlap (AO)** denotes the average of overlaps between all ground truth and estimated bounding boxes. It is computed similarly to the Accuracy but it has a downside that all frames after the first failure receive a zero overlap and are still taken into account, which increases bias and variance of the estimator.

- **Success** is measured as the Intersection over Union of the pixels within the ground truth bounding box BB^{gt} and the predicted ones BB^{tr}

$$S = \frac{\left|BB^{tr} \cap BB^{gt}\right|}{\left|BB^{tr} \cup BB^{gt}\right|}$$

- **Precision** is calculated as the distance in pixels between the centers C^{gt} and C^{tr} of the ground truth bounding box and the tracking result, respectively, according to a certain threshold. The precision P is defined as

$$P = \left\|C^{tr} - C^{gt}\right\|_2$$

  Since the precision metric is sensitive to the resolution of input images, its more common to use the normalized precision P_{norm}

$$P_{norm} = \left\|W\left(C^{tr} - C^{gt}\right)\right\|_2 \quad \text{, where} \quad W = diag\left(BB_x^{gt}, BB_y^{gt}\right)$$

The aforementioned Success and Precision metrics are often referred to as Success Score and Precision Score, respectively. The Success Rate measures the percentage of successfully tracked frames where the overlaps exceed a threshold (e.g. 0.5).

4.2 Benchmarks

We evaluate performance of the proposed tracker on several popular academic benchmarks:

- **VOT-ST2021** [Kristan et al., 2016] consists of 60 short video sequences with challenging scenarios: similar objects, partial occlusions, scale and appearance change to address short-term, causal, model-free trackers. The model performance is evaluated using Accuracy, Robustness and EAO.
- **GOT-10K** [Huang, Zhao, and Huang, 2021] is a benchmark covering a wide range of different objects, their deformations, and occlusions. We evaluate our solution using the official GOT-10K submission page with AO and Robustness metrics.
- **LaSOT** [Fan et al., 2021] contains 280 video segments for long-range tracking evaluation. Each sequence is longer than 80 seconds in average making in the largest densely annotated long-term tracking benchmark. We report the Precision Score and Robustness.
- **NFS** [Kiani Galoogahi et al., 2017] dataset is a long-range benchmark, which has 100 videos (380K frames) captured with now commonly available higher frame rate (240 FPS) cameras from real world scenarios. We'll use the Precision Score and Robustness metrics to evaluate the tracker.

4.3 Tracker Efficiency Benchmark

Mobile devices have a limited amount of both computing power and energy available to execute a program. Most current benchmarks measure only runtime speed without taking into account the energy efficiency of the algorithm, which is equally

important in a real-world scenario. Thus, we introduce the benchmark to estimate the effect of tracking algorithms on mobile device battery and thermal state and its impact on the processing speed over time. It measures the energy efficiency of trackers with online and offline evaluation protocols - the former to estimate the energy consumption for the real-time input stream processing and the latter to measure the processing speed of a constant amount of inputs.

The online evaluation collects energy consumption data by simulating a real-time (30 FPS) camera input to the neural network for 30 minutes. The tracker cannot process more frames than the specified FPS even if its inference speed is faster, and it skips inputs that cannot be processed on-time due to the slower processing speed. We collect battery level, the device's thermal state, and neural network inference speed throughout the whole experiment. The thermal state is defined by Apple in the official Thermal state iOS API [*iOS thermal state* n.d.]. The *high* thermal state refers to a serious and critical thermal device state for which the system's performance is significantly reduced to cool it down. The performance loss due to heat causes trackers to slow down, making it a critical performance metric when deployed to mobile devices. Our benchmark takes care of these issues providing fair comparison.

The offline protocol measures the inference speed of trackers by simulating a constant number of inputs for the processing, similar to processing a media file from a disk. All frames are processed one by one without any inference time restrictions and processing is interrupted if device reaches high temperatures to avoid overheating. Additionally, we perform a model warmup before the experiment, as the first model executions are usually slower. We set the number of warmup iterations and inputs for the processing to 20 and 100, respectively.

In this work, we evaluate trackers on iPhone 7 (A10 Fusion and PowerVR Series7XT GPU), iPhone 8 (A11 Bionic with 3-core GPU), iPhone 11 (A11 Bionic with 4-core GPU), and Google Pixel 4 (Snapdragon 855 and Adreno 640 GPU). All devices are fully charged before the experiment, no background tasks are running, and the display is set to the lowest brightness to reduce the energy consumption of hardware that is not involved in computations.

Chapter 5

Training Data

This work uses multiple publicly available video object tracking datasets to train the proposed tracker.

The **YouTube-BoundingBoxes** [Real et al., 2017] is a large-scale dataset of videos with densely-sampled high-quality single-object bounding box annotations. The dataset consists of approximately **380,000** video segments of 15-20s extracted from 240 000 different publicly available YouTube videos, automatically selected to feature objects in natural settings without editing or post-processing, with a recording quality often akin to that of a hand-held cell phone camera. All these video segments were human-annotated with high precision classifications and bounding boxes at 1 frame per second, in total containing over **5.6M** bounding boxes.

The **LaSOT** [Fan et al., 2021] is a high-quality benchmark consisting of 1,400 sequences with more than **3.5M** frames in total. Each video is annotated manually and accurately at 30 frames per second with a bounding box making it one of the largest densely annotated benchmarks for long-term tracking. Each sequence contains 2,500 frames on average and the dataset represents 70 different object categories.

The **GOT-10k** [Huang, Zhao, and Huang, 2021] is built upon the backbone of WordNet structure [Fellbaum, 1998] and it populates the majority of over 560 classes of moving objects and 87 motion patterns. It contains more than 10,000 of short video sequences with more than **1.5M** manually labeled bounding boxes, annotated at 30 frames per second, enabling unified training and stable evaluation of deep trackers.

The **ImageNet-VID** [Deng et al., 2009] is a benchmark created for video object detection task. It contains 30 object categories which is a subset of the 200 basic-level categories of the object detection task. Each video is densely annotated at 30 frames per second with a set of bounding boxes. Overall, benchmark consists of near **2M** annotations and over 4,000 video sequences.

In addition, similar to other tracking models [Bertinetto et al., 2016a], [Zhu et al., 2018], [Zhang et al., 2020a], we use a part of the **COCO** [Lin et al., 2014] dataset for object detection with 80 different object categories to diversify the training dataset for visual object tracking. In our setup, we set $I_S = I_T$ to let the network efficiently predict the object's location in a larger context.

For each training epoch, we randomly sample 20,000 images from LaSOT [Fan et al., 2021], 120,000 from COCO [Lin et al., 2014], 400,000 from YoutubeBB [Real et al., 2017], 320,000 from GOT-10k [Huang, Zhao, and Huang, 2021] and 310,000 images from the ImageNet dataset [Deng et al., 2009], so, overall, 1,170,000 images are used in each epoch. The validation set is constructed from the GOT-10k public test set, VOT-2018, and NFS.

Chapter 6

Experiments

6.1 Implementation Details

6.1.1 Training

We implemented all of models using PyTorch [Paszke et al., 2019]. The backbone network is initialized using the pretrained weights on ImageNet. All the models are trained using **2 RTX A6000** GPUs, with a total batch size of 256. We use ADAM [Kingma and Ba, 2014] optimizer with a learning rate = $2 * 10^{-4}$ and a plateau learning rate reducer with a reduce factor = 0.5 every 10 epochs when the target metric (mean IoU) stops increasing. Each epoch contains 10^6 image pairs. The training takes 3 days to converge.

From each video sequence in a dataset, we randomly sample a template frame I_T and search frame I_S such that the distance between them is $d = 70$ frames. Starting from the $15th$ epoch, we increase d by 2 every epoch. It allows the network to learn the correlation between objects on easier samples initially and gradually increase complexity as the training proceeds. A dynamic template image is sampled from the video sequence between the static template frame and search image frame. For the negative crop, where possible, we sample it from the same frame as the dynamic template but without overlap with this template crop; otherwise, we sample the negative crop from another video sequence. The value for d was found empirically. It is consistent with the note in TrackingNet [Muller et al., 2018] that any tracker is reliable within 1 second. Our observations are that the appearance of objects does not change dramatically over 2 seconds (60 frames), and we set $d = 70$ as a trade-off between the inference speed and the amount of additionally incorporated temporal information.

6.1.2 Preprocessing

We extract template image crops with an additional offset of 20% around the bounding box. Then, we apply a light shift (up to 8px) and random scale change (up to 5% on both sides) augmentations, pad image to the square size with the mean RGB value of the crop, and resize it to the size of 128x128 pixels. We apply the same augmentations with a more severe shift (up to 48px) and scale (between 65% and 135% from the original image size) for the search and negative images. Next, the search image is resized to 256x256 pixels with the same padding strategy as in the template image.

Finally, we apply random photometric augmentations for both search and template images to increase model generalization and robustness under different lighting and color conditions [Buslaev et al., 2020].

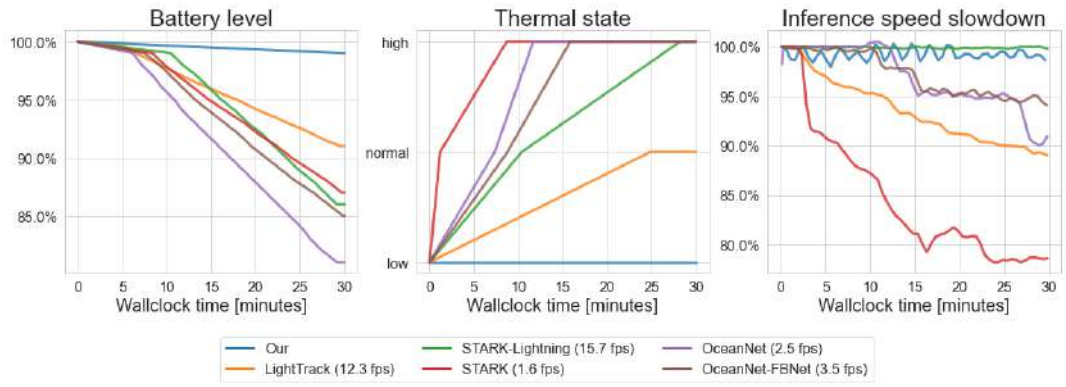


FIGURE 6.1: Online Efficiency Benchmark on iPhone 8: battery consumption, device thermal state, and inference speed degradation over time.

6.1.3 Testing

During inference, tracking follows the same protocols as in [Bertinetto et al., 2016a], [Li et al., 2018]. The static template features of the target object are computed once at the first frame. The dynamic template features are updated every 70 frames and interpolated with the static template features. These features are combined with the search image features in the correlation modules, regression, and classification heads to produce the final output.

6.1.4 Smartphone-based Implementation

The models are trained offline using PyTorch[Paszke et al., 2019] and then ported an optimal model snapshot to mobile devices for inference. All models are executed in *float16* mode for faster execution comparing to *float32* computations. The precision loss of float16 computations is negligible, we observe that the results differ only by $\pm 0.5\%$ depending on the experiment, so all metrics presented in comparisons with SOTA are the same when inferenced on mobile. We use Core ML [*Core ML* n.d.] framework to run trackers on iPhone devices. Core ML is a machine learning API from Apple that optimizes on-device neural network inference by leveraging the CPU, GPU and Neural Engine. For Android devices, we employ TensorFlow Lite [Abadi et al., 2015] which is an open source deep learning framework for on-device inference from Google supporting execution on CPU, GPU and DSP.

6.2 Online Efficiency Benchmark

Figure 6.1 summarizes the online benchmark results on iPhone 8. The upper part of the plot demonstrates the degradation of inference speed over time. We observe that our tracker and STARK-Lightning [Yan et al., 2021d] backbone do not change inference speed over time, while LightTrack [Yan et al., 2021c] and OceanNet [Zhang et al., 2020a] start to process inputs slower. Also, transformer network STARK-S50 degrades significantly and becomes 20% slower after 30 minutes of runtime. The lower part of the figure demonstrates energy efficiency of our tracker compared to other trackers and its negligible impact on device thermal state. STARK-S50 and Ocean overheat device after 10 minutes of execution, LightTrack slightly elevates temperature after 24 minutes, STARK-Lightning overheats device after 27 minutes, while

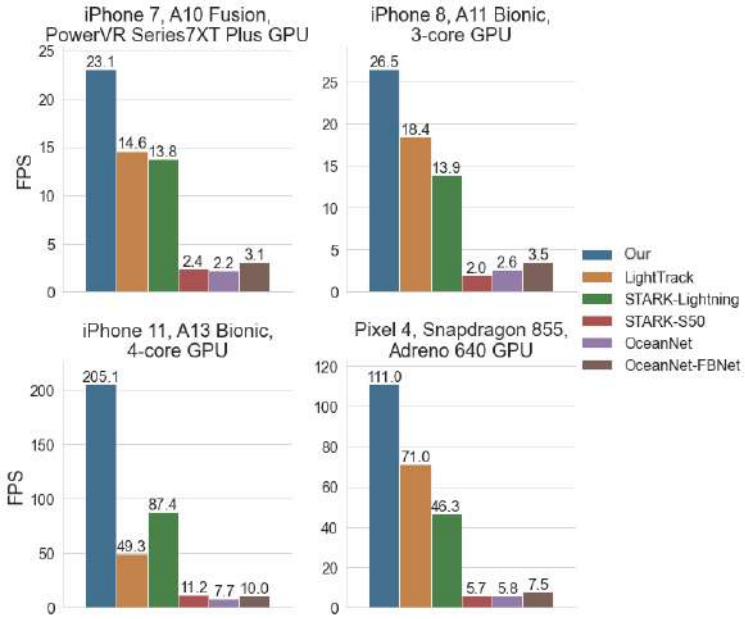


FIGURE 6.2: Offline Efficiency Benchmark: mean FPS on a range of mobile GPU architectures.

our tracker keeps device in a low temperature. Moreover, Ocean with a lightweight backbone FBNet [Wu et al., 2019] is still consuming lots of energy and produces heat due to complex and inefficient decoder.

Additionally, we observe that STARK-Lightning reaches high thermal state without performance drop. Modern devices have a special hardware, called Neural Processing Unit (NPU), designed specifically for neural network inference. The Apple Neural Engine (ANE) is a type of NPU that accelerates neural network operations such as convolutions and matrix multiplies. STARK-Lightning is a transformer based on simple matrix multiplications that are efficiently computed by ANE and thus do not slow down over time.

6.3 Offline Efficiency Benchmark

We summarize the results of offline benchmark in Figure 6.2. We observe that our tracker achieves 1.6 times higher FPS than LightTrack [Yan et al., 2021c] on iPhone 7 (A10 Fusion and PowerVR Series7XT GPU), iPhone 8 (A11 Bionic with 3-core GPU) and Google Pixel 4 (Snapdragon 855 and Adreno 640 GPU). Furthermore, our tracker is more than 4 times faster than LightTrack on iPhone 11 (A11 Bionic with 4-core GPU). our tracker achieves more than 10 times faster inference than OceanNet [Zhang et al., 2020a] and STARK [Yan et al., 2021b] on all aforementioned mobile devices. Such low inference time makes our tracker a very cost-efficient candidate for use in resource-constrained applications. We observe that LightTrack cannot achieve real-time (more than 30 FPS) inference speed on Apple iPhone 7. In contrast, our tracker secures real-time inference time on all mobile devices in scope. The real-time speed of the proposed tracker allows it to be efficiently integrated into applications that run on edge devices with limited computational resources.

	SiamFC++ (GoogleNet) [Xu et al., 2020]	SiamRPN++ (MobileNet-V2) [Li et al., 2019a]	SiamRPN++ (ResNet-50) [Li et al., 2019a]	ATOM [Danelljan et al., 2019]	KYS [Bhat et al., 2020]	Ocean (offline) [Zhang et al., 2020a]	STARK (S50) [Yan et al., 2021b]	STARK (lightning) [Yan et al., 2021d]	LightTrack [Yan et al., 2021c]	**Our**
EAO ↑	0.227	0.235	0.239	0.258	0.274②	0.290①	0.270③	0.226	0.240	0.270③
Accuracy ↑	0.418	0.432	0.438	0.457	0.453	0.479①	0.464③	0.433	0.417	0.471②
Robustness ↑	0.667	0.656	0.668	0.691	0.736①	0.732②	0.719③	0.627	0.684	0.708
iPhone 11 FPS ↑	7.11	6.86	3.49	-	-	7.72	11.2	87.41②	49.13③	205.12①
Parameters (M) ↓	12.71	11.15	53.95	-	-	25.87	23.34	2.28③	1.97②	1.37①
Memory (MB) ↓	24.77	21.63	103.74	-	-	102.81	109.63	6.28③	4.11②	3.00①
Peak memory (MB) ↓	34.17	31.39	192.81	-	-	119.51	295.97	30.69③	9.21①	10.10②

(A) **VOT-ST2021** [Kristan et al., 2016]

	SiamRPN++ (ResNet-50) [Li et al., 2019a]	ATOM [Danelljan et al., 2019]	KYS [Bhat et al., 2020]	Ocean (offline) [Zhang et al., 2020a]	STARK (S50) [Yan et al., 2021b]	LightTrack [Yan et al., 2021c]	**Our**
Average Overlap ↑	0.518	0.556	0.636②	0.592	0.672①	0.611	0.619③
Success Rate ↑	0.618	0.634	0.751②	0.695	0.761①	0.710	0.722③

(B) **GOT-10K** [Huang, Zhao, and Huang, 2021]

	SiamRPN++ (ResNet-50) [Li et al., 2019a]	ATOM [Danelljan et al., 2019]	KYS [Bhat et al., 2020]	Ocean (offline) [Zhang et al., 2020a]	STARK (S50) [Yan et al., 2021b]	STARK (lightning) [Yan et al., 2021d]	LightTrack [Yan et al., 2021c]	**Our**
Success Score ↑	0.503	0.491	0.541③	0.505	0.668①	0.586②	0.523	0.535
Precision Score ↑	0.496	0.483	0.539	0.517	0.701①	0.579②	0.515	0.545③
Success Rate ↑	0.593	0.566	0.640	0.594	0.778①	0.690②	0.596	0.641③

(C) **LaSOT** [Fan et al., 2021]

	SiamRPN++ (ResNet-50) [Li et al., 2019a]	ATOM [Danelljan et al., 2019]	KYS [Bhat et al., 2020]	Ocean (offline) [Zhang et al., 2020a]	STARK (S50) [Yan et al., 2021b]	STARK (lightning) [Yan et al., 2021d]	LightTrack [Yan et al., 2021c]	**Our**
Success Score ↑	0.596	0.592	0.634②	0.573	0.681①	0.628③	0.591	0.614
Precision Score ↑	0.720	0.711	0.766③	0.706	0.825①	0.754	0.730	0.768②
Success Rate ↑	0.748	0.737	0.795③	0.728	0.860①	0.796③	0.743	0.788

(D) **NFS** [Kiani Galoogahi et al., 2017]

TABLE 6.1: Comparison with the state-of-the-art trackers on common benchmarks. ①, ② and ③ indicate the top-3 trackers

6.4 Comparison with the state-of-the-art

We compare the proposed tracker to existing state-of-the-art Siamese [Zhang et al., 2020a; Yan et al., 2021c; Xu et al., 2020; Li et al., 2019a] and DCF [Danelljan et al., 2019; Bhat et al., 2019; Bhat et al., 2020] trackers in terms of model accuracy, robustness and speed. We evaluate performance on two short-term tracking benchmarks: VOT-ST2021[Kristan et al., 2016], GOT-10k[Huang, Zhao, and Huang, 2021] and two long-term tracking benchmarks: LaSOT[Fan et al., 2021], NFS[Kiani Galoogahi et al., 2017].

6.4.1 VOT-ST2021 Benchmark

Table 6.1a reports results on VOT-ST2021. The proposed tracker shows near state-of-the-art performance, outperforming LightTrack [Yan et al., 2021c] and STARK-Lightning [Yan et al., 2021d] by 3% and 4.4% EAO, respectively, while having higher FPS. Also, it is only 2% behind Ocean, yet having more than **18 times fewer parameters** than Ocean tracker and being **26 times faster** at model inference time on iPhone 11. Our tracker demonstrates the same EAO as transformer network STARK-S50 [Yan et al., 2021b] having much fewer parameters and being more efficient on mobile.

Table 6.1a additionally reports model weights memory consumption and peak memory consumption during the forward pass in megabytes. LightTrack and STARK-Lightning model sizes are 4.11MB and 6.28MB, respectively, while our method consumes only 3MB. During the forward pass, the peak memory usage of our tracker is 10.1MB, LightTrack consumes slightly less (9.21MB) by using fewer filters in bounding box regression convolutional layers, and STARK-Lightning has 30.69MB peak memory usage due to memory-consuming self-attention blocks.

6.4.2 GOT-10K Benchmark

Our tracker achieves better results than LightTrack [Yan et al., 2021c] and Ocean [Zhang et al., 2020a], while using 1.4 and 19 times fewer parameters, respectively. Additionally, our method outperforms ATOM [Danelljan et al., 2019], but it is behind STARK-S50 [Yan et al., 2021b] and KYS [Bhat et al., 2020].

6.4.3 LaSOT Benchmark

As presented in Table 6.1c, the Precision Score of our tracker is 3% and 2.8% superior than LightTrack [Yan et al., 2021c] and Ocean [Zhang et al., 2020a], respectively. STARK-S50 is a transformer based tracker and performs better on long term tracking tasks like in LaSOT benchmark, resulting in higher scores than the proposed method, but as reported in Table 6.1a STARK-S50 is much slower and has more parameters.

6.4.4 NFS Benchmark

Table 6.1d presents that our tracker achieves better Success Score (61.4%), being 2.3% and 4.1% higher than LightTrack [Yan et al., 2021c] and Ocean [Zhang et al., 2020a], respectively. The proposed tracker lags behind larger networks like STARK-S50 and KYS similarly to aforementioned benchmarks.

6.5 Qualitative Comparison

The comparison of our tracker with the state-of-the-art methods is presented in Figure 6.3. We display the tracking results of every 200 frames (0 - 1000) on the challenging cases from LaSOT benchmark where the object appearance and scale change throughout the video. The figure demonstrates that our method is more stable than Ocean [Zhang et al., 2020a] and STARK-Lightning [Yan et al., 2021d], while demonstrating slightly worse bounding box predictions than the transformer based tracker STARK-S50 [Yan et al., 2021b], which is more stable in a long-term tracking and more robust to the change of visual appearance.

6.6 Ablation Study

#	Component	EAO↑	Robustness↑	iPhone 11 FPS↑
1	*baseline*	0.236	0.672	122.19
2	+ lower spatial resolution	0.234	0.668	208.41
3	+ pixel-wise fusion block	0.264	0.683	207.72
4	+ dynamic template update	0.270	0.708	205.12

TABLE 6.2: Ablation study on VOT-ST2021 [Kristan et al., 2016].

To verify the efficiency of the proposed method, we evaluate the effects of its different components on the VOT-ST2021 [Kristan et al., 2016] benchmark, as presented in Table 6.2. The baseline model (#1) consists of the FBNet backbone with an increased spatial resolution of the final stage, followed by a plain pixel-wise cross-correlation operation and bounding box prediction network. The performance of the baseline is 0.236 EAO and 0.672 Robustness. In #2, we set the spatial resolution of the last stage to its original value and observe a negligible degradation of EAO while significantly increasing FPS on mobile. Adding our pixel-wise fusion blocks

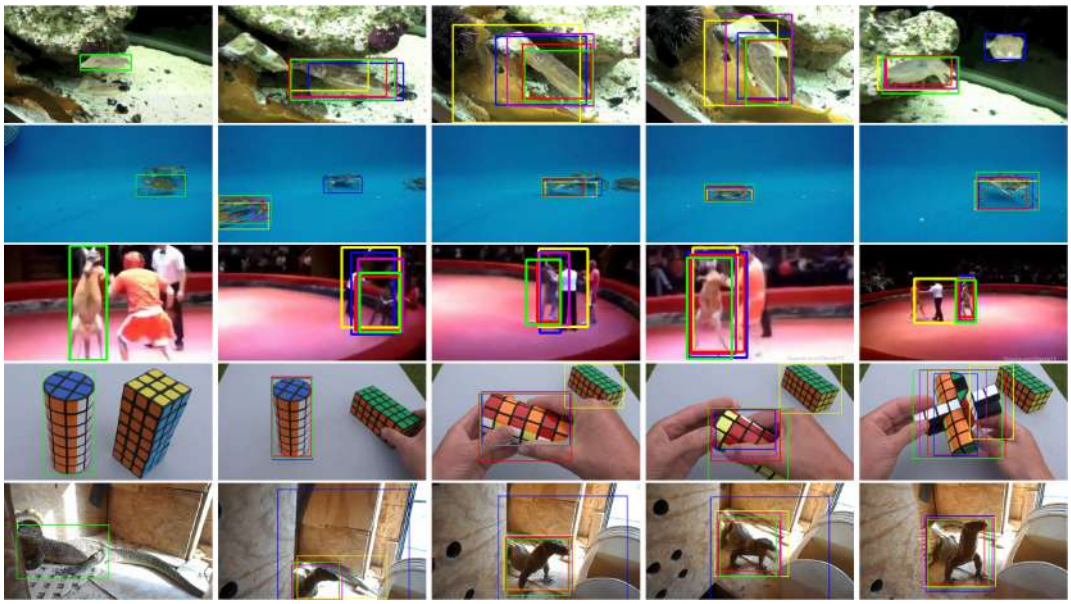


FIGURE 6.3: Qualitative comparison of the proposed tracker with state-of-the-art methods on challenging cases of variations in tracked object appearance from LaSOT benchmark [Fan et al., 2021]. Green: Ground Truth, Red: our tracker, Yellow: STARK Lightning, Blue: Ocean, Purple: Stark-ST50.

(#3) brings a 3% EAO improvement. This indicates that combining search image features and correlation feature maps enhances feature representability and improves tracking accuracy. Furthermore, the proposed dynamic template update module (#4) also brings an improvement of 0.6% in terms of EAO and 2.5% Robustness, showing the effectiveness of this module. The pixel-wise fusion block and dynamic template update brought a significant accuracy improvements while keeping almost the same inference speed. Note that the EAO metrics is calculated w.r.t. *bounding box* tracking.

We additionally compare Focal Loss [Lin et al., 2017] with regular Cross-Entropy as a classification loss term. Table 6.3 demonstrates that our method equipped with Focal Loss (#1) performs better in terms of EAO and Robustness than the same model with the Cross-Entropy loss (#2). Focal Loss addresses the issue of foreground-background class imbalance in the classification score map.

#	Component	EAO↑	Robustness↑
1	our method	0.270	0.708
2	Cross Entropy loss	0.267	0.704

TABLE 6.3: Ablation for Focal and Cross Entropy losses on VOT-ST2021 [Kristan et al., 2016].

Chapter 7

Conclusions

This paper approaches the problem of efficient visual object tracking on mobile devices. We analyzed the related work in the field and outlined the drawbacks of existing state-of-the-art algorithms that limit the use of tracking on mobile devices. Previous works have not paid attention to the efficiency of their solution but concentrated on accuracy instead, resulting in complex neural network architectures. This work proposes a novel ideas and efficiency evaluation protocol to tackle these problems and designs a new tracking algorithm that meets the strict requirements for inference on mobile devices while showing close to state-of-the-art results on popular academic benchmarks (see Section 6.2 and Section 6.4).

7.1 Future Work

As we have shown in Section 6.4, the proposed method outperforms many SOTA algorithms in terms of both accuracy and inference speed on short-term tracking benchmarks. More sophisticated algorithms like STARK [Yan et al., 2021b] are more robust to visual appearance change and occlusions during the long-term tracking due to more complex feature correlation blocks and more computationally expensive online update modules. STARK utilizes a transformer network to capture long-distance pixel relation and works well in capturing temporal information. We suggest further investing in designing a lightweight template update module by combining the proposed lightweight network with a more advanced temporal module.

Bibliography

(N.d.[b]). In: ().

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Bertinetto, Luca et al. (2016a). "Fully-convolutional siamese networks for object tracking". In: *European conference on computer vision*. Springer, pp. 850–865.

Bertinetto, Luca et al. (2016b). "Staple: Complementary learners for real-time tracking". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1401–1409.

Bhat, Goutam et al. (2019). "Learning discriminative model prediction for tracking". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6182–6191.

— (2020). "Know Your Surroundings: Exploiting Scene Information for Object Tracking". In: *CoRR* abs/2003.11014. arXiv: 2003.11014. URL: https://arxiv.org/abs/2003.11014.

Buslaev, Alexander et al. (2020). "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2. ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: https://www.mdpi.com/2078-2489/11/2/125.

Carion, Nicolas et al. (2020). *End-to-End Object Detection with Transformers*. arXiv: 2005.12872 [cs.CV].

Chen, Yiwei et al. (2020a). "AFOD: Adaptive Focused Discriminative Segmentation Tracker". In: *Computer Vision – ECCV 2020 Workshops*. Ed. by Adrien Bartoli and Andrea Fusiello. Cham: Springer International Publishing, pp. 666–682. ISBN: 978-3-030-68238-5.

Chen, Yukang et al. (2019). "Detnas: Backbone search for object detection". In: *Advances in Neural Information Processing Systems* 32, pp. 6642–6652.

Chen, Zedu et al. (2020b). "Siamese box adaptive network for visual tracking". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6668–6677.

Core ML (n.d.). https://developer.apple.com/documentation/coreml.

Danelljan, Martin et al. (2015). "Convolutional features for correlation filter based visual tracking". In: *Proceedings of the IEEE international conference on computer vision workshops*, pp. 58–66.

Danelljan, Martin et al. (2016). "Beyond correlation filters: Learning continuous convolution operators for visual tracking". In: *European conference on computer vision*. Springer, pp. 472–488.

Danelljan, Martin et al. (2017). "Eco: Efficient convolution operators for tracking". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6638–6646.

Danelljan, Martin et al. (2019). "Atom: Accurate tracking by overlap maximization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4660–4669.

Deng, Jia et al. (2009). "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, pp. 248–255.

Duan, Kaiwen et al. (2019). "Centernet: Keypoint triplets for object detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6569–6578.

Fan, Heng and Haibin Ling (2019). "Siamese cascaded region proposal networks for real-time visual tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7952–7961.

Fan, Heng et al. (2021). "LaSOT: A High-quality Large-scale Single Object Tracking Benchmark". In: *Int. J. Comput. Vis.* 129.2, pp. 439–461. DOI: 10.1007/s11263-020-01387-y. URL: https://doi.org/10.1007/s11263-020-01387-y.

Fellbaum, Christiane (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

Gao, Ming et al. (2020). "Manifold Siamese Network: A Novel Visual Tracking ConvNet for Autonomous Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 21.4, pp. 1612–1623. DOI: 10.1109/TITS.2019.2930337.

Guo, Dongyan et al. (2020). "SiamCAR: Siamese fully convolutional classification and regression for visual tracking". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6269–6277.

Held, David, Sebastian Thrun, and Silvio Savarese (2016). "Learning to track at 100 fps with deep regression networks". In: *European conference on computer vision*. Springer, pp. 749–765.

Henriques, João F et al. (2014). "High-speed tracking with kernelized correlation filters". In: *IEEE transactions on pattern analysis and machine intelligence* 37.3, pp. 583–596.

Hoffer, Elad and Nir Ailon (2015). "Deep Metric Learning Using Triplet Network". In: *Similarity-Based Pattern Recognition*. Ed. by Aasa Feragen, Marcello Pelillo, and Marco Loog. Cham: Springer International Publishing, pp. 84–92. ISBN: 978-3-319-24261-3.

Huang, Lianghua, Xin Zhao, and Kaiqi Huang (2021). "GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.5, pp. 1562–1577. DOI: 10.1109/TPAMI.2019.2957464.

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR, pp. 448–456.

iOS thermal state (n.d.). https://developer.apple.com/documentation/foundation/processinfo/thermalstate.

Kiani Galoogahi, Hamed et al. (2017). "Need for speed: A benchmark for higher frame rate object tracking". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1125–1134.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Koch, Gregory et al. (2015). "Siamese neural networks for one-shot image recognition". In.

Kristan, Matej et al. (2016). "A Novel Performance Evaluation Methodology for Single-Target Trackers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.11, pp. 2137–2155. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2516982.

Kristan, Matej et al. (2020). "The eighth visual object tracking VOT2020 challenge results". In: *European Conference on Computer Vision*. Springer, pp. 547–601.

Law, Hei and Jia Deng (2018). "Cornernet: Detecting objects as paired keypoints". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 734–750.

Lee, Chen-Yu, Patrick W. Gallagher, and Zhuowen Tu (2016). "Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 464–472.

Li, Bo et al. (2018). "High Performance Visual Tracking With Siamese Region Proposal Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, Bo et al. (2019a). "Siamrpn++: Evolution of siamese visual tracking with very deep networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4282–4291.

Li, Peixia et al. (2019b). *GradNet: Gradient-Guided Network for Visual Object Tracking*. arXiv: 1909.06800 [cs.CV].

Lin, Tsung-Yi et al. (2014). "Microsoft COCO: Common Objects in Context". In: *ECCV*.

Lin, Tsung-Yi et al. (2017). "Focal Loss for Dense Object Detection". In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007. URL: http://openaccess.thecvf.com/content_ICCV_2017/papers/Lin_Focal_Loss_for_ICCV_2017_paper.pdf.

Ma, Ziang et al. (2020). "RPT: Learning point set representation for Siamese visual tracking". In: *European Conference on Computer Vision*. Springer, pp. 653–665.

Marvasti-Zadeh, Seyed Mojtaba et al. (2021). "Deep learning for visual tracking: A comprehensive survey". In: *IEEE Transactions on Intelligent Transportation Systems*.

Meinhardt, Tim et al. (2021). *TrackFormer: Multi-Object Tracking with Transformers*. arXiv: 2101.02702 [cs.CV].

Muller, Matthias et al. (2018). "Trackingnet: A large-scale dataset and benchmark for object tracking in the wild". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 300–317.

Nam, Hyeonseob and Bohyung Han (2016). "Learning multi-domain convolutional neural networks for visual tracking". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4293–4302.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Peng, Shengyun et al. (2020). "Accurate anchor free tracking". In: *arXiv preprint arXiv:2006.07560*.

Pham, Hieu et al. (2018). "Efficient neural architecture search via parameters sharing". In: *International Conference on Machine Learning*. PMLR, pp. 4095–4104.

Real, Esteban et al. (2017). *YouTube-BoundingBoxes: A Large High-Precision Human-Annotated Data Set for Object Detection in Video*. arXiv: 1702.00824 [cs.CV].

Ren, Shaoqing et al. (2016). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. arXiv: 1506.01497 [cs.CV].

Rezatofighi, Hamid et al. (2019). "Generalized Intersection over Union". In.

Robin, Cyril and Simon Lacroix (2016). "Multi-robot target detection and tracking: taxonomy and survey". In: *Autonomous Robots* 40.4, pp. 729–760.

Shin, Hochul et al. (2019). "Sequential image-based attention network for inferring force estimation without haptic sensor". In: *IEEE Access* 7, pp. 150237–150246.

Sun, Peize et al. (2021). *TransTrack: Multiple Object Tracking with Transformer*. arXiv: 2012.15460 [cs.CV].

Tao, Ran, Efstratios Gavves, and Arnold WM Smeulders (2016). "Siamese instance search for tracking". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1420–1429.

Tian, Zhi et al. (2019). "Fcos: Fully convolutional one-stage object detection". In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9627–9636.

Vojír, Tomás, J. Noskova, and Jiri Matas (2013). "Robust Scale-Adaptive Mean-Shift for Tracking". In: *SCIA*.

Wang, Guangting et al. (2020). *Tracking by Instance Detection: A Meta-Learning Approach*. arXiv: 2004.00830 [cs.CV].

Wang, Qiang et al. (2018). "Learning attentions: residual attentional siamese network for high performance online visual tracking". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4854–4863.

Wu, Bichen et al. (2019). "FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wu, Yi, Jongwoo Lim, and Ming-Hsuan Yang (2013). "Online Object Tracking: A Benchmark". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xing, Junliang, Haizhou Ai, and Shihong Lao (2010). "Multiple human tracking based on multi-view upper-body detection and discriminative learning". In: *2010 20th International Conference on Pattern Recognition*. IEEE, pp. 1698–1701.

Xu, Tianyang et al. (2019). "Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual object tracking". In: *IEEE Transactions on Image Processing* 28.11, pp. 5596–5609.

Xu, Yinda et al. (2020). "SiamFC++: Towards robust and accurate visual tracking with target estimation guidelines". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07, pp. 12549–12556.

Yan, Bin et al. (2021a). "Alpha-refine: Boosting tracking performance by precise bounding box estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5289–5298.

Yan, Bin et al. (2021b). "Learning spatio-temporal transformer for visual tracking". In: *arXiv preprint arXiv:2103.17154*.

Yan, Bin et al. (2021c). "LightTrack: Finding Lightweight Neural Networks for Object Tracking via One-Shot Architecture Search". In: *CVPR 2021*.

Yan, Bin et al. (2021d). *Stark Lightning*. https://github.com/researchmm/Stark.

Yang, Kai et al. (2021). "SiamCorners: Siamese Corner Networks for Visual Tracking". In: *IEEE Transactions on Multimedia*.

Yang, Tianyu and Antoni B. Chan (2018). *Learning Dynamic Memory Networks for Object Tracking*. arXiv: 1803.07268 [cs.CV].

Zhang, Guangcong and Patricio A Vela (2015). "Good features to track for visual slam". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1373–1382.

Zhang, Lichao et al. (2019). *Learning the Model Update for Siamese Trackers*. arXiv: 1908.00855 [cs.CV].

Zhang, Zhipeng and Houwen Peng (2019). "Deeper and wider siamese networks for real-time visual tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4591–4600.

Zhang, Zhipeng et al. (2020a). "Ocean: Object-aware anchor-free tracking". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*. Springer, pp. 771–787.

Zhang, Zhipeng et al. (2020b). "Towards Accurate Pixel-wise Object Tracking by Attention Retrieval". In: *arXiv preprint arXiv:2008.02745*.

Zhang, Zhongzhou and Lei Zhang (2020). "Hard Negative Samples Emphasis Tracker without Anchors". In: *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 4299–4308.

Zheng, Linyu et al. (2020). "Learning feature embeddings for discriminant model based tracking". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*. Springer, pp. 759–775.

Zhou, Xingyi, Dequan Wang, and Philipp Krähenbühl (2019). "Objects as points". In: *arXiv preprint arXiv:1904.07850*.

Zhou, Xingyi, Jiacheng Zhuo, and Philipp Krahenbuhl (2019). "Bottom-up object detection by grouping extreme and center points". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 850–859.

Zhu, Chenchen, Yihui He, and Marios Savvides (2019). "Feature selective anchor-free module for single-shot object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 840–849.

Zhu, Zheng et al. (2018). "Distractor-aware siamese networks for visual object tracking". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 101–117.