FEAR: Fast, Efficient, Accurate and Robust Visual Tracker

Vasyl Borsuk[1,2*], Roman Vei[1,2*], Orest Kupyn[1,2], Tetiana Martyniuk[1,2], Igor Krashenyi[1,2], and Jiři Matas[3]

[1] Ukrainian Catholic University
{borsuk, vey, kupyn, t.martynyuk, igor.krashenyi}@ucu.edu.ua
[2] Piñata Farms, Los Angeles, USA
{vasyl, roman, orest, tetiana, igor}@pinatafarm.com
[3] Visual Recognition Group, Center for Machine Perception, FEE, CTU in Prague
matas@cmp.felk.cvut.cz

Abstract. We present FEAR, a family of **f**ast, **e**fficient, **a**ccurate, and **r**obust Siamese visual trackers. We present a novel and efficient way to benefit from dual-template representation for object model adaption, which incorporates temporal information with only a single learnable parameter. We further improve the tracker architecture with a pixel-wise fusion block. By plugging-in sophisticated backbones with the abovementioned modules, FEAR-M and FEAR-L trackers surpass most Siamese trackers on several academic benchmarks in both accuracy and efficiency. Employed with the lightweight backbone, the optimized version FEAR-XS offers more than 10 times faster tracking than current Siamese trackers while maintaining near state-of-the-art results. FEAR-XS tracker is **2.4x** smaller and **4.3x** faster than LightTrack with superior accuracy. In addition, we expand the definition of the model efficiency by introducing FEAR benchmark that assesses energy consumption and execution speed. We show that energy consumption is a limiting factor for trackers on mobile devices. Source code, pretrained models, and evaluation protocol are available at https://github.com/PinataFarms/FEARTracker.

Keywords: Object tracking

1 Introduction

Visual object tracking is a highly active research area of computer vision with many applications such as autonomous driving [15], surveillance [55], augmented reality [63], and robotics [44]. Building a general system for tracking an arbitrary object in the wild using only information about the location of the object in the first frame is non-trivial due to occlusions, deformations, lighting changes, background cluttering, reappearance, etc. [54]. Real-world scenarios often require models to be deployed on the edge devices with hardware and power limitations, adding further complexity. Thus, developing a robust tracking algorithm has remained a challenge.
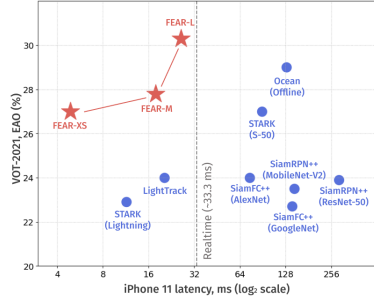
---

* These authors contributed equally to this work.

Fig. 1: **The EAO-Latency trade-off plot**. Compared to other state-of-the-art approaches (shown in blue), the FEAR-XS tracker (in red) achieves superior or comparable quality (EAO) while attaining outstanding speed on mobile devices; FEAR-L (in red) tracker runs in real-time on iPhone 11 and shows the best performance in EAO on VOT-2021.

The recent adoption of deep neural networks, specifically Siamese networks [28], has led to significant progress in visual object tracking [2], [33], [58], [32], [69], [66], [67]. One of the main advantages of Siamese trackers is the possibility of end-to-end offline learning. In contrast, methods incorporating online learning [9], [3], [39] increase computational complexity to an unacceptable extent for real-world scenarios [37].

Current state-of-the-art approaches for visual object tracking achieve high results on several benchmarks [30], [29] at the cost of heavy computational load. Top-tier visual trackers like SiamRPN++ [32] and Ocean [67] exploit complex feature extraction and cross-correlation modules, resulting in 54M parameters and 49 GFLOPs, and 26M parameters and 20 GFLOPs, respectively. Recently, STARK [59] introduced a transformer-based encoder-decoder architecture for visual tracking with 23.3M parameters and 10.5 GFLOPs. The large memory footprint 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 (see Table 1). 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 [59], or online learning modules [3]. Either approach results in extra FLOPs that negatively impact the run-time performance.

We introduce a novel lightweight tracking framework, *FEAR tracker*, that efficiently solves the above-mentioned problems. We develop a single-parameter *dual-template* module which allows to learn the change of the object appearance on the fly without any increase in model complexity, mitigating the memory bottleneck of recently proposed online learning modules [9], [3], [4], [11]. This module predicts the likelihood of the target object being close to the center of the search image, thus allowing to select candidates for the template image update. Furthermore, we interpolate the online selected dynamic template image feature map with the feature map of the original static template image in a learnable way. This allows the model to capture object appearance changes during inference. We optimize the neural network architecture to perform more than 10 times faster than most current Siamese trackers. Additionally, we design an extra lightweight FEAR-XS network that achieves real-time performance on mobile

devices while still surpassing or achieving comparable accuracy to the state-of-the-art deep learning methods.

The main contributions of the paper are:

– A novel **dual-template representation** for object model adaptation. The first template, static, anchors the original visual appearance and thus prevents drift and, consequently, adaptation-induced failures. The other is dynamic; its state reflects the current acquisition conditions and object appearance. Unlike STARK [59], which incorporates additional temporal information by introducing *a separate score prediction head*, we introduce a *parameter-free* similarity module as a template update rule, optimized with the rest of the network. We show that a learned convex combination of the two templates is effective for tracking on multiple benchmarks.

– A lightweight tracker that combines a compact feature extraction network, the dual-template representation, and pixel-wise fusion blocks. The resulting **FEAR-XS tracker** runs at 205 FPS on iPhone 11, 4.2× faster than LightTrack [61] and 26.6× faster than Ocean [67], with high accuracy on multiple benchmarks - no state-of-the-art tracker is at the same time more accurate and faster than any of FEAR trackers. Besides, the algorithm is highly energy-efficient.

– We introduce **FEAR benchmark** - a new tracker efficiency benchmark and protocol. Efficiency is defined in terms of both energy consumption and execution speed. Such aspect of vision algorithms, important in real-world use, has not been benchmarked before. We show that current state-of-the-art trackers show high instantaneous speed when evaluated over a small test set, but slow down over time when processing large number of samples, as the device overheats when the tracker is not energy-efficient. In that sense, FEAR family fills the gap between speed and accuracy for real-time trackers.

2 Related Work

Visual Object Tracking. Conventional tracking benchmarks such as annual VOT challenges [30] and the Online Tracking Benchmark [54] have historically been dominated by hand-crafted features-based solutions [49], [19], [1]. With the rise of deep learning, they lost popularity constituting only 14% of VOT-ST2020 [29] participant models. Lately, short-term visual object tracking task [29] was mostly addressed using either discriminatory correlation filters [10], [3], [9], [68], [57], [7] or Siamese neural networks [67], [33], [32], [69], [58], [66], [18], as well as both combined [36], [65], [62]. Moreover, success of visual transformer networks for image classification [13] has resulted in new high-scoring models [59], [6], [50] for tracking.

Siamese trackers. Trackers based on Siamese correlation networks 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 trackers initially became popular due to their impressive trade-off between accuracy and efficiency [47], [2], [51], [33], [69]; however, they could not keep up with the accuracy of online learning methods [9], [3]. With recent modeling improvements, Siamese-based trackers [59], [67] hold winning positions on the most popular benchmarks [30], [22], [14].

One of the state-of-the-art methods, Ocean [67], incorporates FCOS [48] 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. Another state-of-the-art approach, STARK [59], introduces a transformer-based encoder-decoder in a Siamese fashion: flattened and concatenated search and template feature maps serve as an input to the transformer network.

Neither of the forenamed state-of-the-art architectures explicitly addresses the task of fast, high-quality visual object tracking across the wide variety of GPU architectures.

Recently, LightTrack [61] made a considerable step towards performant tracking on mobile, optimizing for FLOPs as well as model size via NAS [41], [8]. Still, FLOP count does not always reflect actual inference time [52].

Efficient Neural Networks. Designing efficient and lightweight neural networks optimized for inference on mobile devices has attracted much attention in the past few years due to many practical applications. SqueezeNet [23] was one of the first works focusing on reducing the size of the neural network. They introduced an efficient downsampling strategy, extensive usage of 1x1 Convolutional blocks, and a few smaller modules to decrease the network size significantly. Furthermore, SqueezeNext [16] and ShiftNet [53] achieve extra size reduction without any significant drop of accuracy. Recent works focus not only on the size but also on the speed, optimizing FLOP count directly. MobileNets introduce new architecture components: MobileNet [21] uses depth-wise separable convolutions as a lightweight alternative to spatial convolutions, and MobileNet-v2 [45] adds memory-efficient inverted residual layers. ShuffleNet [64] utilizes group convolutions and shuffle operations to reduce the FLOP count further. More recently, FBNet [52] also takes the hardware design into account, creating a family of mobile-optimized CNNs using neural architecture search.

For FEAR trackers, we followed best practices for designing efficient and flexible neural network architecture. For an extremely lightweight version, where possible, we used depth-wise separable convolutions instead of regular ones and designed the network layers such that the Conv-BN-ReLU blocks could be fused at the export step.

3   The method

FEAR tracker is a single, unified model composed of a feature extraction network, a dual-template representation, pixel-wise 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 and passed to the classification and regression subnetworks. Every stage is described in detail further on, see the overview of the FEAR network architecture in Figure 2.

3.1   FEAR Network Architecture

Feature Extraction Network. Efficient tracking pipeline requires a flexible, lightweight, and accurate feature extraction network. Moreover, the outputs of such backbone net-
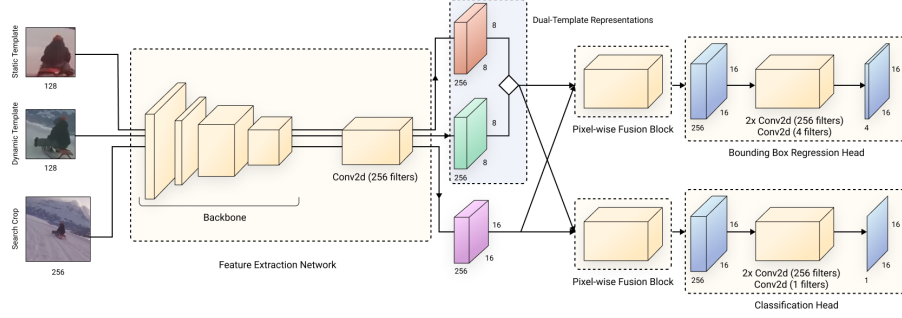
Fig. 2: **The FEAR network architecture** consists of 5 components: feature extraction network, dual-template representation, pixel-wise fusion blocks, and bounding box and classification heads. The CNN backbone extracts feature representations from the template and search images. The dual-template representation allows for a single-parameter dynamic template update (see Fig. 3). The pixel-wise fusion block effectively combines template and search image features (see Fig. 4). The bounding box and classification heads make the final predictions for the box location and its presence, respectively.

work should have high enough spatial resolution to have optimal feature capability of object localization [32] while not increasing the computations for the consecutive layers. Most of the current Siamese trackers [67], [32] increase the spatial resolution of the last feature map, which significantly degrades the performance of successive layers. We observe that keeping the original spatial resolution significantly reduces the computational cost of both backbone and prediction heads, as shown in Table 1.

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

Table 1: GigaFLOPs, per frame, of the FEAR tracker and OceanNet [67] architectures; ↑ indicates the increased spatial resolutions of the backbone. We show in Section 4.4 that upscaling has a negligible effect on accuracy while increasing FLOPs significantly.

We use the first four stages of the neural network pretrained on the ImageNet [12] as a feature extraction module. The FEAR-M tracker adopts the vanilla ResNet-50 [17] as a backbone, and the FEAR-L tracker incorporates the RegNet [56] backbone to pursue the state-of-the-art tracking quality, yet remaining efficient.

The output of the backbone network is a feature map of stride *16* for the template and search images. To map the depth of the output feature map to a constant number

of channels, we use a simple *AdjustLayer* which is a combination of Convolutional and Batch Normalization [24] layers.

To shift towards being more efficient during inference on mobile devices, for the mobile version of our tracker - FEAR-XS - we utilize the FBNet [52] family of models designed via NAS. Table 1 and Figure 5 demonstrate that even a lightweight encoder does not improve the model efficiency of modern trackers due to the complex prediction heads. Thus, designing a lightweight and accurate decoder is still a challenge.

The dual-template representation with a dynamic template update allows the model to capture the appearance changes of objects during inference without the need to perform optimization on the fly. The general scheme of the Dynamic Template Update algorithm is shown in Figure 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:

$$F'_T = (1 - w)F_T + wF_d \qquad (1)$$

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) [46] of F_S by the classification confidence scores; the dual-template embedding e_T is computed as an Average Pooling [31] 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 [20] 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 [59], 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 Section 4, 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.

Pixel-wise fusion block. The cross-correlation module creates a joint representation of the template and search image features. Most existing Siamese trackers use
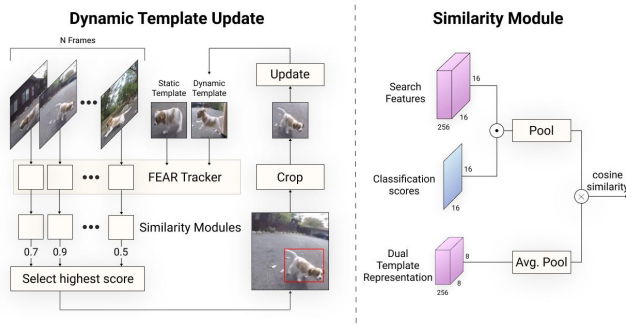
Fig. 3: **Dynamic Template update.** We compare the average-pooled dual-template representation with the search image embedding using cosine similarity, and dynamically update the template representation when object appearance changes dramatically.
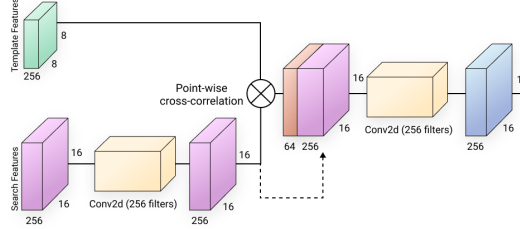

Fig. 4: **The pixel-wise fusion block**. The search and template features are combined using the point-wise cross-correlation module and enriched with search features via concatenation. The output is then forwarded to regression heads.

either simple cross-correlation operation [2], [58], [33] or more lightweight depth-wise cross-correlation [32]. Recently, Alpha-Refine [62] 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 4).

We pass the search image feature map through a 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 a 1x1 Conv-BN-ReLU block to aggregate them. With this approach, learned features are more discriminative and can efficiently encode object position and appearance: see Sec. 4.4 and Tab. 4 for the detailed ablation study. The overall architecture of a pixel-wise fusion block is visualized in Fig. 4.

Classification and Bounding Box Regression Heads. 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 [67], [48]. 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 [67] 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 16x16 score map, where each pixel represents a confidence score of object appearance in the corresponding region of the search crop.

3.2   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 [67], [59], IoU loss [43] and classification loss are used to efficiently train the regression and classification networks jointly. In addition, to train FEAR trackers, we supplement those training objectives with **triplet loss**, which enables performing Dynamic Template Update. As seen in the ablation study, it improves the tracking quality by 0.6% EAO with only a single additional trainable parameter and marginal inference cost (see Table 4). To our knowledge, this is a novel approach in training object trackers.

The triplet loss term is computed from template (e_T), search (e_S), and negative crop (e_N) feature maps:

$$L_t = \max\left\{d(e_T, e_S) - d(e_T, e_N) + \text{margin}, 0))\right\}, \tag{2}$$

where $d(x_i, y_i) = \|x_i - y_i\|_2$. The regression loss term is computed as:

$$L_{reg} = 1 - \sum_i IoU(t_{reg}, p_{reg}), \tag{3}$$

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 [34]:

$$L_c = -(1 - p_t)^\gamma \log(p_t), \quad p_t = \begin{cases} p & \text{if y = 1,} \\ 1 - p & \text{otherwise.} \end{cases} \tag{4}$$

In the above, $y \in \{-1; 1\}$ is a GT class, and $0 \le p \le 1$ is the predicted probability for the class $y = 1$. The overall loss function is a linear combination of the three components with $\lambda_1, \lambda_2, \lambda_3$ being 0.5, 1.0, 1.0, respectively:

$$L = \lambda_1 * L_t + \lambda_2 * L_{reg} + \lambda_3 * L_c. \tag{5}$$

4   Experimental Evaluation

4.1   Implementation Details

Training. We implemented all of models using PyTorch [40]. The backbone network is initialized using the pretrained weights on ImageNet. All the models are trained using

4 RTX A6000 GPUs, with a total batch size of 512. We use ADAM [27] optimizer with a learning rate = $4*10^{-4}$ and a plateau learning rate reducer with a factor = 0.5 every 10 epochs monitoring the target metric (mean IoU). Each epoch contains 10^6 image pairs. The training takes 5 days to converge. For each epoch, we randomly sample 20,000 images from LaSOT [14], 120,000 from COCO [35], 400,000 from YoutubeBB [42], 320,000 from GOT10k [22] and 310,000 images from the ImageNet dataset [12], so, overall, 1,170,000 images are used in each epoch.

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 15*th* 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 [38] 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.

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 [5].

Testing: During inference, tracking follows the same protocols as in [2], [33]. 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.

4.2   Tracker efficiency benchmark

Setup: 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 **FEAR 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
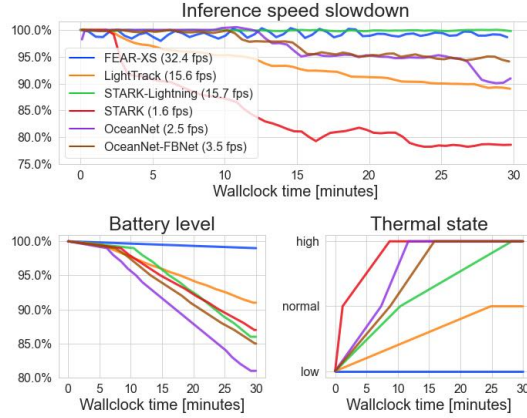
Fig. 5: **Online Efficiency Benchmark on iPhone 8:** battery consumption, device thermal state, and inference speed degradation over time. FEAR-XS tracker does not change the thermal state of the device and has a negligible impact on the battery level. Transformer-based trackers have a battery level drop comparable to the Siamese trackers, reaching a high thermal state in less than 10 minutes of online processing.

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, device's thermal state, and inference speed throughout the whole experiment. The thermal state is defined by Apple in the official Thermal state iOS API [25]. The *high* thermal state refers to a critical thermal state when 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. FEAR benchmark takes care of these issues providing fair comparison (see Fig. 5).

The offline protocol measures the inference speed of trackers by simulating a constant number of random inputs for the processing. All frames are processed one by one without any inference time restrictions.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, iPhone 8 Plus, iPhone 11, and Pixel 4. 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.

We observe that algorithms that reach the high system thermal state get a significant drop in the processing speed due to the smaller amount of processing units available.
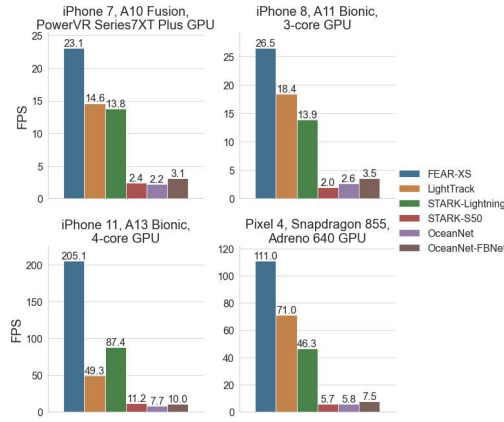
Fig. 6: **Offline Efficiency Benchmark:** mean FPS on a range of mobile GPU architectures. FEAR-XS tracker has superior processing speed on all devices while being an order of magnitude faster on a modern GPU – Apple A13 Bionic.

The results prove that the tracking speed is dependent on the energy efficiency, and both should be taken into account.

Online efficiency benchmark: Fig. 5 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 FEAR-XS tracker and STARK-Lightning [60] backbone do not change inference speed over time, while LightTrack [61] and OceanNet [67] 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 FEAR-XS tracker against competitors 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 FEAR-XS tracker keeps device in a low temperature. Moreover, Ocean with a lightweight backbone FBNet [52] still consumes 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.

Offline efficiency benchmark: We summarize the results of offline benchmark in Figure 6. We observe that FEAR-XS tracker achieves 1.6 times higher FPS than Light-Track [61] 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, FEAR-XS tracker is more than 4 times faster than LightTrack on iPhone 11 (A11 Bionic with 4-core GPU). FEAR-XS tracker achieves more than 10 times

FPS	Success Score	Precision Score	Success Rate
30	0.618	0.753	0.780
240	**0.655**	**0.816**	**0.835**

Table 2: **Extremely High FPS Tracking Matters**. The metrics were computed from the same set of frames on 30 and 240 fps NFS benchmark[26]. FEAR-XS, tracking in over 200 fps, achieves superior performance than trackers limited to 30 fps by incorporating additional temporal information from intermediate frames.

faster inference than OceanNet [67] and STARK [59] on all aforementioned mobile devices. Such low inference time makes FEAR-XS tracker a very cost-efficient candidate for use in resource-constrained applications.

4.3   Comparison with the state-of-the-art

We compare FEAR trackers to existing state-of-the-art Siamese[67], [61], [58], [32] and DCF[9], [3], [4] trackers in terms of model accuracy, robustness and speed. We evaluate performance on two short-term tracking benchmarks: VOT-ST2021[30], GOT-10k[22] and two long-term tracking benchmarks: LaSOT[14], NFS[26]. We provide three version of FEAR tracker: **FEAR-XS**, **FEAR-M** and **FEAR-XL**. The first one is a lightweight network optimized for on-device inference while two latter networks are more heavy and provide more accurate results.

VOT-ST2021 Benchmark: This benchmark 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. Table 3a reports results on VOT-ST2021. It takes both Accuracy (A) and Robustness (R) into account to compute the *bounding box* Expected Average Overlap metric (EAO) [30] which is used to evaluate the overall performance. FEAR-L tracker demonstrates 1.3% higher EAO than Ocean [67] and outperforms trackers with online update, such as ATOM [9] and KYS [4], by 3% EAO. FEAR-XS tracker shows near state-of-the-art performance, outperforming LightTrack [61] and STARK-Lightning [60] 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 (iPhone 11).

Table 3a 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 FEAR-XS consumes only 3MB. During the forward pass, the peak memory usage of FEAR-XS 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.

GOT-10K Benchmark: GOT-10K [22] 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. FEAR-XS tracker achieves better results than

Table 3: **Comparison of FEAR and the state-of-the-art trackers** on common benchmarks: VOT-ST2021[30], GOT-10K[22], LaSOT[14], and NFS[26]. FEAR trackers use much fewer parameters, achieves higher FPS; their accuracy and robustness is on par with the best. ①, ② and ③ indicate the top-3 trackers

	SiamFC++ (GoogleNet) [58]	SiamRPN++ (MobileNet-V2) [32]	SiamRPN++ (ResNet-50) [32]	ATOM [9]	KYS [4]	Ocean (offline) [67]	STARK (S50) [59]	STARK (lightning) [60]	LightTrack [61]	FEAR-XS	FEAR-M	FEAR-L
EAO ↑	0.227	0.235	0.239	0.258	0.274	0.290②	0.270	0.226	0.240	0.270	0.278③	0.303①
Accuracy ↑	0.418	0.432	0.438	0.457	0.453	0.479②	0.464	0.433	0.417	0.471	0.476③	0.501①
Robustness ↑	0.667	0.656	0.668	0.691	0.736②	0.732③	0.719	0.627	0.684	0.708	0.728	0.755①
iPhone 11 FPS ↑	7.11	6.86	3.49	-	-	7.72	11.2	87.41②	49.13	205.12①	56.20③	38.3
Parameters (M) ↓	12.71	11.15	53.95	-	-	25.87	23.34	2.28③	1.97②	1.37①	9.67	33.65
Memory (MB) ↓	24.77	21.63	103.74	-	-	102.81	109.63	6.28③	4.11②	3.00①	18.82	66.24
Peak memory (MB) ↓	34.17	31.39	192.81	-	-	119.51	295.97	30.69	9.21①	10.10②	25.88③	85.97

(a) **VOT-ST2021** [30]

	SiamRPN++ (ResNet-50) [32]	ATOM [9]	KYS [4]	Ocean (offline) [67]	STARK (S50) [59]	LightTrack [61]	FEAR-XS	FEAR-M	FEAR-L
Average Overlap ↑	0.518	0.556	0.636③	0.592	0.672①	0.611	0.619	0.623	0.645②
Success Rate ↑	0.618	0.634	0.751②	0.695	0.761①	0.710	0.722	0.730	0.746③

(b) **GOT-10K** [22]

	SiamRPN++ (ResNet-50) [32]	ATOM [9]	KYS [4]	Ocean (offline) [67]	STARK (S50) [59]	STARK (lightning) [60]	LightTrack [61]	FEAR-XS	FEAR-M	FEAR-L
Success Score ↑	0.503	0.491	0.541	0.505	0.668①	0.586②	0.523	0.535	0.546	0.579③
Precision Score ↑	0.496	0.483	0.539	0.517	0.701①	0.579③	0.515	0.545	0.556	0.609②
Success Rate ↑	0.593	0.566	0.640	0.594	0.778①	0.690②	0.596	0.641	0.638	0.686③

(c) **LaSOT** [14]

	SiamRPN++ (ResNet-50) [32]	ATOM [9]	KYS [4]	Ocean (offline) [67]	STARK (S50) [59]	STARK (lightning) [60]	LightTrack [61]	FEAR-XS	FEAR-M	FEAR-L
Success Score ↑	0.596	0.592	0.634③	0.573	0.681①	0.628	0.591	0.614	0.622	0.658②
Precision Score ↑	0.720	0.711	0.766	0.706	0.825①	0.754	0.730	0.768③	0.745	0.814②
Success Rate ↑	0.748	0.737	0.795	0.728	0.860①	0.796③	0.743	0.788	0.788	0.834②

(d) **NFS** [26]

LightTrack [61] and Ocean [67], while using 1.4 and 19 times fewer parameters, respectively. More details in the Table 3b.

LaSOT Benchmark: LaSOT [14] 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 *Success Score* as well as *Precision Score* and *Success Rate*. As presented in Table 3c, the *Precision Score* of FEAR-XS tracker is 3% and 2.8% superior than LightTrack [61] and Ocean [67], respectively. Besides, the larger FEAR-M and FEAR-L trackers further improve *Success Score* outperforming KYS [4] by 0.5% and 3.8%.

NFS Benchmark: NFS [26] 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. Table 3d presents that FEAR-XS tracker achieves better *Success Score* (61.4%), being 2.3% and 4.1% higher than LightTrack [61] and Ocean [67], respectively. Besides, FEAR-L tracker outperforms KYS [4] by

#	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 4: **FEAR-XS tracker – Ablation study on VOT-ST2021 [30]**.

2.4% *Success Score* and 4.8% *Precision Score*. Additionally, Table 2 reports the impact of extremely high FPS video processing on accuracy, implying the importance of developing a fast tracker capable to process videos in higher FPS.

4.4   Ablation Study

To verify the efficiency of the proposed method, we evaluate the effects of its different components on the VOT-ST2021 [30] benchmark, as presented in Table 4. 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 (#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.

5   Conclusions

In this paper, we introduce the *FEAR tracker* family - an efficient and powerful new Siamese tracking framework that benefits from novel architectural blocks. We validate FEAR trackers performance on several popular academic benchmarks and show that the models near or exceed existing solutions while reducing the computational cost of inference. We demonstrate that the FEAR-XS model attains real-time performance on embedded devices with high energy efficiency. Additionally, we introduce a novel tracker efficiency benchmark, where FEAR trackers demonstrate their energy efficiency and high inference speed, being more efficient and accurate than current state-of-the-art approaches at the same time.

References

1. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.: Staple: Complementary learners for real-time tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1401–1409 (2016) 3

2. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: European conference on computer vision. pp. 850–865. Springer (2016) 2, 3, 7, 9

3. Bhat, G., Danelljan, M., Gool, L.V., Timofte, R.: Learning discriminative model prediction for tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6182–6191 (2019) 2, 3, 12

4. Bhat, G., Danelljan, M., Gool, L.V., Timofte, R.: Know your surroundings: Exploiting scene information for object tracking. In: European Conference on Computer Vision. pp. 205–221. Springer (2020) 2, 12, 13

5. Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M., Kalinin, A.A.: Albumentations: fast and flexible image augmentations. Information **11**(2), 125 (2020) 9

6. Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., Lu, H.: Transformer tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8126–8135 (2021) 3

7. Chen, Y., Xu, J., Yu, J., Wang, Q., Yoo, B., Han, J.J.: Afod: Adaptive focused discriminative segmentation tracker. In: European Conference on Computer Vision. pp. 666–682. Springer (2020) 3

8. Chen, Y., Yang, T., Zhang, X., Meng, G., Xiao, X., Sun, J.: Detnas: Backbone search for object detection. Advances in Neural Information Processing Systems **32**, 6642–6652 (2019) 4

9. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: Atom: Accurate tracking by overlap maximization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4660–4669 (2019) 2, 3, 12, 13

10. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: Eco: Efficient convolution operators for tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6638–6646 (2017) 3

11. Danelljan, M., Gool, L.V., Timofte, R.: Probabilistic regression for visual tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 7183–7192 (2020) 2

12. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) 5, 9

13. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2021) 3

14. Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., Ling, H.: Lasot: A high-quality benchmark for large-scale single object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 3, 9, 12, 13

15. Gao, M., Jin, L., Jiang, Y., Guo, B.: Manifold siamese network: A novel visual tracking convnet for autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems **21**(4), 1612–1623 (2020) 1

16. Gholami, A., Kwon, K., Wu, B., Tai, Z., Yue, X., Jin, P., Zhao, S., Keutzer, K.: Squeezenext: Hardware-aware neural network design. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (June 2018) 4

17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 5

18. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 fps with deep regression networks. In: European conference on computer vision. pp. 749–765. Springer (2016) 3

19. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE transactions on pattern analysis and machine intelligence **37**(3), 583–596 (2014) 3

20. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: International workshop on similarity-based pattern recognition. pp. 84–92. Springer (2015) 6

21. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) 4

22. Huang, L., Zhao, X., Huang, K.: Got-10k: A large high-diversity benchmark for generic object tracking in the wild. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(5), 1562–1577 (2021) 3, 9, 12, 13

23. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. arXiv:1602.07360 (2016) 4

24. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456. PMLR (2015) 6

25. iOS thermal state. https://developer.apple.com/documentation/foundation/processinfo/thermalstate 10

26. Kiani Galoogahi, H., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1125–1134 (2017) 12, 13

27. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 9

28. Koch, G., Zemel, R., Salakhutdinov, R., et al.: Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop. vol. 2, p. 0. Lille (2015) 2

29. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Kämäräinen, J.K., Danelljan, M., Zajc, L.Č., Lukežič, A., Drbohlav, O., et al.: The eighth visual object tracking vot2020 challenge results. In: European Conference on Computer Vision. pp. 547–601. Springer (2020) 2, 3

30. Kristan, M., Matas, J., Leonardis, A., Vojir, T., Pflugfelder, R., Fernandez, G., Nebehay, G., Porikli, F., Čehovin, L.: A novel performance evaluation methodology for single-target trackers. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(11), 2137–2155 (Nov 2016) 2, 3, 12, 13, 14

31. Lee, C.Y., Gallagher, P.W., Tu, Z.: 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 (2016) 6

32. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: 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 (2019) 2, 3, 5, 7, 12, 13

33. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018) 2, 3, 7, 9

34. Lin, T.Y., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 2999–3007 (2017) 8

35. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., Zitnick, L.: Microsoft coco: Common objects in context. In: ECCV (September 2014) 9

36. Ma, Z., Wang, L., Zhang, H., Lu, W., Yin, J.: Rpt: Learning point set representation for siamese visual tracking. In: European Conference on Computer Vision. pp. 653–665. Springer (2020) 3

37. Marvasti-Zadeh, S.M., Cheng, L., Ghanei-Yakhdan, H., Kasaei, S.: Deep learning for visual tracking: A comprehensive survey. IEEE Transactions on Intelligent Transportation Systems (2021) 2

38. Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., Ghanem, B.: 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 (2018) 9

39. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4293–4302 (2016) 2

40. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 32 (2019) 8

41. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameters sharing. In: International Conference on Machine Learning. pp. 4095–4104. PMLR (2018) 4

42. Real, E., Shlens, J., Mazzocchi, S., Pan, X., Vanhoucke, V.: Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5296–5305 (2017) 9

43. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 8

44. Robin, C., Lacroix, S.: Multi-robot target detection and tracking: taxonomy and survey. Autonomous Robots 40(4), 729–760 (2016) 1

45. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018) 4

46. Shin, H., Cho, H., Kim, D., Ko, D.K., Lim, S.C., Hwang, W.: Sequential image-based attention network for inferring force estimation without haptic sensor. IEEE Access 7, 150237–150246 (2019) 6

47. Tao, R., Gavves, E., Smeulders, A.W.: Siamese instance search for tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1420–1429 (2016) 3

48. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9627–9636 (2019) 4, 7

49. Vojír, T., Noskova, J., Matas, J.: Robust scale-adaptive mean-shift for tracking. In: SCIA (2013) 3

50. Wang, N., Zhou, W., Wang, J., Li, H.: Transformer meets tracker: Exploiting temporal context for robust visual tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1571–1580 (2021) 3

51. Wang, Q., Teng, Z., Xing, J., Gao, J., Hu, W., Maybank, S.: 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 (2018) 3

52. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 4, 6, 11

53. Wu, B., Wan, A., Yue, X., Jin, P., Zhao, S., Golmant, N., Gholaminejad, A., Gonzalez, J., Keutzer, K.: Shift: A zero flop, zero parameter alternative to spatial convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9127–9135 (2018) 4

54. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2013) 1, 3

55. Xing, J., Ai, H., Lao, S.: Multiple human tracking based on multi-view upper-body detection and discriminative learning. In: 2010 20th International Conference on Pattern Recognition. pp. 1698–1701. IEEE (2010) 1

56. Xu, J., Pan, Y., Pan, X., Hoi, S., Yi, Z., Xu, Z.: Regnet: self-regulated network for image classification. IEEE Transactions on Neural Networks and Learning Systems (2022) 5

57. Xu, T., Feng, Z.H., Wu, X.J., Kittler, J.: Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual object tracking. IEEE Transactions on Image Processing **28**(11), 5596–5609 (2019) 3

58. Xu, Y., Wang, Z., Li, Z., Yuan, Y., Yu, G.: Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 12549–12556 (2020) 2, 3, 7, 12, 13

59. Yan, B., Peng, H., Fu, J., Wang, D., Lu, H.: Learning spatio-temporal transformer for visual tracking. arXiv preprint arXiv:2103.17154 (2021) 2, 3, 4, 6, 8, 12, 13

60. Yan, B., Peng, H., Fu, J., Wang, D., Lu, H.: Stark lightning. https://github.com/researchmm/Stark (2021) 11, 12, 13

61. Yan, B., Peng, H., Wu, K., Wang, D., Fu, J., Lu, H.: Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search. In: CVPR 2021 (June 2021) 3, 4, 11, 12, 13

62. Yan, B., Zhang, X., Wang, D., Lu, H., Yang, X.: 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 (2021) 3, 7

63. Zhang, G., Vela, P.A.: Good features to track for visual slam. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1373–1382 (2015) 1

64. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6848–6856 (2018) 4

65. Zhang, Z., Li, B., Hu, W., Peng, H.: Towards accurate pixel-wise object tracking by attention retrieval. arXiv preprint arXiv:2008.02745 (2020) 3

66. Zhang, Z., Peng, H.: 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 (2019) 2, 3

67. Zhang, Z., Peng, H., Fu, J., Li, B., Hu, W.: Ocean: Object-aware anchor-free tracking. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16. pp. 771–787. Springer (2020) 2, 3, 4, 5, 7, 8, 11, 12, 13

68. Zheng, L., Tang, M., Chen, Y., Wang, J., Lu, H.: Learning feature embeddings for discriminant model based tracking. In: European Conference on Computer Vision. pp. 759–775. Springer (2020) 3

69. Zhu, Z., Wang, Q., Li, B., Wu, W., Yan, J., Hu, W.: Distractor-aware siamese networks for visual object tracking. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 101–117 (2018) 2, 3