2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

Temporal Feature Enhancement Network with External Memory for Object Detection in Surveillance Video

Masato Fujitake
Dept. of Informatics
The Graduate University for Advanced Studies, SOKENDAI
Tokyo, Japan
Email: fujitake@nii.ac.jp

Akihiro Sugimoto
National Institute of Informatics
Tokyo, Japan
Email: sugimoto@nii.ac.jp

***Abstract*—Video object detection is challenging and essential in practical applications, such as surveillance cameras for traffic control and public security. Unlike the video in natural scenes, the surveillance video tends to contain dense and small objects (typically vehicles) in their appearances. Therefore, existing methods for surveillance object detection utilize still-image object detection approaches with rich feature extractors at the expense of their run-time speeds. The run-time speed, however, becomes essential when the video is being streamed. In this paper, we exploit temporal information in videos to enrich the feature maps, proposing the first temporal attention based external memory network for the live stream of video. Extensive experiments on real-world traffic surveillance benchmarks demonstrate the real-time performance of the proposed model while keeping comparable accuracy with state-of-the-art.**

I. Introduction

Object detection in images or videos is one of the fundamental problems in computer vision. In particular, object detection in videos has been in interest lately since it has a wide range of applications, including robotics, vehicles, smartphone, and surveillance systems. Thanks to deep convolutional neural networks (CNNs), still-image object detectors[1], [2], [3], [4], [5] provide reasonably high-performances in detection in recent years. Directly applying these detectors to videos, however, faces new challenges such as motion blur, occlusion, out-of-focus, and compression artifacts. Therefore, video object detectors[6], [7], [8], [9], [10] have been actively studied to improve the detection performance. Indeed, the number of research on video object detection has recently increased, especially since the introduction to the ImageNet VID[11] challenge, which is on video object detection in natural scenes. This challenge raises additional problems such as the vast size of a dataset, motion blur, partial occlusion, and low quality of video clips. Although this dataset promotes research on object detection in videos, it is not valid for all conditions.

Surveillance object detection, on the other hand, has a significant impact on efficient and effective transportation systems. One of the great surveillance object detection challenges is on the UA-DETRAC detection dataset [12] published as a large scale benchmark for vehicle detection in video. Compared to the ImageNet VID dataset, the UA-DETRAC dataset offers smaller objects and higher image quality. To detect small vehicles in high-resolution frames, some works[13], [14], [15], [16] have been proposed using strong feature extractors[17], [5]. They improve their detection accuracy through enriching feature extractors developed for still-image object detection at the expense of running time; they run in 4-10 fps. When the surveillance video is being streamed, the run-time speed becomes essential for lively detecting objects. Therefore, developing a method that meets both accuracy and run-time speed is desired.

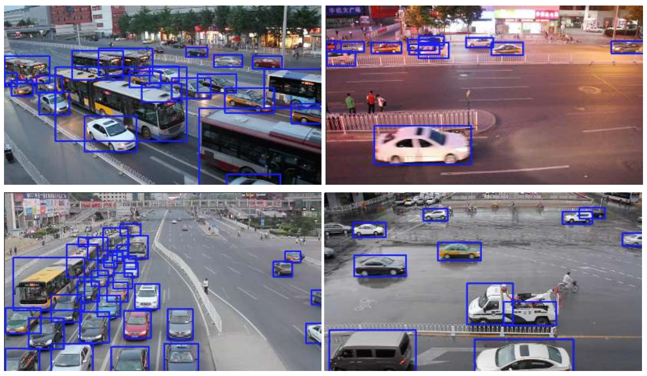
Fig. 1: Examples of the detection results by TFEN on UA-DETRAC. A bounding box is plotted if its confidence score is larger than 0.4.

To achieve accurate real-time object detection in surveillance videos, we propose an encoder-decoder based network, Temporal Feature Enhancement Network (TFEN), that utilizes (i) spatial information from coarse spatial features and (ii) temporal information available from the live stream of video data. The encoder consists of recurrent convolutional units dealing with both spatial and temporal features. The decoder has the external memory to store feature maps generated in the past to utilize temporal information, and exports densely aggregated feature maps using attention weights. In this way, TFEN enriches coarse features with spatial and temporal information so that the trade-off between accuracy and speed is considerably enhanced.

TFEN is connected to the lightweight feature extractor and object detector. In our experiment, we employ MobileNetV2 as the feature extractor and Cascade R-CNN as the object detector, demonstrating real-time performance while keeping comparable accuracy with state-of-the-arts. Fig. 1 shows some detection results by TFEN on the UA-DETRAC. We see that TFEN successfully detects most of the vehicles in different appearances, especially even when heavy and partial occlusions occur, also when cars are far away from the camera.

Unlike other methods [7], [8], [9] that utilize optical flow to warp feature maps across neighboring frames, TFEN relies on only appearance information from image feature extractors. The architecture of TFEN is thus simple to design and enables us to efficiently optimize its loss function because we do not suffer from any disturbance caused by differences between appearance feature and optical flow feature.

II. Related work

A. Generic Object detection

Video object detection has received increased attention recently, where how to utilize temporal information is critical. Many existing methods utilizing temporal information in video can be separated into the box-level approach [6], [18], [19], [20], [21], [22] and the feature-level approach [7], [8], [9], [23].

The box-level approach focuses on how to improve detection accuracy considering temporary consistency within a tracklet. Seq-NMS [18] links high-confidence predictions into sequences across the entire video. TCNN [19], [20], [21] uses optical flow to map detection into neighboring frames and suppresses low-confidence predictions. Since they require side information or access to frames from past to future, they cannot deal with the live video stream.

There are two ways in the feature-level approach. One is to exploit optical flow to have pixel-to-pixel correspondence among nearby frames such as Deep Feature Flow (DFF) [7] and Flow-guided Feature Aggregation (FGFA) [8]. DFF accelerates detection by running the detector on sparse key-frames and using optical flow to generate the remaining feature maps but cannot apply to the live stream of video. FGFA improves detection accuracy by warping and averaging features from nearby frames with adaptive weighting. The other is to store temporal information inside feature maps using a sequential network [23], [24]. Inspired by the above methods of the feature-level approach, we combine feature aggregation [8] and temporally-aware feature maps [23].

B. Surveillance Object Detection

Detecting objects such as vehicles in surveillance videos has its unique challenges because relatively small objects appear in high-resolution and false positives easily appear in background regions.

YOLOv3_SPP [15] and MSVD_SPP [16] are proposed to improve detection accuracy in large scale variation with additional prediction layers and spatial pyramid pooling based on YOLOv3 network [5]. Evolving Boxes (EB) [14] improves its region proposal network with a cascade strategy, which refines object boxes. GP-FRCNN [13], on the other hand, proposes geometric proposals for Faster R-CNN [25], whereby they re-rank the geometric object proposals with an approximate geometric estimate of the scene to remove false positives. Foreground Gating and Background Refining Network (FG-BR Net) [26] incorporates the background subtraction method to ignore a non-interested region efficiently for false-positive elimination. None of these methods, however, performs in real-time because they focus on improving detection accuracy even at the expense of running time. While our method performs comparable detection accuracy thanks to features enriched in TFEN, it keeps the real-time inference on a moderate commercial GPU.

III. Proposed model

A. Architecture

Figure 2 shows the overall architecture of our proposed TFEN at previous time $t-1$ and current time t where TFEN is connected to the feature extractor and the object detector. The skip connection is employed from the feature extractor to the object detector to retain information from the feature extractor. We denote by F_t the feature maps extracted from the feature extractor at time t, which is fed to TFEN. As the lightweight feature extractor, we employ MobileNetV2 [27] because its computational cost is low.

Our proposed TFEN receives extracted feature maps F_t's and enriches them to pass to the object detector (see the rectangle area in pink in Figure 2). It consists of the spatiotemporal encoder and the temporal attention decoder having the external memory. For the frame at time t, the spatiotemporal encoder creates temporally-aware feature maps $\tilde{F}_t$ using recurrent convolutional neural networks [23]. It exploits the spatial attention module BAM [28] and ConvGRU [29]. BAM refines feature maps F_t by inferring simple spatial and channel attention while ConvGRU uses spatiotemporal information for temporally-aware feature maps. The temporal attention decoder, on the other hand, utilizes both the current time feature maps F_t and the external memory which stores the temporally-aware feature generated in the past m frames: $\{\tilde{F}_i\}_{t-m+1 \leq i \leq t} := \{\tilde{F}_t, \tilde{F}_{t-1}, \dots \tilde{F}_{t-m+1}\}$. Then it outputs densely aggregated feature maps according to the attention coefficient calculated in the decoder. We adopt a residual learning scheme [30] to facilitate the gradient flow. The aggregated feature maps (called enhanced feature maps) are fed to the object detector. We detail the encoder and the decoder in the following subsections.

B. Spatiotemporal encoder

Our encoder is designed for extracting spatiotemporal information from feature maps F_t's coming from the feature

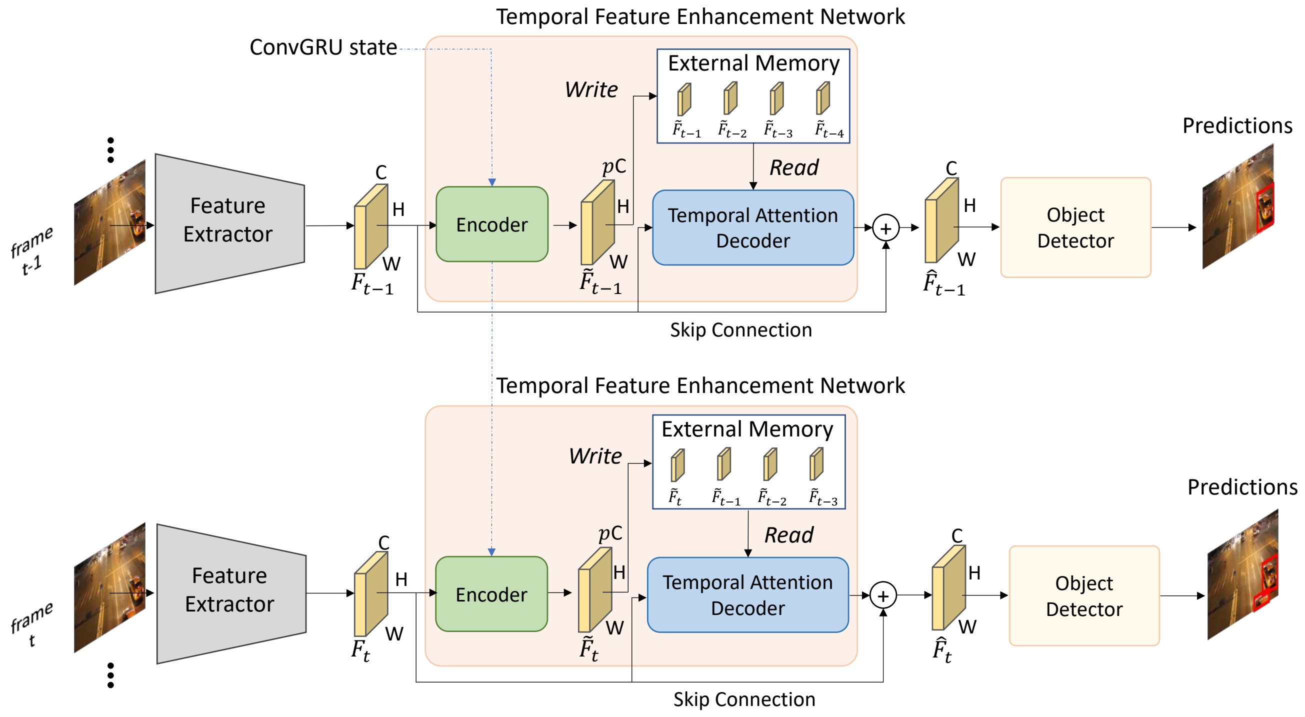

Fig. 2: Architecture of our proposed TFEN.

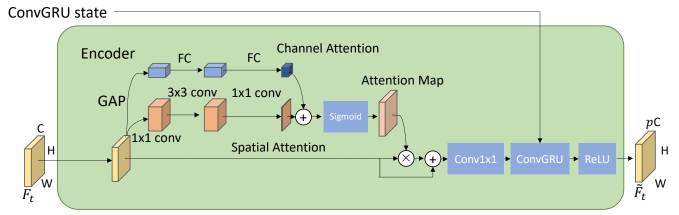

Fig. 3: Architecture of the spatiotemporal encoder.

extractor. It consists of BAM [28] and ConvGRU [29] (Figure 3). BAM is a simple and effective attention module, which infers an attention map along two separate pathways: channel and spatial. ConvGRU is recurrent convolutional units that are able to deal with both spatial and temporal features.

First, for given input feature maps $F_t \in \mathbb{R}^{C\times H\times W}$ at time t, BAM [28] infers spatial attention maps $M(F_t) \in \mathbb{R}^{C\times H\times W}$ where C, H, W denote the number of channels, the horizontal and vertical sizes of feature maps, respectively. The refined feature maps $F_t{}'$ are computed as

$$F_t{}' = F_t + F_t \otimes M(F_t), \tag{1}$$

where $\otimes$ denotes the element-wise multiplication. We introduce the compressibility value p to reduce the channels of F_t' by applying the 1×1 convolution operation to have $F_t'' \in \mathbb{R}^{pC\times H\times W}$. We then feed F_t'' into ConvGRU to store temporal information with hidden state. See [29] for details on ConvGRU. The output of ConvGRU is fed into the Rectified Linear Unit (ReLU) to output temporally-aware feature maps $\tilde{F}_t$ which are saved in the external memory.

C. *Temporal attention decoder*

The temporal attention decoder is most important, and its architecture is shown in Figure 4. Our temporal attention operation is similar to dense feature aggregation [7]. At time t, the decoder performs multiple dense feature maps aggregation by summing all the the temporally-aware feature maps $\{\tilde{F}_i\}_{t-m+1\leq i\leq t}$ based on soft attention weights. The weights determine which time of temporally-aware feature maps should be focused.

Soft attention weights for time are calculated through the tensor computed from $\tilde{F}_t$ and current time feature maps F_t. The 1×1 convolution is first applied to F_t to adjust its size, and then its output is concatenated with $\tilde{F}_t$ in the channel direction. Transforming operation with the stacked convolution layers and ReLU is applied, and then global average pooling

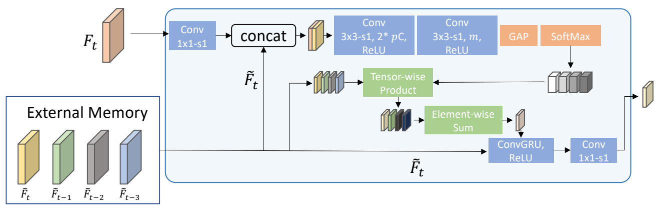

Fig. 4: Architecture of the temopral attention decoder.

(GAP) [31] and the softmax function are applied to have soft attention weights for time. The output channel of the first convolution layer and the second one depicted in Figure 4 are pC and m, respectively.

The computed soft attention weights are used for tensor-wise products with all $\tilde{F}_i$ in $\{\tilde{F}_i\}_{t-m+1\leq i\leq t}$ stored in the external memory. Then, the element-wise summation of tensors and $\tilde{F}_t$ are fed to ConvGRU followed by ReLU. The hidden state of ConvGRU is initialized by $\tilde{F}_t$. The 1×1 convolution operation is next applied to adjust the channels of feature maps to export enhanced feature maps $\hat{F}_t$, which are forwarded to the object detector.

External memory: Our external memory consists of a data buffer and a set of *Write* and *Read* operations to access. The data buffer stores the past temporally-aware feature maps $\{\tilde{F}_i\}_{t-m+1\leq i\leq t}$ where m is the number of frames to be stored.

The data structure inside the memory uses a first-in-first-out queue. Therefore, older temporally-aware feature maps are pushed out over time as new ones are written to the memory. With the *Write* operation, the latest temporally-aware feature maps are en-queued into the buffer after the oldest maps are discarded. The *Write* operation allows the decoder to access all the tenors.

D. Loss function

Since all the modules described above are differentiable, TFEN can be trained in an end-to-end manner. We follow the Cascade R-CNN loss [4] for multi-stage classification and bounding box regression. This is because we employ the conventional cascade R-CNN[4] as the object detector in our experiments.

At each stage, the detector head predicts the classification score and bounding box regression offset for all sampled RoIs. The overall loss function takes the form of multi-task learning:

$$L = \sum_{s=1}^{S}(L^s_{\text{loc}} + L^s_{\text{cls}}), \tag{2}$$

where L^s_{loc} and L^s_{cls} are the losses of the bounding box predictions and classification prediction at stage s, and S is the total number of multi-stages. We follow [4] and set $S = 3$.

IV. Experiments

A. Dataset

The UA-DETRAC dataset [12] contains 100 video sequences corresponding to more than 140,000 frames of real-world traffic scenes. More than 1.2 million vehicles are labeled with bounding boxes in this dataset. The videos are taken at 24 different locations in Beijing and Tianjin in China and recorded at 25 fps, with the resolution of 960×540 pixels.

There are 60 videos in the training set and 40 videos in the test set. However, no validation set is available. We thus split the training set into mini-training and mini-validation sets at the ratio of 8 to 2. We use these mini datasets to determine the hyper-parameters.

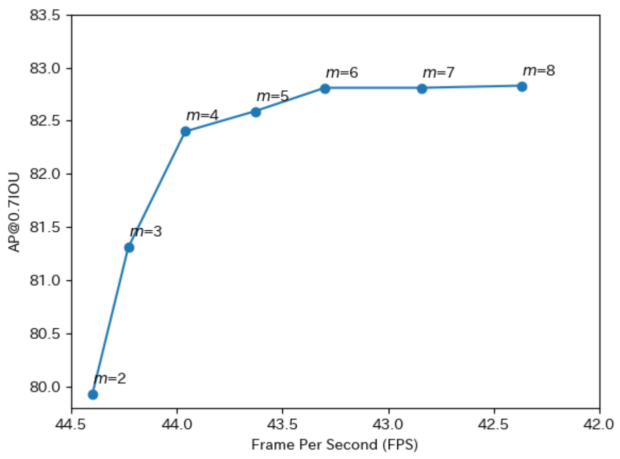

Fig. 5: AP v.s. FPS under different number m of frames to be stored in the external memory.

B. Implementation details

We employ MobileNetV2 [27] as the feature extractor and cascade R-CNN [4] as the object detector. We re-implemented cascade R-CNN [4] network with the pre-trained MobileNetV2 [27] in PyTorch [32] and regard it as our baseline model. We then fine-tuned our baseline model on all the 60 videos in the UA-DETRAC training set. First, we trained the baseline model in 36 epochs using asynchronous gradient descent with 0.9 momentum, 0.0005 weight decay, in the batch size of 4 with 84k static images on 2 GPUs. The initial learning rate was 0.005 and we decreased it by 0.1 after 18 and 30 epochs. For data augmentation, random horizontal flip was adopted during training.

To train TFEN, we initialized the weights of the feature extractor and the object detector with the weights of the fine-tuned baseline model while we randomly initialized the weights of the feature enhancement network. We then trained all the weights together in the end-to-end manner. The initial learning rate was 0.001, and we decreased it in the same way as the baseline model.

We used a PC with Intel 3.9GHz Xeon W-2123 CPU, NVIDIA RTX 2080 Ti GPU with 11 GB Memory, and 64 GB of RAM. The experiments are executed with cuDNN v7.6 and CUDA 10.1. Our proposed TFEN runs in about 29 fps, achieving the real-time level.

C. Number of frames in attention decoder

First, we evaluated the number m of frames to be stored in the external memory. Since 4 frames are maximum using FP32 to accommodate in our GPU memories, we used FP16 in this experiment so that we can accommodate up to 8 frames. We changed m from 2 to 8 and computed the average precision (AP) and fps. We also computed soft attention weights to see temporally-aware feature maps of which frames are really focused to derive the enhanced feature maps. Fig. 5 illustrates

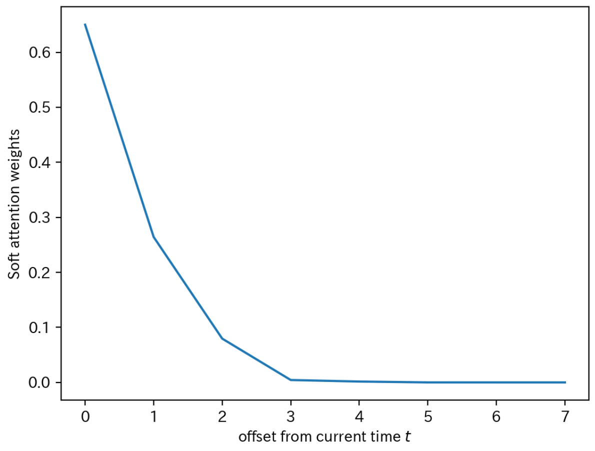

Fig. 6: Soft attention weights used in the temporal decoder.

AP and fps under different m while Fig. 6 shows the average of soft attention weights in overall subset when $m = 8$.

We see that from Fig. 5 the accuracy tends to be improved by increasing the number of frames and is saturated with $m = 6$. The run-time speed, on the other hand, decreases as m increases. We may conclude that $m = 4, 5$ or 6 is good compromise as the trade-off between accuracy and speed.

Figure 6 shows that the weight for the current frame is largest and weights for the last 3 frames are dominant. It also shows that even if we store 8 frames in the external memory, the frames that are really used in the computation are the last 3 or 4 frames. This indicates that storing more than 5 frames in the external memory results in just taking run-time while it does not contribute to improve accuracy. We note that the horizontal axis shows the offset from the current frame, meaning that 0 indicates the current feature map, and 7 indicates the feature map of the last frame in the external memory.

Based on the above observation, we can conclude that $m = 4$ is the best choice in terms of accuracy and speed.

D. Bottleneck dimension

We analyzed impacts of the GRU output channel dimension on accuracy and speed; see in Table I. The compressibility p defines the number of output channels of feature maps in the spatiotemporal encoder. We changed p from 0.5 to 0.1 by 0.1; the maximum amount of the compressibility was 0.5 because of our GPU memory constraint. Accuracy remains almost constant up to $p = 0.4$, then drops. We confirmed that the compressibility p controls the trade-off between detection speed and accuracy of TFEN, and can conclude that $p = 0.5$ is the best choice.

E. Comparison with state-of-the-art results

We set $m = 4$ and $p = 0.5$ according to the above experimental results and compared the average precision (AP) of TFEN with the state-of-the-art methods, see Table II. We remark that we used FP32 because we can accommodate 4 frames in our GPU memories. We trained and evaluated TSSD[24] with UA-DETRAC dataset from the official code, which utilizes temporal information in videos. We note that most of published works in UA-DETRAC benchmark are still-image object detectors. We also remark that CSP [36], RD^2 [37], ExtendNet [37], IMIVD-TF [37] and MYOLO [37] are not available as published papers up to now. Therefore, we used their reported scores and regard them just as reference scores.

TABLE I: AP v.s. FPS under different compressibility p of the output channel dimension.

p	0.5	0.4	0.3	0.2	0.1
AP[%]	82.42	81.62	76.56	74.82	73.41
FPS	29.11	30.31	31.48	33.25	33.39

From Table II, we see that TFEN ranks on the top among all the methods on the easy and sunny subsets. MSVD_SPP[15] achieves best accuracy on almost all subsets while TFEN achieves comparable accuracy with more than 3 times faster. The gap between TFEN and the other methods except for TSSD [24] is significant in speed but not in accuracy. TSSD also runs in real time but its performance is far worse than TFEN.

Figure 7 shows precision-recall curve comparison. As with TFEN, Faster R-CNN and EB have the common point as a two-stage detector; however, we see that TFEN draws a better curve. We also observe that TFEN keeps high-level precision, even though the recall becomes higher.

Figure 8 shows ground-truth and detection results obtained by TFEN, MobileNetV2 based Cascade R-CNN as the baseline model, and TSSD along a sequence of frames on the DETRACT medium subset. The video clip shows that TFEN and TSSD, which exploit temporal information in their models, detect the occluded car behind the bus. However, the baseline model, which is a still-image object detector fails to detect it. This confirms the importance of using temporal information.

F. Ablation study

We evaluated the effectiveness of each component in TFEN to show its necessity. We removed each component of TFEN one by one from the complete model to have ablation models. They are the model w/o TAD (temporal attention decoder), the model w/o SK (skip connection), the model w/o SA (spatial attention), and the model w/o TF (temporally-aware feature maps). Note that the baseline model corresponds to the model dropping TFEN. Performances of ablation models and the baseline model are illustrated in Table III.

Temporal attention decoder: Table III (b) shows the ablation results of replacing the temporal attention decoder with a normal decoder without attention mechanism. This replacing decoder consists of a simple stacked ConvGRU and ReLU, which accepts only current-frame feature maps and has no external memories. Therefore, the model w/o TAD cannot exploit past feature maps except the hidden state. It drops

TABLE II: Comparison of AP scores [%] on UA-DETRAC. Bold faces are the top performance on each subset. The bottom bock lists unpublished methods and thus shows just reference scores. (* is tested by ourselves.)

Method	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny	FPS	GPU
DPM[33]	25.70	34.42	30.29	17.62	24.78	30.91	25.55	31.77	0.17	-
ACF[34]	46.35	54.27	51.52	38.07	58.30	35.29	37.09	66.58	0.67	-
R-CNN[1]	48.95	59.31	54.06	39.47	59.73	39.32	39.06	67.52	0.10	Tesla K40
CompACT[35]	53.23	64.84	58.70	43.16	63.23	46.37	44.21	71.16	0.22	Tesla K40
Faster R-CNN[25]	58.45	82.75	63.05	44.25	62.34	66.29	45.16	69.85	11.0	Titan X
GP-FRCNN[13]	76.57	91.79	80.85	66.05	85.16	81.23	68.59	77.20	4.00	Tesla K40
EB[14]	67.96	89.65	73.12	54.64	72.42	73.93	53.40	83.73	11	Titan X
YOLOv3-SPP[16]	84.96	95.59	**89.95**	75.34	**88.12**	**88.81**	77.46	89.46	6-7	Titan Xp
MSVD_SPP[15]	**85.29**	96.04	89.42	**76.55**	88.00	88.67	**78.90**	88.91	9-10	Titan Xp
FG-BR_Net[26]	79.96	93.49	83.60	70.78	87.36	78.42	70.50	89.89	10	Tesla M40
TSSD[24]*	57.16	81.06	62.07	43.14	57.59	63.87	44.98	67.73	31.78	RTX 2080 Ti
TFEN	82.42	**97.40**	88.90	72.18	87.54	82.41	72.32	**90.78**	**29.11**	RTX 2080 Ti
CSP[36]	77.67	93.65	83.67	64.54	89.66	86.81	61.39	80.63	4	Tesla K40
RD^2[37]	85.35	95.80	89.84	76.64	89.67	86.59	78.17	90.49	n/a	Tesla P40
ExtendNet[37]	83.59	95.46	88.75	73.36	86.89	85.05	76.75	90.77	45.45	Titan X
IMIVD-TF[37]	85.67	96.32	91.17	75.45	87.02	88.93	80.60	89.69	1	n/a
MYOLO[37]	83.50	95.15	88.18	73.99	88.58	83.38	77.06	88.37	7	n/a

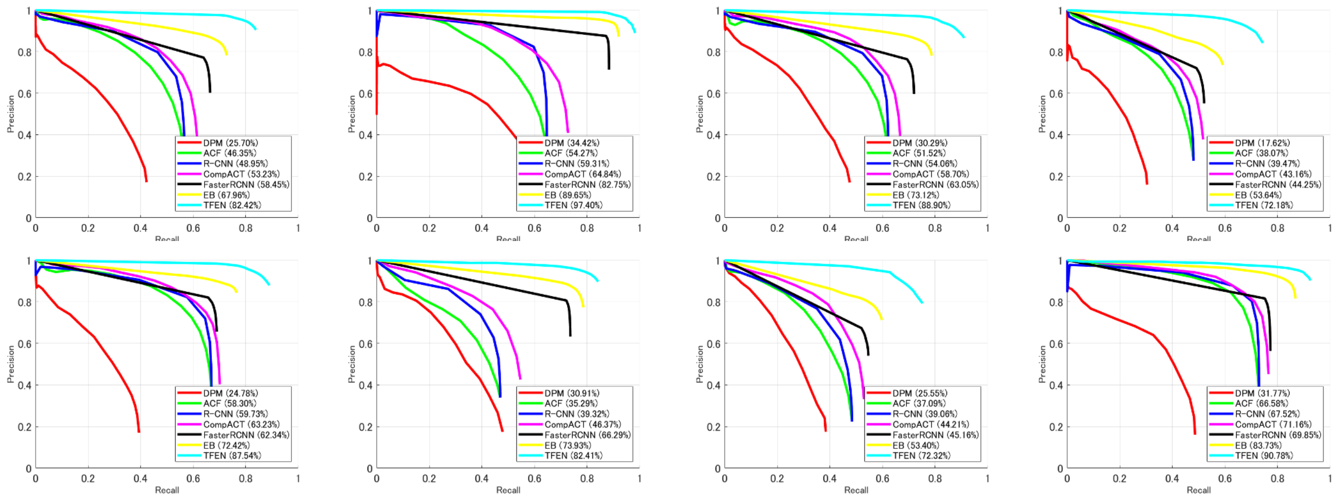

Fig. 7: Comparison of precision-recall curves on UA-DETRAC. Clockwise from top left: Overall; easy; medium; hard; sunny; rainy; night; cloudy sets.

3.16 points of AP for the overall subset. This demonstrates the effectiveness of the temporal attention decoder.

Skip connection: Table III (c) and (f) show that applying skip connection between the feature extractor and the object detector brings consistent gains on all the subsets. Moreover, Table III (a) and (c) reveal that the model w/o SK has lower scores than the baseline model in overall, medium, and hard subsets. We conjecture that the skip connection helps gradient flowing through TFEN and is an essential part of TFEN.

Spatial attention: Table III (d) and (f) show that the spatial attention mechanism used in the spatiotemporal encoder brings additional improvements 1.49%, 0.23%, 2.82%, 5.74% in AP on overall, easy, medium, hard subsets, respectively. This suggests that the spatial attention mechanism is useful for all the subsets, but especially for the hard subset. We confirmed that the hard subset tends to contain dense vehicles or small vehicles. Therefore, we may conclude that the spatial attention mechanism removes useless features around objects and makes feature maps easier to handle in the temporal attention decoder.

Temporally-aware feature maps To verify the necessity of temporally-aware feature maps, we store the feature maps without temporal information in the external memory instead of the temporally-aware feature maps, which is the model of w/o TF. We remark that the model w/o TF uses the encoder only to create attention weights. Table III (e) and (f) show that the temporally-aware feature maps used in the external memory gives additional improvements 3.20%, 2.34%, 4.13%, 6.72% in AP on overall, easy, medium, hard subsets, respectively. This suggests that the temporal information in feature maps is useful for all the subsets. We thus confirmed that both the feature aggregation with temporal attention mechanism and temporal-aware feature maps are necessary for improving the

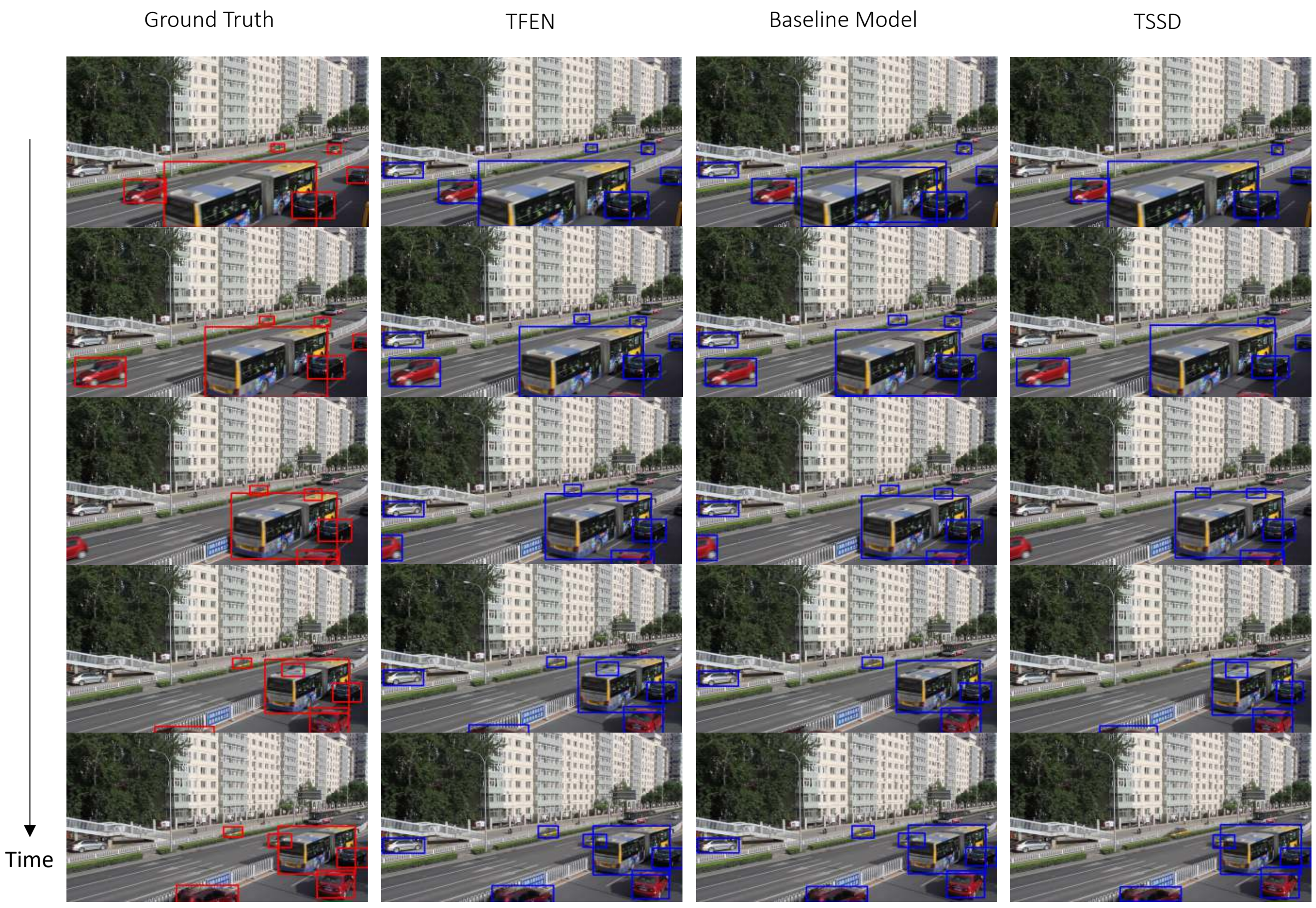

Fig. 8: Example detection results of TFEN, Baseline Model, and TSSD on medium subset.

TABLE III: AP performance [%] of ablation models on UA-DETRAC.

Method	Video	Temporal Attention Decoder	Skip Connection	Spatial Attention	Temporally-aware Feature maps	Overall	Easy	Medium	Hard
(a) baseline model	–	–	–	–	–	73.39	90.92	79.28	60.33
(b) model w/o TAD	✓	–	✓	✓	✓	79.26	95.96	85.83	67.42
(c) model w/o SK	✓	✓	–	✓	✓	72.53	91.26	78.57	59.24
(d) model w/o SA	✓	✓	✓	–	✓	80.93	97.17	86.08	66.44
(e) model w/o TF	✓	✓	✓	✓	–	79.22	95.06	84.77	65.46
(f) (complete) TFEN	✓	✓	✓	✓	✓	82.42	97.40	88.90	72.18

detection accuracy.

Feature maps enhancement: We finally visualize the feature maps before and after TFEN (F_t and $\hat{F_t}$) in Fig. 9 to illustrate how feature maps are enriched. Yellow means higher activation values, whereas dark Mazarin indicates negligible feature activation. We observe that compared with F_t, $\hat{F_t}$ shows stronger responses near regions where vehicles exist, even highly occluded vehicles. We thus see that our feature maps enhancement is practical and improves the detection accuracy.

V. Conclusion

We presented the temporal attention network with the external memory called TFEN for object detection in surveillance video. TFEN exploits spatial and temporal information available to enrich feature maps extracted a lightweight feature extractor and thus achieves real-time performance while keeping comparable accuracy with state-of-the-art methods that do not run in real-time. TFEN demonstrated clear benefits of the attention module based on the external memory, and achieved considerably enhanced trade-off between accuracy and speed.

References

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *CVPR*, 2014, p. 580–587. 1, 6

[2] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” in *NIPS*, p. 379–387. 1

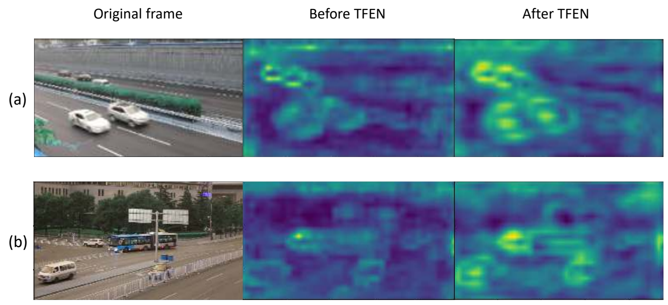

Fig. 9: Feature activation before and after TFEN. For visualization, we summed up the feature maps in the channel direction, then mapped their values to the interval of 0-255, and up-sampled the feature channel by the bi-linear interpolation.

[3] 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. 1

[4] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *CVPR*, 2017, pp. 6154–6162. 1, 4

[5] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018. 1, 2

[6] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *ICCV*, 2017, pp. 3057–3065. 1, 2

[7] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *CVPR*, 2017, pp. 4141–4150. 1, 2, 3

[8] 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. 1, 2

[9] X. Zhu, J. Dai, L. Yuan, and Y. Wei, "Towards high performance video object detection," in *CVPR*, 2018, pp. 7210–7218. 1, 2

[10] H. Deng, Y. Hua, T. Song, Z. Zhang, Z. Xue, R. Ma, N. Robertson, and H. Guan, "Object guided external memory network for video object detection," in *ICCV*, October 2019. 1

[11] 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. 1

[12] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, 2020. 1, 4

[13] S. Amin and F. Galasso, "Geometric proposals for faster r-cnn," in *AVSS*, 2017, pp. 1–6. 1, 2, 6

[14] L. Wang, Y. Lu, H. Wang, Y. Zheng, H. Ye, and X. Xue, "Evolving boxes for fast vehicle detection," in *ICME*, 2017. 1, 2, 6

[15] K. Kim, P. Kim, Y. Chung, and D. Choi, "Multi-scale detector for accurate vehicle detection in traffic surveillance data," *IEEE Access*, vol. 7, pp. 78 311–78 319, 2019. 1, 2, 5, 6

[16] K. Kim, P. Kim, Y. Chung, and D. Choi, "Performance enhancement of yolov3 by adding prediction layers with spatial pyramid pooling for vehicle detection," in *AVSS*, 2018. 1, 2, 6

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. 1

[18] W. Han, P. Khorrami, T. L. 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. 2

[19] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang, 2017. 2

[20] 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," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2896–2907, 10 2018. 2

[21] K. Kang, W. Ouyang, H. Li, and X. Wang, "Object detection from video tubelets with convolutional neural networks," 2016. 2

[22] 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. 2

[23] M. Liu and M. Zhu, "Mobile video object detection with temporally-aware feature maps," in *CVPR*, 2018, pp. 5686–5695. 2

[24] X. Chen, Z. Wu, and J. Yu, "Tssd: Temporal single-shot detector based on attention and lstm," in *IROS*, 2018. 2, 5, 6

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, vol. 1, 2015, p. 91–99. 2, 6

[26] Z. Fu, Y. Chen, H. Yong, R. Jiang, L. Zhang, and X. Hua, "Foreground gating and background refining network for surveillance object detection," *IEEE Transactions on Image Processing*, vol. 28, no. 12, pp. 6077–6090, 12 2019. 2, 6

[27] 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. 2, 4

[28] J. Park, S. Woo, J.-Y. Lee, and I.-S. Kweon, "Bam: Bottleneck attention module," in *BMVC*, 2018. 2, 3

[29] N. Ballas, Yao, C. J. Pal, and A. C. Courville, "Delving deeper into convolutional networks for learning video representations," *CoRR*, vol. abs/1511.06432, 2015. 2, 3

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778. 2

[31] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013. 4

[32] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017. 4

[33] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *CVPR*, 2010, pp. 2241–2248. 6

[34] P. Dollár, R. D. Appel, S. J. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 1532–1545, 2014. 6

[35] Z. Cai, M. Saberian, and N. Vasconcelos, "Learning complexity-aware cascades for deep pedestrian detection," in *ICCV*, 2015, p. 3361–3369. 6

[36] W. Liu, S. Liao, W. Ren, W. Hu, and Y. Yu, "High-level semantic feature detection: A new perspective for pedestrian detection," in *CVPR*, 2019, pp. 5182–5191. 5, 6

[37] S. Lyu, M.-C. Chang, D. Du, W. Li, Y. Wei, M. Del Coco, P. Carcagnì, A. Schumann, B. Munjal, D.-H. Choi *et al.*, "Ua-detrac 2018: Report of avss2018 & iwt4s challenge on advanced traffic monitoring," in *AVSS*. IEEE, 2018, pp. 1–6. 5, 6