Towards High Performance Video Object Detection for Mobiles

Xizhou Zhu* Jifeng Dai Xingchi Zhu* Yichen Wei Lu Yuan

Microsoft Research Asia
{v-xizzhu,jifdai,v-xizh14,luyuan,yichenw}@microsoft.com

Abstract Despite the recent success of video object detection on Desktop GPUs, its architecture is still far too heavy for mobiles. It is also unclear whether the key principles of sparse feature propagation and multi-frame feature aggregation apply at very limited computational resources. In this paper, we present a light weight network architecture for video object detection on mobiles. Light weight image object detector is applied on sparse key frames. A very small network, Light Flow, is designed for establishing correspondence across frames. A flow-guided GRU module is designed to effectively aggregate features on key frames. For non-key frames, sparse feature propagation is performed. The whole network can be trained end-to-end. The proposed system achieves 60.2% mAP score on ImageNet VID validation at speed of 25.6 fps on mobiles (e.g., HuaWei Mate 8).

1 Introduction

Object detection has achieved significant progress in recent years using deep neural networks [1]. The general trend has been to make deeper and more complicated object detection networks [2,3,4,5,6,7,8,9,10,11] in order to achieve higher accuracy. However, these advances in improving accuracy are not necessarily making networks more efficient with respect to size and speed. In many real world applications, such as robotics, self-driving car, augmented reality, and mobile phone, the object detection tasks need to be carried out in a real-time fashion on a computationally limited platform.

Recently, there has been rising interest in building very small, low latency models that can be easily matched to the design requirements for mobile and embedded vision application, for example, SqueezeNet [12], MobileNet [13], and ShuffleNet [14]. These structures are general, but not specifically designed for object detection tasks. For this purpose, several small deep neural network architectures for object detection in static images are explored, such as YOLO [15], YOLOv2 [11], Tiny YOLO [16], Tiny SSD [17]. However, directly applying these detectors to videos faces new challenges. First, applying the deep networks on all video frames introduces unaffordable computational cost. Second, recognition accuracy suffers from deteriorated appearances in videos that are seldom observed in still images, such as motion blur, video defocus, rare poses, etc.

* This work is done when Xizhou Zhu and Xingchi Zhu are interns at Microsoft Research Asia.

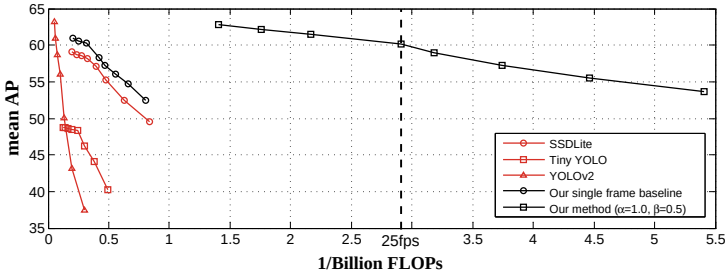

Figure 1. Speed-accuracy trade-off for different lightweight object detectors. Curves are drawn with varying image resolutions. Inference time is evaluated with TensorFlow Lite [18] on a single 2.3GHz Cortex-A72 processor of Huawei Mate 8.

To address these issues, the current best practice [19,20,21] exploits temporal information for speedup and improvement on detection accuracy for videos. On one hand, *sparse feature propagation* is used in [19,21] to save expensive feature computation on most frames. Features on these frames are propagated from sparse key frame cheaply. On the other hand, *multi-frame feature aggregation* is performed in [20,21] to improve feature quality and detection accuracy.

Built on the two principles, the latest work [21] provides a good speed-accuracy tradeoff on Desktop GPUs. Unfortunately, the architecture is not friendly for mobiles. For example, flow estimation, as the key and common component in feature propagation and aggregation [19,20,21], is still far from the demand of real-time computation on mobiles. Aggregation with long-term dependency is also restricted by limited runtime memory of mobiles.

This paper describes a light weight network architecture for mobile video object detection. It is primarily built on the two principles – propagating features on majority non-key frames while computing and aggregating features on sparse key frames. However, we need to carefully redesign both structures for mobiles by considering speed, size and accuracy. On all frames, we present Light Flow, a very small deep neural network to estimate feature flow, which offers instant availability on mobiles. On sparse key frame, we present *flow-guided Gated Recurrent Unit (GRU) based feature aggregation*, an effective aggregation on a memory-limited platform. Additionally, we also exploit a light image object detector for computing features on key frame, which leverage advanced and efficient techniques, such as depthwise separable convolution [22] and Light-Head R-CNN [23].

The proposed techniques are unified to an end-to-end learning system. Comprehensive experiments show that the model steadily pushes forward the performance (speed-accuracy trade-off) envelope, towards high performance video object detection on mobiles. For example, we achieve 60.2% mAP score on ImageNet VID validation at speed of 25.6 frame per second on mobiles (e.g., HuaWei Mate 8). It is one order faster than the best previous effort on fast object detection, with on par accuracy (see Figure 1). To the best of our knowledge, for the first time, we achieve realtime video object detection on mobile with reasonably good accuracy.

2 Revisiting Video Object Detection Baseline

Object detection in static images has achieved significant progress in recent years using deep CNN [1]. State-of-the-art detectors share the similar network architecture, consisting of two conceptual steps. First step is feature network, which extracts a set of convolutional feature maps F over the input image I via a fully convolutional backbone network [24,25,26,27,28,29,30,13,14], denoted as $\mathcal{N}_{feat}(I) = F$. Second step is detection network, which generates detection result y upon the feature maps F, by performing region classification and bounding box regression over either sparse object proposals [2,3,4,5,6,7,8,9] or dense sliding windows [10,15,11,31], via a multi-branched sub-network, namely $\mathcal{N}_{det}(F) = y$. It is randomly initialized and jointly trained with $\mathcal{N}_{feat}$.

Directly applying these detectors to video object detection faces challenges from two aspects. For speed, applying single image detectors on all video frames is not efficient, since the backbone network $\mathcal{N}_{feat}$ is usually deep and slow. For accuracy, detection accuracy suffers from deteriorated appearances in videos that are seldom observed in still images, such as motion blur, video defocus, rare poses.

Current best practice [19,20,21] exploits temporal information via *sparse feature propagation* and *multi-frame feature aggregation* to address the speed and accuracy issues, respectively.

Sparse Feature Propagation Since contents would be very related between consecutive frames, the exhaustive feature extraction is not very necessary to be computed on most frames. Deep feature flow [19] provides an efficient way, which computes expensive feature network at only sparse key frames (e.g., every 10^{th}) and propagates key frame feature maps to majority non-key frames, which results $5\times$ speedup with minor drop in accuracy.

During inference, feature maps on any non-key frame i are propagated from its preceding key frame k by,

$$F_{k\to i} = \mathcal{W}(F_k, M_{i\to k}), \tag{1}$$

where $F_k = \mathcal{N}_{feat}(I_k)$ is the feature of key frame k, and $\mathcal{W}$ represents the differentiable bilinear warping function. The two dimensional motion field $M_{i\to k}$ between two frames I_i and I_k is estimated through a flow network $\mathcal{N}_{flow}(I_k, I_i) = M_{i\to k}$, which is much cheaper than $\mathcal{N}_{feat}$.

Multi-frame Feature Aggregation To improve detection accuracy, flow-guided feature aggregation (FGFA) [20] aggregates feature maps from nearby frames, which are aligned well through the estimated flow.

The aggregated feature maps $\hat{F}_i$ at frame i is obtained as a weighted average of nearby frames feature maps,

$$\hat{F}_i = \sum_{k\in[i-r,i+r]} W_{k\to i} \odot F_{k\to i}, \tag{2}$$

where $\odot$ denotes element-wise multiplication, and the weight $W_{k\to i}$ is adaptively computed as the similarity between the propagated feature maps $F_{k\to i}$ and the feature map F_i at frame i.

To avoid dense aggregation on all frames, [21] suggested *sparsely recursive feature aggregation*, which operates only on sparse key frames. Such a way retain the feature quality from aggregation but reduce the computational cost as well. Specifically, given two succeeding key frames k and k', the aggregated feature at frame k' is computed by,

$$\hat{F}_{k'} = W_{k\to k'} \odot \hat{F}_{k\to k'} + W_{k'\to k'} \odot F_{k'}. \tag{3}$$

2.1 Practice for Mobiles

As the two principles, *sparse feature propagation* and *multi-frame feature aggregation*, yield the best practice towards high performance (speed and accuracy trade-off) video object detection [21] on Desktop GPUs. Instead, there are very limited computational capability and runtime memory on mobiles. Therefore, what are principles for mobiles should be explored.

- Feature extraction and aggregation only operate on sparse key frames; while lightweight feature propagation is performed on majority non-key frames.
- Flow estimation is the key to feature propagation and aggregation. However, flow networks $\mathcal{N}_{flow}$ used in [19,20,21] are still far from the demand of real-time processing on mobiles. Specifically, FlowNet [32] is 11.8× FLOPs of MobileNet [13] under the same input resolutions. Even the smallest FlowNet Inception used in [19] is 1.6× more FLOPs. A more cheaper $\mathcal{N}_{flow}$ is so necessary.
- Feature aggregation should be operated on aligned feature maps according to flow. Otherwise, displacements caused by large object motion would cause severe errors to aggregation. Long-term dependency in aggregation is also favoured because more temporal information can be fused together for better feature quality.
- The backbone network of single image detector should be as small as possible, since we need it to compute features on sparse key frame.

3 Model Architecture for Mobiles

Based on the above principles, we design a much smaller network architecture for mobile video object detection. Inference pipeline is illustrated in Figure 2.

Given a key frame k' and its proceeding key frame k, feature maps are first extracted by $F_{k'} = \mathcal{N}_{feat}(I_{k'})$, and then aggregated with its proceeding key frame aggregated feature maps $\hat{F}_k$ by,

$$\hat{F}_{k'} = \mathcal{G}(F_{k'}, \hat{F}_k, M_{k'\to k}), \tag{4}$$

where $\mathcal{G}$ is a flow-guided feature aggregation function. The *detection network* $\mathcal{N}_{det}$ is applied on $\hat{F}_{k'}$ to get detection predictions for the key frame k'.

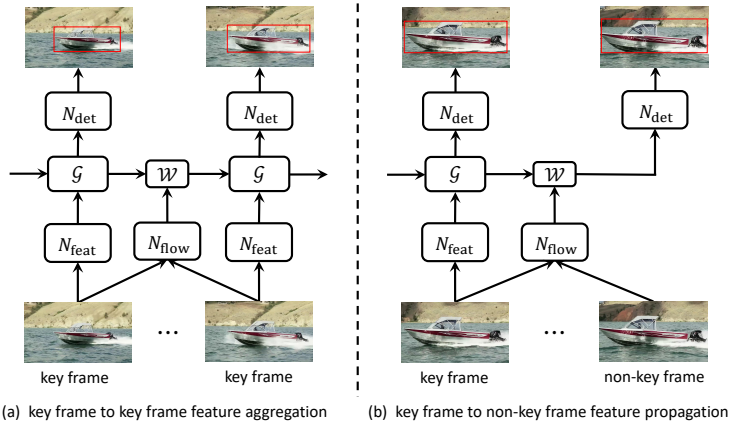

Figure 2. Illustration of video object detection for mobile by the proposed method.

Given a non-key frame i, the feature propagation from key frame k to frame i is denoted as,

$$\hat{F}_{k\to i} = \mathcal{W}(\hat{F}_k, M_{i\to k}), \tag{5}$$

where $\hat{F}_k$ is the aggregated feature maps of key frame k, and $\mathcal{W}$ represents the differentiable bilinear warping function also used in [19]. And then the *detection network* $\mathcal{N}_{det}$ is applied on $\hat{F}_{k\to i}$ to get detection predictions for the non-key frame i.

Next, we will describe two new techniques which are specially designed for mobiles, including Light Flow, a more efficient flow network for mobiles, and a *flow-guided GRU based feature aggregation* for better modeling long-term dependency, yielding better quality and accuracy.

3.1 Light Flow

FlowNet [32] is originally proposed for pixel-level optical flow estimation. It is designed in a encoder-decoder mode followed by multi-resolution optical flow predictors. Two input RGB frames are concatenated to form a 6-channels input. In encoder, the input is converted into a bundle of feature maps in spatial dimensions to 1/64 of input size through a class of convolutional layers. In decoder, the feature maps are fed to multiple deconvolution layers to achieve the high resolution flow prediction. After each deconvolution layer, the feature maps are concatenated with the last feature maps in encoder, which share the same spatial resolution and an upsampled coarse flow prediction. Multiple optical flow predictors follow each concatenated feature maps in decoder. Loss functions are applied to each predictor, but only the finest prediction is used during inference.

To speedup *flow network* $\mathcal{N}_{flow}$ greatly, we present Light Flow, a more light weight flow network with several deliberate designs based on FlowNet [32]. It only causes minor drop in accuracy (15% increasing in end-point error) but significantly speeds up by nearly 65× theoretically (see Table. 2).

In encoder part, convolution is always the bottleneck of computation. Motivated by MobileNet [13], we replace all convolutions to 3×3 depthwise separable convolutions [22] (each 3×3 depthwise convolution followed by a 1×1 pointwise convolution). Compared with standard 3×3 convolution, the computation cost of 3×3 depthwise separable convolution is reduced by $8 \sim 9$ times with a slight drop in accuracy [13].

In decoder part, each deconvolution operation is replaced by a nearest-neighbor upsampling followed by a depthwise separable convolution. [33] replaces deconvolution with nearest-neighbor upsampling followed by a standard convolution to address checkerboard artifacts caused by deconvolution. By contrast, We leverage this idea, and further replace the standard convolution with depthwise separable convolution, to reduce computation cost.

Finally, we adopt a simple and effective way to consider multi-resolution predictions. It is inspired by FCN [34] which fuses multi-resolution semantic segmentation prediction as the final prediction in a explicit summation way. Unlike [32], we do not use only the finest optical flow prediction as final prediction during inference. Instead, multi-resolution predictions are up-sampled to the same spatial resolution with the finest prediction, and then are averaged as the final prediction. Also, during training, only a single loss function is applied on the averaged optical flow prediction instead of multiple loss functions after each prediction. Such a way can reduce end-point error by nearly 10%.

Architecture and Implementation Network of Light Flow is illustrated in Table. 1. Each convolution operation is followed by batch normalization [35] and Leaky ReLU nonlinearity [36] with slope fixed as 0.1. Following [32,37], Light Flow is pre-trained on the Flying Chairs dataset. For training Light Flow, Adam [38] with a weight decay of 0.00004 is used as optimization method. 70k iterations are performed on 4 GPUs, with each GPU holding 64 image pairs. A warm-up learning rate scheme is used in which we first train with a learning rate of 0.001 for 10k iterations. Then we train with learning rate of 0.01 for 20k iterations and divided it by 2 every 10k iterations.

When applying Light Flow for our method, to get further speedup, two modifications are made. First, following [19,20,21], Light Flow is applied on images with half input resolution of the feature network, and has an output stride of 4. As the feature network has an output stride of 16, the flow field is downsampled to match the resolution of the feature maps. Second, since Light Flow is very small and has comparable computation with the detection network $\mathcal{N}_{det}$, sparse feature propagation is applied on the intermediate feature maps of the detection network (see Section 3.3, the 256-d feature maps in RPN [5], and the 490-d feature maps in Light-Head R-CNN [23]), to further reduce computations for non-key frame.

3.2 Flow-guided GRU based Feature Aggregation

Previous works [20,21] have showed that feature aggregation plays an important role on improving detection accuracy. It should be explored how to learn com-

Name	Filter Shape	Stride	Output size	Input
Encoder				
Images			512×384×6	
Conv1_dw	3 × 3 × 6 dw	2	256×192×6	Images
Conv1	1 × 1 × 6 × 32	1	256×192×32	Conv1_dw
Conv2_dw	3 × 3 × 32 dw	2	128×96×32	Conv1
Conv2	1 × 1 × 32 × 64	1	128×96×64	Conv2_dw
Conv3_dw	3 × 3 × 64 dw	2	64×48×64	Conv2
Conv3	1 × 1 × 64 × 128	1	64×48×128	Conv3_dw
Conv4a_dw	3 × 3 × 128 dw	2	32×24×128	Conv3
Conv4a	1 × 1 × 128 × 256	1	32×24×256	Conv4a_dw
Conv4b_dw	3 × 3 × 256 dw	1	32×24×256	Conv4a
Conv4b	1 × 1 × 256 × 256	1	32×24×256	Conv4b_dw
Conv5a_dw	3 × 3 × 256 dw	2	16×12×256	Conv4b
Conv5a	1 × 1 × 256 × 512	1	16×12×512	Conv5a_dw
Conv5b_dw	3 × 3 × 512 dw	1	16×12×512	Conv5a
Conv5b	1 × 1 × 512 × 512	1	16×12×512	Conv5b_dw
Conv6a_dw	3 × 3 × 512 dw	2	8×6×512	Conv5b
Conv6a	1 × 1 × 512 × 1024	1	8×6×1024	Conv6a_dw
Conv6b_dw	3 × 3 × 1024 dw	1	8×6×1024	Conv6a
Conv6b	1 × 1 × 1024 × 1024	1	8×6×1024	Conv6b_dw
Decoder				
Conv7_dw	3 × 3 × 1024 dw	1	8×6×1024	Conv6b
Conv7	1 × 1 × 1024 × 256	1	8×6×256	Conv7_dw
Conv8_dw	3 × 3 × 768 dw	1	16×12×768	[Conv7↑, Conv5b]
Conv8	1 × 1 × 768 × 128	1	16×12×128	Conv8_dw
Conv9_dw	3 × 3 × 384 dw	1	32×24×384	[Conv8↑, Conv4b]
Conv9	1 × 1 × 384 × 64	1	32×24×64	Conv9_dw
Conv10_dw	3 × 3 × 192 dw	1	64×48×192	[Conv9↑, Conv3]
Conv10	1 × 1 × 192 × 32	1	64×48×32	Conv10_dw
Conv11_dw	3 × 3 × 96 dw	1	128×96×96	[Conv10↑, Conv2]
Conv11	1 × 1 × 96 × 16	1	128×96×16	Conv11_dw
Optical Flow Predictors				
Conv12_dw	3 × 3 × 256 dw	1	8×6×256	Conv7
Conv12	1 × 1 × 256 × 2	1	8×6×2	Conv12_dw
Conv13_dw	3 × 3 × 128 dw	1	16×12×128	Conv8
Conv13	1 × 1 × 128 × 2	1	16×12×2	Conv13_dw
Conv14_dw	3 × 3 × 64 dw	1	32×24×64	Conv9
Conv14	1 × 1 × 64 × 2	1	32×24×2	Conv14_dw
Conv15_dw	3 × 3 × 32 dw	1	64×48×32	Conv10
Conv15	1 × 1 × 32 × 2	1	64×48×2	Conv15_dw
Conv16_dw	3 × 3 × 16 dw	1	128×96×16	Conv11
Conv16	1 × 1 × 16 × 2	1	128×96×2	Conv16_dw
Multiple Optical Flow Predictions Fusion				
Average			128×96×2	Conv12↑↑↑↑ + Conv13↑↑↑ + Conv14↑↑ + Conv15↑ + Conv16

Table 1. The details of the Light Flow architecture, 'dw' in filter shape denotes a depthwise separable convolution, ↑ is a 2× nearest neighbor upsampling, $[\cdot,\cdot]$ is the concatenation operation.

plex and long-term temporal dynamics for a wide variety of sequence learning and prediction tasks. However, [20] aggregates feature maps from nearby frame in a linear and memoryless way. Obviously, it only models short-term dependencies. Though recursive aggregation [21] has proven successful on fusing more past frames, it can be difficult to train it to learn long-term dynamics, likely due in part to the vanishing and exploding gradients problem that can result from propagating the gradients down through the many layers of the recurrent network.

Recently, [39] has showed that Gated Recurrent Unit (GRU) [40] is more powerful in modeling long-term dependencies than LSTM [41] and RNN [42], because nonlinearities are incorporated into the network state updates. Inspired by this work, we incorporate convolutional GRU proposed by [43] into flow-guided feature aggregation function $\mathcal{G}$ instead of simply weighted average used in [20,21]. The aggregation function $\mathcal{G}$ in Eq. (4) is computed by,

$$\begin{aligned}
\hat{F}_{k'} &= \mathcal{G}(F_{k'}, \hat{F}_k, M_{k'\to k}) \\
&= (1 - z_t) \odot \hat{F}_{k\to k'} + z_t \odot \phi(W_h \star F_{k'} + U_h \star (r_t \odot \hat{F}_{k\to k'}) + b_h), \\
z_t &= \sigma(W_z \star F_{k'} + U_z \star \hat{F}_{k\to k'} + b_z) \text{ is the update gate}, \\
r_t &= \sigma(W_r \star F_{k'} + U_r \star \hat{F}_{k\to k'} + b_r) \text{ is the reset gate},
\end{aligned} \tag{6}$$

where $\hat{F}_{k\to k'} = \mathcal{W}(\hat{F}_k, M_{k'\to k})$, $W., U., b.$ are parameter tensors and vectors, $\odot$ denotes elementwise multiplication, $\star$ denotes 3×3 convolution, σ is sigmoid function and ϕ is ReLU function.

Compared with the original GRU [40], there are three key differences. First, 3×3 convolution is used instead of fully connected matrix multiplication, since fully connected matrix multiplication is too costly when GRU is applied to image feature maps. Second, ϕ is ReLU function instead of hyperbolic tangent function (tanh) for faster and better convergence. Third, we apply GRU only on sparse key frames (e.g., every 10^{th}) instead of consecutive frames. Since two successive inputs for GRU would be apart 10 frames (1/3 second for a video with 30 fps), object displacements should be taken into account, and thus features should be aligned for GRU aggregation. By contrast, previous works [44,45,46,43] based on either convolutional LSTM or convolutional GRU do not consider such a designing since they operate on consecutive frames instead, where object displacement would be small and neglected.

3.3 Lightweight Key-frame Object Detector

For key frame, we need a lightweight single image object detector, which consists of a feature network and a detection network. For the feature network, we adopt the state-of-the-art lightweight MobileNet [13] as the backbone network, which is designed for mobile recognition tasks. The MobileNet module is pre-trained on ImageNet classification task [47]. For the detection network, RPN [5] and the recently presented Light-Head R-CNN [23] are adopted, because of their light weight. Detailed implementation is illustrated below.

Feature Network We remove the ending average pooling and the fully-connected layer of MobileNet [13], and retain the convolutional layers. Since our input image resolution is very small (e.g., 224×400), we increase feature resolution to get higher performance. First, a 3×3 convolution is applied on top to reduce the feature dimension to 128, and then a nearest-neighbor upsampling is utilized to increase feature stride from 32 to 16. To give more detailed information, a 1×1 convolution with 128 filters is applied to the last feature maps with feature stride 16, and then added to the upsampled 128-d feature maps.

Detection Network RPN [5] and Light-Head R-CNN [23] are applied on the shared 128-d feature maps. In our model, to reduce computation of RPN, 256-d intermediate feature maps was utilized, which is half of originally used in [5]. Three aspect ratios {1:2, 1:1, 2:1} and four scales $\{32^2, 64^2, 128^2, 256^2\}$ for RPN are set to cover objects with different shapes. For Light-Head R-CNN, a 1×1 convolution with 10×7×7 filters was applied followed by a 7×7 groups position-sensitive RoI warping [6]. Then, two sibling fully connected layers are applied on the warped feature to predict RoI classification and regression.

3.4 End-to-end Training

All the modules in the entire architecture, including $\mathcal{N}_{feat}$, $\mathcal{N}_{det}$ and $\mathcal{N}_{flow}$, can be jointly trained for video object detection task. In SGD, $n+1$ nearby video frames, I_i, I_k, I_{k-l}, I_{k-2l}, ..., $I_{k-(n-1)l}$, $0 \leq i-k < l$, are randomly sampled, where key frame duration $l = 10$ and key frame samples $n = 8$ are set for our experiments. In the forward pass, $I_{k-(n-1)l}$ is assumed as a key frame, and the inference pipeline is exactly performed. Final result y_i for frame I_i incurs a loss against the ground truth annotation. All operations are differentiable and thus can be end-to-end trained.

4 Experiments

Experiments are performed on ImageNet VID [47], a large-scale benchmark for video object detection. Following the practice in [48,49], model training and evaluation are performed on the 3,862 training video snippets and the 555 validation video snippets, respectively. The snippets are at frame rates of 25 or 30 fps in general. 30 object categories are involved, which are a subset of ImageNet DET annotated categories.

In training, following [48,49], both the ImageNet VID training set and the ImageNet DET training set are utilized. In each mini-batch of SGD, either $n+1$ nearby video frames from ImageNet VID, or a single image from ImageNet DET, are sampled at 1:1 ratio. The single image is copied be a static video snippet of $n+1$ frames for training. In SGD, 240k iterations are performed on 4 GPUs, with each GPU holding one mini-batch. The learning rates are 10^{-3}, 10^{-4} and 10^{-5} in the first 120k, the middle 60k and the last 60k iterations, respectively.

By default, the key-frame object detector is MobileNet+Light-Head R-CNN, and flow is estimated by Light Flow. The key frame duration length is every 10 frames. In both training and inference, the images are resized to a shorter side of 224 pixels and 112 pixels, for the image recognition network and the flow network, respectively. Inference time is evaluated with TensorFlow Lite [18] on a single 2.3GHz Cortex-A72 processor of Huawei Mate 8. Theoretical computation is counted in FLOPs (floating point operations, note that a multiply-add is counted as 2 operations).

Following the practice in MobileNet [13], two width multipliers, α and β, are introduced for controlling the computational complexity, by adjusting the network width. For each layer (except the final prediction layers) in $\mathcal{N}_{feat}$, $\mathcal{N}_{det}$ and $\mathcal{N}_{flow}$, its output channel number is multiplied by α, α and β, respectively. The resulting network parameter number and theoretical computation change quadratically with the width multiplier. We experiment with $\alpha \in \{1.0, 0.75, 0.5\}$ and $\beta \in \{1.0, 0.75, 0.5\}$. By default, α and β are set as 1.0.

4.1 Ablation Study

Ablation on flow networks The middle panel of Table 2 compares the proposed Light Flow with existing flow estimation networks on the Flying Chairs test set (384 x 512 input resolution). Following the protocol in [32], the accuracy is evaluated by the average end-point error (EPE). Compared with the original FlowNet design in [32], Light Flow ($\beta = 1.0$) can achieve 65.2× theoretical speedup with 14.9× less parameters. The flow estimation accuracy drop is small (15% relative increase in EPE). It is worth noting that it achieves higher accuracy than FlowNet Half and FlowNet Inception utilized in [19], with at least one order less computation overhead. The speed of Light Flow can be further fastened with reduced network width, at certain cost of flow estimation accuracy. Flow estimation would not be a bottleneck in our mobile video object detection system.

Would such a light-weight flow network effectively guide feature propagation? To answer this question, we experiment with integrating different flow networks into our mobile video object detection system. The key-frame object detector is MobileNet+Light-Head R-CNN.

The rightmost panel of Table 2 presents the results. The detection system utilizing Light Flow achieves accuracy very close to that utilizing the heavy-weight FlowNet (61.2% v.s. 61.5%), and is one order faster. Actually, the original FlowNet is so heavy that the detection system with FlowNet is even 2.7× slower than simply applying the MobileNet+Light-Head R-CNN detector on each frame.

Ablation on feature aggregation How important is to exploit flow to align features across frames? To answer this question, we experiment with a degenerated version of our method, where no flow-guided feature propagation is applied before aggregating features across key frames.

flow network	Flying Chairs *test*			ImageNet VID *validation*		
	EPE	params (M)	FLOPs (B)	mAP	params (M)	FLOPs (B)
FlowNet [32]	2.71	38.7	53.48	61.5	45.1	6.41
FlowNet Half [19]	3.53	9.7	14.50	-	-	-
FlowNet Inception [19]	3.68	3.5	7.28	-	-	-
FlowNet 2.0 [37]	1.71	162.5	269.39	-	-	-
1.0 Light Flow	3.14	2.6	0.82	61.2	9.0	0.41
0.75 Light Flow	3.63	1.4	0.48	60.6	7.8	0.37
0.5 Light Flow	4.44	0.7	0.23	60.1	7.1	0.34

Table 2. Ablation of different flow networks for optical flow prediction on Flying Chairs and for video object detection on ImageNet VID.

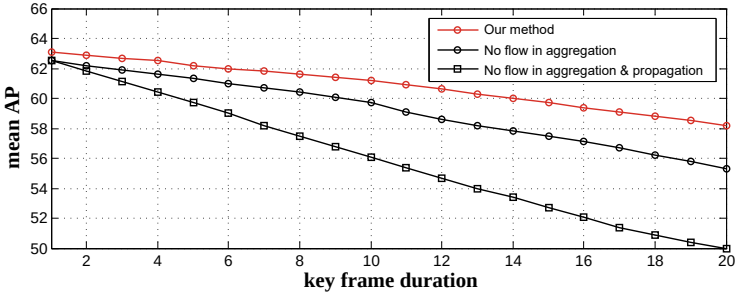

Figure 3. Ablation on the effect of flow guidance in flow-guided GRU and in sparse feature propagation.

Figure 3 shows the speed-accuracy curves of our method with and without flow guidance. The curve is drawn by adjusting the key frame duration l. We can see that the curve with flow guidance surpasses that without flow guidance. The performance gap is more obvious when the key frame duration increases (1.5% mAP score gap at $l = 10$, 2.9% mAP score gap at $l = 20$). This is because the spatial disparity is more obvious when the key frame duration is long. It is worth noting that the accuracy further drops if no flow is applied even for sparse feature propagation on the non-key frames.

Table 3 presents the results of training and inference on frame sequences of varying lengths. We tried training on sequences of 2, 4, 8, 16, and 32 frames. The trained network is either applied on trimmed sequences of the same length as in training, or on the untrimmed video sequences without specific length restriction. The experiment suggests that it is beneficial to train on long sequences, but the gain saturates at length 8. Inference on the untrimmed video sequences leads to accuracy on par with that of trimmed, and can be implemented easier. By default, we train on sequences of length 8, and apply the trained network on untrimmed sequences.

Table 4 further compares the proposed flow-guided GRU method with the feature aggregation approach in [21]. An mAP score of 58.4% is achieved by the aggregation approach in [21], which is comparable with the single frame

train sequence length	2	4	8	16	32
inference trimmed, mAP (%)	59.5	61.0	61.5	61.6	61.5
inference untrimmed, mAP (%)	56.4	60.6	61.2	61.4	61.5

Table 3. Ablation of sequence length in training and inference.

aggregation method	mAP (%)	params (M)	FLOPs (B)
single frame baseline	58.3	5.6	2.39
feature aggregation in [21]	58.4	8.3	0.37
GRU (128-d, default)	61.2	9.0	0.41
GRU (256-d)	62.4	13.0	0.64
GRU (tanh for ϕ)	57.3	9.0	0.41
GRU (stacking 2 layers)	61.4	9.9	0.47
GRU (stacking 3 layers)	60.6	10.8	0.53

Table 4. Ablation on feature aggregation.

baseline at 6.5× theoretical speedup. But it is still 2.8% shy in mAP of utilizing flow-guided GRU, at close computational overhead.

We further studied several design choices in flow-guided GRU. ϕ function with ReLU nonlinearity leads to 3.9% higher mAP score compared to tanh nonlinearity. The ReLU nonlinearity seems to converge faster than tanh in our network. If computation allows, it would be more efficient to increase the accuracy by making the flow-guided GRU module wider (1.2% mAP score increase by enlarging channel width from 128-d to 256-d), other than by stacking multiple layers of the flow-guided GRU module (accuracy drops when stacking 2 or 3 layers).

Accurate Realtime Video Object Detection on Mobile Figure 4 presents the speed-accuracy trade-off curve of our method, drawn with varying key frame duration length l from 1 to 20. Multiple curves are presented, which correspond to networks of different complexity ($\alpha \times \beta \in \{1.0, 0.75, 0.5\} \times \{1.0, 0.75, 0.5\}$). When $l = 1$, the image recognition network is densely applied on each frame, as in the single frame baseline. The difference is flow-guided GRU is applied. The derived accuracy by such dense feature aggregation is noticeably higher than that of the single frame baseline. With increased key frame duration length, the accuracy drops gracefully as the computation overhead relieves. The accuracy of our method at long duration length ($l = 20$) is still on par with that of the single frame baseline, and is 10.6× more computationally efficient. The above observation holds for the curves of networks of different complexity.

As for comparison of different curves, we observe that under adequate computational power, networks of higher complexity ($\alpha = 1.0$) would lead to better speed-accuracy tradeoff. On the other hand, networks of lower complexity ($\alpha = 0.5$) would perform better under limited computational power.

At our mobile test platform, the proposed system achieves an accuracy of 60.2% at speed of 25.6 frames per second ($\alpha = 1.0$, $\beta = 0.5$, $l = 10$). The

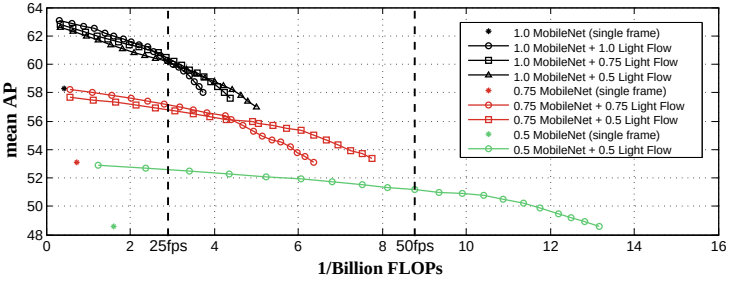

Figure 4. Speed-accuracy trade-off curves of our method utilizing networks of different computational complexity. Curves are drawn with different key frame duration length $l \in \{1, 2, 3, ..., 20\}$.

method	mAP (%)	Params (M)	FLOPs (B)	runtime (fps)
Single frame baseline ($\alpha = 1.0$)	58.3	5.6	2.39	4.0
Single frame baseline ($\alpha = 0.75$)	53.1	3.4	1.36	7.6
Single frame baseline ($\alpha = 0.5$)	48.6	1.7	0.62	16.4
Our method ($\alpha = 1.0$, $\beta = 1.0$)	61.2	9.0	0.41	12.5
Our method ($\alpha = 1.0$, $\beta = 0.75$)	60.8	7.8	0.37	18.2
Our method ($\alpha = 1.0$, $\beta = 0.5$)	60.2	7.1	0.34	25.6
Our method ($\alpha = 0.75$, $\beta = 0.75$)	56.4	5.3	0.23	26.3
Our method ($\alpha = 0.75$, $\beta = 0.5$)	56.0	4.6	0.20	37.0
Our method ($\alpha = 0.5$, $\beta = 0.5$)	51.2	2.6	0.11	52.6

Table 5. Speed-accuracy performance of our method.

accuracy is 51.2% at a frame rate of 50Hz ($\alpha = 0.5$, $\beta = 0.5$, $l = 10$). Table 5 summarizes the results.

5 In Context of Previous Work on Mobile

There are also some other endeavors trying to make object detection efficient enough for devices with limited computational power. They can be mainly classified into two major branches: lightweight image object detectors making the per-frame object detector fast, and mobile video object detectors exploiting temporal information.

5.1 Lightweight Image Object Detector

In spite of the work towards more accurate object detection by exploiting deeper and more complex networks, there are also efforts designing lightweight image object detectors for practical applications. Of them, the improvements of YOLO [15], SSD [10], together with the lastest Light-head R-CNN [23] are of the best speed-accuracy trade-off.

YOLO [15] and SSD [10] are one-stage object detectors, where the detection result is directly produced by the network in a sliding window fashion. YOLO

frames object detection as a regression problem, and a light-weight detection head directly predicts bounding boxes on the whole image. In YOLO and its improvements, like YOLOv2 [11] and Tiny YOLO [16], specifically designed feature extraction networks are utilized for computational efficiency. For SSD, the output space of bounding boxes are discretized into a set of anchor boxes, which are classified by a light-weight detection head. In its improvements, like SSDLite [50] and Tiny SSD [17], more efficient feature extraction networks are also utilized.

Light-head R-CNN [23] is of two-stage, where the object detector is applied on a small set of region proposals. In previous two-stage detectors, either the detection head or its previous layer, is of heavy-weight. In Light-head R-CNN, position-sensitive feature maps [6] are exploited to relief the burden. It shows better speed-accuracy performance than the single-stage detectors.

Lightweight image object detector is an indispensable component for our video object detection system. On top of it, our system can further significantly improve the speed-accuracy trade-off curve. Here we choose to integrate Light-head R-CNN into our system, thanks to its outstanding performance. Other lightweight image object detectors should be generally applicable within our system.

5.2 Mobile Video Object Detector

Despite the practical importance of video object detection on devices with limited computational power, there is scarce literature. Till very recently, there are two latest works seeking to exploit temporal information for addressing this problem.

In Fast YOLO [51], a modified YOLOv2 [11] detector is applied on sparse key frames, and the detected bounding boxes are directly copied to the the non-key frames, as their detection results. Although sparse key frames are exploited for acceleration, no feature aggregation or flow-guided warping is applied. No end-to-end training for video object detection is performed. Without all these important components, its accuracy cannot compete with ours. But direct comparison is difficult, because the paper does not report any accuracy numbers on any datasets for their method, with no public code.

In [44], MobileNet SSDLite [50] is applied densely on all the video frames, and multiple Bottleneck-LSTM layers are applied on the derived image feature maps to aggregate information from multiple frames. It cannot speedup upon the single-frame baseline without sparse key frames. Extending it to exploit sparse key frame features would be non-trival. It would involve feature alignment, which is also lacking in [44]. Its performance also cannot be easily compared with ours. It reports accuracy on a subset of ImageNet VID, where the split is not publicly known. Its code is also not public.

Both two systems cannot compete with the proposed system. They both do not align features across frames. Besides, [51] does not aggregate features from multiple frames for improving accuracy, while [44] does not exploit sparse key

frames for acceleration. Such design choices are vital towards high performance video object detection.

5.3 Comparison on ImageNet VID

Of all the systems discussed in Section 5.1 and Section 5.2, SSDLite [50], Tiny YOLO [16], and YOLOv2 [11] are the most related systems that can be compared at proper effort. They all seek to improve the speed-accuracy trade-off by optimizing the image object detection network. Although they do not report results on ImageNet VID [47], they all public their code fortunately. We first carefully reproduced their results in paper (on PASCAL VOC [52] and COCO [53]), and then trained models on ImageNet VID, also by utilizing ImageNet VID and ImageNet DET train sets. The trained models are applied on each video frame for video object detection. By varying the input image frame size (shorter side in {448, 416, 384, 352, 320, 288, 256, 224} for SSDLite and Tiny YOLO, and {320, 288, 256, 224, 192, 160, 128} for YOLO v2), we can draw their speed-accuracy trade-off curves. The technical report of Fast YOLO [51] is also very related. But it neither reports accuracy nor has public code. We cannot compare with it. Note that the comparison is at the detection system level. We do not dive into the details of varying technical designs.

Figure 1 presents the the speed-accuracy curves of different systems on ImageNet VID validation. For our system, the curve is drawn also by adjusting the image size[1](shorter side for image object detection network in {320, 288, 256, 224, 208, 192, 176, 160}), for fair comparison. The width multipliers α and β are set as 1.0 and 0.5 respectively, and the key frame duration length l is set as 10. Our system surpasses all the existing systems by clear margin. Our method achieves an accuracy of 60.2% at 25.6 fps. Meanwhile, YOLOv2, SSDLite and Tiny YOLO obtain accuracies of 58.7%, 57.1%, and 44.1% at frame rates of 0.3, 3.8 and 2.2 fps respectively. To the best of our knowledge, for the first time, we achieve realtime video object detection on mobile with reasonably good accuracy.

6 Discussion

In this paper, we propose a light weight network for video object detection on mobile devices. We verified that the principals of sparse feature propagation and multi-frame feature aggregation also hold at very limited computational overhead. A very small flow network, Light Flow, is proposed. A flow-guided GRU module is proposed for effective feature aggregation.

A possible issue with the current approach is that there would be short latency in processing online streaming videos. Because the recognition on the key frame is still not fast enough. It would be interesting to study this problem in the future.

[1] the input image resolution of the flow network is kept to be half of the resolution of the image recognition network.

References

1. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: CVPR. (2017) 1, 3
2. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. (2014) 1, 3
3. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: ECCV. (2014) 1, 3
4. Girshick, R.: Fast r-cnn. In: ICCV. (2015) 1, 3
5. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NIPS. (2015) 1, 3, 6, 8, 9
6. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: Object detection via region-based fully convolutional networks. In: NIPS. (2016) 1, 3, 9, 14
7. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR. (2017) 1, 3
8. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV. (2017) 1, 3
9. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: ICCV. (2017) 1, 3
10. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV. (2016) 1, 3, 13
11. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. arXiv preprint arXiv:1612.08242 (2016) 1, 3, 14, 15
12. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016) 1
13. 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) 1, 3, 4, 6, 8, 9, 10
14. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. arXiv preprint arXiv:1707.01083 (2017) 1, 3
15. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR. (2016) 1, 3, 13
16. Redmon, J.: Darknet: Open source neural networks in c. http://pjreddie.com/darknet/ (2013–2016) 1, 14, 15
17. Wong, A., Shafiee, M.J., Li, F., Chwyl, B.: Tiny ssd: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection. arXiv preprint arXiv:1802.06488 (2018) 1, 14
18. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015) Software available from tensorflow.org. 2, 10
19. Zhu, X., Xiong, Y., Dai, J., Yuan, L., Wei, Y.: Deep feature flow for video recognition. In: CVPR. (2017) 2, 3, 4, 5, 6, 10, 11

20. Zhu, X., Wang, Y., Dai, J., Yuan, L., Wei, Y.: Flow-guided feature aggregation for video object detection. In: ICCV. (2017) 2, 3, 4, 6, 8
21. Zhu, X., Dai, J., Yuan, L., Wei, Y.: Towards high performance video object detection. arXiv preprint arXiv:1711.11577 (2017) 2, 3, 4, 6, 8, 11, 12
22. Sifre, L., Mallat, P.: Rigid-motion scattering for image classification. PhD thesis, Citeseer (2014) 2, 6
23. Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., Sun, J.: Light-head r-cnn: In defense of two-stage object detector. arXiv preprint arXiv:1711.07264 (2017) 2, 6, 8, 9, 13, 14
24. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR. (2015) 3
25. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. (2015) 3
26. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016) 3
27. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261 (2016) 3
28. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR. (2017) 3
29. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: CVPR. (2017) 3
30. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: CVPR. (2017) 3
31. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. arXiv preprint arXiv:1708.02002 (2017) 3
32. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., v.d. Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: ICCV. (2015) 4, 5, 6, 10, 11
33. Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Distill (2016) 6
34. Evan Shelhamer, Jonathan Long*, T.D.: Fully convolutional models for semantic segmentation. TPAMI (2016) 6
35. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. (2015) 6
36. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: ICML. (2013) 6
37. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: CVPR. (2017) 6, 11
38. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 6
39. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014) 8
40. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: EMNLP. (2014) 8
41. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8) (1997) 1735–1780 8

42. Elman, J.L.: Finding structure in time. Cognitive science **14**(2) (1990) 179–211 8
43. Ballas, N., Yao, L., Pal, C., Courville, A.: Delving deeper into convolutional networks for learning video representations. In: ICLR. (2016) 8
44. Liu, M., Zhu, M.: Mobile video object detection with temporally-aware feature maps. arXiv preprint arXiv:1711.06368 (2017) 8, 14
45. Xingjian, S., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.c.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: Advances in neural information processing systems. (2015) 802–810 8
46. Li, Z., Gavves, E., Jain, M., Snoek, C.G.: Videolstm convolves, attends and flows for action recognition. arXiv preprint arXiv:1607.01794 (2016) 8
47. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., Li, F.F.: Imagenet large scale visual recognition challenge. In: IJCV. (2015) 8, 9, 15
48. Kang, K., Li, H., Yan, J., Zeng, X., Yang, B., Xiao, T., Zhang, C., Wang, Z., Wang, R., Wang, X., Ouyang, W.: T-cnn: Tubelets with convolutional neural networks for object detection from videos. arXiv preprint arxiv:1604.02532 (2016) 9
49. Lee, B., Erdenee, E., Jin, S., Nam, M.Y., Jung, Y.G., Rhee, P.K.: Multi-class multi-object tracking using changing point detection. In: ECCV. (2016) 9
50. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. arXiv preprint arXiv:1801.04381 (2018) 14, 15
51. Shafiee, M.J., Chywl, B., Li, F., Wong, A.: Fast yolo: A fast you only look once system for real-time embedded object detection in video. arXiv preprint arXiv:1709.05943 (2017) 14, 15
52. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**(2) (2010) 303–338 15
53. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, Springer (2014) 740–755 15