Appendix A. Training Datasets

The **YouTube-BoundingBoxes** [10] is a large-scale dataset of videos. The dataset consists of approximately **380,000** video segments of 15-20s with a recording quality often akin to that of a hand-held cell phone camera.

The **LaSOT** [5] consists of 1,400 sequences with more than **3.5M** frames in total. Each sequence contains 2,500 frames on average and the dataset represents 70 different object categories.

The **GOT-10k** [7] is built upon the backbone of WordNet structure [6] 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** [4] is a benchmark created for video object detection task. It contains 30 object categories. Overall, benchmark consists of near **2M** annotations and over **4,000** video sequences.

In addition, similar to other tracking models [2], [12], [11], we use a part of the **COCO** [8] 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.

Appendix B. Technical details

B.1. Pixel-wise correlation implementation

Classical cross-correlation cannot be executed by most mobile neural network inference engines such as CoreML [3] 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 that is better supported on mobile devices.

Given input image features $\Phi_S$ and template image features $\Phi_T$ flattened along the spatial dimensions to shapes $C \times WH$ and $C \times wh$ respectively, we compute pixel-wise cross-correlation features $\Phi_{corr}$ as:

$$\Phi_{corr} = \Phi_T^T \Phi_S$$  \hspace{1cm} (1)

The resulting $\Phi_{corr}$ will be a tensor of shape $wh \times WH$.

B.2. Smartphone-based Implementation

The models are trained offline using PyTorch [9] and then ported with 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 $\pm0.5\%$ depending on the experiment.
We use Core ML [3] framework to run FEAR tracker 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 [1] which is an open-source deep learning framework for on-device inference from Google supporting execution on CPU, GPU and DSP.

Appendix C. Qualitative comparison

The comparison of FEAR tracker with the state-of-the-art methods is presented in Figure 1. 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.

Fig. 1: Qualitative comparison of FEAR tracker with state-of-the-art methods on challenging cases of variations in tracked object appearance from LaSOT benchmark [5]. **Green:** Ground Truth, **Red:** FEAR-L, **Yellow:** STARK Lightning, **Blue:** Ocean, **Purple:** Stark-ST50.
References

3. Core ML. https://developer.apple.com/documentation/coreml 1, 2  