REAL-TIME OBJECT DETECTION BY FEATURE MAP FORECAST FOR LIVE STREAMING VIDEO

Masato Fujitake[†] *and Akihiro Sugimoto*[‡]

[†]Dept. of Informatics, The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan
[‡]National Institute of Informatics, Tokyo, Japan
{fujitake, sugimoto}@nii.ac.jp

ABSTRACT

This paper proposes a method that jointly learns to detect objects at the current frame and forecast the next frame's future feature map. Previous offline detectors have shown the effectiveness of utilizing future information in video object detection; however, we cannot take such an approach when dealing with live streaming videos. In contrast, we utilize the forecast feature map with the current and past frame feature maps for object detection, where forecast feature maps are learned using observation of the present and past frames. To maintain a reliable forecast, we introduce a scheduler network, which decides whether we use the forecast feature map as input or extract the feature map from the next frame. Evaluations of our proposed model on the ImageNet VID dataset demonstrate the superior performance of our model against the public benchmark at similar architectures, with achieving 65.7% mAP at 38.9 fps.

Index Terms— feature map forecast, scheduler, video object detection, deep learning.

1. INTRODUCTION

Video object detection has received increasing attention as it sees immense potential in real-world applications. Nevertheless, extending successful image-based object detectors [1] to the video domain remains challenging due to motion blur, sudden occlusions, and rare pose. To improve accuracy, most of the prior work attempts to leverage the unique characteristic of videos [2, 3], which is the temporal consistency. It is shown in [3, 4] that the usage of current and past frames, as well as future frames, can stabilize the detection. However, considering the application to live streaming videos such as smartphones and robotics, we cannot take advantage of future frames. Therefore, some studies [5, 6, 7] have attempted to stabilize the detection with only past and present information.

Recent psychology literature has shown that humans build a mental image of the future, including future actions before initiating muscle movements or motor controls, and this mental image affects the current results [8]. Inspired by this insight, we introduce the concept of the forecast as an exploitation of the future to improve detection accuracy in live-stream videos. Figure 1 intuitively shows the characteristic of our approach against the existing one. Unlike other models that exploit both past and current information(Fig 1(a)), our model (Fig 1(b)) forecasts the feature map at the next frame and utilizes it with the current and past feature maps. We accomplish the proposed model by jointly learning to forecast the future feature map and object detection.

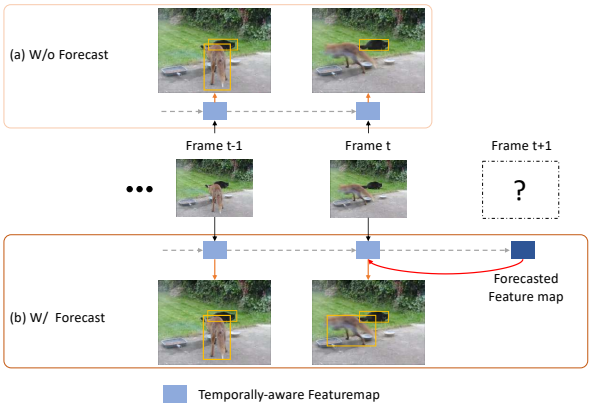

Fig. 1. Video object detection through the anticipated feature map. Current live streaming video object detector approaches (a) store historical information of feature maps to acquire stable detection results. In our proposed model (b), we jointly learn the future feature map prediction to support the detection task at the current frame.

Detection from forecast feature maps reduces the processing cost of the backbone and thus increases processing speed, on the one hand. On the other hand, we have to load images in the video at appropriate timing to maintain the reliability of forecast feature maps, which is difficult to determine in advance. For this purpose, we propose a scheduler network that decides whether we read the next actual frame, or we exploit the forecast feature map. In this way, our model improves the processing speed by using the forecast feature map without significant performance loss. Experiments on ImageNet VID dataset show that our method achieves 65.7 mAP, outperform-

978-1-6654-3864-3/21/$31.00 ©2021 IEEE

ing competitive state-of-the-art methods.

2. RELATED WORK

Object detection in videos: Due to the huge variety of videos under different scenarios, it is not trivial to generalize image-based detector's success into the video domain. A primary focus of recent methods towards addressing video object detection is to improve detection performance by exploiting the temporal information. Such methods could be roughly categorized into off-line and live-stream methods.

The off-line video object detectors allow us to exploit all the available information, including side [9] or future information[10, 3], to stabilize the detection accuracy at the current frame. Indeed, these methods have shown improved detection performance by leveraging complementary and future information. However, their effectiveness is limited due to their inability to operate in a real-time& online fashion.

On the other hand, live-streaming object detection methods have been studied in recent years under application scenarios such as smartphones and robotics. In [11, 12], key-frame arrangement is introduced to fasten detection by using flow networks or correlation map. Incorporating temporal information inside feature maps [5, 6, 7] is viable for reasonable detection accuracy with running in real-time (>25 FPS). Indeed, our model has a similar structure to the aforementioned methods but is different in that it forecasts and exploits future information.

Future predictions in videos: Forecasting the future in video content is mainly explored in the next-frame video prediction task, which tries to predict what happens next in images or a few frames. For video prediction, a convolutional LSTM [13] is proposed to store past information in a convolutional neural network, and developed further inspired by the idea of neuroscience [14]. While these studies aim to generate precise future images themselves, our proposed method focuses on generating the future feature map that is effective for detection.

3. PROPOSED METHOD

Our goal is to produce frame-by-frame detection $\{D_t\}_{t=1}^{T}$ for a given live streamed video with the length of T, where D_t is a list of bounding box locations and class predictions corresponding to the frame at time t, i.e., I_t. Note that in a live streaming setting, detection D_t is generated using only frames up to t. Normally, the object detection model can be viewed as a composed function $D_t = N_{\mathrm{det}}(N_{\mathrm{feat}}(I_t))$, where N_{feat} and N_{det} represent a feature extractor and an object detector model such as SSD [15]. We use the recurrent-network based SSDLite architecture [5] as the baseline model and insert our proposed modules, i.e., encoder module and scheduler module, into between the extractor and the detector.

Figure 2 depicts the proposed framework dealing with frames at time t and $t+1$. It consists of the feature extractor, the encoder module, the scheduler module, and the object detector. The feature extractor and object detector are the recurrent-network based SSDLite architecture [5]. Our encoder module generates forecast feature maps from the output of the feature extractor while the scheduler module decides whether to leverage the forecast feature map (forecast operation) at the next frame or load a new image (read operation).

3.1. Encoder module for feature map forecast

Our encoder module has two identical encoders: $Encoder_1$ and $Encoder_2$. The architecture of each encoder consists of the spatial attention network [16] followed by the bottleneck LSTM [5]. This allows us to recurrently retain past states and to adapt the limited capacity of the bottleneck LSTM. We stack the two encoders to generate forecast feature map $\acute{F}_{t+1}$ at the next frame and convert it into the feature map at the current frame. $Encoder_1$ takes the role of the forecast while $Encoder_2$ for the conversion.

At time t, $Encoder_1$ receives the feature map F_t from the feature extractor and outputs forecast feature map $\acute{F}_{t+1}$ for the next frame. To train $Encoder_1$, we use the forecast loss so that $\acute{F}_{t+1}$ becomes close to (actual) feature map F_{t+1} (see Section 3.3). $Encoder_2$, on the other hand, receives $\acute{F}_{t+1}$ and outputs $\hat{F}_t$ as the converted feature map at time t. $Encoder_2$ incorporates the temporal information into the feature map, allowing the detector to be aware of the temporal information.

In the inference phase, F_t and $\hat{F}_t$ are then element-wisely averaged to have feature map $\tilde{F}_t$ which is fed to the object detector. By doing so, this simple architecture enables to leverage the forecast feature map and to stabilize the object detection at the current frame.

3.2. Scheduler module

If the forecast feature map is reliably well-generated from the current feature map, we can use it as the alternative of the (actual) feature map (obtained by the feature extractor) for the detection at the next frame. This tends to happen as long as there is no significant change from the current frame to the next frame. However, it is better to use an (actual) feature map if the forecast feature map is not reliable. To determine which way should be taken, we propose the scheduler module. The scheduler module aims to determine whether to utilize the forecast feature map or extract the feature map by loading the actual frame for the next frame detection.

Following [17] and utilizing the correlation of feature maps, we design the architecture of the scheduler module, as shown in Fig. 3. The module receives $\acute{F}_{t+1}$ and $\tilde{F}_t$ and exports the score of 1 or 0 with its confidence, indicating to exploit the forecast feature map (1) (forecast operation) or to read a new image (0) (read operation). If the confidence score

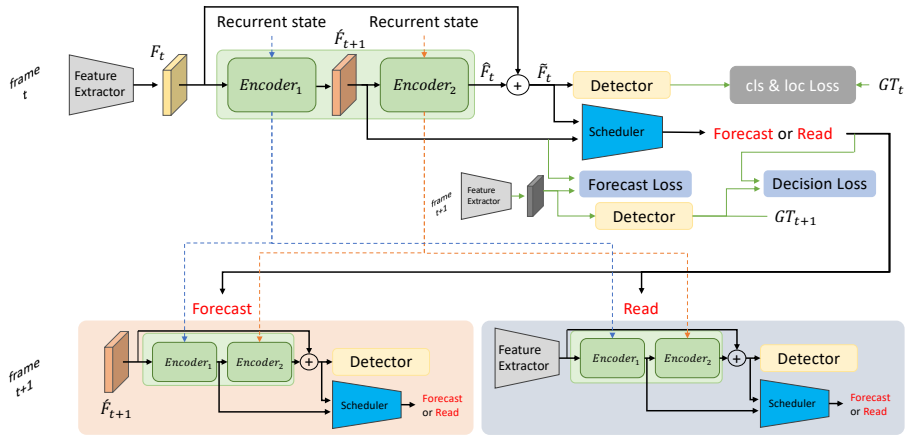

Fig. 2. The architecture of our proposed model. It consists of the feature extractor, the encoder, the scheduler, and the object detector. The encoder predicts the future feature map at the next frame and the current temporally-aware feature map. The scheduler decides to whether exploit the forecast feature map or extract the actual feature map at the next frame. The black arrows show the information flow used both during training and inference, and the green arrows show the flow for training only.

Fig. 3. Scheduler network. The output feature map of the correlation layer is followed by two convolutional layers and a fc layer with a 2-way softmax.

of the forecast operation exceeds threshold p (given beforehand), the forecast operation is executed. Otherwise, the read operation is performed.

The binary classification loss is adopted to train the scheduler module where the ground truth is generated using the next frame detection D_{t+1} and its corresponding ground truth GT_{t+1}.

3.3. Loss function

We design a multi-task objective function to train our model. Namely, we use a localization loss L_{loc}, a classification loss L_{cls}, a forecast loss L_{for}, and a decision loss L_{dec} all together:

$$L = \frac{1}{M}(\alpha L_{\text{loc}} + \beta L_{\text{cls}}) + \gamma L_{\text{for}} + \lambda L_{\text{dec}}, \qquad (1)$$

where M is the number of matched bounding boxes. We exactly follow [15] to define L_{loc} and L_{cls}. Note that We set the hyper parameters to be $\alpha = 1, \beta = 1, \gamma = 1, \lambda = 0.7$ in experiments.

Forecast Loss: To optimize $Encoder_1$ to generate the forecast feature map, we supervise $Encoder_1$ using the mean squared error between the forecast feature map $\acute{F}_{t+1}$ and the (actual) one F_{t+t}. Then, L_{for} can be given as

$$L_{\text{for}} = \frac{1}{n}\frac{1}{m}\frac{1}{l}\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l}||\acute{F}_{t+1}(i,j,k) - F_t(i,j,k)||^2, \qquad (2)$$

where n, m, and l are, respectively, the width, height, and channels of feature maps.

Decision Loss: The decision loss is developed to train the scheduler module. It has the form of a simple binary cross entropy:

$$L_{\text{dec}} = -y_t \log(p_t) - (1 - y_t)\log(1 - p_t), \qquad (3)$$

where, p_t and y_t are the output score of the scheduler module at time t and the ground truth generated using the next frame detection D_{t+1} and its corresponding ground truth GT_{t+1} (See Section 3.4 for details).

3.4. Training

The whole training pipeline is depicted in Fig 2. Fundamentally, our training procedure is the same as the usual video object detection [5, 6, 7]; however, there are three major differences.

The first difference exists in training $Encoder_1$. At time t, $Encoder_1$ forecasts $\acute{F}_{t+1}$ while its ground truth F_{t+1} is available at time $t+1$. Therefore, unlike existing works, we need to generate a batch containing one extra frame in addition to the video length to be trained.

The second difference is how to train the scheduler module. It is most important to train the scheduler module so that its output is (almost) the same as the ground truth y_t. This depends on the accuracy of the object detector, and thus generating y_t during training is required. We use the detection result D_{t+1} and its corresponding ground truth GT_{t+1} to

generate the ground truth for the scheduler. If all the ground-truth bounding boxes GT_{t+1} are matched with D_{t+1} (IOU over 0.7, for example), the y_t is labeled as 1; 0 otherwise.

The last difference is how we combine two feature maps: F_t and $\hat{F}_t$. In the training phase, we do not propagate the recurrent state simply to generate D_{t+1}. We thus use probabilistic connections in the averaging operation in the training phase, which leads to output F_t, $\hat{F}_t$, or their averaged feature map randomly. This allows the object detector not to depend on the temporally-aware feature maps.

All of the training is performed jointly, but the encoder module and the detector are trained first just for stability.

3.5. Testing

The flow of inference follows the black arrows in Fig 2. At the time t, the model simultaneously performs forecasting a feature map and detecting objects at the current frame from $\tilde{F}_t$. Also, using $\acute{F}_{t+1}$ and $\hat{F}_t$, the scheduler module decides to whether use $\acute{F}_{t+1}$ or read the next frame for the detection at the next frame. The video's initial frame is image loading, but the scheduler's function of inferring in subsequent frames will continue.

4. EXPERIMENTS

4.1. Dataset and evaluation setup

We used ImageNet VID dataset [18] for validation. It is a large-scale benchmark for the task of video object detection with 30 categories, consisting of 3,862 videos in the training set and 555 videos in the validation set. Following the protocols widely adopted in [3, 19], we evaluate our method on the validation set and use the mean average precision (mAP) as the evaluation metrics.

4.2. Implementation details

We used PyTorch and a PC with Xeon W-2123 CPU, NVIDIA RTX 2080 Ti GPU, cuDNN v7.6, and CUDA 10.1.

Architecture: We adapt SSDLite [5, 15, 20] architecture to the proposed model. We employ MobileNetV2 [20] as the feature extractor because of its computational efficiency and use the feature map before its average pool layer as F_t.

Data augmentation: In addition to the data augmentation proposed in [15, 7], we employ a more extended one to alleviate the potential over-fitting problem. To augment the motion of objects, we recombine videos by selecting frames at equal intervals instead of training with consecutive frames. To be more specific, for each video in a batch, thinning parameter q (integer) is randomly selected from the interval $\left[0, \min(\lfloor(\frac{l-1}{n}) - 1\rfloor, r)\right]$, where l and n are the video's length and the number of training frames. Since Imagenet VID has some videos whose length is too long, we truncate the video length using r, which is set to be 25. Then, the training video is reconstructed from the original video according to q. This augmentation gives us the improvement of 0.8, resulting in 65.2 in mAP.

Following the idea of [21], we train the model without the thinning operation from the last two epoch. This operation contributed to an additional 0.5 point increase in accuracy to achieve 65.7 in mAP.

Training details: Our training procedure consists of two phases: (1) we pre-train our baseline model following the protocols [5, 3] with additional ImageNet DET dataset [18]. (2) We injected the encoder and scheduler modules into the baseline model while we randomly initialized the weights of the additional modules. We train the model on sequences of 10 frames and use a batch size of 12 and SGD. The encoder module, the backbone, and detector are trained with an initial learning rate of 10^{-4} and 10^{-2}, respectively, and their decay rate of 0.1 at the 18th, 30th epochs. From the 25th epoch, the scheduler module is involved in training with an initial learning rate of 10^{-3} and a decay rate of 0.1 at the 30th, the 35th epoch. We then trained all the weights together in an end-to-end manner until the end of training at the 40th epoch.

4.3. Comparison with state-of-the-art methods

We compared our proposed model with state-of-the-art methods for real-time and live streaming video object detection. They are Chen et al. [6], Zhu et al. [11], and Liu et al. [7]. In this comparison, the threshold of the scheduler module was set to 1.0.

As shown in Table 1, the proposed model achieves the best performance. It achieves 65.7% mAP, 0.3% higher than the strongest competitor [6], which uses VGG-16, a stronger feature extractor. Compared with [7], which uses the same feature extractor as ours, the proposed method achieved 4.3 points improvement. This is mainly thanks to the introduction of the feature map forecast and our end-to-end joint training.

Table 1 also shows the runtime of the methods. Our method runs at about 39 fps, achieving the real-time level. We see that our method runs 12 fps faster than Chen et al. [6] in the same GPU proposal. We note that [7, 11] are developed for mobile devices, and thus runtime comparison with them is just for a reference.

Figure 4 visualizes object detection results and outputs of the scheduler module where frames used by the forecast feature maps are surrounded by the red rectangle. The scheduler module tends to leverage the forecast feature map when the motion of objects is easy to predict while frequently decides to read new images on difficult scenes. We confirm that reasonable detection is realized using forecast feature maps.

4.4. Detailed analysis

We conduct experiments to evaluate the impact of key components of our model on the final performance.

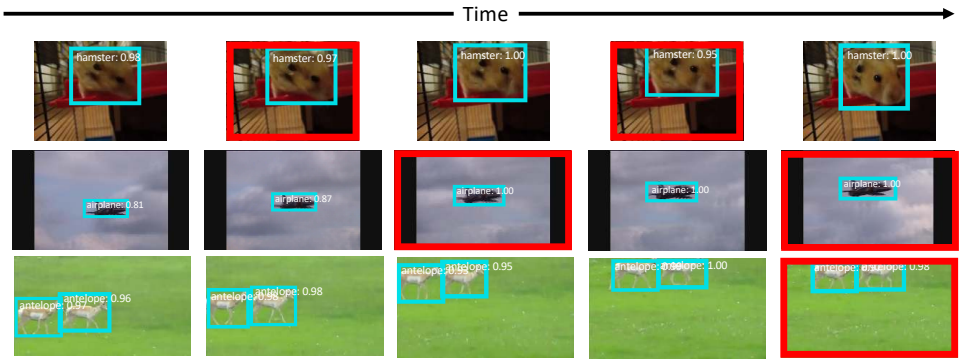

Fig. 4. Visualization of example detection and the corresponding scheduler results on Imagenet VID validation (best view in color). We set $p = 0.5$ in the scheduler module. Frames where the forecast feature map is used are specified in red; otherwise the real frame is adopted.

Table 1. Perfomance comparison with state-of-the-art end-to-end video object detection models on ImageNet VID validaton set. α is the hyper parameter of MobileNet.

Methods	Backbone	mAP	Device	FPS
Chen et al. [6]	VGG-16	65.4	Titan X	27
Zhu et al. [11](α = 1.0)	MobileNet	61.2	Mate 8	13
Liu et al. [7](α = 1.0)	MobileNetV2	61.4	Pixel 3	27
Ours (α = 1.0)	MobileNetV2	**65.7**	RTX 2080 Ti	39

Effectiveness of forecast: We set $p = 1.0$ in the scheduler module and generated two ablation models: the model w/o forecast and the model w/o combination. The model w/o forecast is obtained by dropping the forecast training and the model w/o combination is obtained by dropping the skip connection between F_t and $\hat{F}_t$. We remark that the encoder module of the model w/o forecast is used only to stabilize the feature map from the past, resulting in similar to methods [5, 6]. We also remark that in the model w/o combination, $\hat{F}_t$ is directly fed to the object detector.

Performances of the ablation models and the baseline model are illustrated in Table 2. Note that we follow a detailed evaluation metric [3] to evaluate the performance on the categories of slow, medium, and fast objects, where these three categories are divided by their average IoU scores between objects across nearby frames.

From Table 2, we see that simultaneously training the forecast and the detection improves the overall accuracy by 1.6 points. This gain mainly comes from the Fast category (2.9 point improvement). We thus reason that the larger the object's movement, the more important it is to forecast the future state. We also see that only forecasting the future alone is not sufficient. Indeed, we observe that from model w/o combination, utilizing F_t for the feature maps to be fed to the object detector is also essential.

Effectiveness of scheduler module: Figure 5 illustrates mAP and fps under different threshold p in the scheduler module. Note that "alternates" indicates the model using a fixed scheduling rule where the model reads the image at every odd frame and utilizes forecast feature maps at every even frame. We confirm that lowering p accelerates the processing speed while dropping the accuracy. This is because the model tends not to compute forecast feature maps from the image. We also confirm that the adaptive scheduling is superior to the fixed rule. We thus conclude that the scheduler module has the capability of controlling the accuracy and speed in a flexible way.

Figure 6 shows the errors in terms of the heat map between actual feature maps and forecast feature maps when using the scheduler module or "alternates". The heat map turns more yellow when errors become large. We see that "alternates" results in more errors. This can be understood because it does not have enough uptake. By contrast, the scheduler module takes on the actual images until they are stable, so the errors are smaller. Fig. 6 also shows that the errors are more likely to occur near target objects having uncertainty for their future locations.

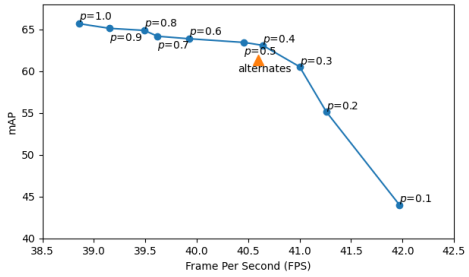

Fig. 5. mAP v.s. FPS trade-off comparison under different threshold p in the scheduler module.

5. CONCLUSION

We have presented a video object detection model that jointly learns to detect objects in the current frame and forecast the next frame's feature map. Although existing offline methods have shown the usefulness of utilizing future information for video object detection, future information cannot be used for

Fig. 6. Visualization of the feature activation error between the predicted feature map and the actual feature map at the next time-step. Frames that use the future forecast feature map for detection are surrounded by the red rectangle ($p = 0.5$).

Table 2. Effectiveness of components in the proposed model.

Methods	Current information	Forecast	mAP	Slow(mAP)	Medium(mAP)	Fast(mAP)
Baseline	-	-	60.6	68.7	58.8	41.6
(a) Complete model	✓	✓	**65.7**	**73.9**	**64.4**	**47.4**
(b) Model w/o forecast	✓	-	64.1	72.5	62.7	44.5
(c) Model w/o combination	-	✓	62.4	70.8	61.0	43.9

live-stream detectors. We thus introduced (1) forecasting the future in terms of the feature map and (2) adaptive scheduling to increase the processing speed while keeping accuracy in detection. The effectiveness of our proposed model was supported by validation on the ImageNet VID dataset.

6. REFERENCES

[1] G. Ross, "Fast r-cnn," in *ICCV*, 2015, p. 1440–1448.

[2] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *CVPR*, 2017, pp. 4141–4150.

[3] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *ICCV*, 2017, pp. 408–417.

[4] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang, "T-cnn: Tubelets with convolutional neural networks for object detection from videos," *TCSVT*, vol. 28, no. 10, pp. 2896–2907, 10 2018.

[5] M. Liu and M. Zhu, "Mobile video object detection with temporally-aware feature maps," in *CVPR*, 2018, pp. 5686–5695.

[6] X. Chen, Z. Wu, and J. Yu, "Tssd: Temporal single-shot detector based on attention and lstm," in *IROS*, 2018.

[7] M. Liu, M. Zhu, M. White, Y. Li, and D. Kalenichenko, "Looking fast and slow: Memory-guided mobile video object detection," in *ICCV*, 2019.

[8] L. Moritz and K. Stefan, "Toward a general psychological model of tension and suspense," *Frontiers in Psychology*, vol. 6, pp. 79, 2015.

[9] K. Kai, L. Hongsheng, X. Tong, O. Wanli, Y. Junjie, LXihui, and W. Xiaogang, "Object detection in videos with tubelet proposal networks," 2017, pp. 889–897.

[10] W. Han, P. Khorrami, T. Le Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang, "Seq-nms for video object detection," *ArXiv*, vol. abs/1602.08465, 2016.

[11] X. Zhu, J. Dai, X. Zhu, Y. Wei, and L. Yuan, "Towards high performance video object detection for mobiles," *ArXiv*, vol. abs/1804.05830, 2018.

[12] W. Yang, B. Liu, W. Li, and N. Yu, "Tracking assisted faster video object detection," in *ICME*, 2019, pp. 1750–1755.

[13] S. Xingjian, C. Zhourong, W. Hao, Y. Dit-Yan, W. Wai-kin, and W. Wang-chun, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *NIPS*, 2015, pp. 802–810.

[14] W. Lotter, G. Kreiman, and D. D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," *ArXiv*, vol. abs/1605.08104, 2016.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016.

[16] W. Sanghyun, P. Jongchan, L. Joon-Young, and K. I. So, "Cbam: Convolutional block attention module," in *ECCV*, 2018.

[17] H. Luo, W. Xie, X. Wang, and W. Zeng, "Detect or track: Towards cost-effective video object detection/tracking," *ArXiv*, vol. abs/1811.05340, 2018.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[19] X Zhu, J. Dai, L. Yuan, and Y. Wei, "Towards high performance video object detection," in *CVPR*, 2018, pp. 7210–7218.

[20] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.

[21] T. Hugo, V. Andrea, D. Matthijs, and J. Herve, "Fixing the train-test resolution discrepancy," in *NIPS*, 2019, vol. 32, pp. 8252–8262.