Feature Enhancement using Spatio-Temporal Information for Video Object Detection

by

Masato FUJITAKE

Dissertation

submitted to the Department of Informatics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

The Graduate University for Advanced Studies, SOKENDAI
March 2022

ABSTRACT

Video is an essential resource because of its ability to hold space and time information. Therefore, in the field of computer vision, a lot of research is conducted to extract various information, especially object detection in videos. It is expected to be applied to real-world applications such as surveillance cameras and robotics.

Object detection consists of two processes: extracting the feature maps for detection from the video frames and detecting objects from them. In object detection in video, detectors for still images are sometimes applied to each frame. However, it is difficult to achieve stable detection due to apparent changes with time in the video, which leads to fluctuations of detection confidence score, and false-positive and false-negative detection results. Previous research tried to solve them by incorporating temporal information in the detection stage. However, the effect was limited since the feature maps obtained from the frames are deteriorated due to the changes in appearance, and it is difficult to detect objects from them. Therefore, it is essential to enhance feature maps with temporal information before the detection stage.

Research on feature maps suitable for detection has been conducted mainly in offline methods that employ all future, current, and past information, and there have been few online methods, which do not rely on future information, aimed at real-world applications such as surveillance cameras and robots. In addition, for such applications, not only the accuracy but also the processing speed for real-time is essential. Previous works have proposed stabilizing the detection by propagating the past information from the last frame or a specific nearby keyframe for real-time processing in online settings. However, they have not yet achieved stable detection due to the limited use of temporal information. Therefore, this dissertation studies feature enhancement methods for real-time and online video object detection that utilizes more temporal and spatial information.

To enhance feature maps for video object detection, we studied two aspects. One is to refine a feature map by aggregation, and the other is to enhance a feature map through prediction. First, we propose two new feature map aggregation

methods: frame-level feature map aggregation and element-level feature map aggregation. Feature map aggregation differs from previous real-time and online methods in that it directly exploits multiple past feature maps. It has been studied in offline methods and can provide stable feature maps; however, it requires more processing time due to the computation of the weight between detection and each surrounding frame. Therefore, in frame-level feature map aggregation, we propose to refine the feature map by calculating which past frames should be focused on in a one-shot manner, which runs in real-time. To aggregate past features directly, we extend the detector with external memory. We experimentally show that frame-level feature map aggregation can suppress the issue of object confidence score fluctuations in time. At the element-level, the idea of the frame-level method is further extended. Each element of the feature map is refined considering local and global spatial information and short- and long-range temporal information; however, such dense aggregation takes much time to calculate in general. Therefore, we propose a novel sparse aggregation method to reduce computation processing time for feature aggregation. Furthermore, we also propose an adaptive feature update strategy in external memory to hold long-term information. Finally, we achieve state-of-the-art performance in an online detector that maintains real-time performance. We also show that the proposed method significantly reduces false-positive and false-negative detection results, which are challenges in video object detection.

Next, we propose a novel feature map enhancement approach through prediction. The prediction-based approach differs from the feature map aggregation approach in that it does not utilize external memory but enhances the performance of the model itself. Therefore, it is suitable for conditions under strict GPU memory constraints, such as robotics. The prediction-based approach employs a future prediction task, which requires deep knowledge of objects, such as motion, to forecast the future clearly. The detector enhances the feature maps for stable object detection by learning features through prediction during the training phase. We leveraged the prediction from different perspectives: forecasts for the next and the next several frames. First, we propose a detector that jointly learns detecting objects and forecasting the next-frame feature map. This prediction approach is suitable for extending the recurrent neural network object detectors, and experiments show the effectiveness of learning features through the next frame forecast. Next, we propose a video object detection framework based on stochastic future prediction to leverage more extended time. The next several frames prediction is difficult to predict due to the future uncertainty; therefore, our model learns features by predicting the sampled and possible future. Experiments have shown the effectiveness of leveraging the stochastic long-term prediction for video object detection.

Acknowledgements

First and foremost, I would like to devote my thankful heart and wish to my supervisor, Prof. Akihiro Sugimoto, whose both devotion and expertise make an incredible contribution to the achievement of this dissertation. I cannot find a way to credit all his patient guidance, valuable suggestions, and enthusiastic encouragement in his supervision of my doctorate. It is very fortunate to have the opportunity to be supervised by him, who has deep experience in my research area, who is always gentle and cares about my research progress.

I would like to express my deep gratitude to all committee members, including Prof. Kazuya Kodama, Prof. Satoshi Ikehata, Prof. Shinichi Satoh, and Prof. Imari Satoh, who guided and helped my research process. Without your valuable comments, my Ph.D. would take longer to finish. In addition, I thank all my friends who are always encouraging me to keep studying.

Last but not least, I have to give special thanks to my parents and my older sister for encouraging me and inspiring me in carrying out scientific research for all these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Video is a vital resource that holds information about both time and space. Increasingly, many videos are being made year by year with the spread of smartphones, robotics cameras, surveillance cameras, and cameras for automatic driving. Therefore, it is essential to extract information from the video for robotics [50, 188], factory automation [35, 64], and human support [60, 65, 168], rather than letting it end up as data. Understanding videos and extracting information are researched from various angles, such as video recognition [95], action recognition [148, 157] and anomaly detection [118].

One of the most fundamental but essential understandings in videos is recognizing what an object is and where it is in a video frame. In order to find out what is in the frame, i.e., to identify objects in an image, object detection has been studied in computer vision. Object detection has made tremendous progress in recent years [15, 18, 112, 130, 131, 173], benefiting from the development of deep learning and convolutional neural networks [81, 141, 172]. The famous object detection benchmark MS COCO [107] reached 61.3% accuracy in 2021 [173] from 19.7% in 2015 [70]. These developments mean that object detection technology can now be used in the real-world, and its utility value is attracting attention not only in robotics [26, 50] and automated driving [67] but also in various fields such as medical care [168] and agriculture [135].

Object detection consists of two stages: feature extraction and detection. In feature extraction, important information for detection is obtained as features from images using a backbone such as VGG [141] or ResNet [81]. The features are generally referred to as a feature map. Next, the detection stage estimates object's location and class from the obtained feature map. For detection, mechanisms such as SSD [112] and Faster R-CNN [131] have been proposed. The detectors output the detected objects' positions and their classes with confidence scores for the given image by going through these stages. The confidence score indicates the degree of certainty the detector estimated the detected object. The detector must detect the target with high confidence because the results are filtered at a certain threshold.

When it comes to detecting objects in a video with the detectors [112, 131], it is difficult to achieve stable detection due to apparent changes caused by the temporal changes in the video. In general, to apply object detection to video, the detectors trained on still images are applied on each video frame for detection. However, it does not provide stable detection in practice since the changes in appearance with time lead to phenomena such as motion blur and out-of-focus, making detection difficult. More specifically, the change produces several issues of video object detection: detection confidence score fluctuation, false-negative detection, and false-positive detection. As shown in Figure 1.1, the confidence scores of the detected object fluctuate as the changes in appearance even slightly over time. Figure 1.2 shows the occurrence of false-negative detection and false-positive detection. False-negative detection is a case where a detection target is present but cannot be detected. False-positive detection mainly consists of background false-positive and class false-positive detection. The background false-positive one is that a model detects an object where there is no target object, and the class false-positive one is that a model detects the target as a different class. Thus, overcoming these problems caused by the apparent changes is necessary to achieve stable detection in video object detection.

Recently, feature extraction considering temporal information is expected to be an approach to address those problems. In order to stably detect objects in a video, methods that take temporal information into account at each stage of the object detector have been studied. The existing research initially tried to enhance detection by introducing temporal information into the detection results [77, 165] or the detection stages [54, 90, 91]. However, these methods only slightly improved since they do not work well unless the bounding boxes are detected in most frames. The problem is that the features necessary for detection could not be sufficiently obtained due to degraded appearances caused by

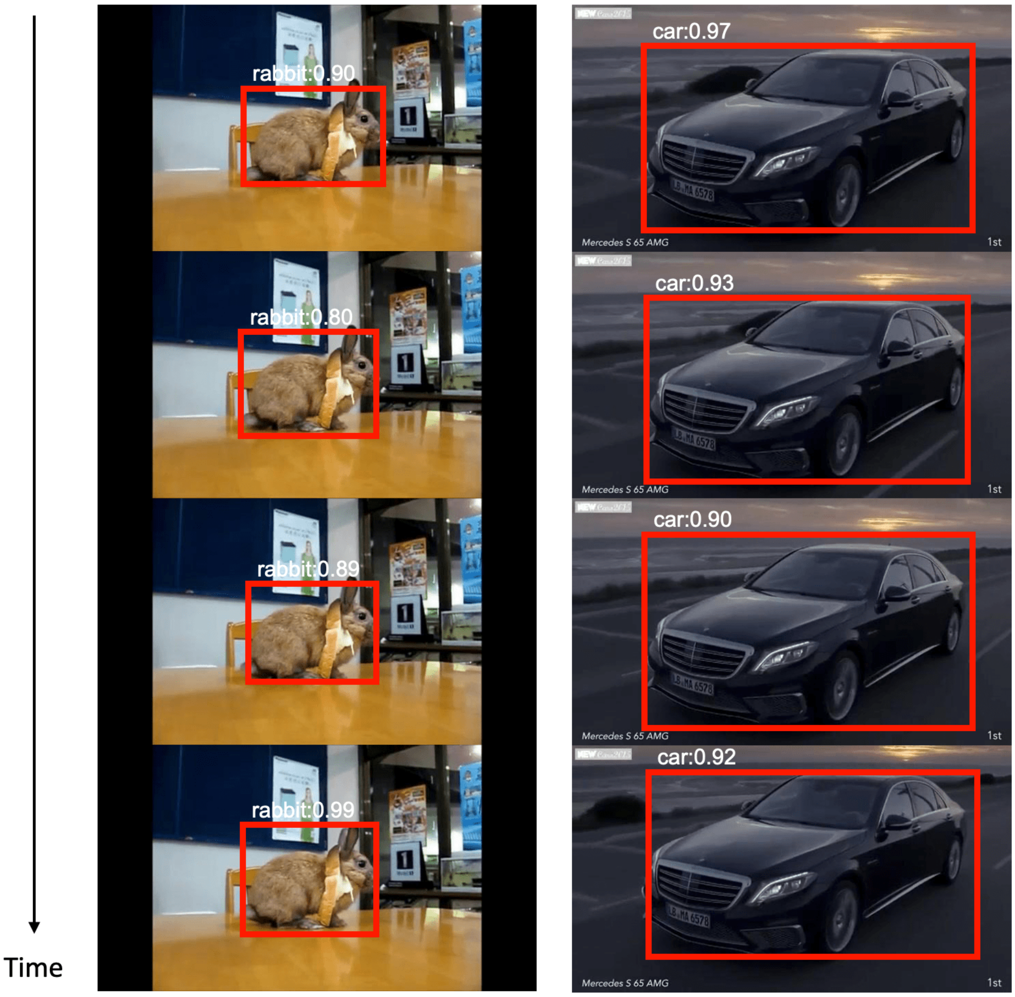


Figure 1.1: Examples of detection confidence score fluctuations with Faster R-CNN due to apparent changes with time. Small changes in appearance, such as luminance, drastically affect the detection confidence in a still image detector.

the apparent temporal changes [191]. Therefore, to obtain sufficient features for detection before the detection stage, it is necessary to enhance the feature map by including temporal information in the feature extraction stage. In video object detection, it is crucial to research generating and enhancing robust features from degraded ones for stable object detection in videos by utilizing temporal information to deal with fundamental issues and has become the mainstream approach [29, 68, 191, 192].

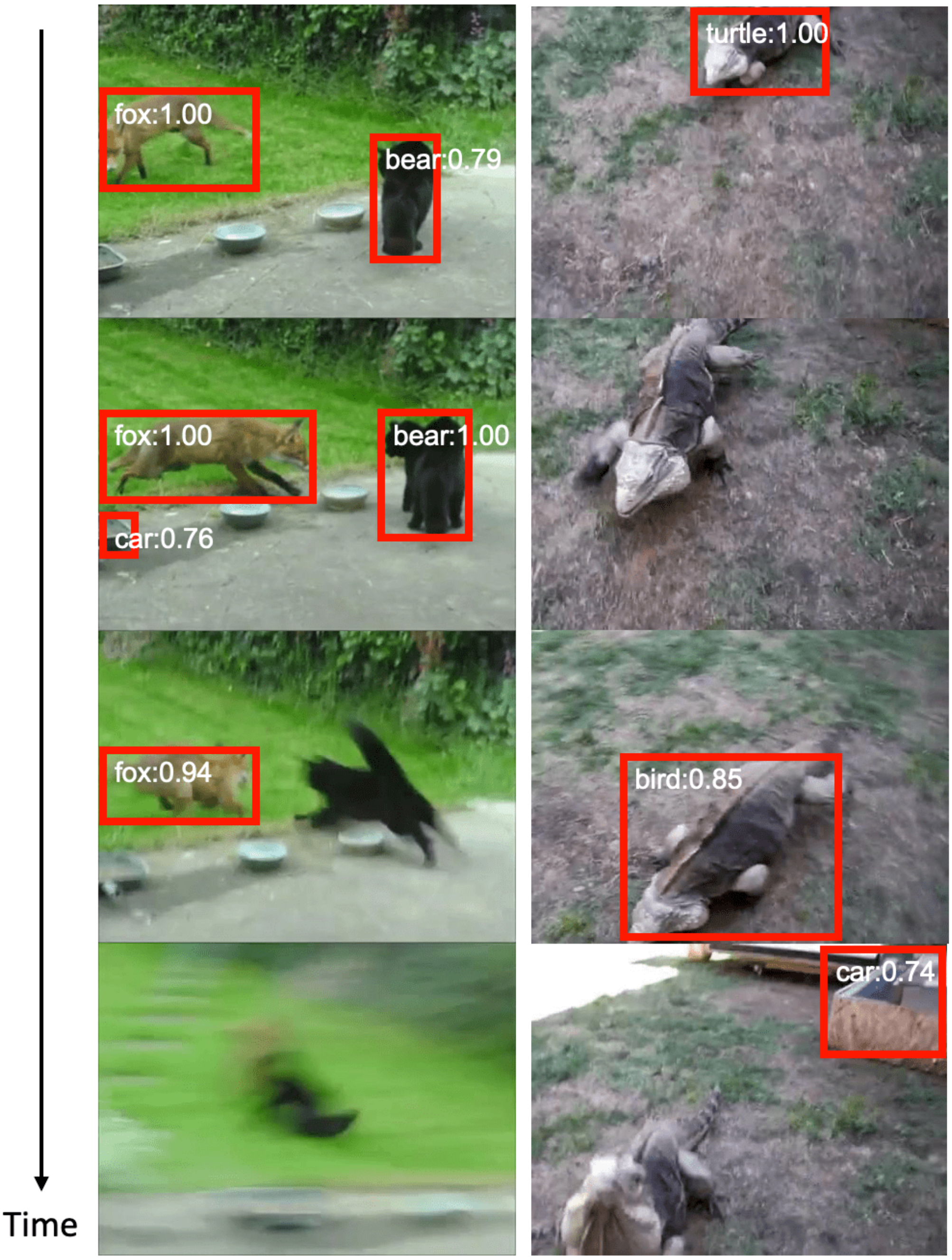


Figure 1.2: Examples of false-negative and false-positive detection with Faster R-CNN due to apparent changes with time. The changes cause the problems, such as detecting an object that is not the target of detection or failing to detect an object in a scene that should be detected.

1.2 Motivations

Research on enhancing feature maps for detecting objects in videos has been actively studied in recent years [29, 42, 125, 143, 191, 192], and can be divided into offline and online settings, depending on how the temporal information is used for detection. Moreover, their target can be divided into accuracy-oriented and speed-oriented.

In terms of detection settings, many methods have been studied in offline settings and a few studies in online ones in the field of research. The offline methods use past and future information when detecting objects in the current frame [29, 42, 125, 191]. On the other hand, future information is not available in online methods [26, 192] intended for live streaming videos. Since accuracy is one of the most crucial factors in the research field, offline methods have been mainly proposed, and online ones have not been sufficiently studied.

In addition, since there is a trade-off between accuracy and speed, the previous works have been divided into the pursuit of accuracy and speed. The accuracy-oriented research is to refine the feature map quality of the detection frame by using the features obtained from the surrounding frames [29, 158, 167, 191]. Improving the feature map can overcome the video challenges and increase detection accuracy. However, due to the high processing cost, these works tend to be slow operation speed, mostly one ˜ten frames per second. On the other hand, the speed-oriented research is to improve the speed performance by eliminating the heavy feature extraction process because the neighboring frames have redundant information [23, 83, 192]. However, the accuracy of these methods tends to be lower than that of still image detectors [192].

When real-world applications are considered, such as surveillance cameras, robotics, and medical fields[135, 168], it is essential to detect objects in online settings because future information cannot be utilized. Furthermore, while stable detection is vital for practical applications, real-time processing performance is also essential. However, research from this perspective has not been sufficiently discussed.

Two approaches have been proposed under the online constraint within real-time processing to leverage a nearby specific keyframe to achieve high accuracy video object detection. The warp-based approach [190, 192] proposed stabilizing the current feature map by propagating features by flow information [190, 192] from a nearby keyframe in the past. The recurrent neural network-based approach [26, 27, 110, 111] propagates temporal information by accumulating the last frame through the recurrent neural network. Although these approaches tried to detect objects stably, they do not result in stable detection since they only rely on the limited temporal information, a past nearby frame. Therefore, an approach that utilizes multiple frames to utilize temporal information more effectively is expected for stable detection.

Inspired by the above discussion, this dissertation researches feature enhancement learning methods, which utilize more temporal information, for online video object detection that improves accuracy within real-time speed. In particular, we consider the feature aggregation approach to learn the assembly of features from multiple past frames and the prediction-based feature enhancement approach, which learns enhancement through future prediction. In the following sections, we describe our research scope in more detail.

1.3 Problem Statement

Video object detection provides problems such as fluctuation of detection confidence score false-negative and false-positive detections because of apparent changes with time. Due to the degraded appearance scenes caused by the changes, detectors cannot obtain enough features for detection. Therefore, it is crucial to enhance the feature map by utilizing temporal information for stable detection. Many feature map enhancement methods have been studied in offline settings that utilize future information and cannot be applied to real-world applications such as surveillance cameras and robotics since they require online settings for live stream videos. Moreover, real-time processing is also an essential factor in such applications. In the previous studies that satisfy those requirements, the use of temporal information is limited and stable detection has not been achieved. Therefore, it is vital to develop online feature map enhancement methods that utilize more temporal information in real-time for stable video object detection in consideration of real-world applications. In our dissertation, we address this issue.

1.4 Dissertation Focus and Main Contributions

This dissertation focuses on enhancing feature maps for video object detection, exploiting spatiotemporal information under real-time processing and live-stream video constraints. We remark that real-time processing is over 15 frames per second on commercial graphics boards, such as RTX 2080 Ti, whose single-precision performance is about 13 TFLOPS in this dissertation. Under the constraints, existing research utilizes flow information to warp features [190, 192] or recurrent neural networks to propagate the past information to the current frame [26, 110, 111]. Unfortunately, they only propagate temporal information from a specific frame, last or nearby keyframe. Therefore, these methods cannot achieve stable object detection and are insufficient in accuracy.

Unlike the existing methods, this dissertation attempts to enhance features by utilizing multiple frames from various perspectives to improve accuracy. For this purpose, we present two approaches to the problem: the feature map aggregation approach and the prediction-based feature map enhancement approach.

First, we propose feature map aggregation approaches for real-time online video object detection. Since feature map aggregation is generally computationally time-consuming, it has been studied only the offline methods [9, 158, 191] and has not been proposed for real-time situations. To aggregate features in real-time, we propose aggregation of feature maps from two different aspects. The first is enhancement through frame-level feature map aggregation (in Chapter 3). We adaptively refine the feature map from the viewpoint of which past frames should be focused to aggregate information. In the conventional offline methods [86, 158, 191], each frame is weighted by similarity, which takes time. On the contrary, we propose a real-time method to weigh the multiple frames at once. We also introduce an external memory to retain the past feature maps. Experiments show that the proposed method is more beneficial than the previous approach for the issue of detection confidence fluctuation. Next, we deepen the frame-level aggregation to the element-level (in Chapter 4). Focusing on the element-level allows it to aggregate for object misalignment over a more extended period. However, element-level aggregation is generally time-consuming. Thus, we propose a sparse aggregation method considering video redundancy. Experiments have shown that refining the feature map by directly accessing past frames using external memory can effectively improve the accuracy in real-time. In addition, we experimentally confirm that it is effective for the problems of false-negative and false-positive detection in video object detection.

Second, we propose prediction-based feature map enhancement methods (in Chapter 5). The feature aggregation approach relies on external memory to store the past features, which requires GPU memory. However, there are memory limitations in some situations, such as robotics. Thus, we present feature map enchantment through prediction learning to improve accuracy without increasing memory usage. Predicting the future requires detailed knowledge of object dynamics, such as motion. Incorporating predictions into video object detection, we improve performance without increasing the model size by utilizing future information during training. We fuse detection and future prediction from two perspectives: the next and the next several frames forecasts. In the next frame prediction, we introduce joint learning of object detection and the next-frame feature map prediction by extending the existing recurrent neural network object detector. Experiments validate the clear benefit of feature enhancement through prediction. We propose a video object detection framework based on a probabilistic future forecast to leverage extended temporal information in future prediction in the next several frames forecasts. This experiment employs ten successive future frames. It is difficult to predict the long-term future clearly; therefore, we leverage stochastic future prediction, which samples one possible future. Our model learns video nature by predicting the possible future, utilized for object detection.

In this way, this research effectively exploits space and time to learn to enhance features for video object detection and improve accuracy while maintaining real-time performance. Following this research direction, the main contributions in the dissertation are as follows:

- We propose a new efficient frame-level feature map aggregation method named Temporal Feature Enhancement Network (TFEN) [61] for real-time online video object detection. The basic idea is to aggregate multiple past features in the external memory by computing their weights adaptively in a one-shot manner from recurrent units.

- We propose a new element-level feature map aggregation method, Video-aware Sparse Transformer with Attention-guided Memory (VSTAM), aggregating a more extended period for real-time video object detection. The proposed method refines the feature map at the element-level by considering local and global space and temporal short-long range information. We sparsely aggregate features in time and space within real-time by proposing Video Sparse Transformer (VST). We also propose an adaptive external memory update strategy to hold frames vital for feature aggregation.

- We propose two new prediction-based feature map enhancement methods, Real-time Object Detector by Feature Map Forecast (ROD-FMF) [62] and Video Representation learning through Prediction (VRP) [63] for strict memory limitation. The prediction-based approach stands on recurrent neural network detectors with less memory consumption than methods with external memory. Predicting the future, i.e., model forecasts how the next or the more forward frames go on, enhances feature maps. To utilize the future prediction, we examined two perspectives, the next frame prediction by ROD-FMF and the next several frames prediction by VRP. In ROD-FMF, we propose extending the existing recurrent detector to predict the feature map of the next frame while learning the detection. Next, in VRP, we propose a framework that integrates probabilistic future prediction and video object detection. Since predicting the successive future frames, ten frames forward is difficult due to future uncertainty; we introduce probabilistic sampling to predict a possible future. Moreover, we propose a two-step training method to utilize future predictions stably: pretraining a model only the future prediction and finetuning it to the detection.

1.5 Organization of the Dissertation

The remaining of this dissertation is organized as follows:

- *Chapter 2* provides an overview of still image and video object detection methods and datasets. In addition, we provide a survey of recent video topics that are closely relevant to this dissertation.

- *Chapter 3* presents feature aggregation method at frame-level, Temporal Feature Enhancement Network (TFEN), an efficient algorithm that computes which past frame to attention in external memory for aggregation.

- *Chapter 4* presents feature aggregation method at element-level, Video-aware Sparse Transformer with Attention-guided Memory (VSTAM), which sparsely aggregates features in temporal short- and long-term and spatial local and global information to process in real-time.

- *Chapter 5* presents two prediction-based feature enchantment approaches, which learn enrichment through future prediction. We present Real-time Object Detector by Feature Map Forecast (ROD-FMF) for the next frame prediction. Regarding the next several frames prediction, we present Video Representation learning through Prediction (VRP), which learns enhancement by generating stochastic frame prediction.

- *Chapter 6* concludes this dissertation by summarizing our contributions and discussing future works.

Chapter 2

Literature Review

The research on Video Object Detection (VOD) is based on object detection in still images. Since this dissertation focuses on enhancing the feature maps of video object detection, we briefly review still image object detection in Section 2.1. We then review video object detection works deeply in Section 2.2. We also review related tasks to video object detection, Multi-object Tracking (MOT) and Video Instance Segmentation (VIS) in Section 2.3.

2.1 Object Detection in Images

Object detection in still images is the task of estimating the location and the category of objects in a given image. In traditional object detection, the models were built as an ensemble of hand-crafted feature extractors such as the scale-invariant feature transform (SIFT) [116], and the histogram of oriented gradients (HOG) [39]. After the deep learning method, AlexNet [96], won the ImageNet Image Classification Competition championship by an overwhelming margin compared to traditional methods, deep convolutional neural networks (CNNs) have been extensively studied [81, 141, 172]. Subsequently, CNNs are also applied to the field of object detection and outperformed traditional approaches [39, 55]. Object detection in still images with deep learning algorithms has made remarkable progress in the past few years [70, 71, 112, 128, 129, 130, 131].

Object detection based on deep learning can be divided into two main groups. The first is called two-stage detectors [15, 80, 131] as shown in Figure 2.1, which estimate the candidate regions of objects from the features obtained by CNNs and then perform

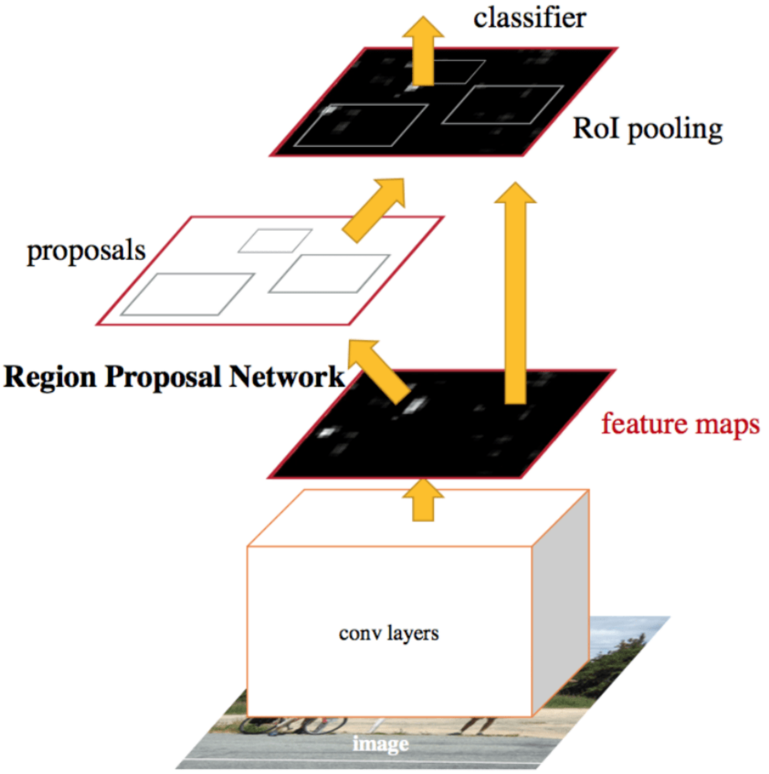


Figure 2.1: Pipeline of a two-stage detector (figure is adapted from [131]). The two-stage detector obtains a feature map from the image and then estimates the candidate regions of the object from the feature map using the Region Proposal Network. It then extracts the corresponding features in the area and performs a classification task to detect the object. Due to several processes, the runtime is generally slow; however, the accuracy tends to be high.

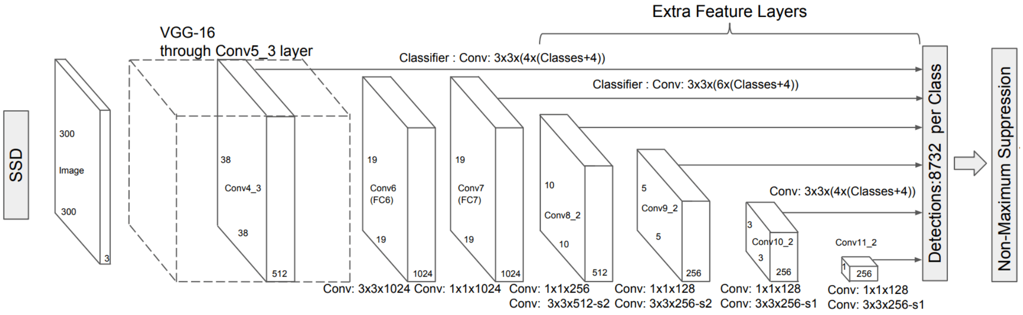


Figure 2.2: Pipeline of a one-stage detector (figure is adapted from [112]). Unlike the two-step detectors, the one-step detector performs classification and localization directly from the feature map obtained from the image. Because it has fewer processes than a two-stage model, it can run faster but tends to be lower accuracy.

classification and localization on them. The second is called one-stage detectors, as shown in Figure 2.2, which perform classification and localization directly from features. In general, two-stage detectors tend to be more accurate because their classifier does not need to consider unnecessary objects in advance thanks to the region proposals, but they are slower in processing speed due to the multi-stages. On the other hand, the one-stage detectors are faster but tend to be less accurate. Based on these approaches, research recently proposes improvements on existing components such as loss for class imbalance problems [106], architecture for utilization of detailed information [105], and hand-crafted parameter-free [18, 51, 97] towards high-performance detectors. We build our feature map refinement frameworks on top of [112, 131] and extend them to the video object detection task.

Especially in surveillance video, some methods have been proposed to improve the detection performance without considering temporal information. Evolving Boxes (EB) [156] improves its region proposal network with a cascade strategy, refining object boxes. GP-FRCNN [140], on the other hand, proposes geometric proposals for Faster R-CNN [131], 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) [59] incorporates the background subtraction method to ignore a non-interested region efficiently for false-positive elimination. They improve their accuracy performance; however, none of them can operate in real-time due to the high cost of computations.

2.2 Object Detection in Videos

Since a video is a collection of consecutive images, a still image object detector [112, 131] can be applied to each video frame to detect objects in the video. However, as discussed in Section 1.1, due to the apparent changes with time, which produces issues such as detection confidence score fluctuation and false-negative detections, the still image detectors cannot detect objects well in videos. To cope with the challenges in video domains, research can be categorized into two primary approaches. The first is to improve the detection process and post-processing, called box-level, based on bounding boxes. The second method is called feature-level because it exploits feature maps. We will first review the box-level method and then the feature-level method.

2.2.1 Box-level

In order to address the challenges of video object detection issues, methods of incorporating temporal information into the detection results or the detection stage of the still object detectors were initially proposed. For example, SeqNMS [77] links bounding boxes across frames with IoU threshold and re-rank the linked bounding boxes to deal with detection confidence score fluctuations. TCN [91] introduces tubelet modules into the detection stage and applies a temporal convolutional network to embed temporal information to improve the detection across frames. T-CNN [90] applies image object detectors to generate results and then uses optical flow to associate the detected results. D&T [54] proposes tracking loss to improve detection accuracy by combining tracking approaches. However, these methods are unsuccessful since they assumed that detection was possible in most frames. It is challenging to detect objects with insufficient features for detection from the degraded appearance caused by the apparent changes with time. Besides, most of the works [77, 90, 91] can not be trained end-to-end, and their performances are still sub-optimal.

2.2.2 Feature-level

Improving feature maps by propagating temporal information has recently become a mainstream approach in video object detection to cope with degraded appearance due to the changes in appearance over time. It can be broadly classified into offline algorithms that pursue accuracy and online ones that consider live stream videos. Since accuracy is one of the essential factors in the research field, many studies have been conducted as offline methods. Thus, this section first reviews offline algorithms and then reviews online ones.

Offline Methods

The main focus of research in the offline algorithms for feature map refinement is how to exploit temporal information, including future information, to improve accuracy. They can be classified into three categories: local, global, and combination aggregation, according to how they handle temporal information.

Local aggregation methods [9, 44, 68, 108, 158, 171, 191] usually focus on propagating features from nearby frames, less than ten frames, on video sequences. In order to refine the whole current frame feature map, FGFA [191] exploits optical flow information to have pixel-to-pixel correspondence among nearby frames. MANet [158] further utilizes flow information to capture object motion. STSN [9], on the other hand, uses deformable convolutions across time to align the features from adjacent frames. STMN [171] computes the correlation between neighboring frames and introduces a memory module to aggregate their features. They have shown the importance of feature maps from nearby frames; however, they require additional network and training costs for flow information. Thus, some research relies on only apparent features from the feature extractors without flow information. FFAVOD [125] proposes aggregating the apparent features of the surrounding frames by fusing channel-wisely. TF-Blender [36] proposes a method to compute the interrelationships between neighboring frames and enhance the entire feature map in all aggregated frames instead of improving only the detection frame.

Recently, to aggregate features more robustly against object misalignment, the object-level feature aggregation approach [44, 68, 75, 94] has been proposed to use only the candidate regions features of objects rather than exploit the entire feature map. RDN [44] focuses on the object relations to align object features. OFAVOD [68] proposes pairwise feature aggregation, which aligns object features based on context. EBFA [75] proposes a temporal and spatial alignment module to refine object-wise features. TM-VOD [94] offers temporally-aware region proposals to extract object-wise features robustly for object-wise refinement.

On the contrary, global aggregation methods [76, 167] rely on long-range semantic information in a video or among videos. They can deal with degraded appearance scenes that continue for multiple frames, such as motion blur. In such cases, since aggregating information from neighboring frames does not provide practical temporal information, propagation from the distant frames before and after the moments can be used. SELSA [167] proposed sequence-level semantics object-level aggregation, which utilizes spectral clustering to capture the object relations in the whole video. Furthermore, HVR-Net [76] proposes an Inter-Video Proposal Relation module to consider the relation of object-level features among different videos, in addition to the intra-proposal association in a single video. However, since they treat all features in the temporal information equally, locally essential features may be ignored.

Unlike the methods that exploit features locally or globally, the combination aggregation method leverages both local and global features but deals with them separately. MEGA [29] introduces a memory module to keep both local and global object-level features separately to enhance the visual representation of the current frame. The offline aggregation methods achieve high detection accuracy; however, they use computationally expensive feature aggregation and future information, which impede live streaming detection.

Online Methods

Although online methods are similar to the offline ones in terms of feature aggregation except for future information, they focus on efficiently utilizing temporal information and are classified into three categories. The first methods aim for high accuracy in online fashion without focusing on speed. The second category is the methods that aim to improve speed performance by utilizing the redundancy of videos; however, they sacrifice detection accuracy. The third is methods to improve detection accuracy while keeping real-time speed performance. Our research belongs to the third category.

The first group of methods aims at high accuracy under the limitation of not utilizing future information. THP [189] exploits optical flow to improve accuracy by propagating information from keyframes. In addition, they propose to determine non-keyframes adaptively and to update the flow information partially. OGEMN [42] proposes to aggregate features by using external memory to keep long-term information. It introduces the updating rule in the external memory to keep the feature maps that maximize the detected objects' confidence. MAMBA [143], which is an extension of OGEMN, proposes a simple random update rule based on video redundancy. Some works [74, 88] point out applying optical flow directly onto image-based features leads to a mismatch in high-level features. Therefore, LSTS [88] proposes learnable spatiotemporal sampling to learn semantic-level correspondences among adjacent frame features. PSLA [74] offers a progressive sparse local attention module to propagate local information from the previous frames based on appearance features. Since their improving approaches require high computational costs, they cannot run on commercial graphics boards in real-time.

The second group of methods aims to take advantage of the redundancy of video to increase speed performance. DFF [192] utilizes a high-cost feature extractor for keyframes and a lightweight optical flow for non-keyframes to propagate temporal information and achieve high speed. DorT [117] proposes a network that decides whether to use detection or tracking to speed up. Adascale [31] proposes a scale regressor network that estimates the subsequent frame resolution adaptively to improve the tradeoff between accuracy and speed. Flow-guided [190] proposes to stabilize the flow information between frames by passing through a recurrent neural network. Memory-guided [111] proposes to exploit high-cost and low-cost feature extractors for keyframes and non-key frames, respectively, to boost speed performance. LWDN [87] propagates high-level features from keyframes by focusing on the weighted local parts. Attention-guided [179] proposes an adaptive key and non-keyframe prediction approach for fast detection with subtle decline and offers a feature propagation module by attention mechanism from a keyframe. Although these methods can speed up processing, they are inferior in accuracy.

In recent years, attention has been focused on improving the trade-off between accuracy and speed for real-world applications such as robotics [26, 27, 110]. As a result, methods in the third group have been proposed to improve accuracy while running in real-time. LSTM-SSD [110] offers bottleneck architecture of the convolutional recurrent neural network to propagate temporal information. TSSD [26] propagate feature maps across frames with convolutional recurrent neural networks using spatial attention. TSSD-OTA [27] extends TSSD with tracking networks for better accuracy. Heatmap-guided [175] propagates the previous reliable detection in the form of the heatmap to boost the results of the subsequent frame. LMP [193] proposes separate temporal propagation modules to handle temporal information of different times separately. They tried to employ temporal information to stabilize detection; however, most concentrate on propagating temporal information from a particular frame (the last frame or a nearby keyframe) to the current one. Thus, they lack the consideration of utilizing more temporal information effectively.

Unlike the existing methods, our approaches employ multiple frames to refine features. First, in the feature map aggregation approach, we aggregate multiple past feature maps directly to refine the current frame feature map by introducing external memory to keep them. However, the existing methods aggregating feature maps proposed in offline methods are computationally expensive, so we propose two cost-effective aggregation methods towards real-time performance. First, in frame-level aggregation, instead of weighting each frame by similarity, we exploit recurrent neural networks to weigh past frames in a one-shot manner by an attention mechanism. Very recently, a method of aggregating past features directly for real-time online video object detection has been proposed [86]. Still, the processing speed is not fast because it conducts the weighting for each frame. Next, in element-level aggregation, we propose to refine the feature map details by aggregating features element-wisely to deal with the object misalignment in multiple frames. However, it takes a long time to compute aggregating features at all elements from past frames. For this reason, we propose sparse element-wise aggregation to achieve real-time speed, considering the video redundancy. The details of frame-level and element-level aggregation are described in Chapter 3 and Chapter 4, respectively.

Secondly, we leverage the next future frame or multiple successive future ones during the training phase to enhance feature maps through future predictions. While the existing recurrent neural network methods [26, 110, 111] focus on propagating information from the past to the present, we introduce predictions to learn about the knowledge of objects, such as motion. The details are explained in Chapter 5.

2.3 Related Topics to Video Object Detection

Multi-object tracking and video instance segmentation are related tasks in video object detection. In this section, we review them briefly.

2.3.1 Multi Object Tracking

Multi-object tracking (MOT) is a task to track multiple objects in a given video, and the goal is to assign the same ID to each tracked object. It is a similar task in terms of continuously detecting objects in the video, but its purpose is different from video object detection. In the pre-deep learning era, trackers often exploits flow fields [85, 132], Kalman filters [10, 24] or Intersection-over-Union (IoU) [10, 11] for association. The proposed methods are simple and fast but fail very easily in challenging scenarios.

With the success of deep learning, many models have recently leveraged appearance features to associate objects [33]. For instance, DeepSORT [165] takes the provided detections by still-image detectors and associates them using an offline trained deep re-identification (Reid) model and a Kalman filter model. SiameseCNN [98] exploits a

Siamese network to learn the similarity between a pair of detections directly. Moreover, Deep Affinity Network [144] employs a Siamese network that takes two video frames as input, extracts multi-scale appearance embeddings, and outputs the similarity scores between all pairs of detections.

More recently, the success of multi-task learning in deep neural networks [20, 27, 147] has led to models that jointly learn detection and tracking tasks. The tracker [32] adapts the Faster RCNN [131] to estimate the bounding box location in a new frame from the preceding frame. CenterTrack [187] and D&T [54] extend the still-image detectors [37, 51] to compute correlation maps between high-level feature maps of consecutive frames to estimate inter-frame offsets between bounding boxes. To directly optimize the MOT model, Xu et al. [174] presents an end-to-end MOT training framework, using a differentiable approximation of MOT metrics in the loss functions. To further integrate the still-image detector and tracking methods, Reid branchs [21, 162, 184] are proposed to extract object embeddings for the association.

2.3.2 Video Instance Segmentation

Recently, a new task, video instance segmentation (VIS) [154, 176], has been proposed, which is a combination of object detection, instance segmentation, and object tracking across frames. To solve this problem, MaskTrack R-CNN [176] extends Mask R-CNN [80] with a pair-wise identity branch to solve the instance association. SipMask-VIS [16] propose one-stage models for VIS, which is the similar pipeline [176]. MaskProp [8] introduces a mask propagation branch on the multistage framework [22] that propagates instance masks from one frame to another. In [102], a modified variational auto-encoder was proposed to learn spatial interdependence and motion continuity in the video. STEm-Seg [4] and TrackR-CNN [154] treats the video clip as 3D spatial-temporal volume and segments objects in a bottomup fashion. VisTR [161] naturally extends DETR [18] for VIS task in a query-based end-to-end fashion. CrossVIS [178] proposes pair-wise global instance embeddings to tackle the association of instance segmentation. They focus only on how to associate objects in a video and do not consider the significance of the feature refinement.

This dissertation also examines the effect of temporal feature map enhancement on the more complex task of video instance segmentation. Recently, CompFeat [58] introduced features aggregation at frame-level and object-level with temporal and spatial context information. TF-Blender [36] proposed feature adjustment to enrich the representation of every neighboring feature map. Although CompFeat and TF-Blender improve accuracy performance, they only consider feature aggregation from nearby frames.

2.4 Benchmark Datasets

In this section, we describe benchmark datasets for object detection in videos. Then, we describe the dataset used for our experiments.

The early days of video object detection began with specific tasks such as person counting [57] and person [48] and car detection [67] for automated driving. However, they were limited by the small number of video scenes [49, 67], the small number of detection targets [49], and the fact that only one frame, known as keyframe, was annotated in every few frames [67].

Therefore, ImageNetVID [134] was proposed using videos collected from the Internet, aiming at a general-purpose dataset for video object detection and has become the de facto standard dataset nowadays. The dataset is split into a training set and a validation set, containing 3862 and 555 video snippets. The dataset consists of 30 classes of categories, annotated on all frames captured at 25 or 30 fps. Those classes are composed of subclasses of ImageNet DET [134], a dataset for still image object detection. Most of the methods recently studied for video object detection have been validated on this dataset.

The Youtube-BB [127] was proposed as a dataset that includes rich variations in videos due to the relatively small size and diversity of datasets in ImageNetVID. YouTube-BB is a large-scale dataset, which is human-annotated at one frame per second on videos from YouTube. It contains a total of 380,000 videos with 23 object categories, which is a subset of the COCO dataset [107]. While Youtube-BB is high diversity, it has not been sufficiently validated by the methods due to the small number of classes and its errors relative to a fully human-annotated dataset.

Table 2.1: Benchmark datasets for object detection and instance segmentation in videos. “D” denotes the detection task, and “I” denotes the instance segmentation task.

Name	Task	Scene	Classes	Number of videos (train/ val/ test)	Annotations	Year
Caltech [49]	D	Driving	1	5/-/5	all-frame	2012
KITTI [67]	D	Driving	3	7,481/-/7,518	key-frame	2012
ImageNetVID [134]	D	Generic	30	3,862/ 555/ -	all-frame	2015
UA-DETRAC [164]	D	Traffic	4	60/-/40	all-frame	2015
YouTube-BB [127]	D	Generic	23	304,000/38,000/38,000	key-frame	2017
EPIC-KITCHENS-55 [40]	D	Egocentric	290	272/106/54	key-frame	2018
UAVDT [50]	D	Drone	3	29/ -/ 21	all-frame	2018
VisDrone-VID [188]	D	Drone	10	56/ 7/ 16	all-frame	2019
YouTube-VIS [176]	I	Generic	40	2,238/302/343	key-frame	2019
KITTI MOTS [154]	I	Driving	2	12/9/-	all-frame	2019

While research has been conducted on video object detection such as pedestrian detection using in-vehicle cameras [49], UA-DETRAC [164] proposes video object detection from fixed-point surveillance cameras for intelligent traffic systems. It offers new challenges compared to general video object detection datasets from the Internet [127, 134] since the size of an object is smaller and things are denser due to the surveillance camera perspective. UA-DETRAC is a dataset containing various traffic patterns and climatic conditions captured from 24 different locations at 25 fps with all frames annotated and consists of 60 and 40 videos for training and testing, respectively.

Moreover, with the recent development of robotics and drone technology, new challenges have arisen in video object detection. From the drone's point of view, the camera has a smaller detection target and a wider angle of view than conventional video object detection, resulting in a high density of detection targets. To tackle the challenges, new datasets for drone viewpoints have been proposed in recent years. UAVDT [50] is an early dataset that consists of 29 and 21 videos for training and testing, respectively, in three categories. Recently, an extension of the UAVDT dataset, VisDrone-VID dataset [188], has been proposed. It contains 79 sequences with ten object categories such as cars, vans, and pedestrians, three non-overlapping subsets, 56 training video clips, seven validation video clips, and 16 test video clips. The annotations are available for all frames. These sequences have been shot in different cities under different weather and lighting conditions.

EPIC-KITCHENS-55 [40] is a dataset proposed for video understanding from a first-person perspective and consists of 432 videos captured at 60 fps by multiple participants using a head-mounted camera in a kitchen. Several tasks are proposed for video understanding: action recognition, action anticipation, and video object detection. However, in the revised version of EPIC-KITCHENS-100 [41], the video object detection task has been deprecated due to the addition of other tasks such as Unsupervised Domain Adaptation for Action Recognition. We do not evaluate it on this dataset because there are few methods to assess them on KITCHENS, and it is not sufficiently comparable.

In very recent years, a more complex video object detection task, video instance segmentation (VIS), has been proposed to simultaneously perform instance segmentation, tracking, and detection in videos. The proposed datasets are mainly for on-vehicle cameras [154] and Internet videos [176], and they have attracted much attention recently.

Figure 2.3: Frame examples of ImageNet VID dataset [134].

YouTube-VIS-2019 [176] is the first dataset for video instance segmentation, which has a 40-category label set. There are 2,238 training videos, 302 validation videos, and 343 test videos with precise annotation intervals of 5 keyframes. KITTI MOTS [154] has 12, and 9 videos for training and validation, respectively, and all frames are annotated.

This dissertation evaluates the proposed methods on datasets with different characteristics to check their effectiveness. First, we investigate the generic performance of all methods using ImageNet VID, a de-facto standard for video object detection with typical scenes. Then, in addition to ImageNet VID, we investigate UA-DETRAC or VisDrone-VID to validate if they are effective for other challenges such as small objects and more accurate localization. In addition, we test our feature map enhancement on YouTube-VIS to see if it is effective for video instance segmentation, which requires more precise feature map refinement due to the generation of masks instead of bounding boxes. However, we utilize only the video object detection dataset for methods that cannot be trained with keyframe annotations due to the training method or are challenging to apply instance segmentation and tracking due to the structure of the model. Figure 2.3, 2.4, 2.5 and 2.6 show the reference images of ImageNet VID, UA-DETRAC, VisDrone-VID and YouTube-VIS, respectively.

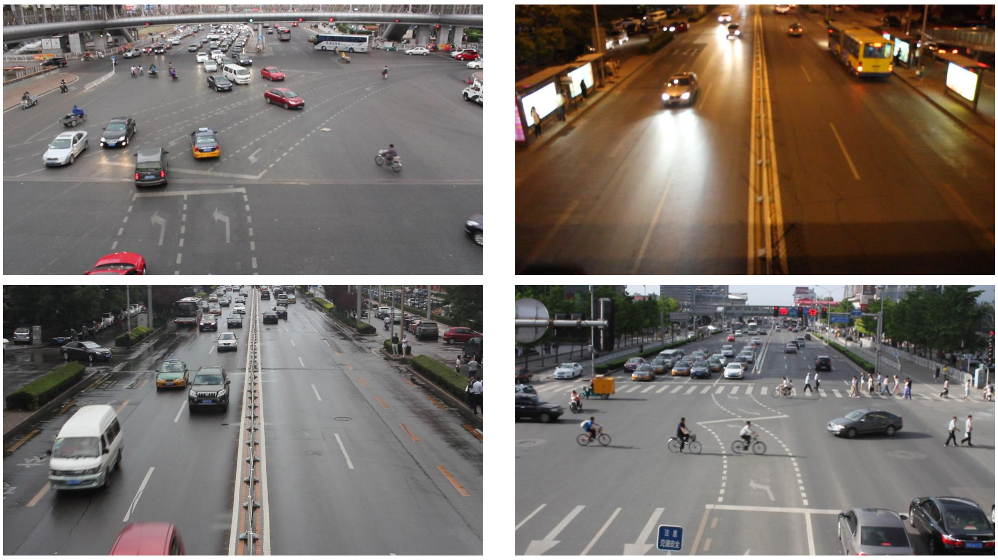


Figure 2.4: Frame examples of UA-DETRAC dataset [164].

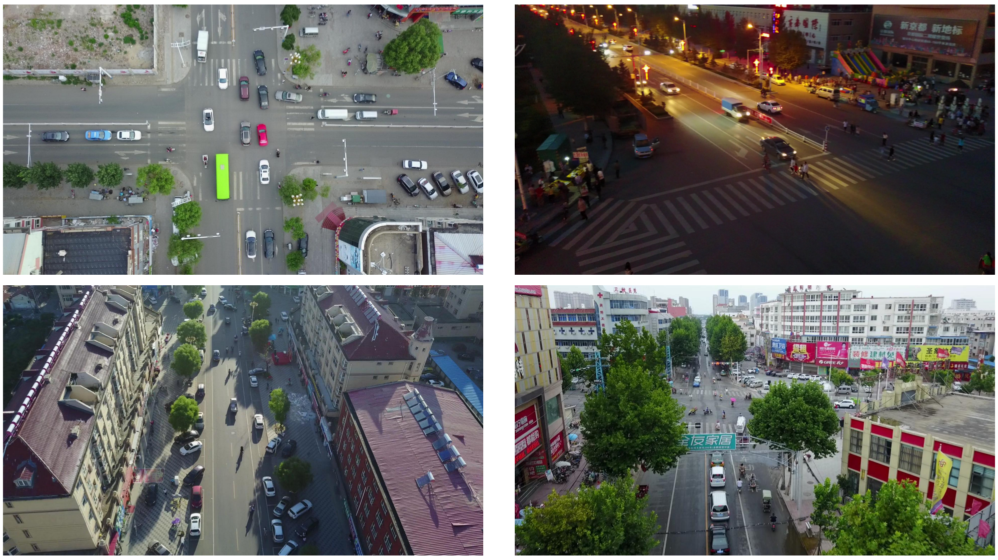


Figure 2.5: Frame examples of VisDrone-VID2019 dataset [188].

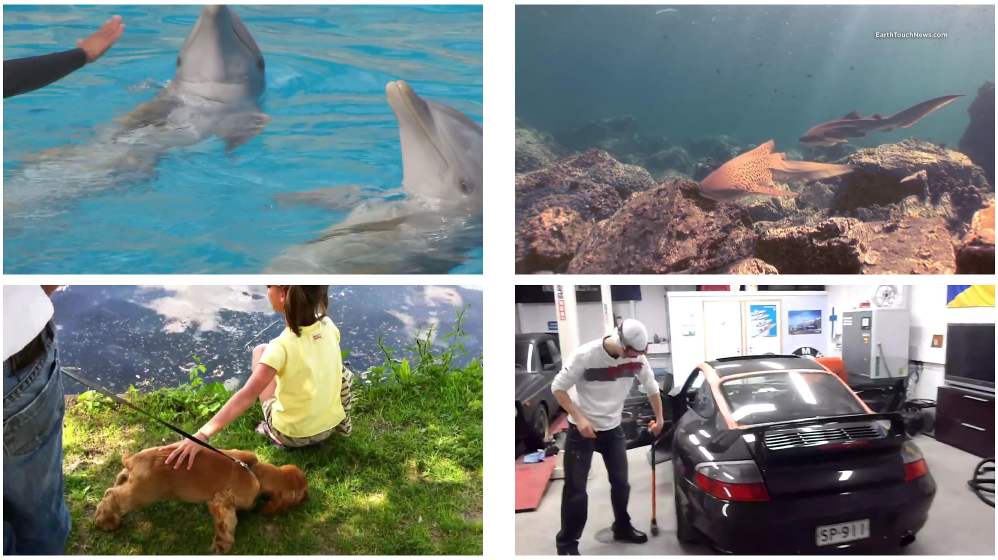


Figure 2.6: Frame examples of YouTube-VIS2019 dataset [79].

2.5 Evaluation Metrics

The accuracy evaluation of video object detection [134, 164, 188] is conducted on a frame-by-frame basis, using the same metrics as for still images. For object detection in still images, the metric, mean Average Precision (mAP), is utilized, which provides a single number of performances in terms of regression and classification accuracy. Precision is derived by Intersection over Union (IoU), the ratio of the area of overlap and area of union between ground truth and predicted bounding box. A threshold is set to determine whether the detection is correct. If the IoU is more than a threshold, it is classified as True Positive (TP), while the IoU below it is classified as False Positive (FP). When the model fails, the object present in the ground truth is detected as False Negative (FN). Precision measures the percentage of correct predictions, while recall measures accurate predictions with respect to ground truth. Precision and recall is defined as follows:

$$precision = \frac{TP}{TP + FP} \tag{2.1}$$

$$recall = \frac{TP}{TP + FN} \tag{2.2}$$

According to the above equations, average precision is computed for each category separately. The mean of average precision of all classes, called mean average precision (mAP), is used to compare the performance between detectors, which acts as a single metric for final evaluation.

For the ImageNet VID dataset, a threshold of 0.5 is used for the IoU. The UA-DETRAC dataset uses an IoU threshold of 0.7 for more accurate localization. We remark that although UA-DETRAC has four categories, the evaluation stage deals with them as one class; therefore, single category accuracy (AP) is used. The VisDrone-VID dataset employs the average of the mAPs measured in increments of 0.05 from 0.5 to 0.95 for the IoU threshold proposed in the COCO, which is called AP.

Chapter 3

Frame-level Feature Aggregation

3.1 Introduction

Video object detectors [42, 54, 189, 191, 192] have been actively studied to detect objects stably against appearance changes with time by utilizing temporal consistency over a video. Some methods [89, 191] have been proposed to stabilize detection by utilizing not only past and present frames but also future ones. However, in the case of live streaming videos, future information cannot be utilized. Besides, improving the still-image detectors with geometrical constraints [140] or the cascade strategy [156] has been proposed for frame-by-frame object detection. Such approaches improve the accuracy while they are limited to running in real-time. To deal with real-time and live-stream video, some methods have been proposed using recurrent neural network [26, 110] and flow information [189]. They tried to improve accuracy in real-time by utilizing temporal information from the last or a specific nearby keyframe in the past. However, the limited temporal information has not achieved sufficient accuracy when real applications are considered.

To achieve accurate real-time object detection in live-stream videos, we aim to generate an enriched feature map by aggregating coarse feature maps in previous multiple frames, extracted using a lightweight feature extractor, using an attention mechanism effectively. To this end, 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

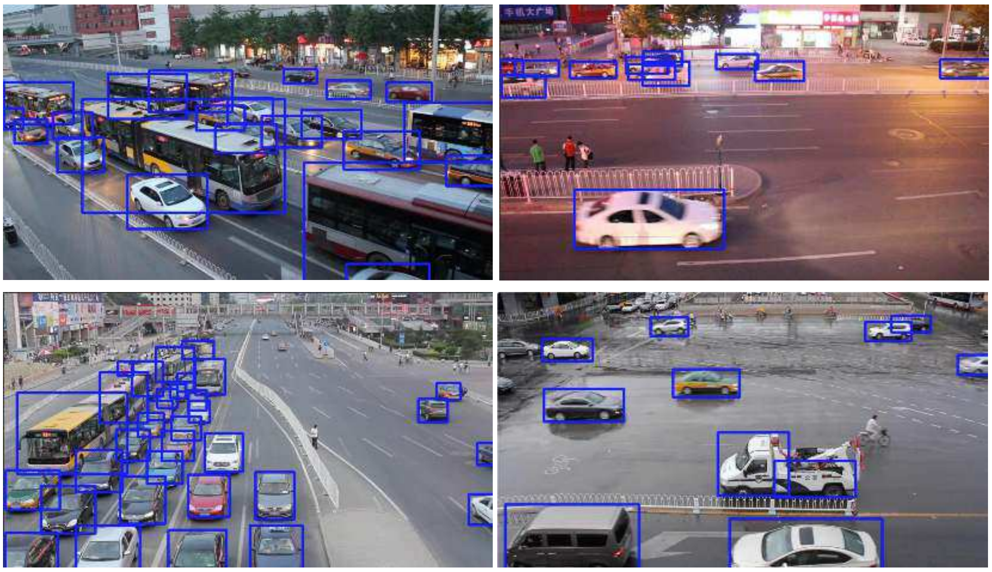


Figure 3.1: Examples of the detection results by TFEN on the UA-DETRAC. A bounding box is plotted if its confidence score is larger than 0.4.

features. The decoder has the external memory to store feature maps generated to utilize temporal information and exports a densely aggregated feature map using attention weights in a one-shot manner to compute in real-time. In this way, TFEN enriches coarse features with spatial and temporal information so that the trade-off between accuracy and speed is considerably enhanced. We evaluate TFEN on the UA-DETRAC dataset [164] and the ImageNet VID dataset [134] using MobileNetV2 [136] as the feature extractor and Cascade R-CNN as the object detector. Experimental results demonstrate that TFEN performs in real-time while keeping comparable accuracy with state-of-the-art. Figure 3.1 and Figure 3.10 show some detection results by TFEN on the UA-DETRAC and the Imagenet VID. We see that TFEN successfully detects most objects in different scenes, especially even when heavy and partial occlusions occur. Additionally, unlike other methods [191, 192] that utilize optical flow to warp feature maps across neighboring frames, TFEN entirely relies on only appearance information from image feature extractors. The architecture of TFEN is thus simple to design. Moreover, it enables us to optimize its loss function efficiently because we do not suffer from any disturbance caused by differences between appearance and optical flow features.

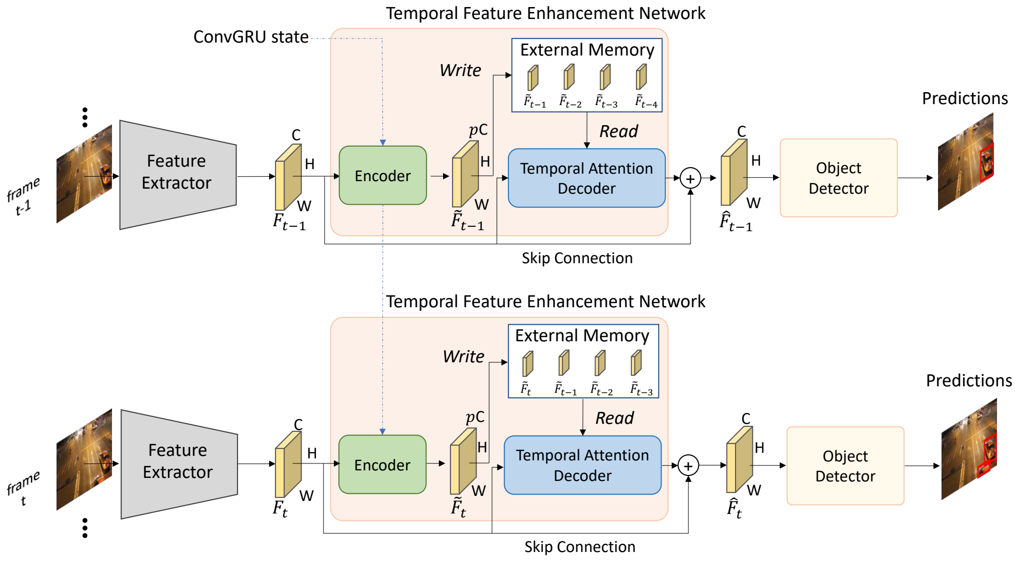


Figure 3.2: Architecture of our proposed TFEN.

3.2 Proposed Method

3.2.1 Architecture

Figure 3.2 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 map extracted from the feature extractor at time t, which is fed to TFEN. As the lightweight feature extractor, we employ MobileNetV2 because its computational cost is low.

Our proposed TFEN receives the extracted feature map F_t and enriches them to pass to the object detector (see the rectangle area in pink in Figure 3.2). 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 map $\tilde{F}_t$ using recurrent convolutional neural networks similarly to [110]. It exploits the spatial attention module BAM [122] and ConvGRU [6]. BAM refines the feature map 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 features generated in the past m frames: $\{\tilde{F}_i\}_{t-m+1\leq i\leq t} := \{\tilde{F}_t, \tilde{F}_{t-1}, \ldots \tilde{F}_{t-m+1}\}$. Then it

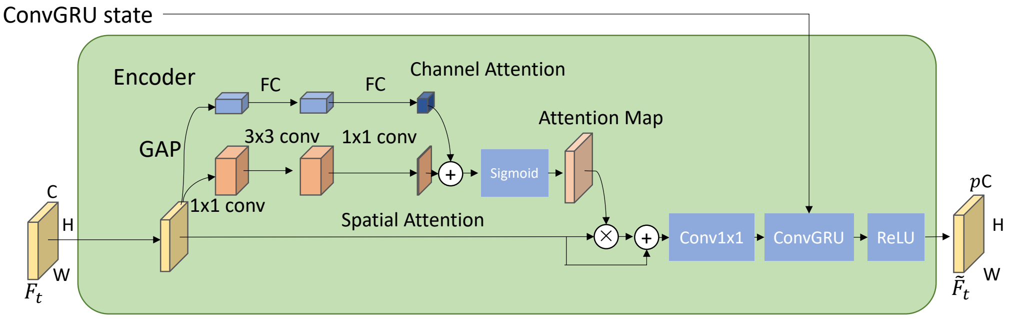


Figure 3.3: Architecture of the spatiotemporal encoder.

outputs a densely aggregated feature map according to the attention coefficient calculated in the decoder. In preliminary experiments, we found that the encoder and decoder were challenging to train the module stable. We adopt a residual learning scheme [81] to facilitate the gradient flow. The aggregated feature map (called enhanced feature map) is fed to the object detector. We detail the encoder and the decoder in the following subsections.

3.2.2 Spatiotemporal Encoder

Our encoder is designed for extracting spatiotemporal information from the feature map F_t coming from the feature extractor. As shown in Figure 3.3, it consists of BAM [122] and ConvGRU [6]. 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 the input feature map $F_t \in \mathbb{R}^{C\times H\times W}$ at the time t, BAM 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 the feature map, respectively. The refined feature map ${F_t}'$ is computed as

$$ {F_t}' = F_t + F_t \otimes M(F_t), \tag{3.1} $$

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 [6] for details on ConvGRU. The output of ConvGRU is fed into the Rectified Linear Unit (ReLU) to output the temporally-aware feature map $\tilde{F}_t$ which is saved in the external memory.

3.2.3 Temporal Attention Decoder

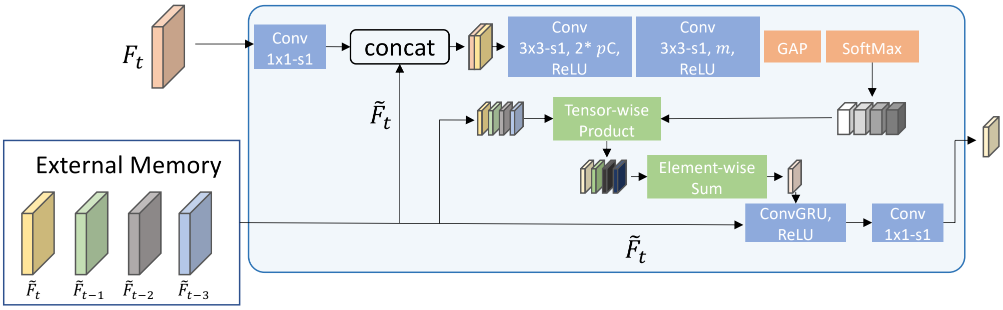


Figure 3.4: Architecture of the temporal attention decoder.

The temporal attention decoder is the most important, and its architecture is shown in Figure 3.4. Our temporal attention operation is similar to dense feature aggregation [7]. At the time t, the decoder performs dense feature map aggregation by summing all 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 map 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 (GAP) [104] 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 3.4 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 the feature map to export the enhanced feature map $\hat{F}_t$, which is forwarded to the object detector.

3.2.4 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 $\{\tilde{F}_i\}_{t-m+1 \leq i \leq t}$

[†] Although the objects can be spatially displaced between frames, the impact of displaced objects can be negligible in short-term aggregation. Indeed, we used $m = 4$ in our experiments, meaning about 120 *msec*. Moreover, the aggregated feature map is used as a state of ConvGRU and, thus, the effect of displaced objects across frames on the feature map is indirect and insignificant.

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 map is en-queued into the buffer after the oldest one is discarded. The *Write* operation allows the decoder to access all the tensors.

3.2.5 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 proposed in [15] for multi-stage classification and bounding box regression. This is because we employ the conventional cascade R-CNN 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:

$$L = \sum_{s=1}^{S}(L_{\text{loc}}^{s} + L_{\text{cls}}^{s}), \tag{3.2}$$

where L_{loc}^{s} and L_{cls}^{s} 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 [15] and set $S = 3$.

3.3 Experiments

3.3.1 Benchmark Datasets and Metrics

We evaluated the proposed model on two datasets. One is the ImageNet VID dataset [134] for natural scenes, and the other is the UA-DETRAC dataset [164] for surveillance. The ImageNet VID dataset [134] provides various challenges for general purposes such as motion blur and occlusion. The UA-DETRAC dataset [164] was published as a large-scale benchmark for vehicle detection in video. On the other hand, it offers challenges such as smaller object sizes and higher resolution frames. These datasets are annotated for all frames and are suitable for our method. (see Section 2.4 for more details about datasets.)

The evaluation metric for the UA-DETRAC dataset follows the average precision (AP) score proposed in the PASCAL VOC challenge [53] and uses the IoU threshold of 0.7. Note that while the PASCAL VOC challenge uses the IoU threshold of 0.5, the UA-DETRAC dataset requires object detection at a more precise location. For the ImageNet VID dataset's evaluation, the detection accuracy is measured by the mean average precision (mAP) at the IoU threshold of 0.5. (see Section 2.5 for more details about mAP.)

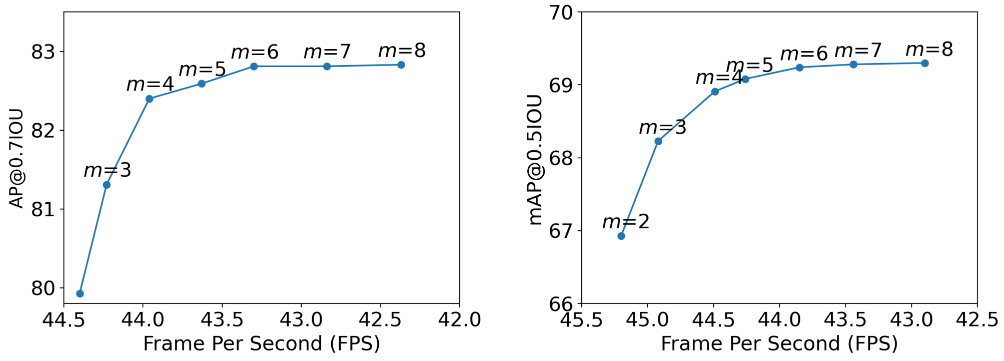
3.3.2 Implementation Details

We employed MobileNetV2 [136] as the feature extractor and cascade R-CNN [15] as the object detector. We re-implemented the cascade R-CNN network with the pre-trained MobileNetV2 in PyTorch [123] and regarded it as our baseline model.

In order to train TFEN, we use a multistep training strategy. Namely, we first fine-tune our baseline model to the dataset domain as a static image detector. For data augmentation, a random horizontal flip was adopted during training. In the next step, we initialize the weights of the feature extractor and the object detector with the weights of the fine-tuned baseline model while we randomly initialize the weights of the feature enhancement network. We then train all the weights together in an end-to-end manner.

In the first step, we fine-tuned our baseline model on all the 60 videos in UA-DETRAC training set for the UA-DETRAC dataset. For ImageNet VID dataset, ImageNet DET dataset [134] was employed as training assistance. The 30 categories in VID dataset are a subset of the 200 categories in the DET dataset. Therefore, following the practicals [54, 66], we trained the model with VID and DET (only using the data from the 30 VID classes). We trained it in 36 epochs using asynchronous gradient descent with 0.9 momentum, 0.0005 weight decay, in a batch size of 4 images on 2 GPUs for both datasets. The initial learning rate was 0.005 and 0.01 on UA-DETRAC dataset and ImageNet VID dataset, respectively. And we decreased the rate by 0.1 after 18 and 30 epochs.

In the second step, we fine-tuned the model injected TFEN in a temporal manner. The initial learning rate was set to 0.001 for both datasets, and we decreased it in the same way as the baseline model. We adapted the random horizontal flip for the UA-DETRAC dataset for the data augmentation. Following the common data augmentation



(a) UA-DETRAC `test`

(b) ImageNet VID `val`

Figure 3.5: Accuracy v.s. FPS under different number m of frames to be stored in the external memory.

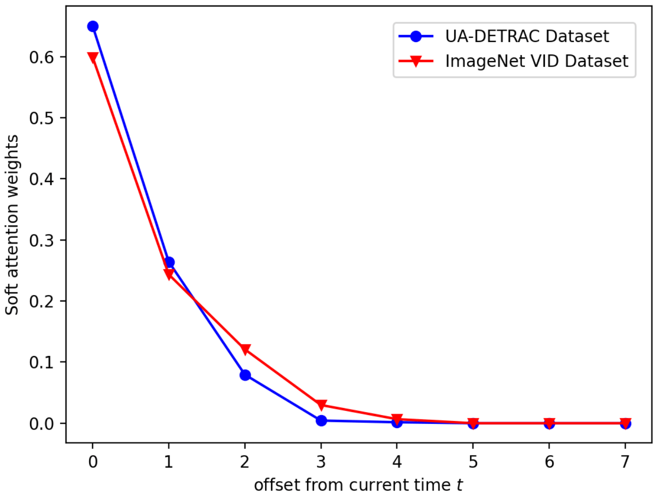


Figure 3.6: Soft attention weights used in the temporal decoder.

in the ImageNet VID dataset, we adapted random sample crop, random horizontal flip, and photometric distortions as in [111, 112] for the ImageNet VID dataset.

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 ($m = 4$) runs in 29.11 and 29.02 fps on UA-DETRAC and ImageNet VID datasets, respectively, and consumes about 1.9 GiB of GPU memory for all components, including external memory.

3.3.3 Model Design Analysis

We first experimentally investigated the optimal channel compressibility and the number of frames stored in the external memory in terms of the trade-off between speed and

Table 3.1: AP v.s. FPS under different compressibility p of the output channel dimension on UA-DETRAC `test`.

p	1.0	0.7	0.5	0.3	0.1
AP[%]	84.12	82.77	82.40	76.53	73.42
FPS	40.92	42.53	43.96	46.13	49.32

Table 3.2: mAP v.s. FPS under different compressibility p of the output channel dimension on ImageNet VID `val`.

p	1.0	0.7	0.5	0.3	0.1
mAP[%]	69.4	69.1	68.9	67.4	65.1
FPS	41.43	43.02	44.96	46.65	49.87

accuracy. This allows us to fix the parameters not determined through training TFEN.

Bottleneck Dimension

We analyzed the impacts of the ConvGRU output channel dimension on accuracy and speed. In this experiment, we changed the compressibility p, which defines the number of output channels of the feature map in the spatiotemporal encoder, from 1.0 to 0.1. We remark that we fixed the number m of frames in the external memory to four and used FP16 due to our GPU memory constraint. We also remark that the processing speed of models using FP16 tends to be faster than FP32 since the computation can be done efficiently using Tensor Core units on GPUs.

Tables 3.1 and 3.2 show the impacts on accuracy and speed under different p on the UA-DETRAC and ImageNet VID datasets. We observe that the accuracy is decreased by compressing the feature map while the processing time and the model capacity are reduced. We also see that the accuracy remains almost constant up to $p = 0.5$, then drops. This is applied to both datasets. Accordingly, we confirm that when aggregating feature maps from the past, it is unnecessary to use the actual feature maps obtained from the frames and that it is possible to reduce the weight to some extent. From this experiment, we may conclude that the compressibility p controls the trade-off between detection speed and accuracy of TFEN and that $p = 0.5$ is the best choice.

Number of Frames in Attention Decoder

We evaluated the number m of frames to be stored in the external memory. Since four 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 accuracy (AP or mAP) and fps. We also computed soft attention weights to see temporally-aware feature maps of which frames are really focused on to derive the enhanced feature map.

Figure 3.5 illustrates accuracy and fps under different m on the UA-DETRAC and ImageNet VID datasets, respectively. Figure 3.6 shows the average of soft attention weights when $m = 8$. 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.

We see that from Figure 3.5 the accuracy tends to be improved by increasing the number of frames and is saturated with $m = 6$ on both the datasets. On the other hand, the run-time speed decreases as m increases. We may conclude that $m = 4, 5$ or 6 is a good compromise as the trade-off between accuracy and speed.

Figure 3.6 shows that the weight for the current frame is most significant, which is represented as 0 in the horizontal axis, and weights for the last 3 and 4 frames are dominant. It also shows that even if we store eight frames in the external memory, the frames that are really used in the computation are the last 3 or 4 frames.

Figure 3.7 visualizes some detection results by TFEN ($m = 2, 4, 6, 8$) where lots of blur is present due to object or camera motion. We can see that the detection's confidence and location become more stable by aggregating over longer periods ($m = 8$ is better than $m = 4, 6$, for example). However, when comparing the improvement between $m = 2$ and $m = 4$ with that between $m = 4$ and $m = 8$, we see that the improvement between $m = 4$ and $m = 8$ is smaller and is not significant.

The above observation indicates that storing more than five frames in the external memory results in just taking run-time while it does not contribute to improving accuracy much. Accordingly, we can conclude that in practice, $m = 4$ is the best choice in terms of accuracy and speed.

3.3.4 Comparison with State-of-the-Art

We set $m = 4$ and $p = 0.5$ according to the above experimental results and compared the average precision (AP) and the mean average precision (mAP) of TFEN with the state-of-the-art methods.

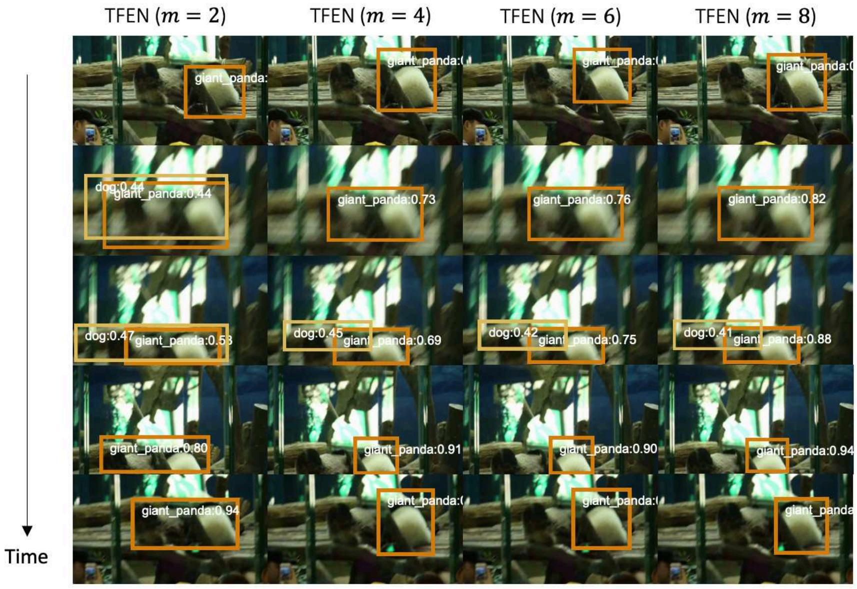


Figure 3.7: Example detection results of TFEN ($m = 2, 4, 6, 8$) for frames with lots of blur on ImageNet VID `val`. A bounding box is plotted if its confidence score is larger than 0.4.

Table 3.3: Performance comparison with state-of-the-art end-to-end models for live-streaming videos on ImageNet VID `val`.

Method	Components				Performances	
	Backbone	Feature Aggregation?	Attention?	RNN?	mAP	FPS (Device)
D&T [54]	ResNet-101 [81]				78.7	8 (Titan X)
LSTM-SSD [110]	MobileNetV1 [84]			✓	54.4	15 (Pixel 2)
Memory-guided [111]	MobileNetV2 [136]			✓	61.4	24 (Pixel 3)
Flow-guided [190]	MobileNetV1 [84]			✓	61.2	13 (Mate 8)
LMP [193]	MobileNetV2 [136]		✓	✓	64.2	29 (GTX 1060)
TSSD [26]	VGG-16 [141]		✓	✓	64.8	27 (Titan X)
TSSD(-OTA) [27]	VGG-16 [141]		✓	✓	65.4	21 (Titan X)
VOD-MT [86]	VGG-16 [141]	✓	✓	✓	71.0	18 (–)
TFEN ($m = 4$)	MobileNetV2 [136]	✓	✓	✓	68.9	29 (2080 Ti)
TFEN ($m = 6$)	MobileNetV2 [136]	✓	✓	✓	69.2	28 (2080 Ti)
TFEN ($m = 4$) w/ SSD	VGG-16 [141]	✓	✓	✓	70.6	25 (2080 Ti)

Comparison on ImageNet VID

Table 3.3 shows the performance comparison with other object detection methods for live-streaming videos on ImageNet VID. We present models of TFEN ($m = 4, 6$) trained using FP32. TFEN($m = 6$) was trained with half of the batch size for memory allocation[‡]. In terms of accuracy, D&T, which employs the tracking operation, surpasses all methods; however, it cannot run in real-time because of the high cost of simultaneous detection and tracking and the heavy backbone. Some methods using ResNet-101 [81]

[‡] We confirm that when using FP32, the performances in accuracy and run-time speed at $m = 4$ and $m = 6$ are almost the same as the case where we use FP16.

or VGG [141] as the backbone achieve higher mAP, but using such heavy backbones is not suitable in the context of real-time object detection in live-streaming video.

We see that methods exploiting MobileNetV1 [84] or MobileNetV2 [136] as the backbone realize real-time object detection thanks to the lightness of the backbone; however, they tend to achieve lower mAP compared to methods having richer feature extractors. In contrast, TFEN achieves the highest accuracy among the methods using MobileNet (either V1 or V2), beating the second place by about four points [193]. This is because TFEN exploits feature aggregation, attention mechanism, and recurrent neural networks altogether, and, furthermore, their combination brings the improvement of accuracy.

We see that although VOD-MT [86] outperforms TFEN, its backbone is heavier than that of TFEN. As a result, VOD-MT runs at only 18 fps while TFEN does at 29 fps. To fairly compare TFEN with VOD-MT, we used VGG-16 [141] as the backbone and SDD as the detector for TFEN, resulting in the difference from VOD-MT is the only feature aggregation module (the last line in Table 3.3). We see that TFEN processes more than 7 fps faster than VOD-MT under the same backbone and detector while achieving comparable accuracy. This faster processing time comes from the one-shot manner aggregation of TFEN. We also see that a lightweight feature extractor, MobileNetV2, speeds up the processing time even more. We confirm that TFEN is more suitable for applications where run-time speed is essential.

Comparison on UA-DETRAC

Table 3.4 shows the performance comparison on UA-DETRAC. We trained and evaluated TSSD [26] with the UA-DETRAC dataset from the official code. We re-implemented VOD-MT by ourselves based on [86] (referred to as VOD-MT$\star$) because the code is not publicly available. We remark that most of the published works on UA-DETRAC benchmarks are for still-image object detection. We also remark that CSP, RD^2, ExtendNet, IMIVD-TF, and MYOLO are not available as published papers up to now, and thus we used their reported scores and regard them just as reference scores.

FFAVOD-SpotNet performs best among the methods, which is an offline algorithm utilizing future information; however, it uses a heavy backbone (namely, Hourglass-104 [97] and cannot run in real-time. Except for the offline method [125], Table 3.4

Table 3.4: Comparison of AP scores [%] on UA-DETRAC `test` under various environmental conditions. Bold faces are the top performance on each subset. Methods in the first block are for still images. The methods in the second block are for videos. The bottom block lists unpublished methods and thus shows just the reference scores. (* is tested by ourselves; ⋆ is our own implementation.)

Method	Backbone	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny	FPS	GPU
DPM [55]	–	25.70	34.42	30.29	17.62	24.78	30.91	25.55	31.77	0.17	–
ACF [47]	–	46.35	54.27	51.52	38.07	58.30	35.29	37.09	66.58	0.67	–
R-CNN [71]	–	48.95	59.31	54.06	39.47	59.73	39.32	39.06	67.52	0.10	Tesla K40
CompACT [14]	–	53.23	64.84	58.70	43.16	63.23	46.37	44.21	71.16	0.22	Tesla K40
Faster R-CNN [131]	VGG-16 [141]	58.45	82.75	63.05	44.25	62.34	66.29	45.16	69.85	11	Titan X
GP-FRCNN [140]	VGG-M [141]	76.57	91.79	80.85	66.05	85.16	81.23	68.59	77.20	4	Tesla K40
EB [156]	VGG-16 [141]	67.96	89.65	73.12	54.64	72.42	73.93	53.40	83.73	11	Titan X
YOLOv3-SPP [92]	Darknet-53 [130]	84.96	95.59	89.95	75.34	88.12	88.81	77.46	89.46	6-7	Titan Xp
MSVD_SPP [93]	Darknet-53 [130]	85.29	96.04	89.42	76.55	88.00	88.67	78.90	88.91	9-10	Titan Xp
FG–BR_Net [59]	ResNet-18 [81]	79.96	93.49	83.60	70.78	87.36	78.42	70.50	89.89	10	Tesla M40
SpotNet [124]	Hourglass-104 [97]	86.80	97.58	92.57	76.58	89.38	89.53	80.93	91.42	–	GTX 1080 Ti
3D-DETNET [101]	Darknet-Conv23 [101]	53.30	66.66	59.26	43.22	63.30	52.90	44.27	71.26	26	–
RN-VID [126]	VGG-16 [141]	70.57	87.50	75.53	58.04	80.69	69.56	56.15	83.15	–	–
FFAVOD-SpotNet [125]	U-Net [125]	**88.10**	**97.82**	**92.84**	**79.14**	**91.25**	**89.55**	**82.85**	**91.72**	–	–
LMP* [193]	MobileNetV2 [136]	55.11	79.98	60.31	40.09	56.82	61.32	43.32	65.16	30	RTX 2080 Ti
TSSD* [26]	VGG-16 [141]	57.16	81.06	62.07	43.14	57.59	63.87	44.98	67.73	**32**	RTX 2080 Ti
VOD-MT⋆ [86]	VGG-16 [141]	67.22	82.81	74.36	55.29	71.43	66.79	64.16	70.82	14	RTX 2080 Ti
TFEN	MobileNetV2 [136]	82.42	97.40	88.90	72.18	87.54	82.41	72.32	90.78	29	RTX 2080 Ti
CSP [113]	ResNet-50 [81]	77.67	93.65	83.67	64.54	89.66	86.81	61.39	80.63	4	Tesla K40
RD^2 [119]	–	85.35	95.80	89.84	76.64	89.67	86.59	78.17	90.49	–	Tesla P40
ExtendNet [119]	–	83.59	95.46	88.75	73.36	86.89	85.05	76.75	90.77	45	Titan X
IMIVD-TF [119]	–	85.67	96.32	91.17	75.45	87.02	88.93	80.60	89.69	1	–
MYOLO [119]	–	83.50	95.15	88.18	73.99	88.58	83.38	77.06	88.37	7	–

shows that TFEN ranks at the top among all methods except for SpotNet on the easy and sunny subsets. We also see that TFEN outperforms VOD-MT⋆ by large margins in accuracy and speed. The gap between TFEN and the other methods except for TSSD is significant in speed but not in accuracy. TSSD also runs in real-time, but its performance is far worse than TFEN.

Figure 3.8 shows precision-recall curve comparison. As with TFEN, Faster R-CNN and EB have a 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.

3.3.5 Qualitative Comparison

Figure 3.9 shows detection results obtained by TFEN, the baseline model (MobileNetV2 based Cascade R-CNN), and TSSD along a sequence of frames on the UA-DETRAC dataset. It also shows ground truth. The video clip shows that both TFEN and TSSD successfully detect the occluded car behind the bus while the baseline model fails. This confirms the importance of using temporal information because the baseline model does not use temporal information at all.

Figure 3.10 illustrates some visualized examples of the detection result by the baseline, TSSD and TFEN. First, we can see that the baseline model and TSSD are not stable

Table 3.5: Performance comparison of ablation models

Method	Components					UA-DETRAC				Imagenet VID
	Video	Temporal Attention Decoder	Skip Connection	Spatial Attention	Temporally-aware Feature map	Overall	Easy	Medium	Hard	mAP
(a) baseline model						73.39	90.92	79.28	60.33	64.1
(b) model w/o TAD	✓		✓	✓	✓	79.26	95.96	85.83	67.42	67.8
(c) model w/o SK	✓	✓		✓	✓	72.53	91.26	78.57	59.24	64.3
(d) model w/o SA	✓	✓	✓		✓	80.93	97.17	86.08	66.44	68.5
(e) model w/o TF	✓	✓	✓	✓		79.22	95.06	84.77	65.46	67.7
(f) (complete) TFEN	✓	✓	✓	✓	✓	82.42	97.40	88.90	72.18	68.9

in detection, as it sometimes incorrectly labels the target with another class or fails to detect the target even when the target is not moving much. On the other hand, the detection results by TFEN are more stable, and the confidence score tends to be higher. It is thanks to incorporating feature aggregation in TFEN.

3.3.6 Detailed Analysis

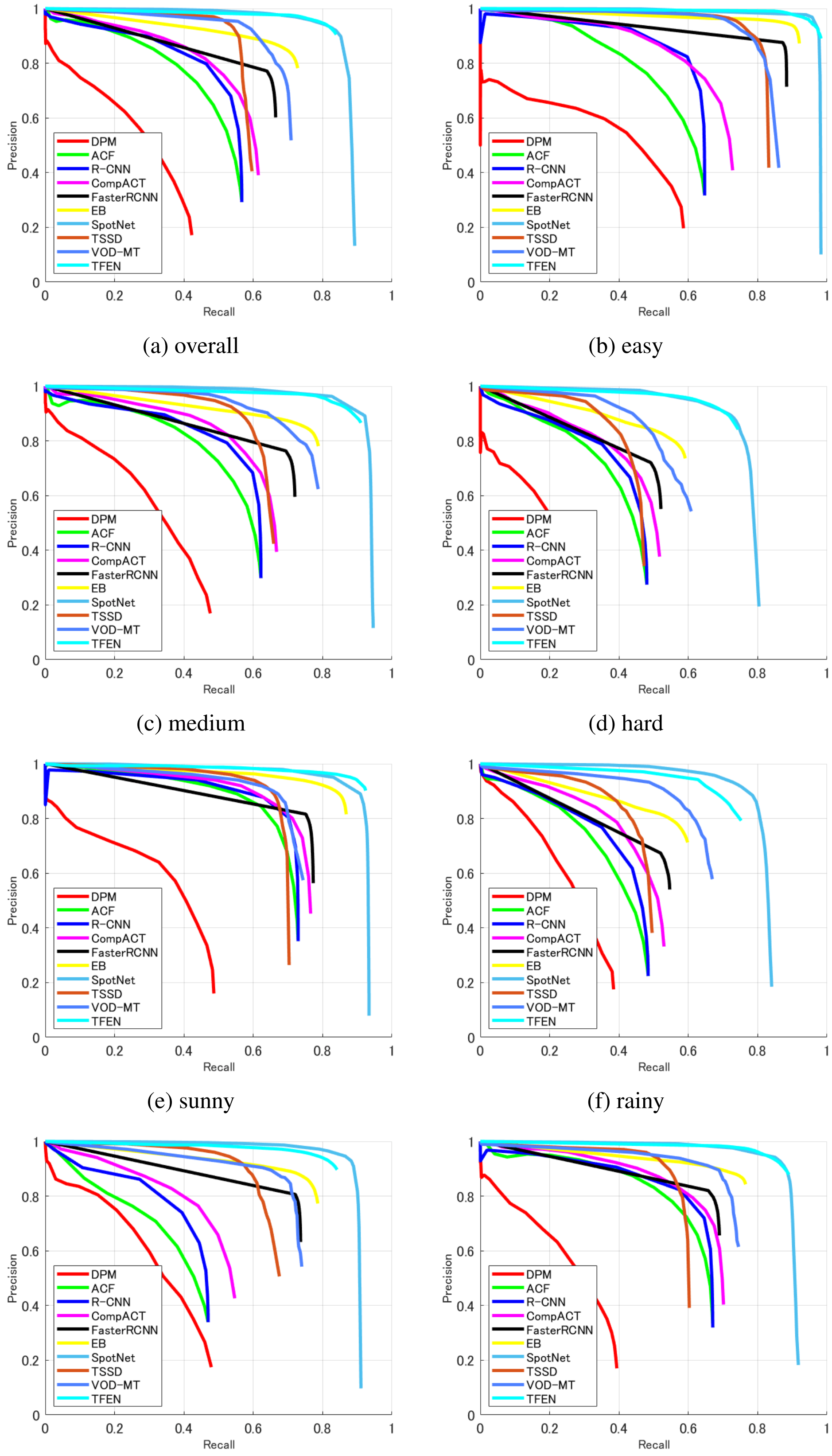
We evaluated the effectiveness of each component in TFEN to show its necessity on both datasets. We removed each component of TFEN one by one from the complete model to have ablation models. They are the model w/o TAD (temporal attention decoder), model w/o SK (skip connection), model w/o SA (spatial attention), and model w/o TF (temporally-aware feature maps). Note that the baseline model corresponds to the model dropping TFEN. Performances of the ablation models and the baseline model are illustrated in Table 3.5.

Temporal Attention Decoder

The model w/o TAD (Table 3.5 (b)) shows the ablation results of replacing the temporal attention decoder with a standard decoder without an attention mechanism. This replacing decoder consists of a simple stacked ConvGRU and ReLU, which receives only current-frame feature maps and has no external memories; the model w/o TAD thus cannot exploit past feature maps except the hidden state. The performance of the model w/o TAD drops 3.16 points of AP and 1.3 points of mAP for the overall subset and mAP, respectively. This demonstrates the effectiveness of the temporal attention decoder.

Skip Connection

Table 3.5 (c) and (f) show that applying skip connection between the feature extractor and the object detector brings consistent gains on all the sets. Moreover, Table 3.5 (a) and (c) reveal that the model w/o SK has lower scores than the baseline model in overall,



(a) overall

(b) easy

(c) medium

(d) hard

(e) sunny

(f) rainy

(g) night

(h) cloudy

Figure 3.8: Comparison of precision-recall curves on each subset of UA-DETRAC `test`. We show results by the default models provided by the dataset providers [164] and results by SpotNet, TSSD, and VOD-MT.

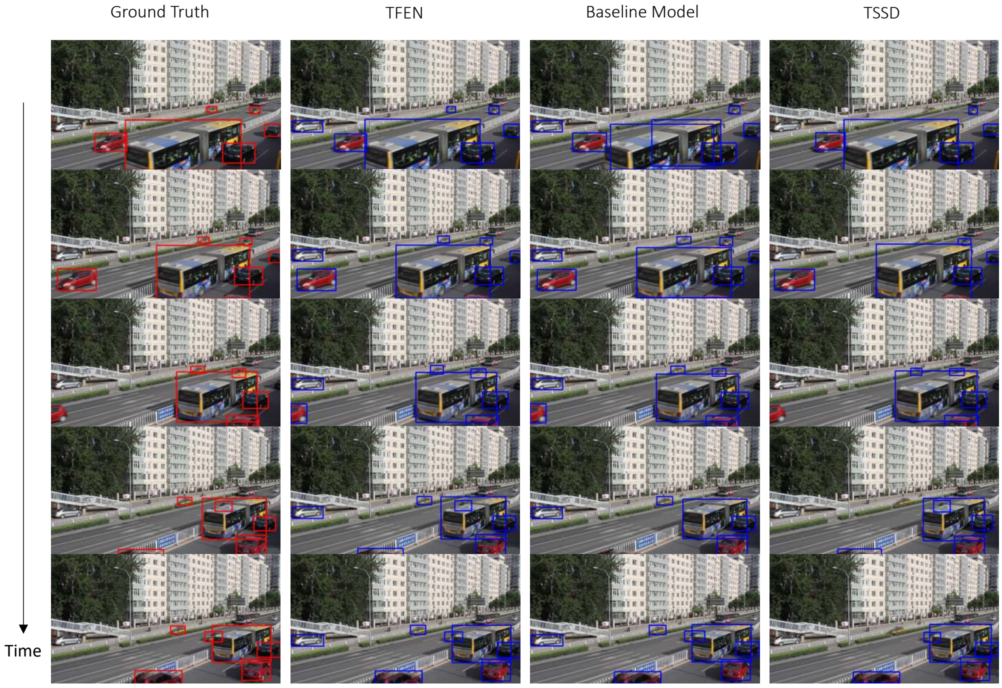


Figure 3.9: Example detection results of TFEN, Baseline Model, and TSSD on UA-DETRAC dataset.

medium, and hard subsets on the UA-DETRAC dataset. On the ImageNet VID dataset, the model w/o SK slightly outperforms the baseline model but achieves the lowest score among all the ablation models. We conjecture that the skip connection helps gradient flow through TFEN and is an essential part of TFEN.

Spatial Attention

Table 3.5 (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. We also see 0.4 point improvements on ImageNet VID. This suggests that the spatial attention mechanism is useful for all the subsets, but especially for the hard subset. We confirm 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.

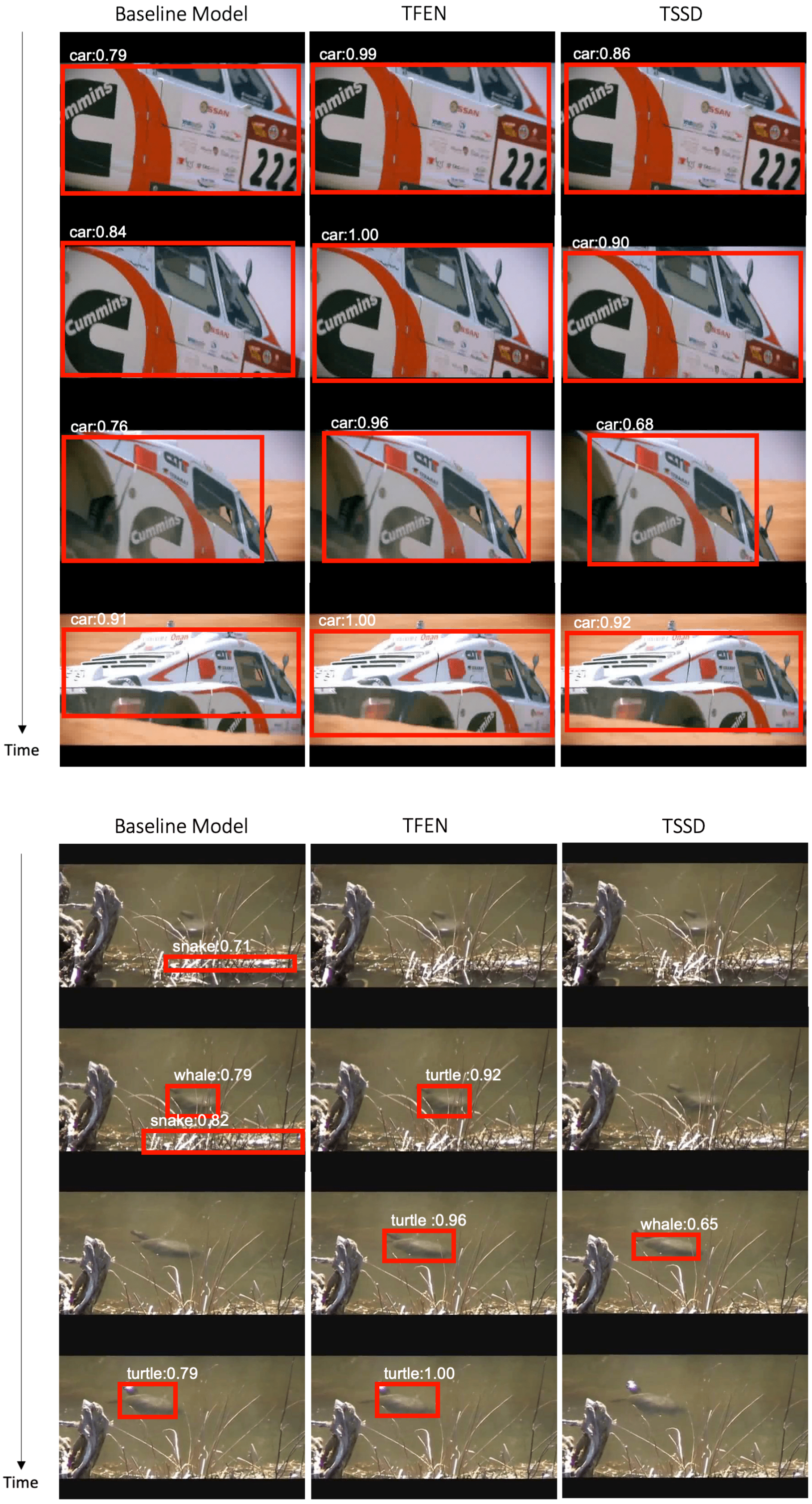


Figure 3.10: Example detection results of TFEN, TSSD and Baseline Model on ImageNet VID `val`. The left column of each video corresponds to the baseline model, the middle one corresponds to the proposed model, TFEN, and the right one corresponds to TSSD. A bounding box is plotted if its confidence score is larger than 0.4.

Temporally-aware Feature Map

To verify the necessity of the temporally-aware feature map, we store the feature maps without temporal information in the external memory instead of the temporally-aware feature map, 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 3.5 (e) and (f) show that temporal information in the external memory gives additional improvements 3.20%, 2.34%, 4.13%, 6.72%, 1.2% on overall, easy, medium, hard subsets, and ImageNet VID, respectively. This suggests that the temporal information stored in the external memory is useful for all the sets. We thus confirm that both the feature aggregation with temporal attention mechanism and the temporal-aware feature map are necessary for improving the detection accuracy.

Table 3.6: Effectiveness of temporal attention mechanism on ImageNet VID `val` and UA-DETRAC `test`.

Methods	ImageNet VID mAP	UA-DETRAC AP	FPS
Attention (ours)	68.9	82.4	29.1
Weighted averaging	63.3	79.8	31.3
Cos similarity	65.4	81.0	27.8

Attention based Feature Aggregation

We validated our introduced attention mechanism in feature aggregation by replacing it with other aggregation ways on the ImageNet VID and UA-DETRAC dataset. Table 3.6 shows the results. Our proposed attention mechanism for feature aggregation, dynamic weighting, is denoted by attention. Weighted averaging [110] is a static weighting way for feature aggregation where the weights of frames are determined so that the previous and current frames are weighted 1:3, and all other frames are set to 1. Cos-similarity [191] is another dynamic weighting way where the weights of the current and past frames are dynamically computed using cosine similarity and the softmax function. We confirmed the effectiveness of the TFEN component of the proposed method by replacing it with other aggregation methods.

We see from Table 3.6 that the dynamic weighting ways are more accurate than the static weighting as the weights are changed for each video and that our introduced attention is most beneficial. We also observe that attention is superior in terms of run-time.

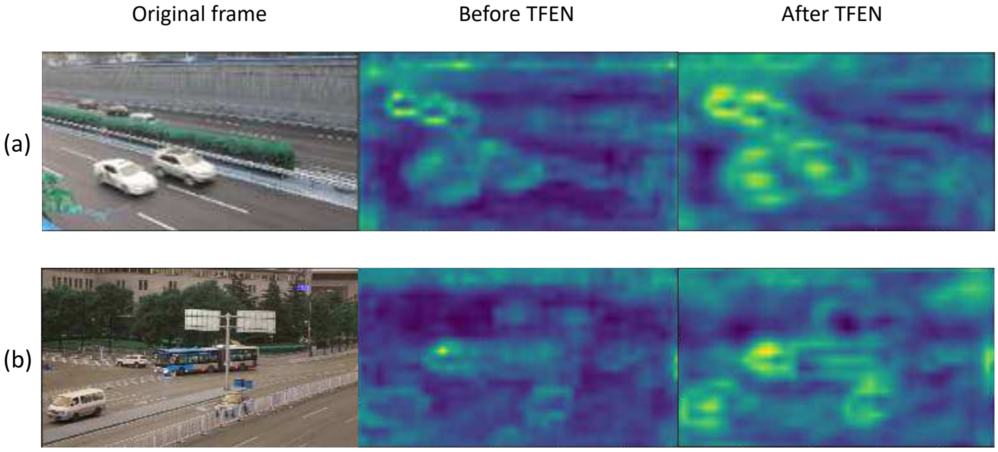


Figure 3.11: Feature activation before and after TFEN. For visualization, we summed up the feature maps in the channel direction, then mapped their values to the interval of 0-255, and up-sampled the feature channel by the bi-linear interpolation.

This is because it predicts the weights in a one-shot manner, not calculating weights one by one.

Analysis of Detection Confidence Score Fluctuation

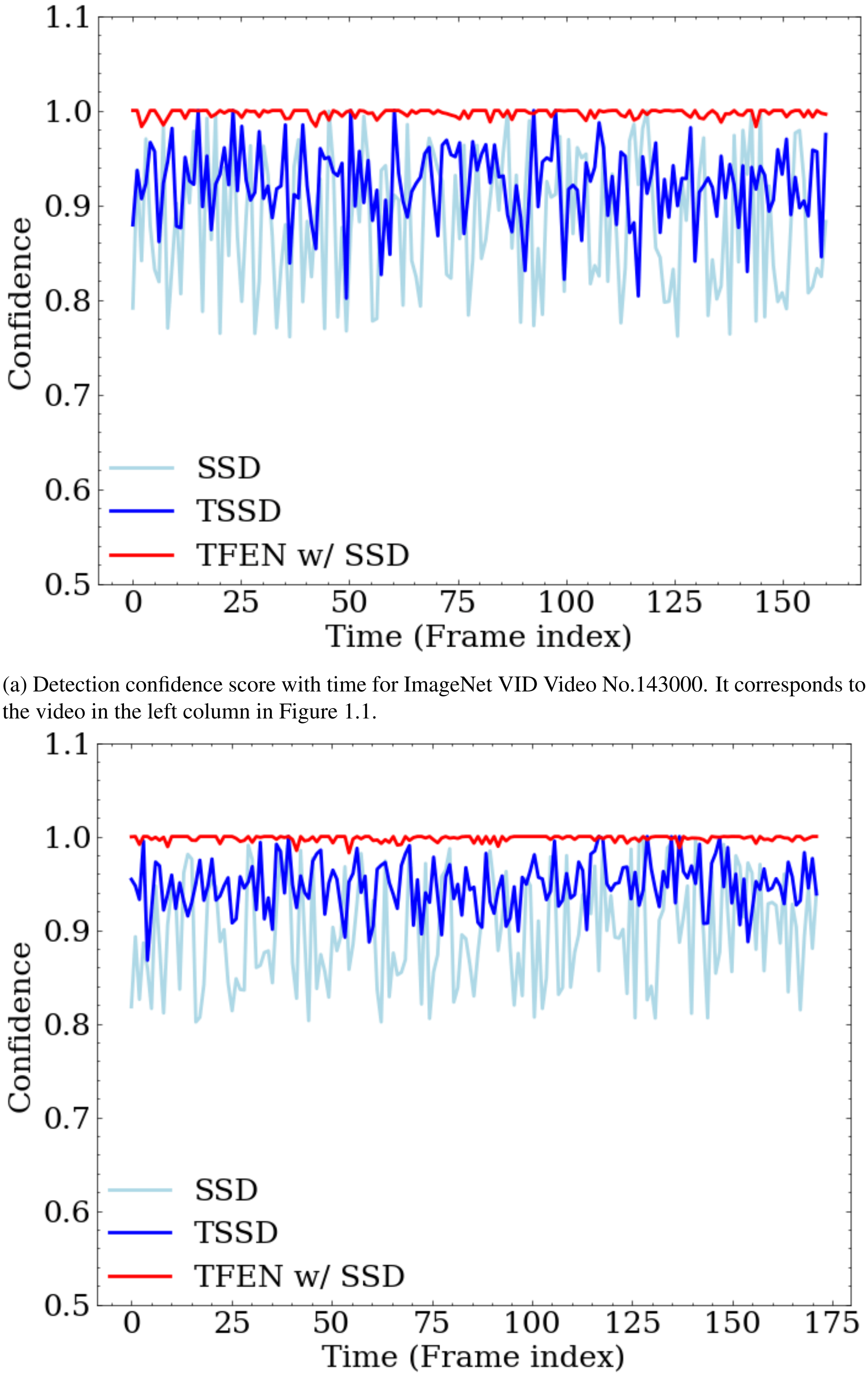
We examined how the proposed method differs from existing recurrent neural network-based methods regarding the fluctuation of detection confidence score. Figure 3.12 shows the confidence fluctuation with time for the baseline single-image SSD detector [112], the recurrent neural network-based TSSD [26], and the TFEN with SSD. In the baseline, the detection confidence score fluctuates greatly, and it can be confirmed that the fluctuation decreases with TSSD. In addition, TFEN significantly reduces it and maintains a high score. We see that enhancing the feature maps by feature aggregation is essential for detection with low fluctuation and high confidence score.

Feature Map Enhancement

We finally visualize the feature maps before and after TFEN (F_t and $\hat{F_t}$) in Figure 3.11 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 enhancement is practical and improves the detection accuracy.

3.4 Conclusion

We presented the temporal attention network with the external memory called TFEN for real-time object detection in live-streaming video. TFEN exploits the spatial and temporal information to enrich the feature map extracted using a lightweight feature extractor. While the feature aggregation approach has been used only for offline video object detectors, TFEN deals with frames in the memory with an efficient attention mechanism to realize online object detection in live-streaming videos. Compared with recent feature aggregation methods that aggregate relevant frames to focus on a frame by frame, TFEN utilizes temporally-aware feature maps to efficiently compute attention weights in a one-shot manner, which leads to real-time object detection. We confirmed that TFEN achieves real-time performance while keeping comparable accuracy with state-of-the-art methods on publicly available datasets. In addition, we showed that the proposed method detects objects with higher and more stable confidence scores than the existing methods, which is a challenge in video object detection. TFEN demonstrates the attention module's clear benefits based on the external memory and achieves a considerably enhanced trade-off between accuracy and speed.



(a) Detection confidence score with time for ImageNet VID Video No.143000. It corresponds to the video in the left column in Figure 1.1.

(b) Detection confidence score with time for ImageNet VID Video No.33000. It corresponds to the video in the right column in Figure 1.1.

Figure 3.12: Example of changes in detection confidence score over time.

Chapter 4

Element-level Feature Aggregation

4.1 Introduction

From two perspectives, this chapter deepens TFEN, a frame-level feature map aggregation. First, TFEN aggregates features by superimposing feature maps with frame-level weighting regarding the aggregation method. However, this method cannot deal with the significant movement of objects between frames. Therefore, we propose aggregating features at the element level to obtain more detailed information. Next, TFEN updates the external memory on a first-in-first-out rule without considering the importance of each frame. However, since each video is dynamic, it is difficult to deal with the sim-

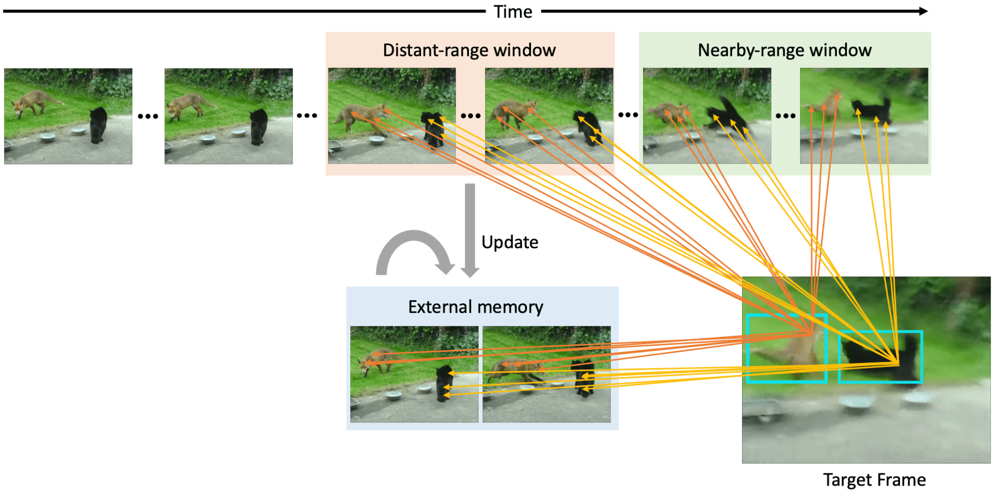


Figure 4.1: An example of the behavior of the VSTAM framework. It collects related information from the past frames spatiotemporally to refine the target frame's representation, including those in external memory. The orange and yellow arrows represent the highly related positions between frames.

ple rules. Therefore, we propose a method to update the external memory dynamically based on the importance of each frame. We explain the details from each perspective.

In video object detection, how to propagate features from surrounding frames to the current one is a vital issue, and many existing methods utilize only local information in both time and space [110, 158, 191]. FGFA [191] and MANet [158] proposed to utilize flow networks whereas TSSD-OTA [27] and some methods [110, 111] exploited recurrent neural networks to propagate features from neighboring frames. Although these methods show accuracy improvement, their refinements are limited for two reasons. First, they consider only local spatial information, such as the pixel-to-pixel transformation of flow-networks and the convolutional kernel of recurrent networks. They do not consider global information such as object relations and context in frames. Second, they utilize only nearby frames, meaning that information over long-term frames is missing. They are robust against a few successive deteriorated frames for issues such as motion blur, but if the problem persists for some time, it becomes challenging to deal with them using short-term frames. Thus, existing methods [27, 191, 192] suffer from a lack of capturing global aspects of video in both space and time to refine a feature map.

The memory consumption cost becomes crucial to capture the dependencies of distant frames. Simply using a static sliding window [9, 34, 191] is not a good choice. This is because a longer-range window to cover distant frames is memory-consuming. A static window may fail to capture videos' dynamic nature because the changes of objects are different at a time in each video. Instead, adaptively holding the most vital frame features identified through attention as an extended memory is more preferable. Such adaptation also reduces the memory-consumption cost since irrelevant frame features are not involved.

Based on the above observations, we propose Video-aware Sparse Transformer with Attention-guided Memory (VSTAM) that captures long-term dependencies in space and time (called "video-aware" in this paper) through aggregating features locally and globally in *both space and time* at the same level to obtain element-wise features. Aggregating features at the element level allows the model to cope with object misalignment flexibly. VSTAM possesses an attention-guided external memory that adaptively holds the most vital frame features. Figure 4.1 illustrates the concept of "element-wise feature aggregation with attention-guided memory", in which features associated with each element of the feature map are widely and appropriately aggregated from multiple locations and frames with sparse attention, and the more vital frame features are sequentially retained in the external memory based on attention. Even if some objects or frames are degraded, VSTAM appropriately uses features from other locations in other frames for aggregation. Moreover, if valuable frame features are in the sliding window, they can be updated to the external memory and aggregated. VSTAM does not suffer from computational expense and memory-intense.

Despite its simplicity, VSTAM achieves outperformance surprisingly on ImageNet VID dataset [134], UA-DETRAC [164] and VisDrone-VID against SOTAs. We furthermore demonstrate the applicability of VSTAM to a more complex task, i.e., Video Instance Segmentation, which requires more precise features for masks, showing comparable detection accuracy against SOTAs.

4.2 Related Work

This section reviews related works on element aggregation-based methods for attention-based extended memory in terms of “feature aggregation,” “external memory,” and “pre-training.” For the related works on video object detection in general, please refer to Section 2.2.

4.2.1 Feature Aggregation for Video Object Detection

Feature aggregation from nearby frames is essential to tackle video issues. Optical flow is used to align and warp features extracted from adjacent frames for aggregation [191, 192]. This approach, however, heavily relies on the accuracy of motion estimation. STSN [9] directly predicts sampling locations in different frames without using optical flow. STMN [171] proposed a spatial-temporal memory module to model long-term appearance and movement changes. TSSD-OTA [27] proposed an attentional recurrent neural network to refine a feature map. However, aggregation from only adjacent frames makes detection difficult when blurring occurs over multiple frames.

To overcome feature aggregation limitation using only nearby frames, using object-wise long-range temporal information was recently proposed. SELSA [167] considered the semantic impact between related object candidate regions in all the frames. RDN [44] distilled relation through repeatedly refining supportive object proposals with high confidences and used them to upgrade object-wise features. MEGA [29] considered both local and global aggregation to enhance the feature representation. However, these methods result in object-wise aggregation after detection, which can be difficult if detection is impossible. On the contrary, our proposed VSTAM considers element-wise features from spatial local-area and short-range temporal information to spatial global-area and long-range temporal information for aggregation.

4.2.2 External Memory for Video Object Detection

Video object detection using external memory has been studied in recent years. There are two main categories regarding their updating strategy. The first category employs the first in first out strategy, which utilizes the external memory to simply extend past features at instance-level [29] and frame-level [61]. However, those methods remove features in the external memory from the old one and simply update it with the new feature map.

The second one dynamically updates the external memory depending on the specific sampling strategy, and our approach falls into this category. OGEMN [42] proposes a top-down object-guided strategy, which computes features that give a high confidence level belonging to the detected objects and selects the higher ones to store. MAMBA [143] employs a random sampling strategy considering video redundancy and proposes feature-wise deleting to remove redundant features for efficient computation. Our method differs from the methods above in that it selects feature maps based on the sum of the attention weights of the aggregated elements at the frame level in a more straightforwardly bottom-up manner.

4.2.3 Transformer Network

The transformer [151] is a novel architecture for learning sequence data dependency. The vanilla transformer [151] and its similar one, non-local [160], are powerful models; however, they suffer from computational costs when they come to large tensors due to the large sequence length and resolution. Some attempts are reported to reduce the cost by making the transformer's self-attention map sparse [30, 137, 183, 185]. Sparse masks, such as slide windows, enable a transformer to abbreviate computation on no mask [30, 183]. Though these masks perform well on NLP tasks [183, 185] and a single

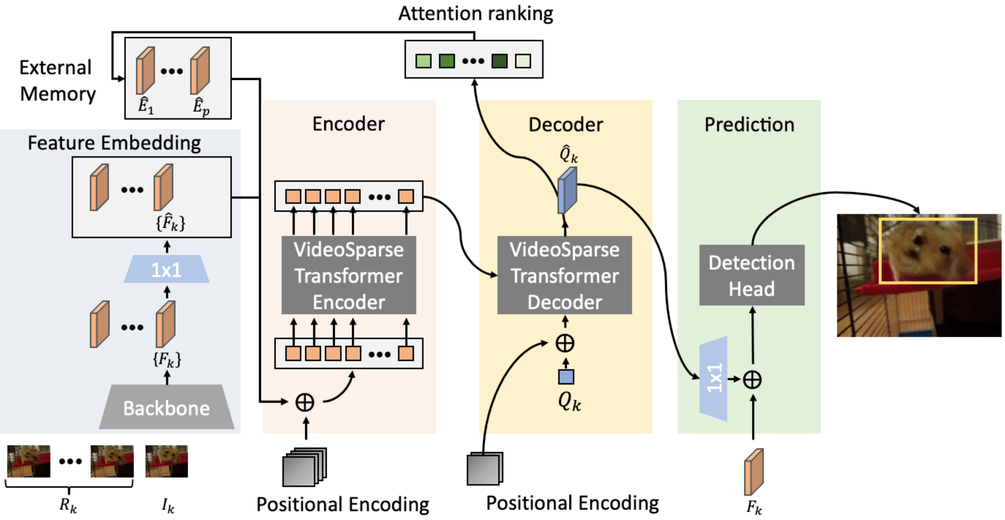


Figure 4.2: The architecture of the proposed Video-aware sparse transformer with attention-guided memory (VSTAM).

image [30], we cannot directly apply them to the video sequence due to spatial and temporal constraints. Our proposed *video-aware sparse attention* properly captures long-range dependencies in space and time with reasonable computational cost and memory consumption.

Recently, SSTVOS [52] has been proposed by making a transformer sparse in video object segmentation (VOS). However, it considers only temporally and spatially local elements and is vulnerable to significant object motion. Our method also considers the object moving in distant frames by incorporating randomness.

4.2.4 Pretraining

Pretraining is one area that has been received much attention in recent years and has shown progressive results in the natural language [46] and image recognition domains [28, 78]. Pretraining for video has been researched in video recognition where the time domain is essential, and methods such as video order estimation [56, 100] and pace estimation [155] have been proposed. However, pretraining for VOD has not been exploited where spatial and temporal information is essential. We propose an effective pretraining method for VOD through the object motion estimation and reconstruction tasks.

4.3 Proposed Method

4.3.1 Overview

Our proposed VSTAM is depicted in Figure 4.2, which is notably simple. It contains five components: feature embedding with frame selection, encoder, decoder, detection networks, and external memory. First, to feed short- and long-range temporal information, we effectively collect both nearby and distant frames. The feature embedding module then extracts feature maps. The feature maps are further compressed and flattened into one dimension [18]. Then, they are concatenated in the timeline order with the feature maps in the external memory to have a one-dimensional sequence. Then, the sequence together with positional encoding is passed to the encoder module to exploit the long-range sequential dependency among frames. Next, the high-level encoded feature map and the positioned frame query are passed into the decoder module for aggregation to have video-aware enriched features. They are passed to the detection network for object detection. Finally, the feature maps to be held are selected for the subsequent time detection. The external memory is updated based on the attention weight of each frame in the decoder to utilize the feature map of essential frames in the distant frame window.

4.3.2 Frame Selection from Short- and Long-ranges

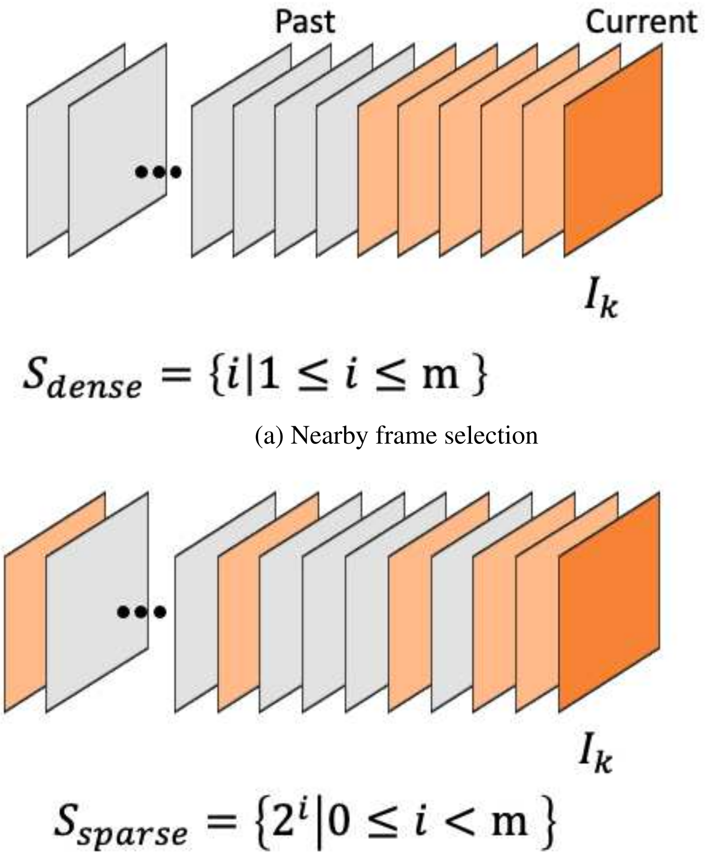
Given video frames $\{I_t\}_{t=1}^{T} \in \mathbb{R}^{H_0 \times W_0 \times C_0}$, where T is the length of a video and H_0, W_0 and C_0 respectively denote height, width, and number of channels, our goal is to detect objects in the current frame (at time k) I_k with reference frames R_k where $|R_k| = m$ for a given m. Reference frames are used for aggregation to have the enriched features of the current frames.

To capture the long-term temporal dependencies of a video, we need to collect reference frames from short- and long-term periods. For a given m (the number of past frames used for aggregation), we define the set S_{sparse} of differences of time from the current frame time as follows: $S_{\text{sparse}} = \{2^i | 0 \leq i < m\}$ (Fig. 4.3b). We then define $R_k = \{I_{k-n} | n \in S_{\text{sparse}}\}$. In this way, we can effectively collect nearby and distant frames as reference frames. I_k and R_k are fed to the feature embedding module.

Our collected reference frames consist of nearby frames that complement the blur in a short time densely and frames that are less affected by rare poses sparsely. Compared with the standard dense sampling [191] (Fig. 4.3a), our collection way allows us to obtain a broader range of information with the same number of reference frames. We remark that we use “nearby frames” to refer to the last five frames and “distant frames” to refer to the frames after that since in existing works [61, 191] focusing only on the nearby frames, the weights were applied only up to five frames.

4.3.3 Feature Embedding

Given the selected frames R_k and I_k, the feature embedding module extracts feature maps $\{F_k\}$. We utilize a shared-weighted ResNet [81] or ResNeXt [172]. Following DETR [18], we use a 1×1 convolution to reduce the channel dimension of feature maps $\{F_k\} \in \mathbb{R}^{H \times W \times C}$ from C to a smaller dimension d, creating new feature maps $\{\hat{F}_k\} \in \mathbb{R}^{H \times W \times d}$. We then collapse the spatial dimensions of $\{\hat{F}_k\}$ into one dimension, resulting in $HW \times d$ feature maps. The external memory contains additional flattened


Past
Current
I_k
$S_{dense} = \{i|1 \leq i \leq \mathrm{m}\}$
I_k
$S_{sparse} = \{2^i|0 \leq i < \mathrm{m}\}$

(a) Nearby frame selection

(b) Nearby-distant frame selection

Figure 4.3: The reference frame selection for feature aggregation from a video clip. m is the number of past frames used for aggregate. Dark and light orange indicate the current frame and the selected reference ones, respectively.

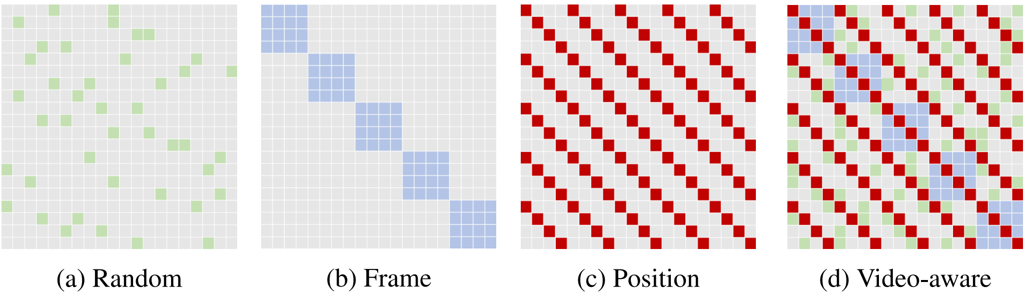


Figure 4.4: Visualization examples of the sparse attention. Here, we assume a video with five consecutive frames, each frame possessing a 2×2 feature element. Gray color indicates the absence of attention. (a) random attention, (b) frame-wise attention, which cares only in self-frame, (c) Position attention, which focuses on the same position of each frame, (d) the combined attention map of Video-aware Sparse Transformer.

feature maps $\{\hat{E}_q\}_{q=1}^{p}$. The newly sampled feature maps and the feature maps stored in the external memory are concatenated in the order of the timeline. The number of frames is $L = p + m + 1$. In this way, we obtain a feature sequence $Z \in \mathbb{R}^{LHW \times d}$.

4.3.4 Video-aware Sparse Transformer

Based on the vanilla transformer [151] which exploits a self-attention mechanism to learn the elements' dependencies and gather information for an input sequence, we develop a video-aware sparse transformer (VST) so that it aggregates information from multiple frame feature maps. A vanilla transformer considers all elements; however, considering all elements of the video sequence is unnecessary because of redundancy involved in a video (ex., objects may appear at similar positions for a certain period in multiple frames). We thus follow recent work [30, 185] in NLP that makes self-attention sparse and samples elements more efficiently for VST.

Video-aware Sparse Self-Attention

To realize video-aware sparse attention, the video-aware sparse attention masking operation $M(\cdot)$ is implemented on self-attention [151] based on the below consideration. The modified formulation of a uni-head sparse self-attention can be formulated as:

$$\text{SparseAttention}(Q, K, V) = \text{softmax}(M(\frac{QK^T}{\sqrt{d_k}}))V, \tag{4.1}$$

where $K \in \mathbb{R}^{l \times d_k}$, $V \in \mathbb{R}^{l \times d_v}$, $Q \in \mathbb{R}^{l \times d_k}$ are the key, value, and the query, respectively. l is the length of input sequence, d_v and d_k are the embedding dimension of the value and key. A sparse mask $M \in [0, 1]^{l \times l}$ is defined as

$$M(i, j) = \begin{cases} 1 & \text{if query } i \text{ attends to key } j, \\ 0 & \text{otherwise,.} \end{cases} \tag{4.2}$$

Video-aware sparse attention is designed taking into account the following considerations. First, to refer to all the elements in a frame globally and locally, we introduce the frame attention (Fig. 4.4b). It allows the self-attention to refer only to each frame's elements, thus improving the feature map while considering its spatial context. However, since it lacks temporal information, we introduce two types of sparse masks: random and position attention.

As the name implies, random attention (Fig. 4.4a) masks a certain percentage of the elements, allowing access to a wide range of features. Different from the original random attention [183], we mask each frame with a random probability r instead of the entire sequence. This is because a video's information is divided into frames.

Although random attention enables us to obtain information from multiple frames, it cannot sometimes aggregate features reliably when objects remain in a specific area over multiple frames. To reliably extract information from around the same location over multiple frames, we introduce position attention (Fig. 4.4c). It plays the role of aggregating features from the corresponding location in the temporal direction. It only considers the same position at each frame; it is sensitive to object motion. Therefore, we applied a mask like a 3×3 dilated convolution kernel (Rate = 2) [181] to each element to give the position attention to a wide field of view for robustness against object motion. The combined masks of frame, random, and position are video-aware sparse attention (Fig. 4.4d) and applied to the self-attention of transformer [151]. We exploit this sparse transformer for both the encoder and decoder.

Encoder and Decoder

The encoder and decoder follow the original layered architecture of the transformer [151] except for its self-attention. We replace the standard self-attention with video-aware sparse attention. Given the positioned and flattened feature sequence Z, we obtain the embedded sequence $\hat{Z} \in \mathbb{R}^{LHW \times d}$ via the encoder.

The function of the decoder is to generate a video-aware refined feature map. To decode at each element of the feature map, a query sequence $Q_k \in \mathbb{R}^{HW \times d}$ is required and obtained by flattening embedded feature map $\hat{F}_k$. Then, the decoder outputs the video-aware refined feature sequence $\hat{Q}_k$ using the query $\hat{Z}$ and embedded Q_k sequence.

4.3.5 Detection

Thanks to the decoder, we have refined sequence $\hat{Q}_k$. To utilize it for detection, we expand its spatial dimension. It contains the local and global information of the video sequence; however, it loses the detailed information due to the compression of 1×1 convolution operation in the feature embedding process. Therefore, we decompress $\hat{Q}_k$ with 1×1 deconvolution operation in the channel direction to generate the feature map $\tilde{Q}_k \in \mathbb{R}^{H \times W \times C}$. Then, we merge both feature maps $\tilde{Q}_k$ and F_k by the element-wise sum to acquire the final refined feature map for detection.

4.3.6 External Memory

To store the feature map of most vital frames adaptively in the external memory, we select them based on the importance of each frame according to their attention weights. They are already computed when VST aggregates each element, indicating the importance of the features. Therefore, we measure the importance of each frame by accumulating the weights for each element in each frame. This is the "Attention ranking" shown in Figure 4.2, and we keep up to the p-th feature map as $\{\hat{E}_q\}$ in external memory, arranged in the cumulative order of attention weights. We remark that the feature map candidates to be stored in the external memory are already ones there and the distant frames newly loaded in the sliding window. The role of the external memory is to hold long-term features and deal with scenes that are difficult to detect by using neighboring frames, so adjacent frames are not to be stored.

4.3.7 Pretraining

We introduce a pretraining for VOD to train the model more effectively. We use only feature embeddings, encoders, and decoders with no external memory for pretraining and let them learn the spatial and temporal context of the video through simple tasks. The proposed pretraining consists of the "reconstruction task" to acquire spatial information in a frame and the "average-motionIoU task" to obtain temporal information in

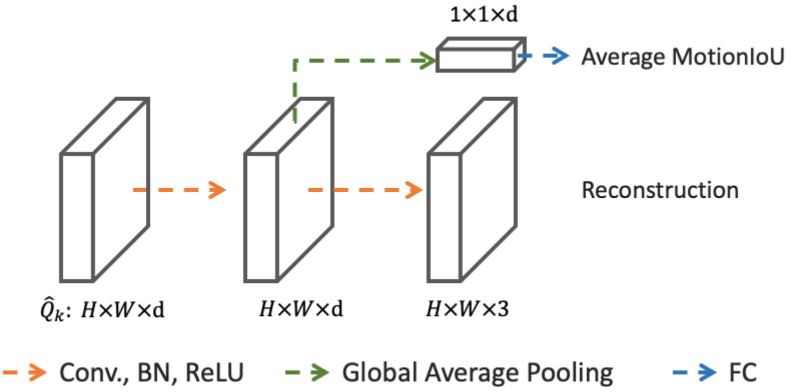


Figure 4.5: The sub-network for pretraining.

a video. The reconstruction task generates the current frame image like an autoencoder. The average-motionIoU task estimates the average of motionIoU for each object at the current frame. MotionIoU is proposed for evaluating moving objects in [191], the average of the IoU deviations of the same object over ± 10 frames from the current frame. Each of them is a spatial and temporal supervisory task essential for VOD.

To perform the two pretraining tasks, we attached the pretraining sub-network shown in Figure 4.5 to the $\hat{Q}_k$ obtained by the decoder. Using $\hat{Q}_k$ as input, we perform average-motionIoU and reconstruction after convolution, batch normalization, and ReLU, respectively. After the pretraining is completed, the head attached for pretraining is removed, and then the network for detection is connected.

We design a multi-task objective function to train our model. Namely, we utilize l2 loss for both motion IoU estimation loss $L_{\text{MotionIoU}}$ and reconstruction loss L_{reconst}, and the total loss is defined as follows:

$$L_{\text{pretrain}} = \alpha L_{\text{MotionIoU}} + \beta L_{\text{reconst}}, \tag{4.3}$$

where α and β are weighting parameters to balance the optimization of both the tasks.

4.4 Experiments

4.4.1 Benchmark Datasets and Metrics

We evaluated the performance of our method on tree public dataset: ImageNet VID [134], UA-DETRAC [164] and VisDrone-VID2019 [188] (see Section 3.3.1 for more details about ImageNet VID and UA-DETRAC.) The VisDrone-VID2019 dataset [188] was published as a large-scale detection benchmark for a drone viewpoint in the video. In addition to the usual video challenges such as motion blur, it offers additional challenges such as smaller object sizes, a large number of detection targets, and higher resolution frames.

We evaluated the performance using mean average precision (mAP) and average precision (AP) for ImageNet VID and UA-DETRAC, respectively, (cf. Section 3.3.1). VisDrone-VID2019 requires more precise localization accuracy. We use AP, AP_{50}, AP_{75}, AR_{1}, AR_{10}, AR_{100}, and AR_{500} metrics for evaluation, similar to that in MS COCO [107].

4.4.2 Network Architecture

Feature Extractor

We mainly conduct experiments with ResNet-50 [81] pretrained on ImageNet [43], if not otherwise noted. Following a common practice [158, 191], we enlarge the resolution of feature maps by modifying the stride of the first convolution block in the last stage of convolution, namely *conv5*, from 2 to 1. Besides, we set the dilation of these convolutional layers to 2 to retain the receptive field size.

Detection Network

We exploit Faster R-CNN [131] as our detection module. For a fair comparison, we follow the commonly employed setting [29, 68, 167]. Specifically, we leverage 12 anchor with 4 scales $\{64^2, 128^2, 256^2, 512^2, \}$ and 3 aspect ratios $\{1:2, 1:1, 2:1\}$ for regression and classification. During training and inference, 3000 and 300 candidate boxes are generated in previous and post non-maximum suppression (NMS), respectively.

VSTAM

We set the frame selection (Fig. 4.3b) at training and inference stages with temporal window size $m = 5$ in VSTAM. Unless otherwise noted, we conducted our experiments with external memory set to $q = 2$. Therefore, we utilize $L = 8$ frames, including the target frame for each batch in total. In the embedding process, we set the compressed dimension d of the feature map to 128. In VST, we utilize multi-heads attention with the number of heads $h = 8$. The number of layers in the encoder and decoder is set to 4, respectively. For the sparse attention, we set a random ratio with $r = 10\%$ at each frame.

4.4.3 Implementation Details

We implement VSTAM mainly on detectron2 [170]. The whole architecture is trained on two RTX 3090 GPUs with ADAMW [114]. Although we trained our model on two RTX 3090 GPUs, we evaluated the speed performance on Titan RTX GPUs for a fair comparison with other methods.

Pretraining Stage

We train VSTAM, whose detection head is replaced by the sub-network for pretraining, to sample a consecutive 6-frame with the same sparse frame selection. We resize frames to 512×512 and random-flip the whole video clip horizontally for data augmentation, following the video recognition pretraining protocol [3]. The batch size is eight, and the initial learning rate is 10^{-4}. All the weights are randomly initialized. The learning rate is divided by 10 for every six epochs, and the training process is stopped after 18 epochs. The hyperparameters α and β for L_{pretain} are set to 1 and 0.5, respectively.

Detection Training Stage

For ImageNet VID, we train VSTAM on a combination of ImageNet VID and DET datasets [134] following common protocols in [9, 29, 167]. For the DET dataset, we select the same 30 classes as in the VID dataset and follow the data augmentation strategy proposed in [111]. For VisDrone-VID2019 and UA-DETRAC, we utilize only its training dataset.

The input images are resized to have their smaller side to be 600 pixels. Each GPU holds two mini-batches, and each mini-batch contains one set of images or frames. The model with a vanilla transformer was trained with one mini-batch due to memory constraints and adjusted according to the batch size [73]. We train our network for 13 epochs with learning rate decay, dividing by ten at epochs 9 and 12, respectively. The initial learning rate is set to 10^{-4}. At inference, an NMS of 0.5 IoU threshold is adopted to suppress reduplicate detection boxes.

Table 4.1: Performance comparison on ImageNet VID.

Methods	Backbone	Base Detector	mAP		
			Online	Offline	Post-processing
DFF [192]	ResNet-101	R-FCN	73.1	–	–
THP [189]	ResNet-101 + DCN	R-FCN	78.6	–	–
OGEMN [42]	ResNet-101 + DCN	R-FCN	80.0	–	81.6
PLSA [74]	ResNet-101 + DCN	R-FCN	80.0	–	–
LSTS [88]	ResNet-101 + DCN	R-FCN	80.1	–	–
D&T [54]	ResNet-101	Faster R-CNN	80.2	–	–
LRTR [139]	ResNet-101	FPN	81.0	–	–
MEGA [29]	ResNet-101	Faster R-CNN	81.9	82.9	84.5
MAMBA [143]	ResNet-101	Faster R-CNN	84.6	–	–
VSTAM(Ours)	ResNet-101	Faster R-CNN	**85.5**	**86.1**(+0.6)	**86.4**(+0.3)
FGFA [191]	ResNet-101	R-FCN	–	76.3	78.4
MANet [158]	ResNet-101	R-FCN	–	78.1	80.3
STSN [9]	ResNet-101 + DCN	R-FCN	–	78.9	–
SELSA [167]	ResNet-101	Faster R-CNN	–	80.3	82.7
TM-VoD [94]	ResNet-101	Faster R-CNN	–	80.5	–
RDN [44]	ResNet-101	Faster R-CNN	–	81.8	83.8
TransVOD [82]	ResNet-101	Deformable DETR	–	81.9	–
TROI [72]	ResNet-101	Faster R-CNN	–	82.0	–
HVR-Net [76]	ResNet-101	Faster R-CNN	–	83.2	83.8
TF-Blender [36]	ResNet-101	Faster R-CNN	–	83.8	–
OFAVOD [68]	ResNet-101	Faster R-CNN	–	83.9	85.1
DSFNet [103]	ResNet-101	Faster R-CNN	–	84.1	–
EBFA [75]	ResNet-101	Faster R-CNN	–	84.8	–
LRTR [139]	ResNeXt-101	FPN	84.1	–	–
MAMBA [143]	ResNeXt-101	Faster R-CNN	85.4	–	–
VSTAM(Ours)	ResNeXt-101	Faster R-CNN	**86.9**	**87.6**(+0.7)	**88.0**(+0.4)
RDN [44]	ResNeXt-101	Faster R-CNN	–	83.2	–
MEGA [29]	ResNeXt-101	Faster R-CNN	–	84.1	85.4
TM-VoD [94]	ResNeXt-101	Faster R-CNN	–	84.5	–
HVR-Net [76]	ResNeXt-101	Faster R-CNN	–	84.8	85.5
DSFNet [103]	ResNeXt-101	Faster R-CNN	–	85.4	–
OFAVOD [68]	ResNeXt-101	Faster R-CNN	–	86.1	86.9

4.4.4 Comparison with State-of-the-Art

Table 4.2: Comparison of accuracy and runtime on ImageNet-VID `val`. All method employ Faster R-CNN with ResNet-101 and their processing time is measured on Titan RTX. The re-measured speed of SELSA, RDN, and MEGA is reported in [143].

Method	mAP	Runtime (ms)
SELSA [167]	80.3	91.3
RDN [44]	81.8	128.0
MEGA [29]	82.9	182.7
MAMBA [143]	84.6	110.3
Ours	85.5	95.2

Comparison on ImageNet VID

Many methods exploit ResNet-101 [81] and ResNext-101 [172]. We thus report scores using them for a fair comparison. We compare the models separately since the accuracy

Table 4.3: Comparison of External Memory Method on ImageNet-VID `val`. All processing time is measured on RTX Titan. The re-measured speed of OGEMN is reported in [143].

Method	mAP	Runtime (ms)
Base (R-FCN [37])	73.8	46.7
OGEMN [42]	79.3 (+5.5)	89.1
MAMBA [143]	81.6 (+7.8)	90.1
Ours	82.6 (+8.8)	80.2

differs among offline, online, and post-processing cases. For comparison, we show the offline setting results for the case where five frames extend the sliding window into the future ($L = 13$).

Table 4.1 shows the result comparison between state-of-the-art methods on online, offline and post-processing conditions. Among all the methods, VSTAM achieves the best performance on all backbone and conditions. With ResNet-101 backbone, our online model achieves 85.5% mAP, 0.9% absolute improvement over the recent and most powerful competitor MAMBA [143], which utilizes external memory. Compared to FGFA [191], MANet [158] and STSN [9], which aggregate element-wise feature from nearby frames, our proposed method outperform more than 8 points. The gap between the proposed method and the above is feature aggregation with global spatiotemporal context. VSTAM also outperforms some methods [29, 44, 68, 72, 167], which utilize object-wise feature aggregation. The object-wise approaches provide effective improvement but may lack sufficient information near objects. Our method considers element-wise features from distant and nearby frames and local to global in the feature map, leading to the best performance on ImageNet VID.

By replacing the backbone from ResNet-101 to ResNeXt-101, our model achieves a better performance of 86.9% mAP, as expected. In an offline setting, our model achieved an accuracy of 87.6%, making it the first model to achieve precision in the 87% range.

We applied post-processing to the offline model since many methods report offline. For the post-processing method, we adopt Seq-NMS [77], which refine scores of weaker detection from nearby frames. Our method still performs the best, obtaining 86.4% and 88.0% mAP with backbone ResNet-101 and ResNeXt-101, respectively.

Table 4.2 shows accuracy and speed comparison on the same architecture and GPUs. We can see that the proposed method is superior in accuracy while the speed is faster than most methods. Next, we replaced the detector with R-FCN [37] to compare the performance of the methods with the external memory under the same conditions and GPU. As shown in Table 4.3, we confirm that the proposed method is superior in accuracy and speed. OGEMN and MAMBA have two-step frame-wise and object-wise aggregation based on complex update and delete rules, requiring more processing time. On the other hand, the proposed method deals only with feature maps element-wisely and reduces run-time by a simple rule that holds the feature maps most used in the enhancement.

Table 4.4: Performance comparison with the state-of-the-art models on VisDrone-VID2019 `test`.

Methods	AP	AP_{50}	AP_{75}	AR_1	AR_{10}	AR_{100}	AR_{500}
FPN [105]	16.72	39.12	11.80	5.56	20.48	28.42	28.42
CornerNet [97]	16.49	35.79	12.89	9.47	24.07	30.68	30.68
CFE-SSDv2 [186]	21.57	44.75	17.95	11.85	30.46	41.89	44.82
D&T [54]	17.04	35.37	14.11	10.47	25.76	31.86	32.03
FGFA [191]	18.33	39.71	14.39	10.09	26.25	34.49	34.89
DBAI-Det [188]	29.22	58.00	25.34	14.30	35.58	50.75	53.67
Faster R-CNN [131]	14.46	31.80	11.20	8.55	21.31	26.77	26.77
Ours	32.16	60.71	27.52	16.45	38.91	56.21	56.21

Comparison on VisDrone-VID2019

Table 4.4 shows performance comparison. Most method [54, 105, 131, 191] utilize ResNet-101 as base backbone, we employ it in this section. D&T [54] and FGFA [191] are methods to stabilize the detection of the still image detector [131] with temporal information. We see that our model significantly improves the accuracy from the baseline, Faster R-CNN [131], and outperforms the compared methods. DBAI-Det [188] combines heavy backbone, ResNeXt-101 [172], more precise detection head, Cascade R-CNN [15], and several methods [17, 38] for accuracy. However, because it does not utilize temporal information, it is less accurate than our method, which uses a smaller backbone, ResNet-101.

Comparison on UA-DETRAC

The results on the UA-DETRAC dataset are reported in Table 4.5. Our online model achieves 90.3% AP, 2.2% absolute improvement over the recent detector, FFAVOD-SpotNet [125], which utilize temporal information including future frames. As expected,

Table 4.5: Performance comparison on UA-DETRAC `test`. Bold faces are the top performance on each subset.

Method	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny	FPS	GPU
GP-FRCNN [140]	76.57	91.79	80.85	66.05	85.16	81.23	68.59	77.20	4	Tesla K40
EB [156]	67.96	89.65	73.12	54.64	72.42	73.93	53.40	83.73	11	Titan X
YOLOv3-SPP [92]	84.96	95.59	89.95	75.34	88.12	88.81	77.46	89.46	6-7	Titan Xp
MSVD_SPP [93]	85.29	96.04	89.42	76.55	88.00	88.67	78.90	88.91	9-10	Titan Xp
FG–BR_Net [59]	79.96	93.49	83.60	70.78	87.36	78.42	70.50	89.89	10	Tesla M40
SpotNet [124]	86.80	97.58	92.57	76.58	89.38	89.53	80.93	91.42	14	GTX 1080 Ti
3D-DETNET [101]	53.30	66.66	59.26	43.22	63.30	52.90	44.27	71.26	26	–
RN-VID [126]	70.57	87.50	75.53	58.04	80.69	69.56	56.15	83.15	–	–
FFAVOD-SpotNet [125]	88.10	**97.82**	92.84	79.14	91.25	89.55	82.85	91.72	–	–
TFEN [61]	82.42	97.40	88.90	72.18	87.54	82.41	72.32	90.78	**29**	RTX 2080 Ti
Ours w/ ResNet-101	**90.30**	**97.82**	**94.67**	**82.07**	**92.48**	**91.92**	**84.91**	**94.51**	10	RTX Titan
Faster-RCNN w/ ResNet-101	76.18	92.01	79.52	65.27	85.10	73.96	67.26	86.37	18	RTX Titan
Ours w/ ResNet-50	85.13	95.82	88.11	75.13	89.23	86.31	77.00	91.08	18	RTX Titan
Faster-RCNN w/ ResNet-50	73.11	90.91	78.16	60.32	83.23	71.41	63.48	83.01	26	RTX Titan

the proposed method improves the detection performance on different datasets.

4.4.5 Detailed Analysis

To evaluate the effectiveness and superiority of each component in VSTAM, we conduct several ablation experiments. Specifically, we gradually modify online VSTAM with ResNet-50 and compare their differences.

In order to analyze details, we follow a motion-aware evaluation metric in [191] to evaluate the performance on the categories of a slow, medium, and fast objects, where these three categories are divided by their average IoU scores between objects across nearby frames. Slow motion means the case where IoU score is higher than 0.9, and fast motion means that IoU score is lower than 0.7. The medium motion indicates the rest. We note that mAP_s, mAP_m, mAP_f represent mAP(small), mAP(medium), mAP(fast), respectively.

Table 4.6: Impact of components in the proposed method on ImageNet VID `val`, VisDrone-VID `val` and UA-DETRAC `test`.

Components			ImageNet VID	VisDrone-VID	UA-DETRAC
Video Sparse Transformer	External Memory	Pretraining	mAP	AP	AP
			71.7	12.1	73.1
✓			77.1	18.7	81.8
✓	✓		80.0	22.6	84.2
✓	✓	✓	81.3	23.6	85.1
Vanilla transformer	✓	✓	80.1	22.7	83.8

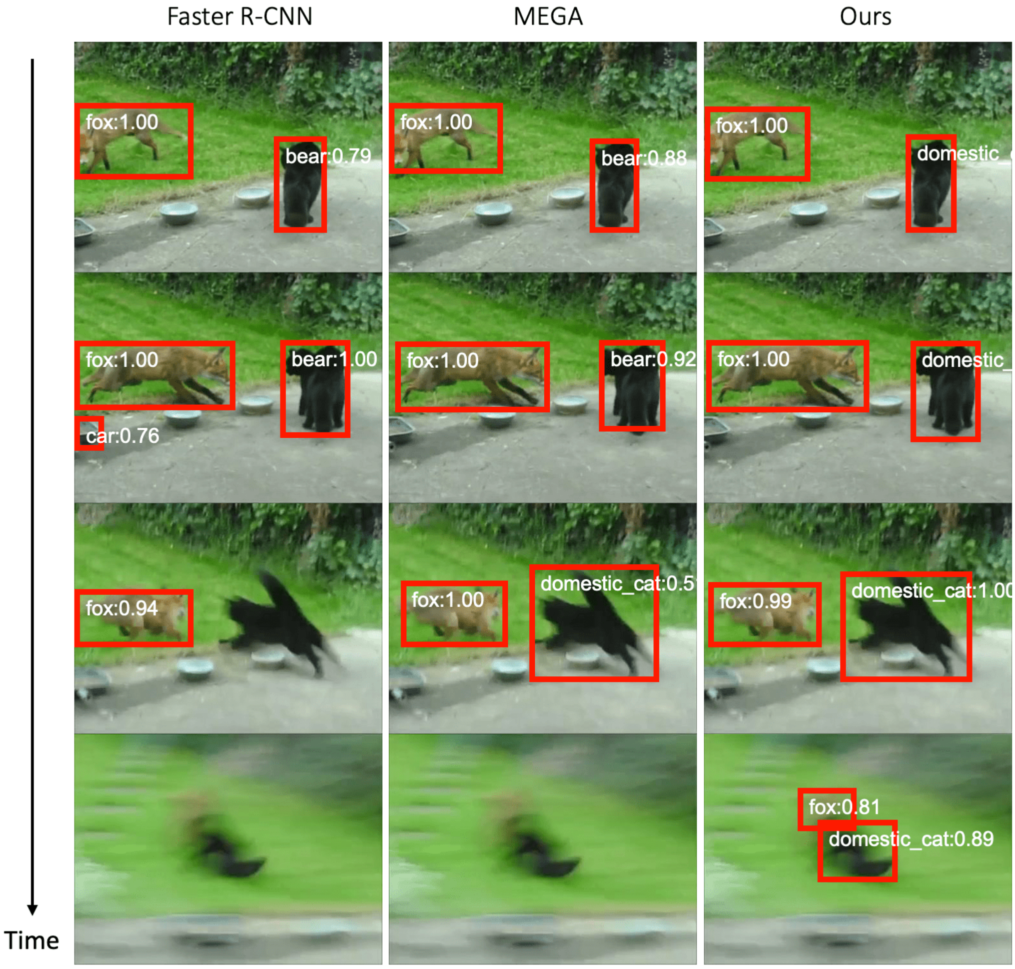


Figure 4.6: Visualized results on ImageNet VID `val`. From left to right: predictions of Faster R-CNN(baseline) [131], MEGA [29] and ours. Best viewed digitally and in color.

Investigation of VSTAM

In this section, we review VSTAM from the proposed component level. To confirm the effectiveness of VASTM, we use Faster R-CNN [131], a single frame detector, as a baseline model and show the effectiveness of the proposed method. Table 4.6 shows the difference of accuracy with the modules on three datasets. First, we can see that the introduction of VST provides a significant gain from the baseline for all different datasets. Therefore, we can confirm that elemental aggregation is effective as feature aggregation for video. Next, we see that introducing external memory for updating by attention improves accuracy. Furthermore, we can confirm that the introduction of pretraining is practical for all datasets. Thus, we can see that all the factors are essential for video object detection.

To check the effectiveness of sparse sampling in VSTAM, VST was replaced by a

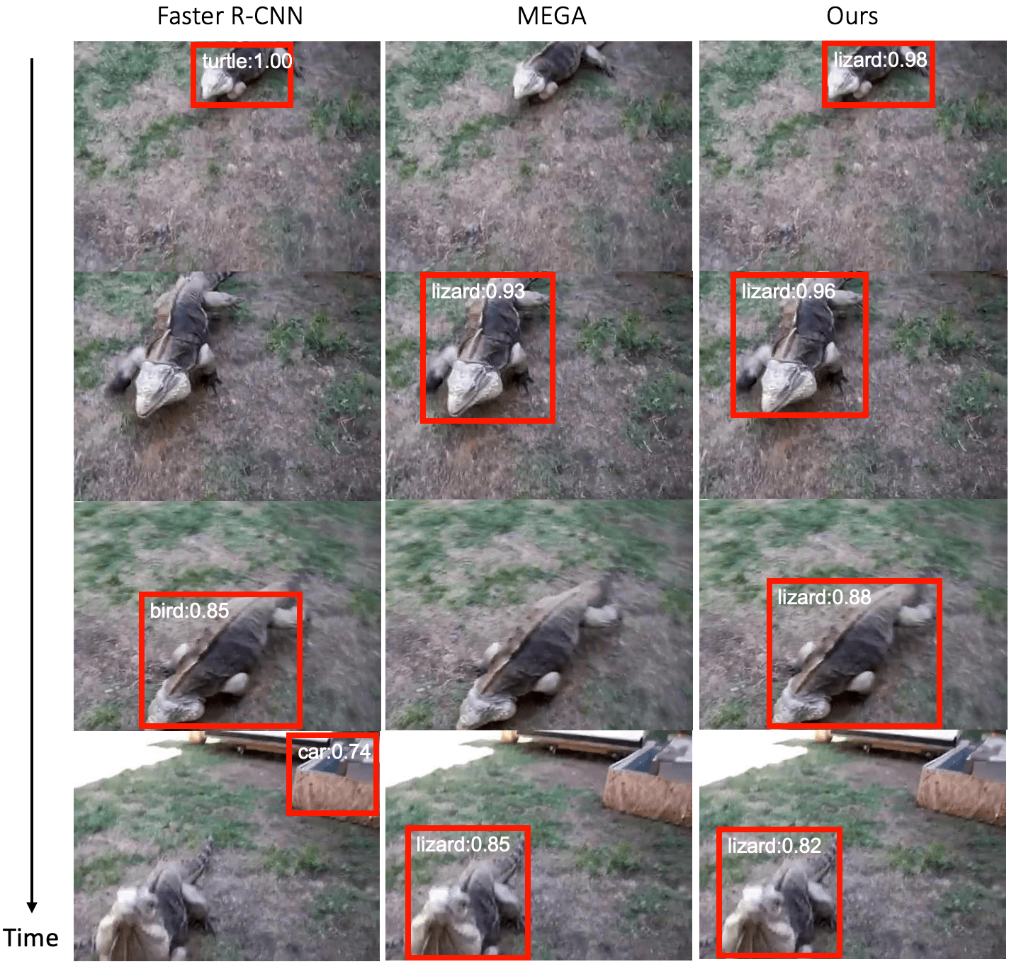
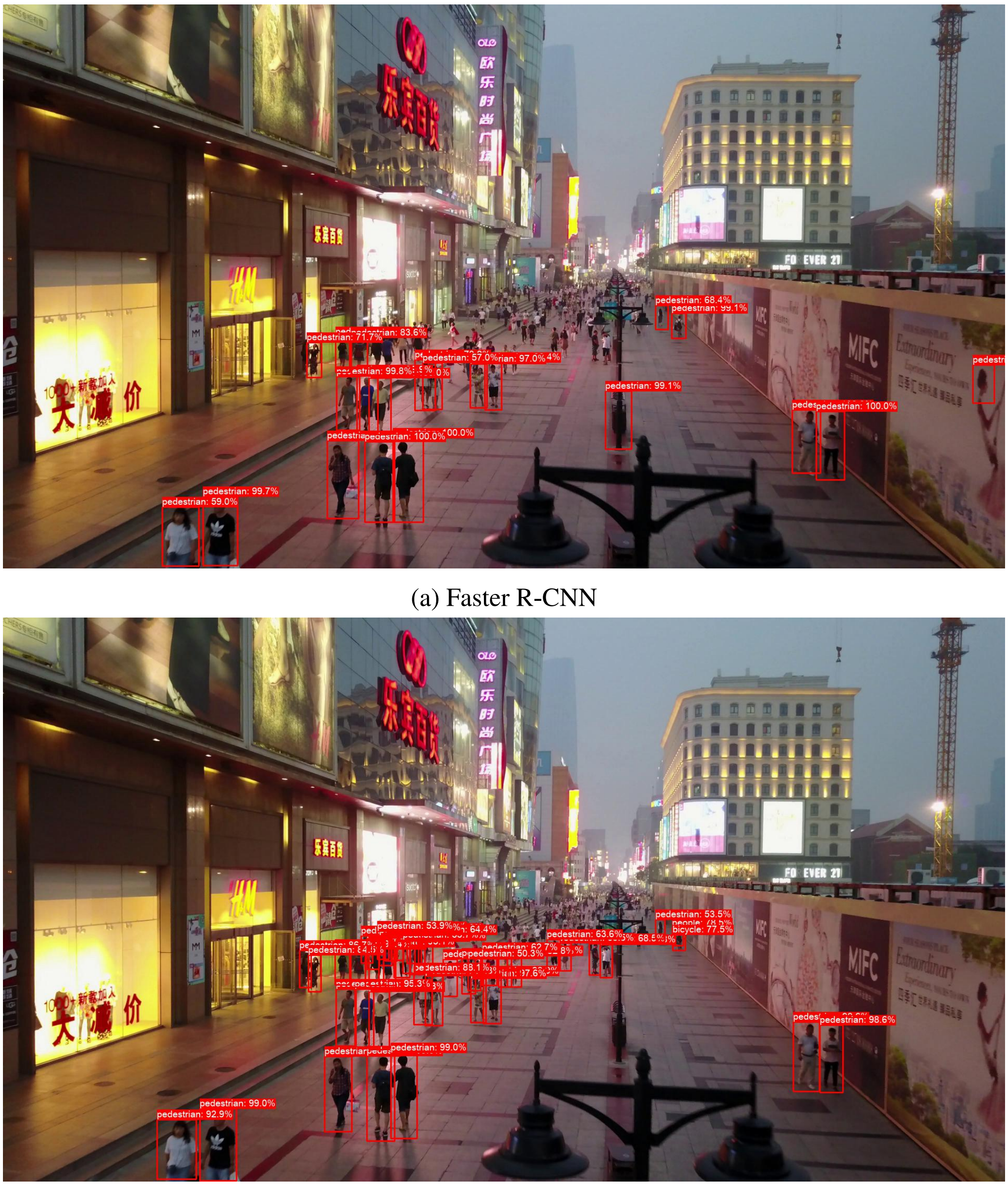


Figure 4.7: Visualized results on ImageNet VID `val`. From left to right: predictions of Faster R-CNN(baseline) [131], MEGA [29] and ours. Best viewed digitally and in color.

vanilla transformer. Our proposed method processes one frame in 52ms. If we replace VST with a vanilla Transformer, the run-time becomes 342ms. VSTAM (w/ VST) and VSTAM (w/ vanilla one) consume 2.1GiB and 7.2GiB memories per frame during inference on FP32. Additionally, 5.1GiB memories are required for each for Faster-RCNN. Our VST offers 658% speed up 70.1% memory reduction. We see significant gains thanks to our sparse sampling.

Figure 4.6 and 4.7 shows the detection results of the baseline [131], MEGA [29] and the proposed method for ImageNet VID. We confirm that the proposed method detects the targets accurately, even in severe scenes, compared to the existing methods. Figure 4.8 and 4.9 shows the detection results of Faster R-CNN(baseline) and the proposed method for VisDrone-VID. It can be seen that the proposed method improves the detection in the deteriorated scenes.



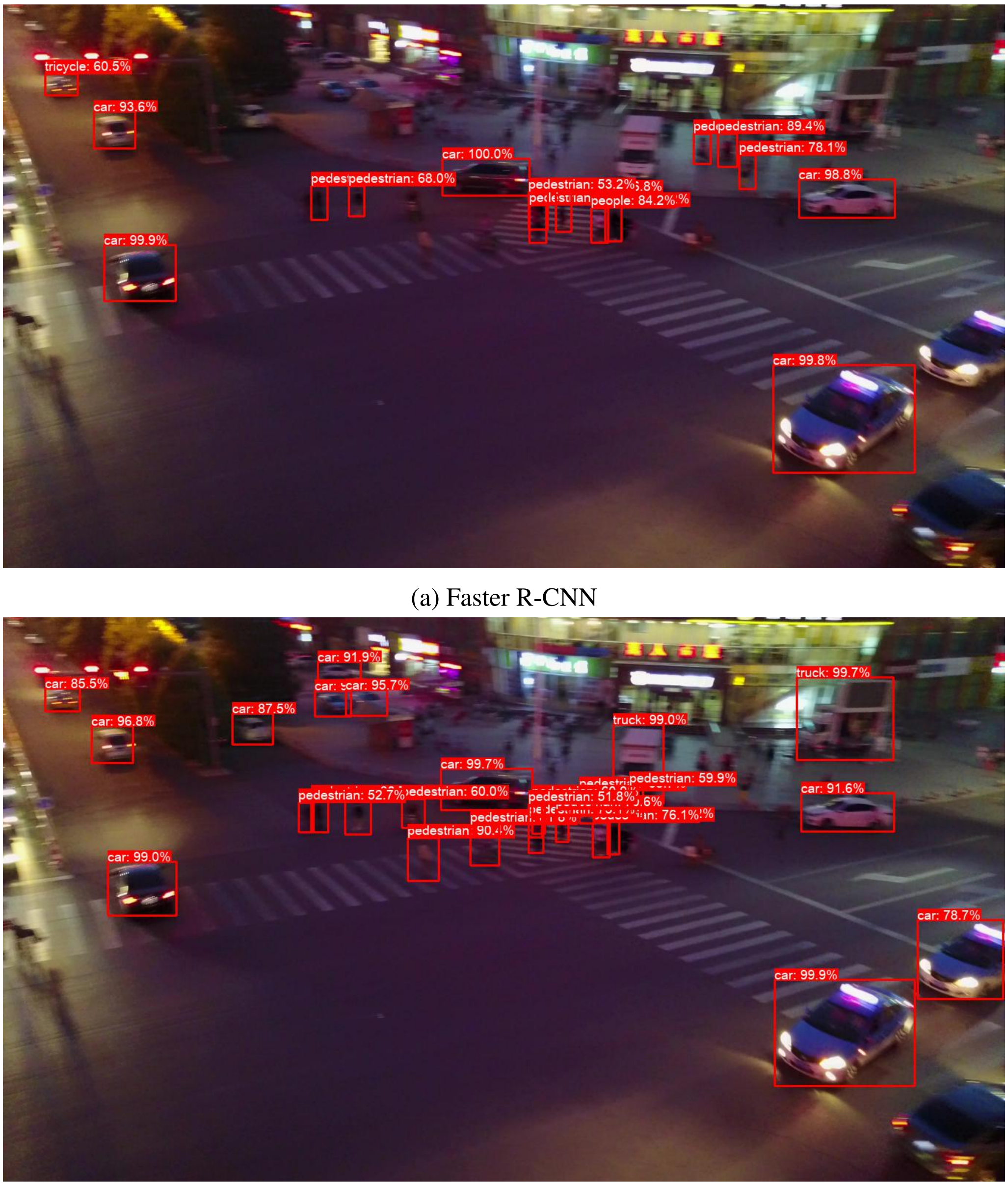
(a) Faster R-CNN

(b) Ours

Figure 4.8: Visualized results for a small objects scene with slight motion on VisDrone-VID `test`. From top to bottom: predictions of Baseline and ours. Best viewed digitally and in color.

Table 4.7: Performance comparison of feature aggregation modules by different frame sampling on ImageNet VID `val`, VisDrone-VID `val` and UA-DETRAC `test`.

Methods	Window-range		ImageNet VID				VisDrone-VID	UA-DETRAC
	Nearby-only	Nearby-Distant	mAP	mAP_s	mAP_m	mAP_f	AP	AP
VST (Ours)		✓	**77.1**	**85.1**	**75.5**	**55.8**	**18.7**	**81.8**
	✓		75.9	84.8	73.9	53.1	17.2	79.2
SSTVOS [52]		✓	74.4	83.3	72.9	50.2	13.8	75.0
	✓		74.7	83.6	73.3	51.3	15.1	75.6
Baseline			71.7	80.7	69.0	47.0	12.1	73.1



(a) Faster R-CNN

(b) Ours

Figure 4.9: Visualized results for a scene with large motion on VisDrone-VID `test`. From top to bottom: predictions of Baseline and ours. Best viewed digitally and in color.

Investigation of Element-wise Aggregation

We examine the effect of element aggregation across different frame sampling. VSTAM aggregates information from a wide range of frame spans to overcome the challenge of videos lasting several frames. We investigate how this frame selection affects the element-level aggregation. In this experiment, we do not include external memory or pretraining.

We compare our results with the recent video object segmentation method, which

Table 4.8: Performance comparison of sparse attention modules on ImageNet VID `val`, VisDrone-VID `val` and UA-DETRAC `test`.

Sparse attention			ImageNet VID				VisDrone-VID	UA-DETRAC
Frame	Position	Random	mAP	mAP$_s$	mAP$_m$	mAP$_f$	AP	AP
✓			73.1	82.4	70.5	48.4	13.6	75.8
	✓		74.0	83.7	71.3	49.3	15.2	79.2
		✓	74.3	83.9	72.1	50.1	13.1	74.8
✓	✓		74.8	84.2	72.9	50.4	16.2	79.4
✓		✓	75.3	84.1	73.4	52.1	14.3	78.8
✓	✓	✓	**77.1**	**85.1**	**75.5**	**55.8**	**18.7**	**81.8**
vanilla transformer			75.9	84.6	73.9	53.2	17.9	80.7
Baseline			71.7	80.7	69.0	47.0	12.1	73.1

proposes a sparse attention-based aggregation method [52] similar to VST. SSTVOS performs element aggregation by focusing on local spatial area and temporal neighborhoods. Since there is no official implementation of SSTVOS, we reproduced it and obtained a result 0.1 pt higher than the paper value, which is used for comparison.

Table 4.7 shows the difference in accuracy between the two types of frame sampling, where "Nearby-only" means dense sampling as shown in Figure 4.3a, and "Nearby-Distant" represents sparse sampling, as shown in Figure 4.3b. Both VST and SSTVOS can improve accuracy from baseline. However, SSTVOS loses accuracy when far frames are included. VST, on the other hand, can improve accuracy by utilizing distant frames rather than aggregating over a short period. In particular, it can be observed that mAP$_f$, which has a large object motion, has a significant gain. This is due to VST's robust sparse attention to object positions, allowing VST to utilize a global view effectively.

Effect of Video Sparse Attention

Next, we investigate the effect of attention on accuracy. Table 4.8 shows the accuracy impact with each attention method. The full attention of the vanilla transformer maintains high accuracy of 75.9% because it considers all elements of the time series frame. Only frame attention, which can only access its frame and cannot aggregate from neighboring frames, results in a significant decrease. Only random attention aggregates feature from each frame to utilize long-range information, but the accuracy is insufficient. The position attention, similar to dilated convolution over multiple frames, aggregates global information, but like random attention, it is not accurate. Combining the frame attention, which accesses locally and globally within a frame, and the random and position, which aggregate features spatiotemporally, improves their accuracy. Moreover,

Table 4.9: Impact of the ratio of random attention on ImageNet VID `val`.

Random r (%)	0	5	10	15	30	50	70	100
mAP (%)	74.8	76.4	77.1	77.1	76.9	76.5	76.3	75.9

Table 4.10: Impact of the number of the layers on ImageNet VID `val`

Layers (L)	1	2	3	4
mAP (%)	80.1	80.7	81.1	81.3

the video-aware sparse transformer, which combines all of them (Frame + Position + Random), is 1.2 points more accurate than the original full attention. Indeed, we confirmed that for video sequences, properly performing the sparse sampling, rather than the dense sampling, achieves higher accuracy. There is much redundant information in video sequences, both in space and time, and it is easier to process them properly if the information is reduced to some extent. Therefore, in video object detection, we confirmed that including computer vision characteristics in the attention mechanism leads to improved accuracy because it omits some of the redundant information in the video rather than accessing all of the information.

Effect of Random Attention

We investigate the impact of the ratio using random attention. Table 4.9 shows the performances under different ratios ($r\%$) using random attention. $r = 0\%$ is identical with using frame attention and position attention only in Table 4.8 while $r = 100\%$ is identical with using the vanilla transformer. We see that the accuracy is improved by increasing r from 5% to 10%, but it gradually decreases from 15% to 100%. This indicates that introducing random attention is practical for feature aggregation, but using random attention too much is not a good way.

Effect of number of layers

The encoder and decoder are essential parts of VSTAM, which are built upon stacked layers. We investigate the influence of the layer number of VSTAM on the performance. Table 4.10 shows that VSTAM performs better with more layers stacked.

Table 4.11: Impact of External Memory on ImageNet VID `val`, VisDrone-VID `val` and UA-DETRAC `test`.

Methods	Update candidate	Additional frames	ImageNet VID mAP	VisDrone-VID AP	UA-DETRAC AP
External Memory	–	0	77.1	18.7	81.8
	Distant	1	78.8	22.0	83.6
	Distant	2	80.0	**22.6**	**84.2**
	Distant	3	**80.1**	22.4	84.0
	Nearby-Distant	2	78.4	20.1	82.8
Extended sliding window	–	1	77.5	18.9	82.2
	–	2	77.7	18.8	82.3

Table 4.12: Performance comparison of pretraining on ImageNet VID `val`, VisDrone-VID `val` and UA-DETRAC `test`.

Method	ImageNetVID				Visdrone-VID	UA-DETRAC
	mAP	mAP$_s$	mAP$_m$	mAP$_f$	AP	AP
Ours w/o pretraining	80.0	86.9	77.7	59.1	22.6	84.2
Ours	81.3	88.2	79.1	60.8	23.6	85.1
Pace Prediction	80.1	87.0	77.5	59.1	22.5	84.4
Ours w/o Motion prediction	80.7	87.7	78.4	59.8	22.9	84.9
Ours w/o Reconstruction	80.5	87.4	78.2	59.5	23.2	84.7

Effect of External Memory

In this section, we examine the effect of external memory. Table 4.11 summarizes the accuracy effects of changing the number of additional frames with and without external memory. To examine the effect of selectively chosen feature maps, we show that the sliding window width increased without using external memory. It can be seen that increasing the number of additional frames also improves the accuracy. However, the gain of the external memory is more prominent, and the number of additional frames of the external memory is saturated after about two frames. Therefore, it is no longer possible to obtain a significant gain. We can also confirm that when nearby frames are included as update candidates, the accuracy decreases compared to only distant frames. It is better to utilize only distant frames as candidates to overcome video issues over a long time.

Investigation of Pretraining

In this section, we discuss the effects of pretraining. Table 4.12 summarizes the effect of pretraining on the accuracy.Ours w/o pretraining means initial values from ImageNet. We also show the results of replacing our model with Pace Prediction [155], which estimates the velocity of frames, proposed in video recognition. First, the effect of pre-

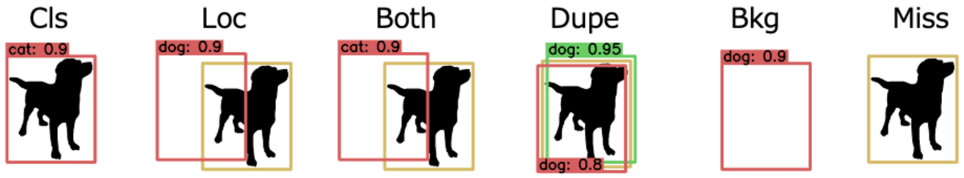


Figure 4.10: Visualized examples of error classification from TIDE [12]. "Cls" represents that the model detected the object but misclassified it into another class. "Loc" means that the model detected the object with lousy localization. "Both" means occurring of both "Cls" and "Loc". "Dupe" represents duplicated detection for an object. "Bkg" means background false-positive detections, while "Miss" means that it does not detect the object even though an object exists there. Best viewed digitally and in color.

training is present on both datasets. On the other hand, we can see that Pace Prediction is less effective, confirming the need for pretraining that suites video detection tasks. In addition, it is essential to train a model with both time and space since the accuracy of each component decreases when either part is removed.

Effect of Feature Map Refinement to RPN

Recent methods [29, 44, 75, 167] in VOD employ object-level features from RPNs [131] and propose refinements, which are robust to position changes of objects over the long term. These methods rely on the detection of RPNs, heavily degrading performance where detection is difficult. In contrast, our method refines the feature map before RPN. We thus evaluate how it affects the RPN in terms of Average Recall (AR). We select top $k = 5, 10, 100$ proposals generated by RPN and calculate $\mathrm{AR_k}$. Table 4.13 shows the difference of Recall in RPN with and without our proposed method. We can see that all the metrics are improved by the proposed method, confirming the effectiveness of the feature improvement before RPN.

Error Analysis

Some scenes are difficult to detect in video object detection due to the appearance changes with time, resulting in detection errors. To see what kind of errors the pro-

Table 4.13: Impact of VSTAM to RPN on ImageNet VID `val`.

VSTAM	$\mathrm{AR_5}$	$\mathrm{AR_{10}}$	$\mathrm{AR_{100}}$
w/o	75.1	81.0	90.2
w/	79.2(+4.1)	86.1 (+5.1)	96.4 (+6.2)

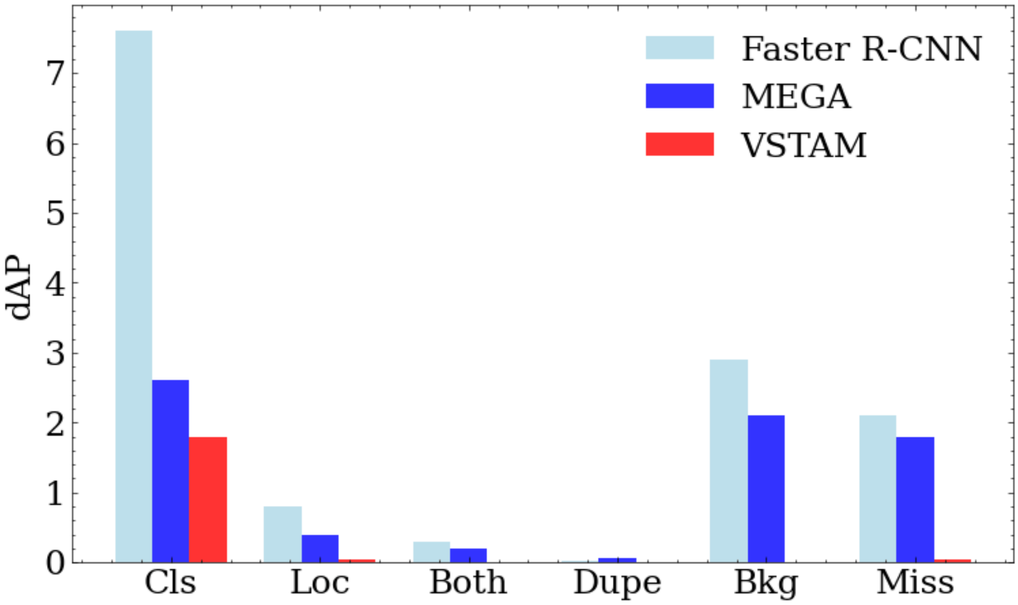


Figure 4.11: Visualized results of error analysis on ImageNet VID `val` by TIDE [12]. From left to right: results of the Faster R-CNN (baseline) [131], MEGA [29] and ours. See Figure 4.10 for the categories of errors. Best viewed digitally and in color.

posed method solves explicitly, we conducted an error analysis using TIDE [12]. It classifies object detection errors into misclassification, incorrect localization, duplicate detection, false-positive detection, and miss, as shown in Figure 4.10. Figure 4.11 shows the error results of the Faster R-CNN(baseline) [131], MEGA [29] and VSTAM on ImageNet VID `val`, where the horizontal axis shows the error categories and the vertical axis shows the amount of error accumulation proposed in TIDE [12].

MEGA improves feature maps only in the object candidate regions after detection and reduces “CLS” errors from the baseline, while many “BKG” and “MISS” errors remain. On the other hand, the proposed method significantly reduces them by performing feature refinement before the object candidate region estimation, thus alleviating the false-negative and false-positive problems, especially background false-positive in video object detection. In addition, the class error is also reduced thanks to our element-wise aggregation.

4.4.6 Video Instance Segmentation Results

To validate the versatility of the proposed method, we also evaluate it on YouTube-VIS dataset [176], which is a more complex task (cf. Section 2.4). The evaluation metric is similar to the standard evaluation one in image instance segmentation, average precision (AP), and average recall (AR) [107]. Specifically, it uses AP, AP_{50}, AP_{75}, AR_1 and

AR_{10} metrics for evaluation. However, the IoU computation is modified from image instance segmentation [107] to compute the spatial-temporal consistency of predicted and ground truth segmentations. It obtains a low IoU if the algorithm detects the object masks successfully but fails to track the objects across frames. Therefore, AR and AP are scored in such a way that tracking is taken into account.

Most of the current VIS methods focus on generating high-quality masks and linking the same objects across frames with features extracted by the backbone like ResNet, while only a few of them pay attention to improving the features.

To apply the proposed method to VIS, we replaced Faster R-CNN with MaskTrack R-CNN [176]. MaskTrack R-CNN is an extension of Mask R-CNN [80] with a tracking branch to link the same object instances between two frames.

The results with ResNet-50 as the backbone are shown in Table 4.14. Our proposed method achieves competitive results on all evaluation metrics. With our proposed method, MaskTrack R-CNN is improved by more than 8.7% on the AP metric. We remark that the AP gap between ours and MaskProp mainly comes from its combination of multiple methods [9, 22] and the high-resolution mask refinement post-processing.

Recently, TF-Blender [36] and TROI [72] have been proposed to improve a feature map. TF-Blender utilizes nearby frames, but it aggregates features at the frame-wise without considering object misalignment, and the gain is limited since instance segmentation requires more precise feature refinement. TROI proposes a temporal ROI alignment to extract ROI features from other frames based on their similarity; however, it is not sufficient for hard-to-detect scenes because the refinement is for the object-level feature map. On the contrary, our approach is based on element-wise aggregation, allowing us to improve the representation more precisely.

Figure 4.12 shows VIS results between baseline (MaskTrack R-CNN [176]) and the proposed method on example frames from the validation set. We see that by refining element-wise feature maps with the temporal information, false-negative detections are reduced, and masks are stabilized.

4.5 Conclusion

We introduced a novel framework VSTAM for video object detection. VSTAM element-wisely refines feature maps by considering spatiotemporal information across long- and

Table 4.14: Performance comparison with the state-of-the-art models on YouTube-VIS2019 `val`. All the methods use ResNet-50 as the backbone.

Methods	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
OSMN [177]	27.5	45.1	29.1	28.6	33.1
DeepSORT [165]	26.1	42.9	26.1	27.8	31.3
FEELVOS [153]	26.9	42.0	29.7	29.9	33.4
MaskProp [8]	40.0	–	42.9	–	–
SipMask [16]	32.5	53.0	33.3	33.5	38.9
STEm-Seg [4]	30.6	50.7	33.5	31.6	37.1
CompFeat [58]	35.3	56.0	38.6	33.1	40.3
VisTR [161]	36.2	59.8	36.9	37.2	42.4
SG-Net [109]	34.8	56.1	36.8	35.8	40.8
CrossVIS [178]	34.8	54.6	37.9	34.0	39.0
VisSTG [159]	36.5	58.6	39.0	35.5	40.8
MaskTrack R-CNN [176]	30.3	51.1	32.6	31.0	35.5
TF-Blender [36] + MaskTrack R-CNN [176]	31.4	52.3	33.5	31.9	36.5
TROI [72] + MaskTrack R-CNN [176]	33.5	57.0	36.6	–	–
VSTAM(Ours) + MaskTrack R-CNN [176]	39.0	61.2	42.9	38.9	47.6

short-term ranges with external memory. We proposed a video-aware sparse transformer (VST) to model a long-range spatially and temporally relation of frames in a video sequence to aggregate features efficiently. In addition, VSTAM significantly improved the accuracy by adaptively storing the feature map of the frame utilized during element aggregation in external memory to deal with the dynamics of video. Extensive evaluations demonstrate that VSTAM performs favorably on several publicly available datasets against SOTAs. Moreover, detailed analysis showed that the proposed method reduced the challenges of false-negative and background false-positive detection due to the apparent changes of video object detection.

For further work, improving the efficiency of feature map management in external memory is considered. Currently, feature maps are stored in external memory on frame-wise management, but we believe that memory-efficient feature map management is necessary to retain only the necessary portions or patch-wise features.

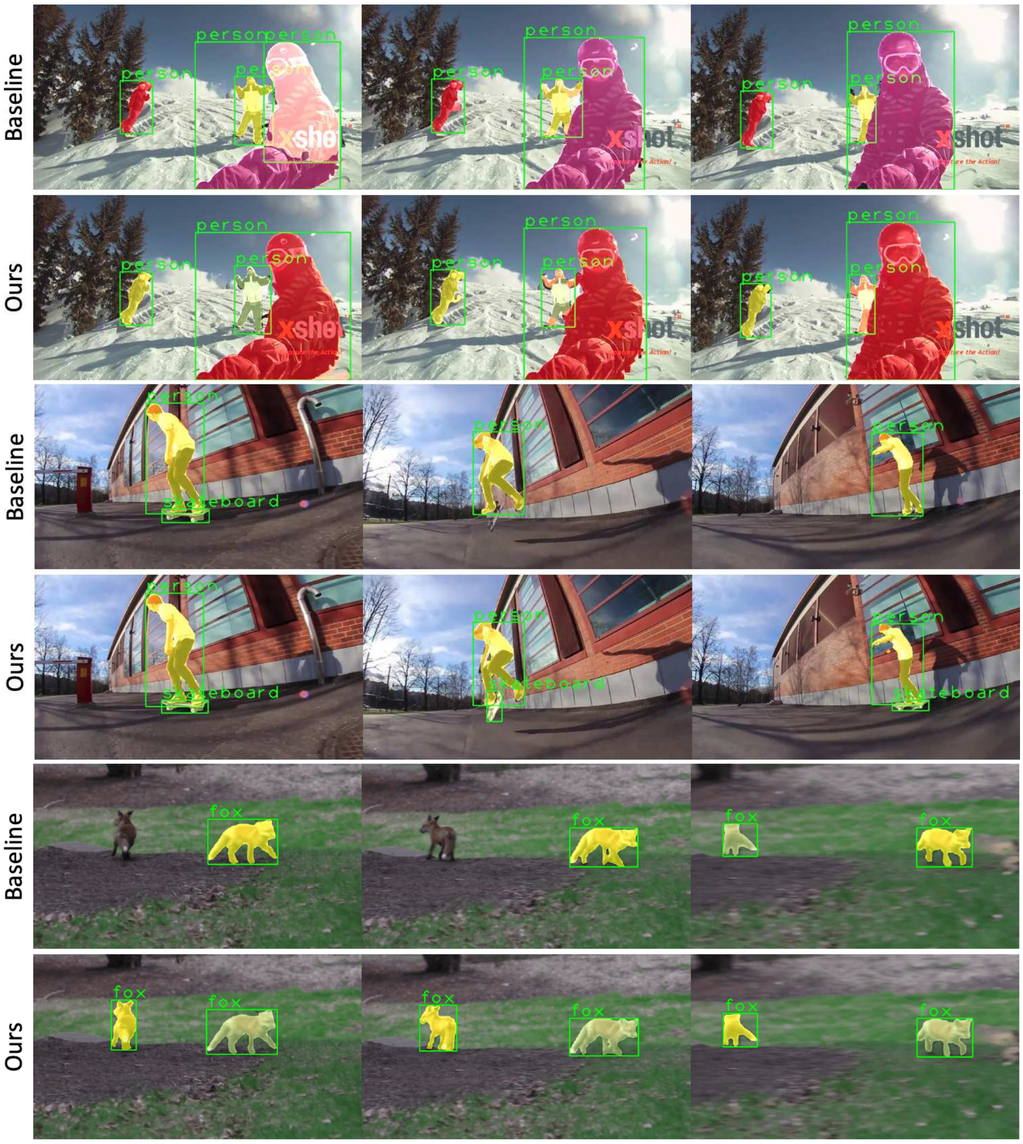


Figure 4.12: Example of visualized results between Baseline (MaskTrack R-CNN [176]) and Ours on YouTube-VIS `val`. Results are plotted if their confidence scores are larger than 0.45. Best viewed digitally and in color.

Chapter 5

Prediction based Feature Enhancement

5.1 Introduction

In order to detect objects robustly in live streaming videos, recurrent neural networks [26, 110, 111] and optical flow methods [190, 192] were proposed to propagate past information. They achieved more stable detection than still-image detectors because of temporal information. However, they are still low accuracy because they utilize only the information of the specific frame, mostly the last or a nearby keyframe. Therefore, the external memory approach [42, 61, 86] has been proposed for online video object detection to extend the temporal information range in recent years. They showed the practical improvement of accuracy by using external memory. However, storing features in external memory consumes a lot of memory banks. For instance, TFEN [61] and VOD-MT [86] use 1.9GiB and 4.3GiB, respectively, including models and external memory. OGEMN [42] and VSTAM (in Chapter 4) consume more than 7GiB memory.

In contrast, there are memory limitations in the circumstances, such as robotics. For example, the Jetson TX1, a GPU device for automated driving and robotics, has 4GiB of memory shared with a CPU, and the memory available for models is around 2.0-2.6GiB[*]. Thus, the external memory approach may not be applicable in environments under strict memory limitations, and a method with low memory usage is needed.

Therefore, we propose a prediction-based feature map enhancement approach to im-

[*] Depending on the operating system used, the memory used is usually between 1.4 and 2.0 GiB.

prove detection accuracy under such a limitation effectively. Recently, in the field of video representation, the task of predicting the future has received much attention [19, 99, 115]. It is necessary to acquire knowledge about the object, its structure, motion, etc., to depict the future accurately. The model can then obtain better feature representations about objects and environments through future prediction.

Based on the idea, we introduce the concept of future prediction, which uses the future frames to improve the object detection accuracy in live stream video. Although the existing approach [26, 110, 111] simply propagates information from the past to the present, our proposed methods actively learn object knowledge through prediction during training to enhance feature maps and improve detection accuracy. This approach does not employ any external memory other than the recurrent neural network states, and we achieve improved accuracy with less than 1.8GiB memory usage, which is under the substantial limitation of 2 GiB.

In order to enhance a feature map through future prediction, we conducted on different temporal perspectives: the next and the next several frames forecasts. We propose a model that simultaneously detects objects and predicts the feature map of the next frame based on the existing propagation model for the next frame prediction. In the future prediction, we examine how it affects the existing detectors by actively predicting the next frame. Moreover, we obtain the pseudo feature map of the next frame by prediction. Suppose the generated feature map is similar to the real one. In that case, we can utilize it for the subsequent frame detection, which results in faster processing speed by the abbreviation of feature extraction. We show the effectiveness of future feature map prediction.

We propose applying the next several frames prediction, ten successive future frames, to video object detection. A deterministic model, such as we used in next-frame feature map prediction, outputs an average of the future candidates. Therefore, it is difficult to predict the future in the successive frames because of the high uncertainty at each time step. Thus, we learn feature representations based on probabilistic future prediction, where we sample and predict the one probable future. We also propose a method to learn only the future prediction first and then fine-tune it to the detection task to predict the problematic future more reliably. We validate in detail how probabilistic future prediction affects object detection.

5.2 Related Work

In this section, we review related works on next frame prediction methods. For the related works on video object detection in general, please refer to Section 2.2. 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. Two main approaches have been proposed to successfully predict future frames from given the past frames: deterministic and stochastic prediction.

The video prediction by deterministic models generates the next frame by using deterministic loss, such as mean-squared-error, along with LSTM [142], ConvLSTM [121], 3D-Convolution [1, 182], and more complex recurrent models [13, 115]. Deterministic models tend to produce blurred images because the output image is the average possible image. For this reason, separating a foreground object from the background has been proposed for more accurate generation [7, 45, 149, 152, 169].

Models with stochastic hidden variables such as VAEs, have been proposed [5, 19, 99] to reduce the uncertainty that increases over time in deterministic models. These models define a prior distribution for a set of latent variables and allow different samples from these latent variables to capture multiple outcomes. It has also been observed that the mean-squared-error loss is based on Gaussian distribution and produces blurred output, so using an adversarial loss with GAN is proposed [99].

While these studies aim to generate detailed future images themselves, our proposed method focuses on generating the future that is effective for detection. Our proposed method utilizes deterministic methods to predict the next-frame feature map and stochastic approaches to forecast actual future frames into video object detection. Recently, CrevNet [182] suggests the video representation learned through video prediction can be directly used for object detection, we discuss in Section 5.6.5.

5.3 Proposed Method through Next Future Prediction

This section proposes a method to improve the accuracy of real-time video object detection using next frame feature prediction. Figure 5.1 intuitively shows the characteristic of our approach against the existing one. Unlike other models that exploit both past and current information (Fig 5.1(a)), our model (Fig 5.1(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.

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 exploit the forecast feature map. In this way, our model improves the processing speed by using the forecast feature map without significant performance loss.

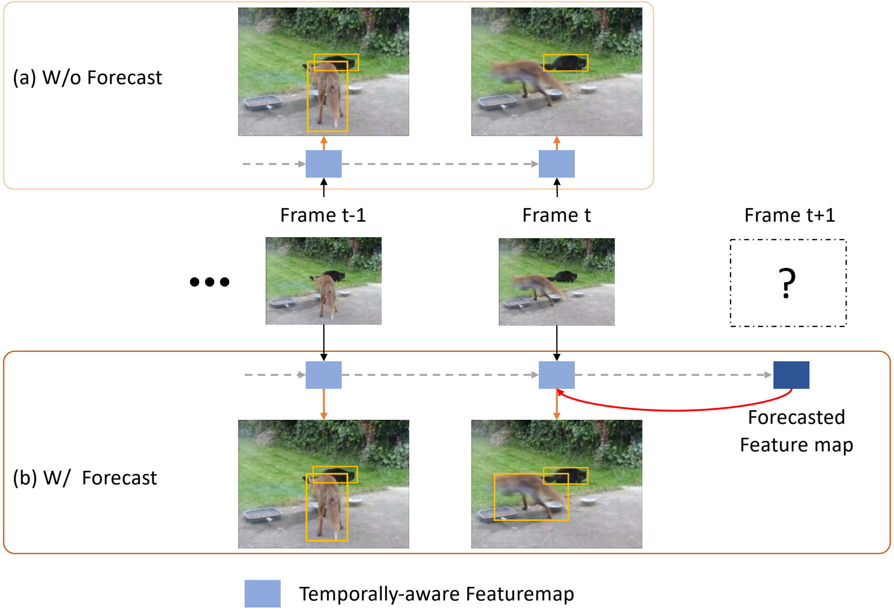


Figure 5.1: Video object detection through the next frame prediction. Current live streaming video object detector approaches (a) store historical information of feature maps to acquire stable detection results. In our proposed model (b), we jointly learn the future feature map prediction to support the detection task at the current frame.

5.3.1 Overview

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 $D_t = N_{\text{det}}(N_{\text{feat}}(I_t))$, where N_{feat} and

N_{det} represent a feature extractor and an object detector model such as SSD [112]. We use the recurrent-network based SSDLite architecture [110] as the baseline model and insert our proposed modules, i.e., encoder module and scheduler module, into between the extractor and the detector.

Figure 5.2 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 [110]. 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).

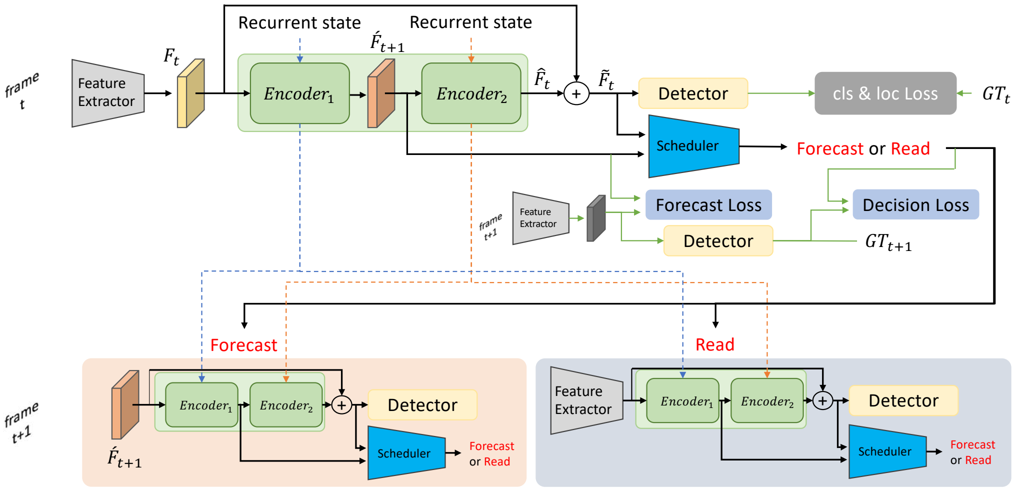


Figure 5.2: The architecture of our proposed model. It consists of the feature extractor, the encoder, the scheduler, and the object detector. The encoder predicts the future feature map at the next frame and the current temporally-aware feature map. The scheduler decides whether to exploit the forecast feature map or extract the actual feature map at the next frame. The black arrows show the information flow used during training and inference, and the green arrows show the flow for training only.

5.3.2 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 [166] followed by the bottleneck LSTM [110]. This allows us to recurrently retain past states and adapt to the limited capacity of the bottleneck LSTM. We stack the two encoders to generate a 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.

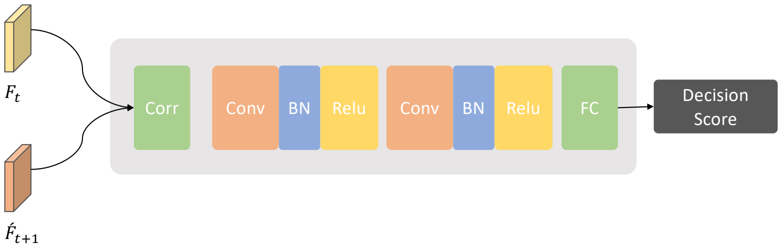


Figure 5.3: Scheduler network. The output feature map of the correlation layer is followed by two convolutional layers and a fc layer with a 2-way softmax.

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 5.3.4). $Encoder_2$, 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 a 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.

5.3.3 Scheduler Module

If the forecast feature map is reliably well-generated from the current feature map, we can use it as the alternative to 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 [117] and utilizing the correlation of feature maps, we design the architecture of the scheduler module, as shown in Figure 5.3. 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 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}.

5.3.4 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:

$$L = \frac{1}{M}(\alpha L_{\text{loc}} + \beta L_{\text{cls}}) + \gamma L_{\text{for}} + \lambda L_{\text{dec}}, \tag{5.1}$$

where M is the number of matched bounding boxes. We exactly follow [112] 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

$$L_{\text{for}} = \frac{1}{n}\frac{1}{m}\frac{1}{l}\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} ||\acute{F}_{t+1}(i,j,k) - F_t(i,j,k)||^2, \tag{5.2}$$

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:

$$L_{\text{dec}} = -y_t \log(p_t) - (1 - y_t)\log(1 - p_t), \tag{5.3}$$

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 5.3.5 for details).

5.3.5 Training

The whole training pipeline is depicted in Figure 5.2. Fundamentally, our training procedure is the same as the usual video object detection [26, 110, 111]; 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.

5.3.6 Testing

The flow of inference follows the black arrows in Fig 5.2. 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.

5.4 Experiments through Next Future Prediction

5.4.1 Benchmark Datasets and Metrics

We evaluated the performance of our method on two public datasets: ImageNet VID [134] and UA-DETRAC [164] (see Section 3.3.1). We evaluated the performance using mean average precision (mAP) and average precision (AP) for ImageNet VID and UA-DETRAC, respectively, (cf. Section 3.3.1).

5.4.2 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 [110, 112] architecture to the proposed model. We employ MobileNet V2 [136] as a feature extractor because of its computational efficiency and use the feature map before its average pool layer as F_t. In total, our proposed method consumes 1.1 GiB of GPU memory for all components during inference.

Data Augmentation

In addition to the data augmentation proposed in [112], we employ a more extended one to alleviate the potential over-fitting problem since it contains data with little change in the positions of objects. 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 and UA-DETRAC have 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 an improvement of 0.8, resulting in 65.2 in mAP for ImageNet VID.

Following the idea of [146], we train the model without the thinning operation from the last two epochs. This operation contributed to an additional 0.5 point increase in

Table 5.1: Performance comparison with state-of-the-art end-to-end video object detection models on ImageNet VID `val`. α is the hyper parameter of MobileNet. The last column shows the runtime (FPS) on our GPU environments. All our results are obtained on RTX 2080 Ti GPUs.

Methods	Backbone	mAP	Published		Our Impl.
			Device	FPS	FPS
LSTM-SSD [110](α = 1.0)	MobileNetV1 [84]	54.4	Pixel 2	15	56
Flow-guided [190](α = 1.0)	MobileNetV1 [84]	61.2	Mate 8	13	–
Memory-guided [111](α = 1.0)	MobileNetV2 [136]	61.4	Pixel 3	27	61
LMP [193](α = 1.0)	MobileNetV2 [136]	64.2	GTX1060	29	30
Ours (α = 1.0)	MobileNetV2 [136]	**65.7**	RTX 2080 Ti	39	–

accuracy to achieve 65.7 in mAP for ImageNet VID.

Training Details

Our training procedure consists of two phases: (1) For Imagenet VID, we pretrain our baseline model following the protocols [110, 191] with additional ImageNet DET dataset [134]. For UA-DETRAC, we pretrain the baseline model as still-images. (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 feature extractor, 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.

5.4.3 Comparison with State-of-the-Art

Comparison on ImageNet VID

We compared our proposed model with state-of-the-art real-time and live streaming video object detection methods. They are LSTM-SSD [110], Flow-guided [190], Memory-guided [111] and LMP [193]. In this comparison, the threshold of the scheduler module was set to 1.0.

As shown in Table 5.1, the proposed model achieves the best performance. It achieves 65.7% mAP, 1.5% higher than the strongest competitor [193], which employs a different local and mid-range propagation strategy to extract past information. This is mainly

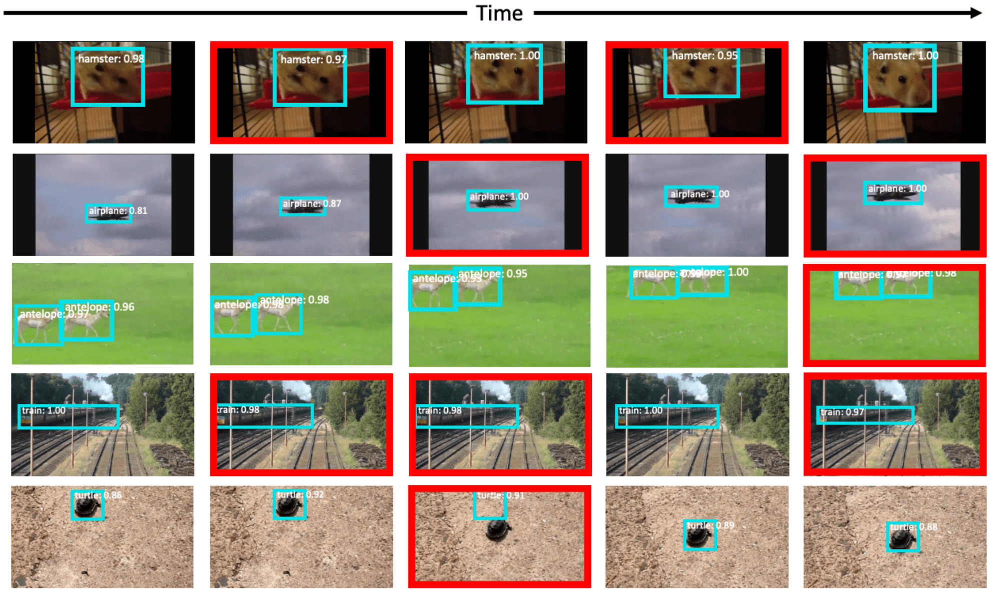


Figure 5.4: Visualization of example detection and the corresponding scheduler results on ImageNet VID `val` (best view in color). We set $p = 0.5$ in the scheduler module. Frames where the forecast feature map is used are specified in red; otherwise the real frame is adopted.

thanks to introducing the feature map forecast and our end-to-end joint training.

Table 5.1 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 about ten fps faster than LMP [193] in the same backbone. We note that [110, 111, 190] are developed for mobile devices, and thus published runtime comparison with them is just for a reference. Therefore, for comparison, we re-measured their implementation [110, 111, 193] on our GPU. Since they are simply propagating past information, they run faster but at the expense of accuracy.

Figure 5.4 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 deciding to read new images on complex scenes. We confirm that reasonable detection is realized using forecast feature maps. Detection using future prediction works well with decisions made by the scheduler, but a failure case by the scheduler is shown in the bottom row. This happens when there is a drastic change in the position of an object in a sequence of frames due to a drastic movement of the camera. When the camera moves gradually, as in the example in the third row, the scheduler does not leverage the prediction until it knows the movement,

Table 5.2: Performance comparison with state-of-the-art real-time detectors on UA-DETRAC `test`. (* is tested by ourselves.)

Method	Backbone	AP	FPS	Memory (GiB)	GPU
LSTM-SSD* [110]	MobileNet [84]	49.2	56	**0.9**	RTX 2080 Ti
Memory-guided* [111]	MobileNetV2 [136]	53.6	**61**	1.1	RTX 2080 Ti
LMP* [193]	MobileNetV2 [136]	55.1	30	1.8	RTX 2080 Ti
Ours	MobileNetV2 [136]	**65.4**	39	1.1	RTX 2080 Ti

but it tends to be weak in scenes where the camera moves rapidly.

Comparison on UA-DETRAC

We compared the real-time processing capability of the methods, which does not rely on external memory, on the UA-DETRAC dataset. Table 5.2 shows the accuracy of the methods. For the non-real-time methods, please refer to Section 3.3.4. LSTM-SSD [110], Memory-guided [111], and LMP [193] only exploit temporal information from the past to the present, and we can confirm that the proposed method, which utilizes future information through prediction, is significantly superior in terms of accuracy. In addition, we can confirm that the accuracy of the proposed method can be improved by using only 1.1 GiB memory, which is less memory than LMP.

5.4.4 Detailed Analysis

To confirm the effectiveness of the proposed method in detail, we conducted ablation studies on ImageNet VID and UA-DETRAC and detailed analysis on ImageNet VID.

Table 5.3: Effectiveness of components in the proposed model

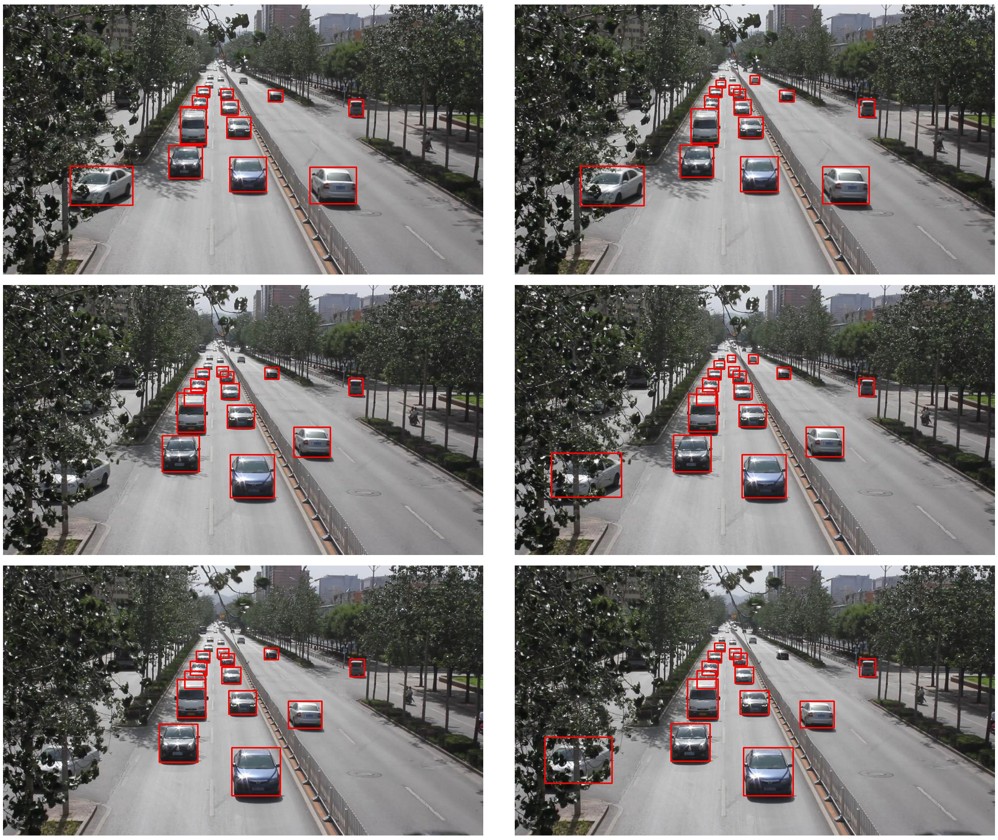
Methods	Components		ImageNet VID				UA-DETRAC
	Current information	Forecast	mAP	mAP_s	mAP_m	mAP_f	AP
(a) Complete model	✓	✓	**65.7**	**73.9**	**64.4**	**47.4**	**65.4**
(b) Model w/o forecast	✓		64.1	72.5	62.7	44.5	54.2
(c) Model w/o combination		✓	62.4	70.8	61.0	43.9	58.6
Baseline			60.6	68.7	58.8	41.6	52.2

Component Analysis

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 [26, 110]. 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 5.3. Note that we follow a detailed evaluation metric [191] 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. (cf. Section 4.4.5).

From Table 5.3, we see that simultaneously training the forecast and the detection improves the overall accuracy by 1.6 points on ImageNet VID. This gain mainly comes from the Fast category (2.9 point improvement). We thus reason that the larger the object's movement, the more critical 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


(a) W/o future prediction

(b) W/ future prediction

Figure 5.5: Visualized results for an occluded scene on UA-DETRAC `test`. Each column shows the detection result of successive frames by the model without forecast and the proposed method. We can confirm that when an occlusion issue occurs on the left side of the frames, the proposed method, which predicts the future motion of the object, robustly detects it. Best viewed digitally and in color.

w/o combination, utilizing F_t for the feature maps to be fed to the object detector is also essential.

In terms of the UA-DETRAC dataset, we can see that, unlike the ImageNet VID, the model w/o forecast has a significant loss of accuracy. It may suggest that the concept of prediction is essential when the future motion of an object such as a car is easily predictable. Figure 5.5 shows the difference in detection results on the UA-DETRAC dataset between the proposed method and the model without forecast, which only propagates past to present information. Under the occlusion scene, the proposed method with future prediction can detect robustly, which confirms the effectiveness of the forecast.

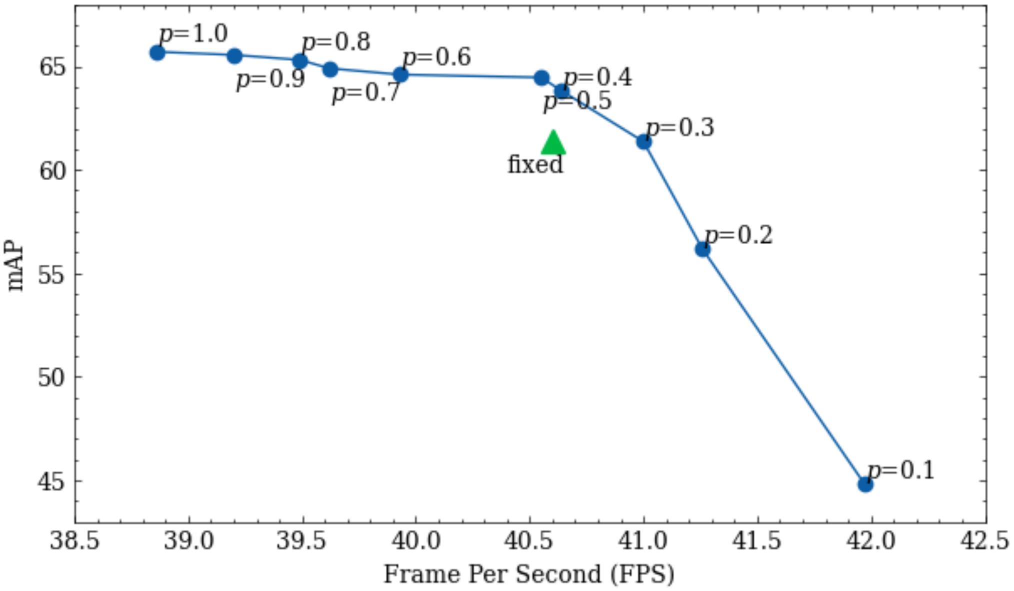


Figure 5.6: mAP v.s. FPS trade-off comparison under different threshold p in the scheduler module.

Effectiveness of Scheduler Module

Figure 5.6 illustrates mAP and fps under different threshold p in the scheduler module on ImageNet VID. Note that “fixed” 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 can flexibly control the accuracy and speed.

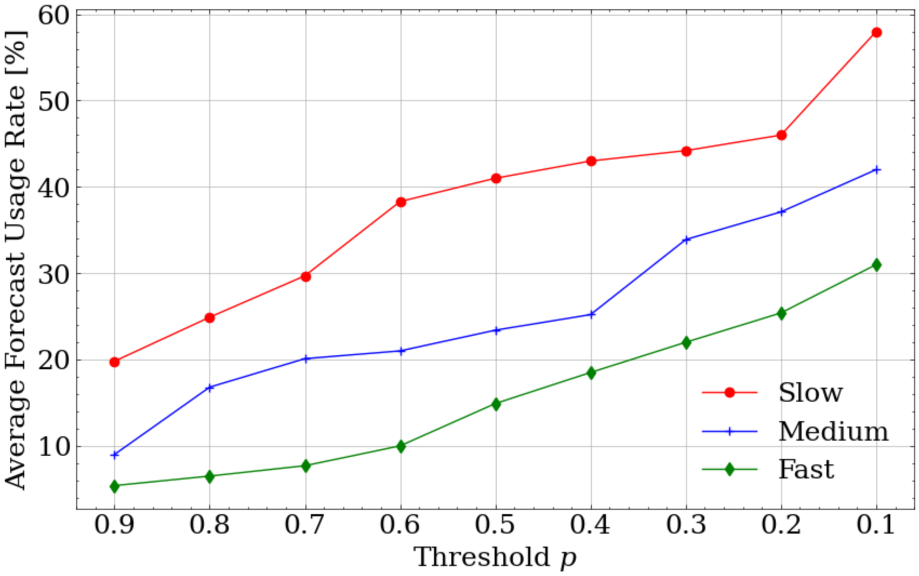


Figure 5.7: Forecast operation usage rate when varying the threshold p. The usage rate of the future prediction operation depends on the movement of the objects, Motion IoU.

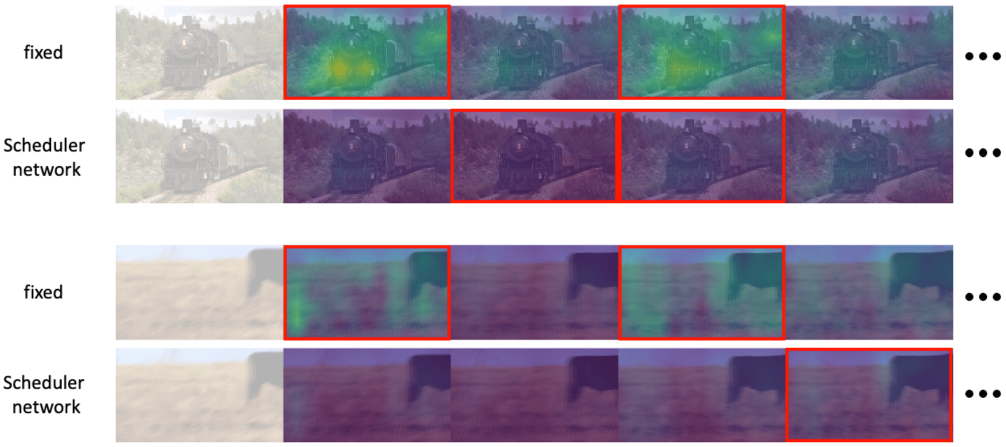


Figure 5.8: Visualization of the feature activation error between the predicted feature map and the actual feature map at the next time-step. Frames that use the future forecast feature map for detection are surrounded by the red rectangle ($p = 0.5$).

To examine more detail in which scenes the scheduler is utilized, the average usage rate of the prediction in a video clip when the threshold p is varied in terms of Motion IoU and shown in Figure 5.7. The average forecast usage rate is defined by the average ratio of the predicted operation in a 9-frame interval during inference over the entire set of videos. It can be seen that as the threshold is increased, the usage rate of the prediction decreases for all metrics, and conversely, as the threshold is decreased, the usage rate increases. Therefore, we confirm that the change of forecast utilization affects speed and accuracy, as shown in Figure 5.6. Moreover, the fact that the usage rate differs for each Motion IoU confirms that the scheduler changes the usage rate adaptively based on the object's motion.

Figure 5.8 shows the errors in terms of the heat map between actual feature maps and forecast feature maps when using the scheduler module or "fixed". The heat map turns more yellow when errors become large. We see that "fixed" 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, making the errors smaller. Figure 5.8 also shows that the errors are more likely to occur near target objects having uncertainty for their future locations.

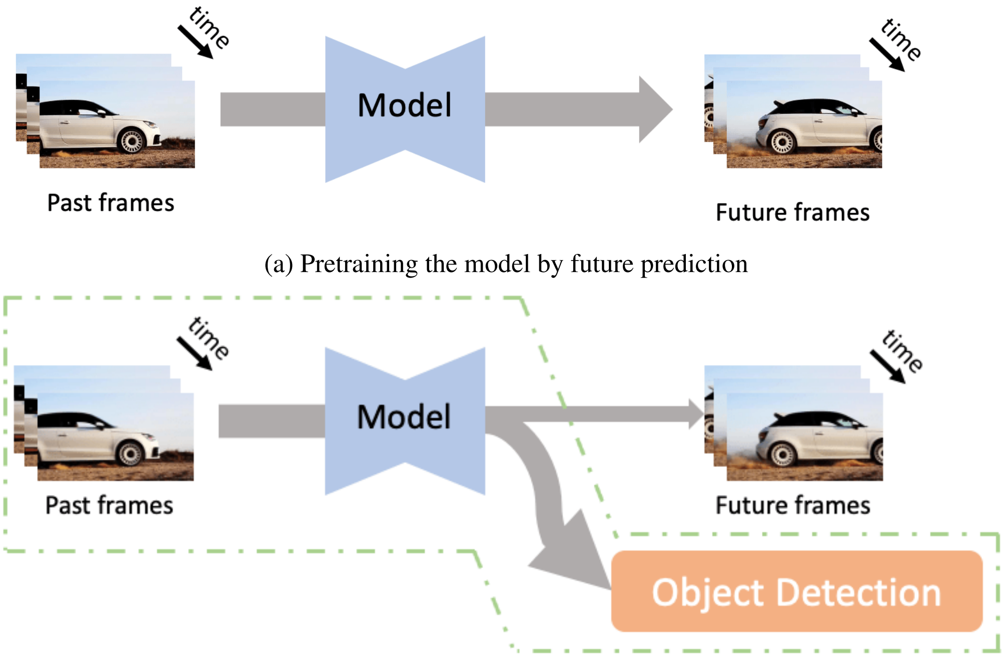
5.5 Proposed Method through Next Several Future Predictions

This section proposes a method to enhance the accuracy of real-time video object detection through the successive several future frame prediction. Future feature map prediction approach [62] is more effective than just stabilizing the feature map with past information [26, 110], since it actively learns how an object moves by predicting a feature map at the next frame. While the method improves accuracy, it has the following limitations: (1) Predicting the next frame feature map is too short-term, so the future information is not effectively utilized in the training phase. (2) The accuracy of predicting the next-frame feature map depends on the feature extractor, and the feature detector itself is acquired in the training phase, which takes time. Hence, it is insufficient to leverage future information to train online video object detection with future feature map prediction [62]. However, if we simply leverage the long-term temporal information, it will be difficult to predict with a definitive model as proposed in the previous work [62] due to the problem of future uncertainty. It is because the deterministic model learns to output the average of multiple future possibilities [99].

Video pretraining methods are recently studied for video recognition tasks where a video representation is learned through such as pace prediction [155] or pseudo-label estimation [69]. Predicting future frames in a video [19, 99] can be regarded as a video pretraining method. It can provide a video representation that effectively learns temporal and spatial information, and can be applied to online object detection. However, since existing studies focus only on generating future frames, proposed model structures are task-specific, and their applicability to other tasks is not well investigated.

Based on the above observations, we propose a framework that utilizes stochastic next-frame video prediction [99] into online video object detection. During training to generate future frames, we can obtain the video representation that can effectively enhance the feature map at the current frame with temporal information from past to several future frames. Stochastic forecasting deals with the uncertainty of the future by sampling one of the future possibilities.

In order to utilize stochastic future prediction for object detection, we propose a two-stage training method. Figure 5.9 shows the two-stage training flow of the pro-



(a) Pretraining the model by future prediction

(b) Fine-tuning the model to a downstream task

Figure 5.9: The overview of our proposed framework. To obtain a video representation, we pretrain the model through the future prediction task. Then, we append a detection module to the trained model and transfer it to the detection task. In the inference, only the dotted line area in (b) is used (the future prediction part is not used).

posed framework. First, we pretrain the encoder-decoder structure of the detector using stochastic next-frame video prediction [99]. In this pretraining step, the video representation, namely, the context of the video is learned. This step is conducted by self-supervised learning with unlabeled videos, and thus our model can exploit large-scale videos with no annotation. Next, we append the object detection module to the decoder as the downstream task and then fine-tune the model for a detector. We remark that although "pretraining" and "fine-tuning" are often used as applying a model trained on one dataset to another dataset in the same task, they mean in this method transferring the model to another task.

5.5.1 Stochastic Adversarial Video Prediction

We build our video object detection framework on Stochastic Adversarial Video Prediction (SAVP) [99]. Our model is modified from SAVP so that it is able not only to predict future frames, but also to detect objects in videos. This section briefly describes SAVP.

SAVP [99] combines GANs and VAEs. VAEs produce diverse images while sam-

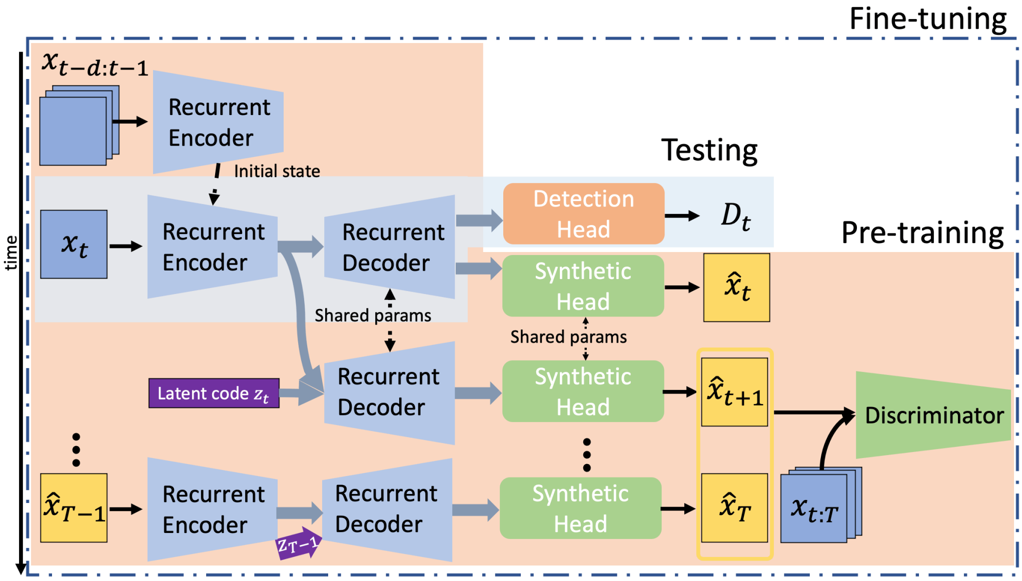


Figure 5.10: The architecture of the proposed model. The model is pretrained using the components in the orange-colored area, and then fine-tuned using the whole components. The inference corresponds to blue-colored area.

pling but tend to produce blurry images, while GANs produce clear images but suffer from producing diverse images. Combining VAEs and GANs thus benefits from their complementary strengths.

SAVP consists of a generator G and a discriminator D. G is with an encoder-decoder structure conditioned with past frames, and from a current frame and its latent codes at the time, generates the next frame while D optimizes G adversarially. SAVP possesses two distributions for sampling the latent code: the prior distribution and the posterior one, approximated by the learned encoder, in which the posterior distribution is parameterized by a conditional Gaussian distribution using frames of adjacent time steps. At test time, a random latent code z is sampled from the prior distribution independently at each time step. The generator G takes the previous frame and z, and then synthesizes a next-step future frame. To generate the next frame, the frame generated in the previous step is fed into G again, and the generation is repeated. During training, G is optimized to predict videos that match the distribution of actual videos using the discriminator D. The historical state is accessed via the recurrent neural networks in the generator G. SAVP also uses separate discriminators D and D^{VAE}, depending on the distribution used to sample the latent code.

5.5.2 Pipeline

The overall pipeline architecture of our proposed framework is depicted in Figure 5.10. The proposed method consists of five major components. They are (1) recurrent encoder RE for feature extraction from each frame, (4) a synthetic head for generating an image from the decoded feature map, and (5) a discriminator for discriminating the generated future frames from the actual future ones.

We have two training steps and one inference step. During "Pretraining", the model acquires the feature representation of videos by self-supervised learning through predicting future frames. The training method is essentially the same as SAVP, but reconstruction of the current frame is also performed for the "fine-tuning" step. Optimization of the generated future frames is performed using a GAN. "Fine-tuning" transfers the model to our downstream task, i.e., detection, using the feature representations acquired in the pretraining step. The difference from pretraining is that the detection head is appended to the decoder.

During "Inference", we do not generate future frames but detect objects. Generating future prediction frames contributes to acquiring video representations during training, but is not necessary for detection.

5.5.3 Our Prediction and Detection Network

The task of future prediction takes the current frame x_t (at time t) as input with the past d frames $\{c_i\}_{i=t-d}^{t-1}$ as the context, and predicts the future frame $\hat{x}_{t+1}$ at time t. However, video object detection requires a detection result in the current frame x_t at the input time t (the output from the decoder must be at the current time t). Therefore, to combine video object detection and future prediction, the model must be able to decode feature maps for the current frame and the next frame separately.

We design an encode-decoder network to decode the encoded features at each time independently. The network has the recurrent encoder RE and recurrent decoder RD as shown in Figure 5.10. The RE and RD are based on ResNet [81] and Feature Pyramid Network (FPN) [105] respectively. Two-layered ConvLSTMs [138] are added to the outputs of each ResNet block (C3, C4, C5, P3, P4, P5). The roles of ConvLSTMs in RE and RD are different. ConvLSTMs in RE are used to generate the temporally-aware feature map [110], whereas those in RD are used to propagate the information over time.

Using the feature map obtained from the encoder at time t, the decoder at time t together with the synthetic head at time t first reconstructs the current frame $\hat{x}_t$. Then, it samples the latent code z_t and decodes the feature map with z_t to generate the next frame $\hat{x}_{t+1}$. The states of the recurrent neural network in the decoder are propagated to account for time-stamp information. The decoders at time t and $t+1$ share their weights except the ConvLSTMs states. In order to generate realistic images from the decoder, we append the shared-weighted synthetic heads on top of the output of P3. The synthetic head is a simple network consisting of a convolution layer with a 3-size kernel and 2-stride, instance normalization [150], and ReLu, stacked together twice. The output of the final convolution layer is set to 3 dimensions for RGB images. To optimize the generated image, the corresponding ground truth image is resized to the P3 feature map resolution size.

To enable stochastic sampling for the future frame generation, RD is conditioned on time-varying latent codes. Those codes are sampled at training. Each latent code z_t is a 16-dimensional vector, and is passed through a fully-connected LSTM to facilitate correlations in time of the latent variables. The encoded latent codes are then converted to match the 256 input dimensions of FPN, and added channel-wisely to the all input of FPN during lateral connections. Thus, the latent codes are added to the input of FPN when generating the future frame stochastically, but not when generating the current frame. FPN works as an ordinary object detection module.

5.5.4 Pretraining Loss

The current frame image is first reconstructed to enable the model to transfer it to both detection in the fine-tuning step and future frame prediction. Then, the decoder timestamp is increased to generate future frames. The loss function for the pretraining step is almost identical to that for the SAVP training [99]. The only difference is that our loss involves the current frame reconstruction. We use $d = 10$ frames for initialization to predict future as proposed in [99].

The loss function of SAVP is defined with four weights λ_i ($i = \{1, \ldots, 4\}$) as follows:

$$\mathcal{L}_{\text{savp}} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_{\text{KL}} + \lambda_3 \mathcal{L}_{\text{GAN}} + \lambda_4 \mathcal{L}_{\text{GAN}}^{\text{VAE}}, \tag{5.4}$$

where $\mathcal{L}_1$ is the L1 norm between the forecasted frames and the ground truth, $\mathcal{L}_{\mathrm{KL}}$ is the KL divergence between the prior and posterior distribution, $\mathcal{L}_{\mathrm{GAN}}$ is adversarial loss for discriminator, and $\mathcal{L}_{\mathrm{GAN}}^{\mathrm{VAE}}$ is analogous to $\mathcal{L}_{\mathrm{GAN}}$ except which use latent codes sampled from the posterior distribution. See [99] for details of these terms. In order to optimize the reconstructed current frame $\hat{x}_t$ from the actual frame x_t at the time t, we employ L1 norm. The loss function $\mathcal{L}_{\mathrm{video}}$ for our pretraining is defined as:

$$\mathcal{L}_{\mathrm{video}} = \mathcal{L}_{\mathrm{savp}} + \lambda_5 ||x_t - \hat{x}_t||_1, \tag{5.5}$$

where λ_5 is the weights for reconstruction loss of the current frame.

5.5.5 Fine-tuning Loss

The pretrained model will be optimized to be a model for detection. Through fine-tuning, we train the whole weights to fit for detection. Fine-tuning is the same as pretraining except for detection loss and input. The loss for detection defined in FCOS [145] is set to $\mathcal{L}_{\mathrm{det}}$. As a result, the loss function for the current frame in fine-tuning is

$$\mathcal{L}_{\mathrm{finetune}} = \alpha \mathcal{L}_{\mathrm{video}} + \beta \mathcal{L}_{\mathrm{det}}, \tag{5.6}$$

where α and β is the balance weights for $\mathcal{L}_{\mathrm{video}}$ and $\mathcal{L}_{\mathrm{det}}$, respectively.

The difference of input is that in detection, the previous frame may not be obtained due to the timing of video loading, so we select a random value from $[0\ 10]$ (we set $d = 10$) and use the frame corresponding to the value.

5.5.6 Inference Step

The encoder and decoder inherit the ConvLSTMs state from the previous frame except for the initial frame, and the detection is performed sequentially using the encoder-decoder with input frames. It is important to note that we neither reconstruct the current frame nor predict future frames at the inference step. This is because the fine-tuned model already acquires the video representation for detection. Removing reconstruction and prediction functions also contributes to faster inference.

5.6 Experiments through Next Several Future Predictions

5.6.1 Benchmark Datasets and Metrics

We evaluated the performance of our method on three public datasets: ImageNet VID [134], UA-DETRAC [164] and VisDrone-VID2019 [188] (see Section 4.4.1 for more details about datasets.) We evaluated the performance using mean average precision (mAP), AP and AP for ImageNet VID, UA-DETRAC and VisDrone-VID, respectively (cf. Section 4.4.1).

5.6.2 Implementation Details

We employ FCOS [145] as the baseline object detector in our proposed model. On ImageNet VID, we use ResNet-50 and ResNet-101 [81] with FPN [105] for the backbone and insert ConvLSTMs as described in Section 5.5.3. We utilize ResNet-101 as a backbone on Visdrone and UA-DETRAC dataset for a fair comparison with other methods. We follow the hyper-parameters of FCOS [145] and the modifications [2]. The input images are resized to have their smaller side to be 512 pixels on ImageNet VID and UA-DETRAC, and 800 pixels on VisDrone-VID2019, respectively.

For pretraining, we follow SAVP [99] with SGD and a batch size of 16 with pretrained weights of ImageNet. We set $\lambda_1 = 0.25$, $\lambda_2 = 0.0375$, $\lambda_3 = 0.3$, $\lambda_4 = 0.3$, and $\lambda_5 = 0.25$ empirically and use 16 dimensions of latent codes. We utilize the discriminator D as proposed in [99]. We train our model for ten epochs to predict a 10-frame forward future in total, with the learning rate of 10^{-4} and 10^{-5} in the first six and the last two epochs, respectively.

For fine-tuning, we set $\alpha = 1.0$, $\beta = 1.0$. We then train the pretrained model for five epochs, with the learning rate of 10^{-4} and 10^{-5} in the first 3.3 and the last epoch, respectively. Although we trained our model on two RTX 3090 GPUs, we evaluated the speed performance on two 2080 Ti GPUs for a fair comparison with other methods. In total, our proposed method with ResNet-50 and ResNet-101 consumes 1.5 and 1.8 GiB of GPU memory for all components during inference, respectively.

Table 5.4: Performance comparison with the state-of-the-art online and real-time detectors on ImageNet VID `val`. (* is tested by ourselves.)

Models	Backbone	Base Detector	mAP	FPS	Device
LMP [193]	MobileNetV2 [136]	RetinaNet [106]	64.2	29	GTX 1060
TSSD-OTA [27]	VGG-16 [141]	SSD [112]	65.4	21	Titan X
ROD-FMF [62]	MobileNetV2 [136]	SSD [112]	65.7	39	2080 Ti
VOD-MT [86]	VGG-16 [141]	SSD [112]	71.0	18	–
CrevNet* [182]	CrevNet-48-2 [182]	SSD [112]	60.1	1.8	2080 Ti
DFF [192]	ResNet-101 [81]	R-FCN [37]	73.1	20	K40
AdaScale [31]	ResNet-101 [81]	R-FCN [37]	75.5	21	1080 Ti
Attention-guided [179]	ResNet-101 [81]	R-FCN [37]	73.7	22	1080 Ti
Heatmap-guided [175]	ResNet-101 [81]	CenterNet [51]	76.7	37	–
Ours	ResNet-50 [81]	FCOS [145]	73.1	**54**	2080 Ti
Ours	ResNet-101 [81]	FCOS [145]	**78.0**	39	2080 Ti

5.6.3 Comparison with State-of-the-Art

Comparison on ImageNet VID

We compare our method against several online video object detectors. Table 5.4 shows their performance comparison. We observe that with the ResNet-101 backbone, our method surpasses the strong competitive detector, Heatmap-guided [175] with faster processing speed.

Our method runs at 54 and 39 fps, positioning at the first and second place with ResNet-50 and ResNet-101, respectively. Despite the high-speed processing, it achieves an accuracy of 73.1 mAP, which may be sufficient for reasonable detection accuracy. A comparison between detectors capable of real-time processing confirms that our method runs at high speed while maintaining high accuracy.

In particular, when compared with VOD-MT [86], which is close in accuracy, we see that our method runs more than twice faster. VOT-MT aggregates past feature maps to stabilize detection, and takes time for each frame aggregation. In contrast, our method learns video representation in advance and transfers it to the detection task, so that no aggregation is required for detection. This difference brings the high accuracy of our method with excellent speed. More detailed comparison with VOT-MT is presented in Section 5.6.4.

In terms of GPU memory consumption, the proposed method consumes 1.5GiB for ResNet-50 and 1.8GiB for ResNet-101. On the other hand, VOD-MT and CrevNet consume 4.3GiB and 3.9GiB, respectively, indicating that the proposed method can operate with low memory. Therefore, the proposed method achieves high accuracy and low

Table 5.5: Performance comparison on UA-DETRAC `test`. Bold faces are the top performance on each subset. Methods in the first block are for still images. The methods in the second block are for videos. (* is tested by ourselves; $\star$ is our own implementation.)

Method	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny	FPS	GPU
GP-FRCNN [140]	76.57	91.79	80.85	66.05	85.16	81.23	68.59	77.20	4	Tesla K40
EB [156]	67.96	89.65	73.12	54.64	72.42	73.93	53.40	83.73	11	Titan X
YOLOv3-SPP [92]	84.96	95.59	89.95	75.34	88.12	88.81	77.46	89.46	6-7	Titan Xp
MSVD_SPP [93]	85.29	96.04	89.42	76.55	88.00	88.67	78.90	88.91	9-10	Titan Xp
FG–BR_Net [59]	79.96	93.49	83.60	70.78	87.36	78.42	70.50	89.89	10	Tesla M40
SpotNet [124]	86.80	97.58	92.57	76.58	89.38	89.53	80.93	91.42	14	GTX 1080 Ti
RetinaNet [106]	69.14	86.82	73.70	56.74	79.88	66.57	55.21	82.09	–	–
3D-DETNET [101]	53.30	66.66	59.26	43.22	63.30	52.90	44.27	71.26	26	–
RN-VID [126]	70.57	87.50	75.53	58.04	80.69	69.56	56.15	83.15	–	–
FFAVOD-SpotNet [125]	**88.10**	**97.82**	**92.84**	**79.14**	**91.25**	**89.55**	**82.85**	**91.72**	–	–
FFAVOD-RetinaNet [125]	70.57	87.50	75.53	58.04	80.69	69.56	56.15	83.60	–	–
TSSD* [26]	57.16	81.06	62.07	43.14	57.59	63.87	44.98	67.73	32	RTX 2080 Ti
VOD-MT$\star$ [86]	67.22	82.81	74.36	55.29	71.43	66.79	64.16	70.82	14	RTX 2080 Ti
TFEN [61]	82.42	97.40	88.90	72.18	87.54	82.41	72.32	90.78	29	RTX 2080 Ti
Ours	80.91	96.52	86.73	71.02	87.31	81.30	71.19	90.10	**38**	RTX 2080 Ti
FCOS* [145]	70.01	86.91	74.17	57.27	80.13	67.16	55.92	82.47	41	RTX 2080 Ti

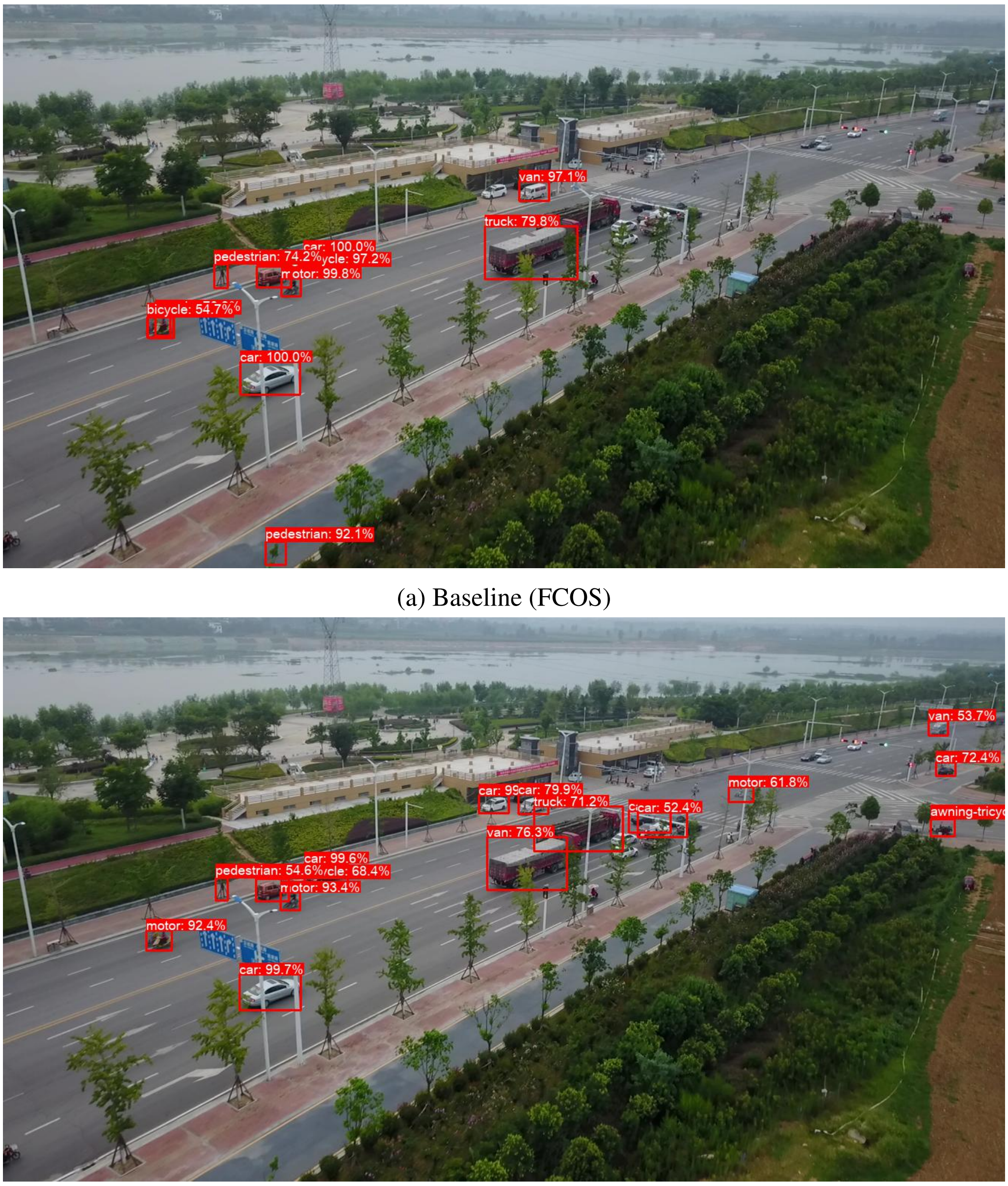
Table 5.6: Performance comparison with the state-of-the-art models on VisDrone-VID2019 `test`.

Methods	AP	AP$_{50}$	AP$_{75}$	AR$_{1}$	AR$_{10}$	AR$_{100}$	AR$_{500}$
Faster R-CNN [131]	14.46	31.80	11.20	8.55	21.31	26.77	26.77
D&T [54]	17.04	35.37	14.11	10.47	25.76	31.86	32.03
FGFA [191]	18.33	39.71	14.39	10.09	26.25	34.49	34.89
FCOS(baseline) [145]	15.12	32.42	11.44	9.01	22.29	26.98	26.98
Ours	21.82	49.01	16.83	12.91	30.22	41.28	41.28
DBAI-Det [188]	29.22	58.00	25.34	14.30	35.58	50.75	53.67

memory by training through future prediction.

Comparison on UA-DETRAC

The results on the UA-DETRAC dataset are reported in Table 5.5. We confirm that the proposed method ranks the fastest among several detectors except for the baseline, about four times faster than FG-BR_Net [59], which produces close accuracy. As for methods using temporal information, FFAVOD-SpotNet [125], which fuses the feature maps of the still image detector SpotNet with past and future feature maps, is superior in terms of accuracy. Moreover, FFAVOD has a gain of about 1.4 pt when applied to RetinaNet, similar to our method baseline, FCOS. In contrast, our method has a significant gain of about 11 pt without using future information, confirming that the effect of future prediction is substantial.



(a) Baseline (FCOS)

(b) Ours

Figure 5.11: Visualized results for subtle motion blur scene on VisDrone-VID `test`. From top to bottom: predictions of Baseline (FCOS) and ours. Best viewed digitally and in color.

Comparison on VisDrone-VID

Table 5.6 shows performance comparison. D&T [54] and FGFA [191] are methods to stabilize the detections of the still image detector [131] with temporal information. We see that our model significantly improves the accuracy from the baseline and outperforms the compared methods. Our model runs at 23 fps. As a reference, we show DBAI-Det [188] since it is ranked in first place at the VisDrone-VID2019 competition. Note that DBAI-Det [188] combines heavy backbone [172] and several methods [15, 38] for

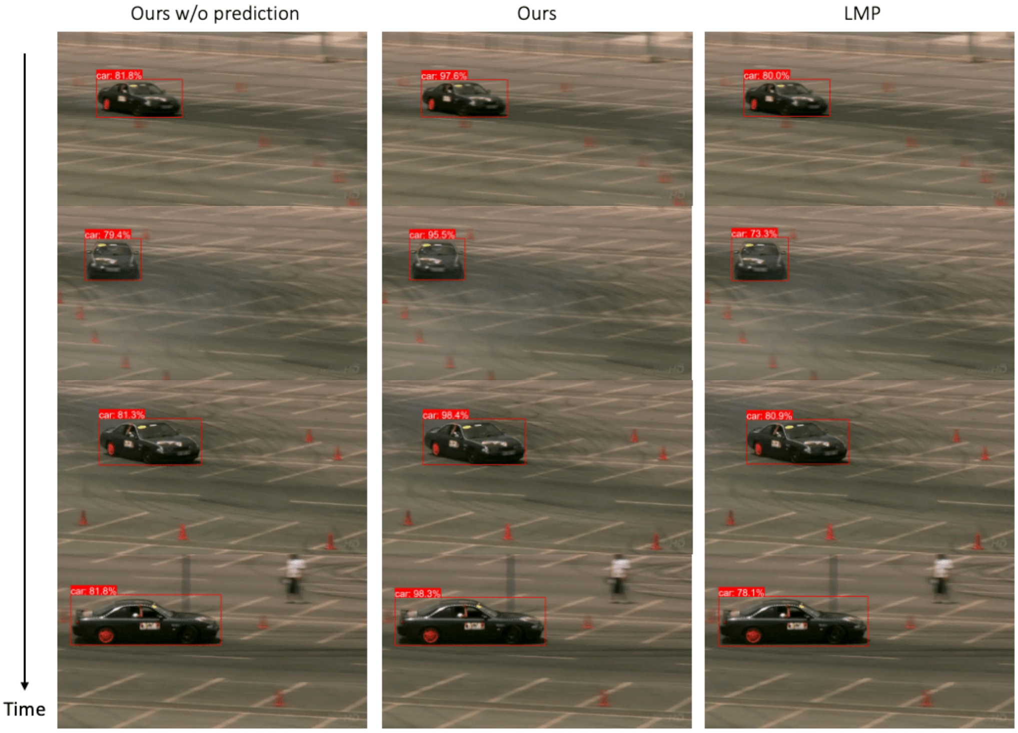


Figure 5.12: Visualization results on ImageNet VID `val`. From left to right: the predictions of the model w/o prediction, the complete model, and LMP [193]. The proposed method continues to detect the object on degraded scenes with a high degree of confidence.

accuracy, and runs at less than 1 fps.

Figure 5.11 visualize the comparison of the baseline and the proposed method on VisDrone-VID scene. We see that our method detect target object robustly compared to the baseline.

5.6.4 Detailed Analysis

To confirm the effectiveness of the proposed method in detail, we conducted ablation studies on ImageNet VID and UA-DETRAC and detailed analysis on ImageNet VID. We follow a motion-aware evaluation metric in [191] to evaluate the performance on

Table 5.7: Ablation study of our model on ImageNet VID `val` and UA-DETRAC `test`.

Methods	ImageNet VID					UA-DETRAC
	mAP	mAP_s	mAP_m	mAP_f	FPS	AP
FCOS (baseline) [145]	68.7	79.3	68.5	43.6	**56**	70.0
model w/o prediction (+ConvLSTMs only)	70.1	80.0	68.7	44.5	54	71.6
model w/o pretraing	71.6	83.1	69.0	47.1	54	77.2
complete model	**73.1**	**83.3**	**71.7**	**52.7**	54	**80.9**
complete model w/o GAN	71.7	81.5	69.6	48.2	54	72.5
complete model w/o VAE	70.7	80.9	69.2	44.9	54	73.1

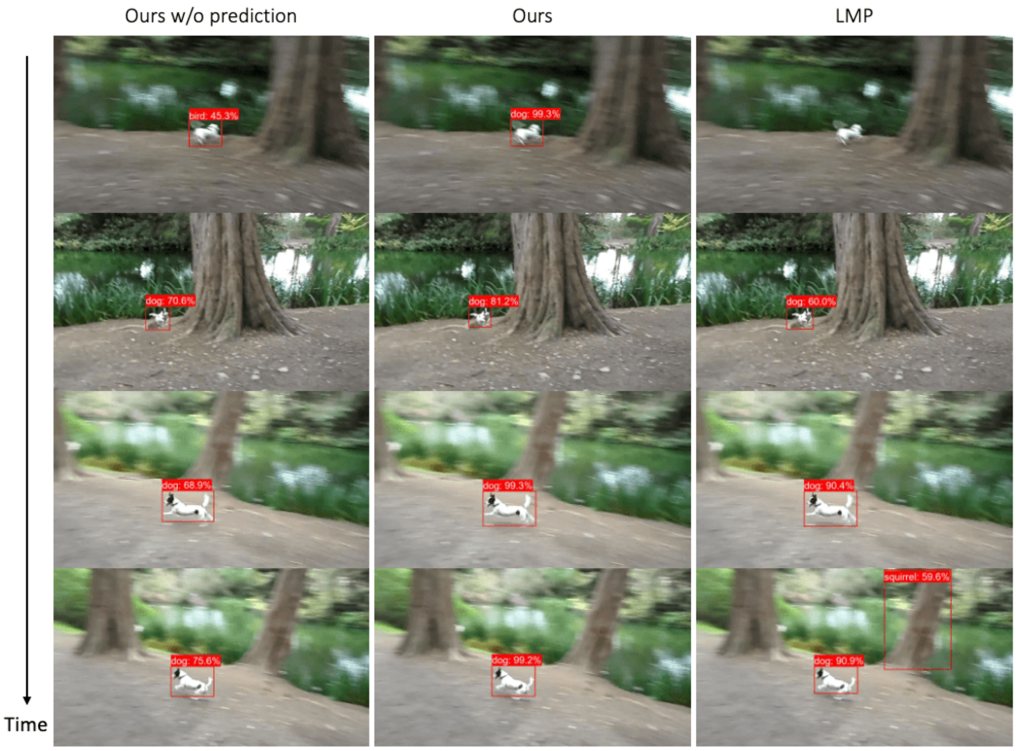


Figure 5.13: Visualization results on ImageNet VID `val`. From left to right: the predictions of the model w/o prediction, the complete model, and LMP [193]. The proposed method provides stable detection without class error even when the object moves rapidly.

the categories of slow, medium, and fast objects, where these three categories are divided by their average IoU scores between objects across nearby frames. (see details in Section 4.4.5.)

Component Ablation Analysis on ImageNet VID and UA-DETRAC

We evaluate the impact of key components of our model on the detection accuracy; see Table 5.7. Model w/o prediction exploits from past to current frames such as [27] (we append ConvLSTMs only), and model w/o pretraining represents the model is trained for future prediction and object detection simultaneously without pretraining, and the number of training iterations is the same as for pretraining. The lower part of Table 5.7 shows the models without using VAEs or GANs in the SAVP [99] part. We see that model w/o pretraining outperforms model w/o prediction, meaning that the accuracy is improved by training the recurrent neural network to predict the future, rather than simply propagating features from the past to the present. It is important to emphasize that the significant gain in UA-DETRAC is future prediction, which is considered to be an easily predictable object. We also see that our complete model significantly outper-

Table 5.8: Performance comparison of VOD modules with VOD-MT [86] on RetinaNet and ResNeXt-101 on ImageNet VID `val`.

Methods	mAP	mAP$_s$	mAP$_m$	mAP$_f$	FPS
RetinaNet [106]	77.9	87.3	74.5	55.7	**9.1**
VOD-MT [86]	79.2	88.2	76.0	57.5	6.4
Ours	**81.5**	**89.2**	**80.2**	**63.4**	8.7

forms the model w/o pretraining. This indicates that learning the video representation through prediction and transferring it to detection is a meaningful procedure to improve detection accuracy. This is also supported by the fact that the improved accuracy for fast objects is remarkable because video representations for fast objects are more sensitive to changes in context, such as motion. When GANs or VAEs are ablated, the accuracy drops, confirming that they are both critical for using long-term future predictions.

Figure 5.12 and 5.13 show the detection results of the model w/o prediction, ours and the competitive model, LMP [193]. All models utilize the same backbone and architecture for a fair comparison. We see that our method provides more stable detection than the model w/o prediction. In particular, our method is found to be robust to blurring and suppresses class switching, achieving a higher score as seen in Table 5.7.

Impact analysis on detection confidence score

We examined how the proposed method differs from the existing approach, which propagates temporal information from the past to the current time regarding the detection confidence score. Figure 5.14 shows the confidence scores with time for the model w/o prediction, the complete model, and LMP [193]. The score is given for objects detected with the correct label and zeroes for false positives or missing detections. We confirm that the proposed method maintains a higher score with smaller fluctuation than the existing approach. In addition, it correctly detects objects longer intervals even in scenes that are difficult to detect, as shown in Figure 5.13. We see that feature map enhancement through prediction is essential for stable detection.

Fair Comparison with the Same Baseline

To make the comparison more fairly with the closely performing method VOT-MT [86], we follow the same configuration of the detector and feature extractor as VOD-MT. Table 5.8 shows the accuracy and speed comparison of VOD-MT under the same detector,

Table 5.9: Impact of the number of the future prediction on ImageNet VID `val`.

predicted frames (T)	1	3	5	7	10	15	20
mAP	71.9	72.3	72.5	72.8	73.1	72.9	73.2

feature extractor, and input frame size. Both methods have improved accuracy from the base detector, but our method achieves higher accuracy and faster processing speed. This is because our method just uses robust feature representations for the inference that are learned during training, while VOT-MT generates robust feature maps at each inference.

Effect of KL Divergence

We change KL loss weight λ_2 to see how the weighting for VAE affects the detection and generation. Figure 5.15 shows under different λ_2, the detection accuracy and Structural Similarity Index Measure (SSIM) [163]) computed with the ground truth in 10 future frames. When λ_2 is large (weighting for VAE is large), KL loss prevents the generation as the regularizer (producing poor SSIM). As λ_2 becomes small, however, the detection accuracy and the generated image become high until a certain point. Then, as λ_2 becomes even smaller, KL loss does not work well for detection while rendering becomes better until some point and then gradually degraded. Therefore, there is the trade-off and λ_2 that compromises detection and generation, which should be learned as a well-balanced point. The frame generation accuracy does not increase when the KL loss works well because images are generated stochastically, and they are structurally different from the actual future ones, as seen at the bottom of the row.

Impact of Prediction of Future Frame

We investigate how much the successive future predictions affect the detection accuracy. Table 5.9 shows the detection accuracy under different number of generated frames during training. We see that the accuracy gradually improves with longer future predictions and becomes saturated with about ten frame predictions. We confirm that ten frames are effective and sufficient for the prediction.

Comparison of Stochastic and Deterministic Future Predictions

The proposed method learns feature representation by stochastically predicting what is likely to happen in the future using VAE. We evaluate how stochastic prediction af-

Table 5.10: Performance comparison under different pretraining datasets on ImageNet VID `val`.

Pretraining dataset	ImageNet VID	YouTube-BB	BDD100K
mAP	73.1	(+3.5) 76.6	(+2.1) 75.2

fects the acquisition of video representations with respect to the size of the training set (Fig. 5.16). As a comparison of deterministic prediction, we also show the result without using VAE. Here, we change the ratio of training data against the training set of ImageNet VID from 0.5 to 1.0 by 0.25. The number of iterations in training is adjusted not to be affected by the ratio. Figure 5.16 shows that stochastic prediction tends to be more accurate as of the training data size increases compared to deterministic prediction. We also observe that while the final value of the loss function for deterministic prediction does not change along with the training data size, that for stochastic prediction becomes increased.

Figure 5.17 shows that the deterministic prediction converges to almost the same level regardless of the ratio of data really used for training against the training set of ImageNet VID. In contrast, stochastic prediction converges significantly different levels depending on the ratio. This can be understood as follows: as the data variation increases, the training loss becomes unlikely to drop, resulting in less overfitting to the training videos. This suggests that stochastic prediction leads to increasing the model capacity for detection by avoiding overfitting that arises for deterministic prediction due to redundant training data [167].

Impact of Pretraining Dataset

Our model does not require an annotated dataset for pretraining. In order to investigate how the size and variation of datasets used for pretraining affect detection accuracy, we exploit two video datasets: YouTube-BB [127] and BDD100K [180]. YouTube-BB is a new large-scale natural scene dataset similar to ImageNet VID and consists of about 380,000 15-20 second videos extracted from publicly available YouTube videos. BDD100K is the largest and most diverse open-driving video dataset to date, consisting of 100,000 videos recorded in different weather conditions such as clear, cloudy, and rainy, and at different times of the day and night.

Table 5.10 shows the detection accuracy under different datasets for pretraining.

Table 5.11: Prediction accuracy in SSIM on Caltech Pedestrian dataset. Higher SSIM means better prediction accuracy.

Model	Next-Frame	3rd	6th	9th
CrevNet [182]	**0.92**	**0.83**	**0.73**	**0.67**
Ours	0.89	0.79	0.69	0.65

Table 5.12: Detection accuracy in mAP on KITTI

Methods	Car			Pedestrian			Cyclist			mAP
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	
CrevNet [182]	91.9	91.8	86.0	**89.7**	**83.2**	75.8	87.3	80.9	72.2	84.3
Ours	**95.3**	**93.3**	**91.1**	88.8	80.5	**75.9**	**89.1**	**81.2**	**73.1**	**85.4**

Without any effort, pretraining with the YouTube-BB dataset improves the accuracy by 3.5 points. This is a significant gain without increasing any cost of inference. Pretraining with BDD100K also shows the improvement in accuracy by 2.1 points. This interestingly indicates that even pretraining on a completely different-looking dataset improves accuracy. These observations mean that there is excellent potential for accuracy improvement by using an immense amount of training data for learning video representations through future prediction.

5.6.5 Discussion

CrevNet [182] is proposed recently as a deterministic model to generate future images. It focuses on predicting future frames as accurately as possible by minimizing information loss during feature extraction, but its application to detection is also suggested. Here, we evaluate the accuracy of future image generation and detection to see whether deterministically predicting accurate future images is really required for accurate detection. We follow the evaluation in CrevNet [182]. To be more specific, we pretrained our model for video prediction on KITTI [67]. The accuracy of future image generation by the pretrained model is then evaluated on the Caltech Pedestrian dataset [48] using SSIM [163]. Next, we fine-tuned the model using the detection data on the KITTI. Since the training set of the KITTI dataset provides unlabeled frames of the previous three frames for each annotated detection frame and no future frames, the fine-tuning step of our method is purely for the detection part.

Tables 5.11 and 5.12 show accuracy of generating future images on Caltech, and the detection accuracy on KITTI. We see that while our method is less accurate than

Table 5.13: Performance comparison with the state-of-the-art models on YouTube-VIS2019 `val`. All of the methods use ResNet-50 as backbone.

Methods	Online?	FPS	AP	AP$_{50}$	AP$_{75}$	AR$_{1}$	AR$_{10}$
OSMN [177]	✓		27.5	45.1	29.1	28.6	33.1
DeepSORT [165]	✓		26.1	42.9	26.1	27.8	31.3
FEELVOS [153]			26.9	42.0	29.7	29.9	33.4
MaskTrack R-CNN [176]	✓	10	30.3	51.1	32.6	31.0	35.5
MaskProp [8]			40.0	–	42.9	–	–
SipMask [16]	✓	34	32.5	53.0	33.3	33.5	38.9
STEm-Seg [4]		4	30.6	50.7	33.5	31.6	37.1
CompFeat [58]	✓		35.3	56.0	38.6	33.1	40.3
VisTR [161]		30	36.2	59.8	36.9	37.2	42.4
CrossVIS [178]	✓	40	34.8	54.6	37.9	34.0	39.0
VisSTG [159]	✓	22	36.5	58.6	39.0	35.5	40.8
SG-Net [109]	✓	23	34.8	56.1	36.8	35.8	40.8
TF-Blender [36]+SG-Net [109]		21	35.7	57.1	37.6	36.6	42.0
Ours+SG-Net [109]	✓	21	36.3	54.7	40.1	36.7	43.6

CrevNet in terms of generating future images in both the short and long-term, it is better in terms of detection; our method significantly outperforms CrevNet. We reason that the video representation learned for generating accurate future images does not match the representation for object detection. Therefore, some uncertainty in video representation brought by stochastic prediction is needed to increase the model capacity for other tasks. Fine-tuning to object detection makes use of this capacity to adjust the video representation to detection.

5.6.6 Video Instance Segmentation Results

Since our method is applicable to methods that are annotated only on keyframes, we evaluated it on the YouTube-VIS dataset [176] (See Section 4.4.6 for more details.) To support instance segmentation and tracking, we used the SG-Net [109] method based on the one-stage detector. Table 5.13 shows the results of the YouTube-VIS validation set. Our method improves the accuracy by 1.5 points over SG-Net. While most of the methods are proposed for tracking and mask propagation, recently, TF-Blender has been proposed to improve the accuracy by improving the feature map in the surrounding frames. This is an improvement of 0.9 points over the base model, which confirms the effectiveness of our feature map enhancement.

Figure 5.18 shows the detection results of SG-Net and the proposed method. We can see that the proposed method improves the accuracy of mask generation and classification.

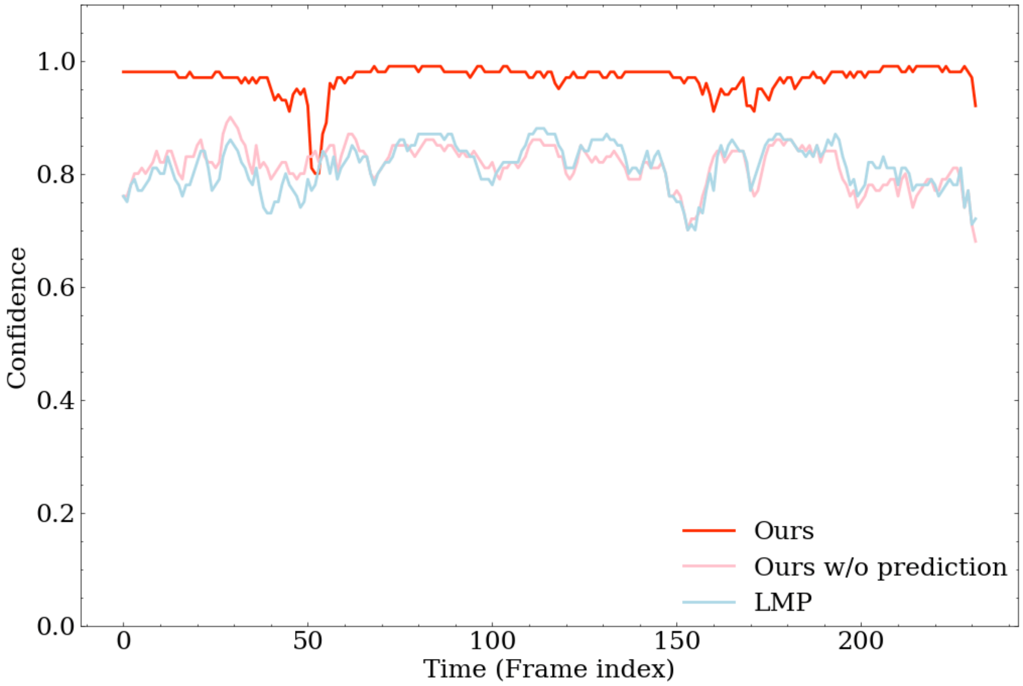
5.7 Conclusion

For stable object detection in a live-stream video that can be processed in real-time under memory limitations, we proposed to improve accuracy by feature map enhancement through prediction. We proposed two frameworks that utilize the next and the following several frames predictions for online video object detection to enhance feature maps.

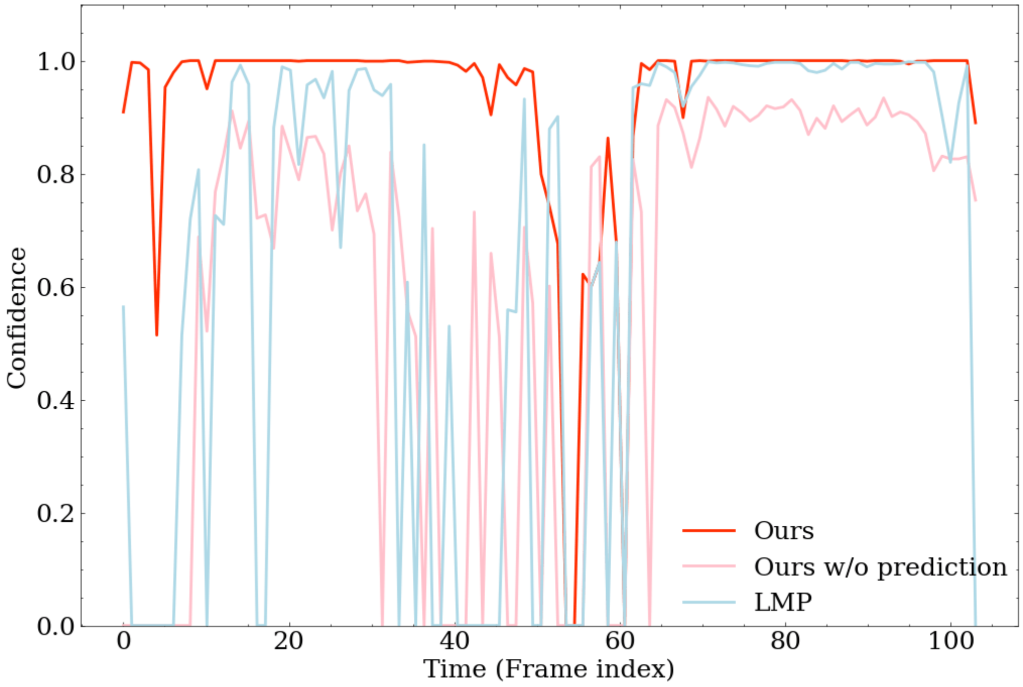
In the next frame future prediction approach, we present a video object detection model that jointly learns to detect objects in the current frame and predict the next frame's feature map. We also improved the processing speed using the predicted feature map for the next-frame detection and introduced an adaptive scheduler to stabilize the detection. Our experiments show that detection accuracy is improved through next-frame feature map prediction and that the processing speed is improved while only a slight loss of the detection accuracy using the scheduler.

Next, we proposed feature enhancement learning in terms of the next several frames prediction. To employ the prediction, we introduced a probabilistic future generation framework. To utilize the future forecast more effectively, we proposed a method that first learns the video representation by predicting future frames and fine-tunes the model for object detection by optimizing the detector module added as a downstream task. Our experiments have shown that the proposed detection model can be robust against the apparent change with time, leading to higher accuracy. We also showed that the detection performance improves with the size of the pretraining data.

Currently, we employed a fixed probability distribution for videos and the exact sampling at each network layer to design a stochastic future prediction model. However, videos in the real world are much more diverse and represented differently at each layer. Exploring further expressive probability distribution with sampling by the hierarchy is left for future work.



(a) Detection confidence score with time with the model w/o prediction, the complete model, and LMP [193]. It corresponds to the video in Figure 5.12.



(b) Detection confidence score with time with the model w/o prediction, the complete model, and LMP [193]. It corresponds to the video in Figure 5.13.

Figure 5.14: Example of changes in detection confidence score over time.

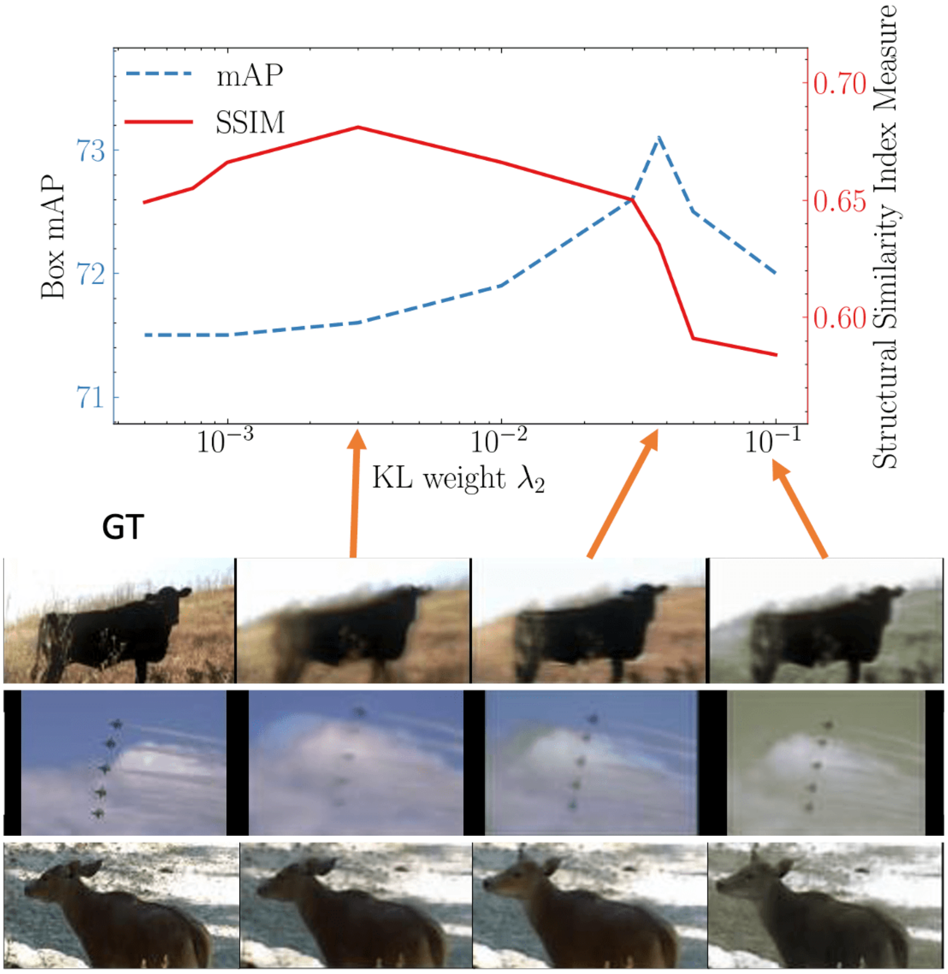


Figure 5.15: Effect of varying the KL loss weights on the detection and generation accuracy, showing the synthesized 10th frames corresponding to the weights and the corresponding ground truth.

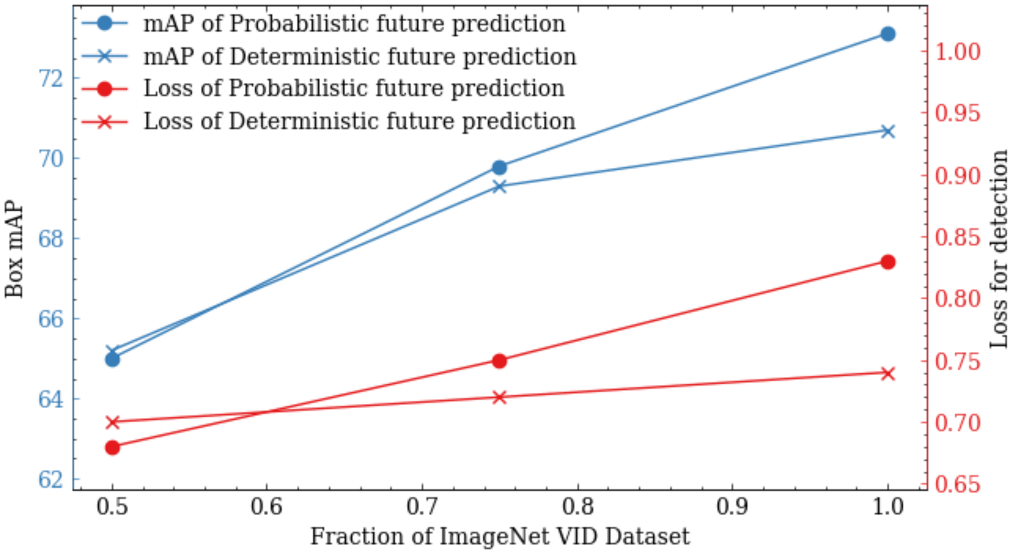


Figure 5.16: Accuracy impact of different methods of generating future forecasts

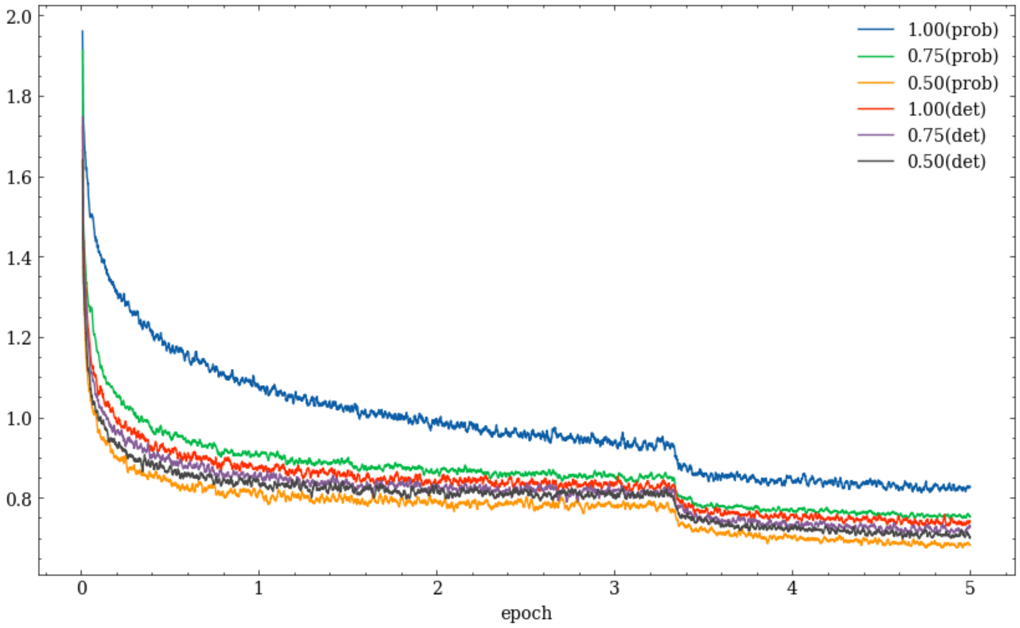


Figure 5.17: Visualization of training losses on several settings. The numbers indicate the ratio of data used for training against the training set of ImageNet VID, and “prob” and “det” indicate stochastic prediction and deterministic one.

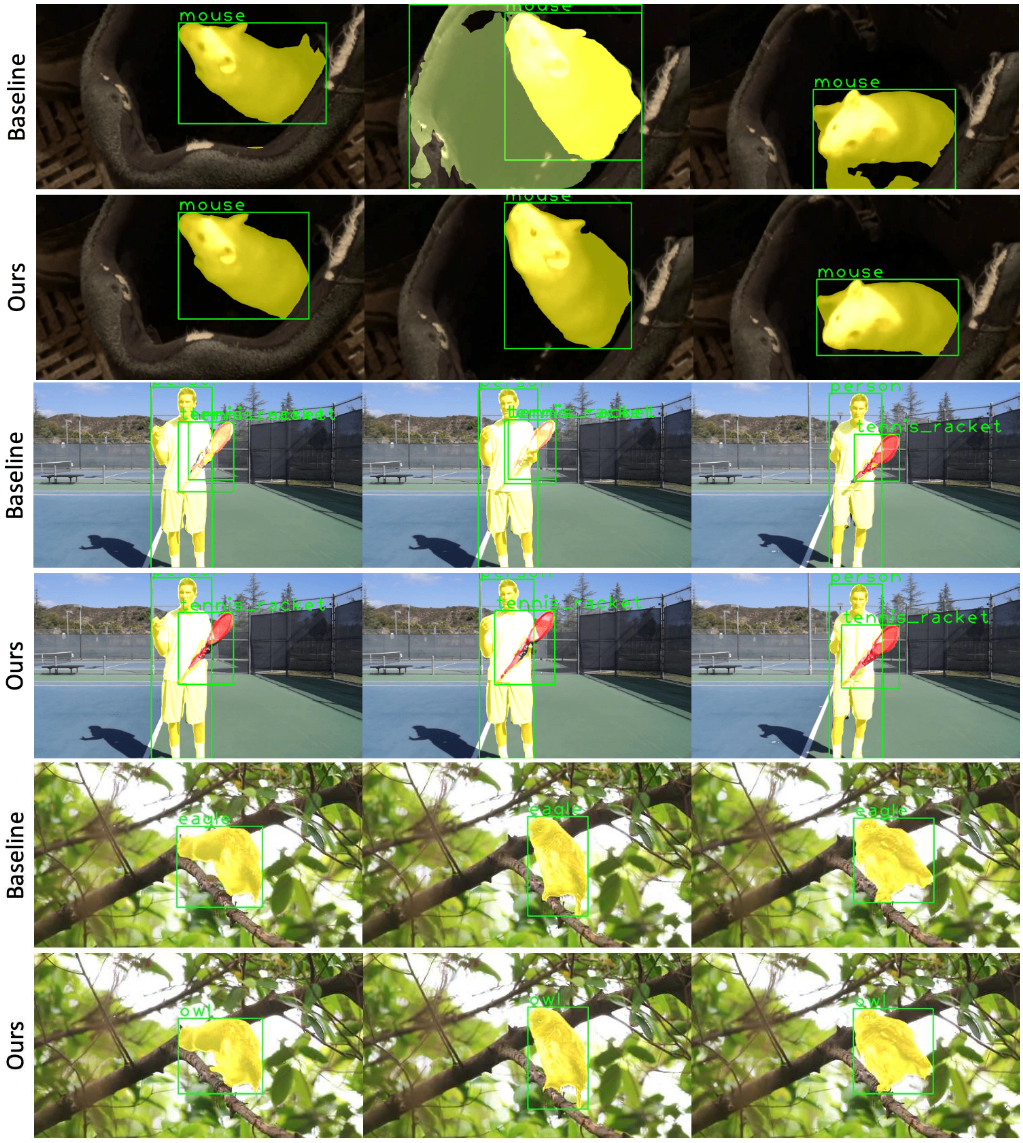


Figure 5.18: Visualized results comparison the baseline (SG-Net [109]) and ours on YouTube-VIS `val`. Results are plotted if their confidence score is larger than 0.45. Best viewed digitally and in color.

Chapter 6

Conclusion

6.1 Summary

Detecting objects in a video is a fundamental technique for obtaining information from video and is essential for real-world applications such as surveillance cameras and robotics. However, in video object detection, appearance changes with time cause detection confidence fluctuation and false-negative and false-positive detection problems. They make it difficult to perform stable detection using still image detectors. To tackle them, the previous studies tried to stabilize detection by incorporating temporal information into the detection result or the detection stage. However, they do not work well unless bounding boxes are detected in most frames. Since the appearance changes deteriorate the feature map for detection, it is challenging to detect objects accurately. Therefore, it is crucial to enhance the feature map for detection with temporal information before the detection stage, and this approach has been studied as a major trend in recent years.

Considering real-world applications such as robotics and surveillance cameras, video object detection is expected to detect objects in a live stream video within the real-time processing speed as well as high accuracy. However, the video object detection research mainly focuses on the offline setting, which utilizes past, present, and future temporal information. There are few studies for the online setting which does not leverage future information. The existing works, which run in real-time on the online setting, enhance feature maps using only the last or a nearby keyframe; however, the accuracy tends to be low due to the limited temporal information. Therefore, there is a need to study video object detection for real-world applications, which can sufficiently utilize temporal information for stable detection.

This dissertation addresses the challenging issue of utilizing spatiotemporal information to obtain suitable features for detecting objects in a video within real-time processing in the online condition. The previous works proposed to propagate information using the last frame or keyframe. In contrast, we proposed online feature aggregation approaches with external memory, which directly exploits multiple past feature maps. We enhanced the feature maps and improved the detection accuracy by learning how to aggregate the feature maps. Detailed experiments show that the proposed methods dramatically reduce the video object detection problems, such as confidence fluctuation, false-negative, and false-positive detection. Next, we proposed novel prediction-based feature refinement approaches for strict memory conditions. Since the feature aggregation methods use external memory, it may be difficult to introduce them in the environments such as robotics due to GPU memory limitations. In contrast, the prediction-based approach exploits the recurrent neural network without any external memory. This approach enhances the feature map by learning the knowledge of the object through future predictions during training. Thus, we have contributed to two aspects of feature map enhancement learning for video object detection: feature map aggregation and prediction-based feature map enhancement. Figure 6.1 and 6.2 show our contributions to ImageNet VID dataset and UA-DETRAC dataset, respectively. We confirm that the proposed methods are generally superior to the existing methods regarding the trade-off between accuracy and speed on different datasets.

Feature Map Aggregation. To take into account the temporal information of the video as much as possible under the limitation of online processing in real-time, we proposed feature map aggregation methods and obtained information directly from the multiple past frames in the external memory. Feature map aggregation has been mainly studied in offline video object detection, and it is effective in terms of accuracy. However, it requires a lot of processing time since it calculates the similarity for each frame in the past and weighs them accordingly. To address these issues, we proposed two methods: frame-level feature map aggregation and element-level feature map aggregation. We proposed a fast aggregation method in frame-level aggregation, which weighs multiple past frames in a one-shot manner. This method enables us to enhance the feature map effectively within real-time processing time without computing frame-by-frame weights.

In detailed validation, we confirmed that the fluctuation of detection confidence score is smaller than that of the existing approach, and the detection can be performed with higher scores.

Next, we extended the frame-level feature map aggregation approach to dense feature map enhancement at the element-level. Focusing on the element-level makes it possible to aggregate for object misalignment flexibly. However, element-level aggregation is generally time-consuming. Thus, we proposed a video sparse transformer that considers video characteristics and collects information sparsely to solve this problem. Our experiment shows that sparse element-level aggregation effectively improves detection accuracy while reducing computation time and memory. Moreover, experiments show that compared with state-of-the-art methods, the proposed method significantly reduces the problems of false-negative and false-positive detection, which are challenges of video object detection.

Prediction-based Feature Map Enhancement. There are memory limitations in situations where real-time processing and online detectors are required, such as robotics. Therefore, applying feature map aggregation methods with external memory in such an environment is sometimes impractical. For this reason, we proposed prediction-based approaches that utilize future information during training to enhance the feature map. The prediction-based feature map enhancement method is based on an existing recurrent neural network detector and learns features such as the motion of objects through future prediction. We have proposed two different perspectives for prediction: the next and the next several frames. In the next frame prediction, we proposed to let the model predict the feature map of the next frame at the same time as detection. We also pointed out that this method retains the generated next-frame feature map, so it is possible to skip extracting features from the following frame and increase the processing speed by performing detection from it. We proposed an adaptive scheduler to increase the processing speed without sacrificing accuracy and showed its effectiveness. For the next several frames prediction, we proposed to learn to predict successive ten future frames in advance and then transfer the model to the detection task. Because of the diversity of videos, it is not easy to make reliable successive future frame predictions. Thus we introduced stochastic future forecasts and showed their effectiveness in video object detection. We showed that prediction could improve the detection performance without increasing the model size.

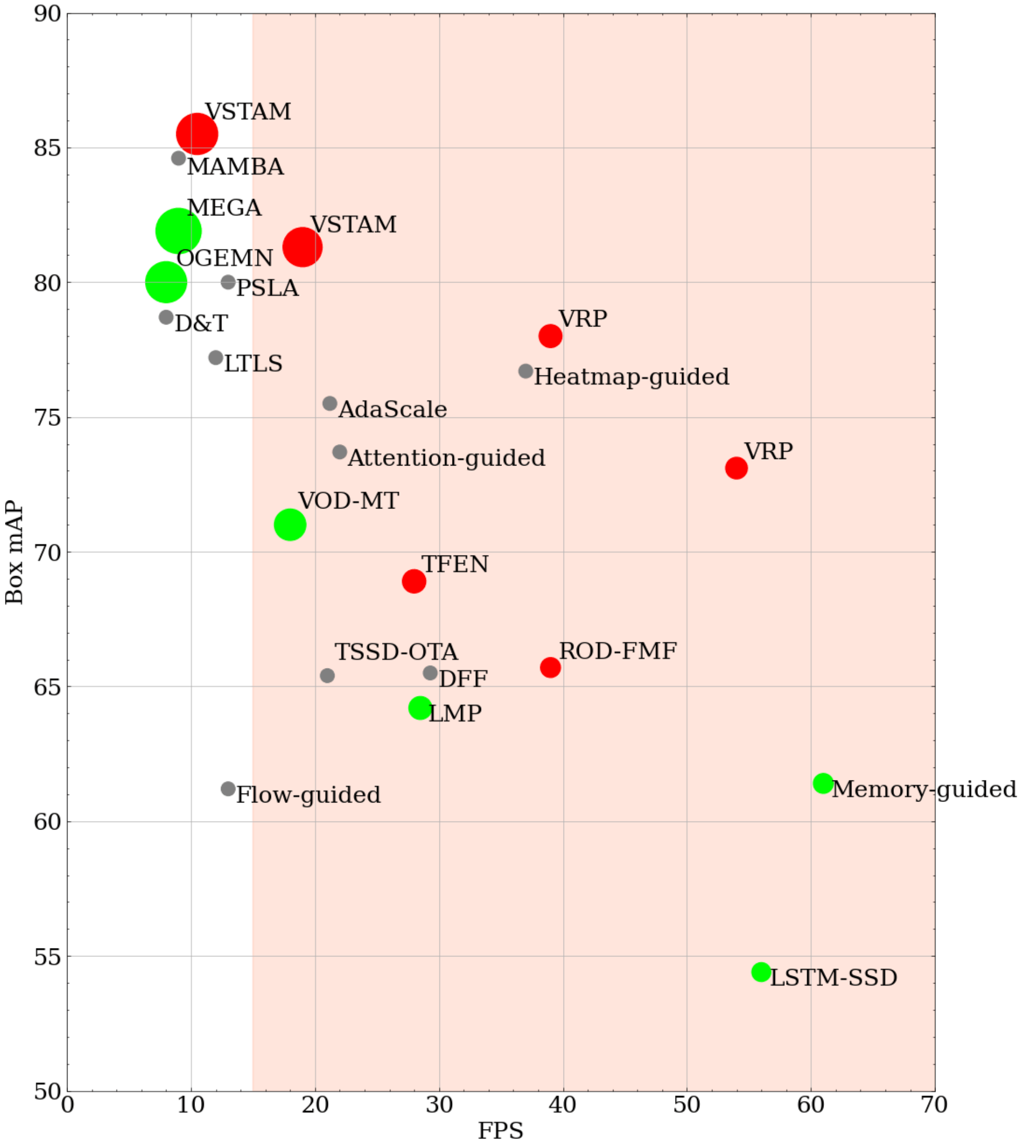


Figure 6.1: Accuracy-speed trade-off across various online detectors on ImageNet VID `val` (our methods are plotted in red). Multiple points for our same-named method show the results when using different backbones. The size of the marker indicates the model size (GiB). Measured competitive methods are shown in green, and methods with unknown model sizes are shown in gray. Also, the processing time of LSTM-SSD [110] and Memory-guided [111] was reported on a smartphone in the papers. LTLS uses high-end GPUs for speedup; we re-measured them on a GTX 2080 Ti using their implementation. The red area shows the general real-time processing performance.

6.2 Future Work

We plan to extend our work to the following for more robust to apparent changes with time:

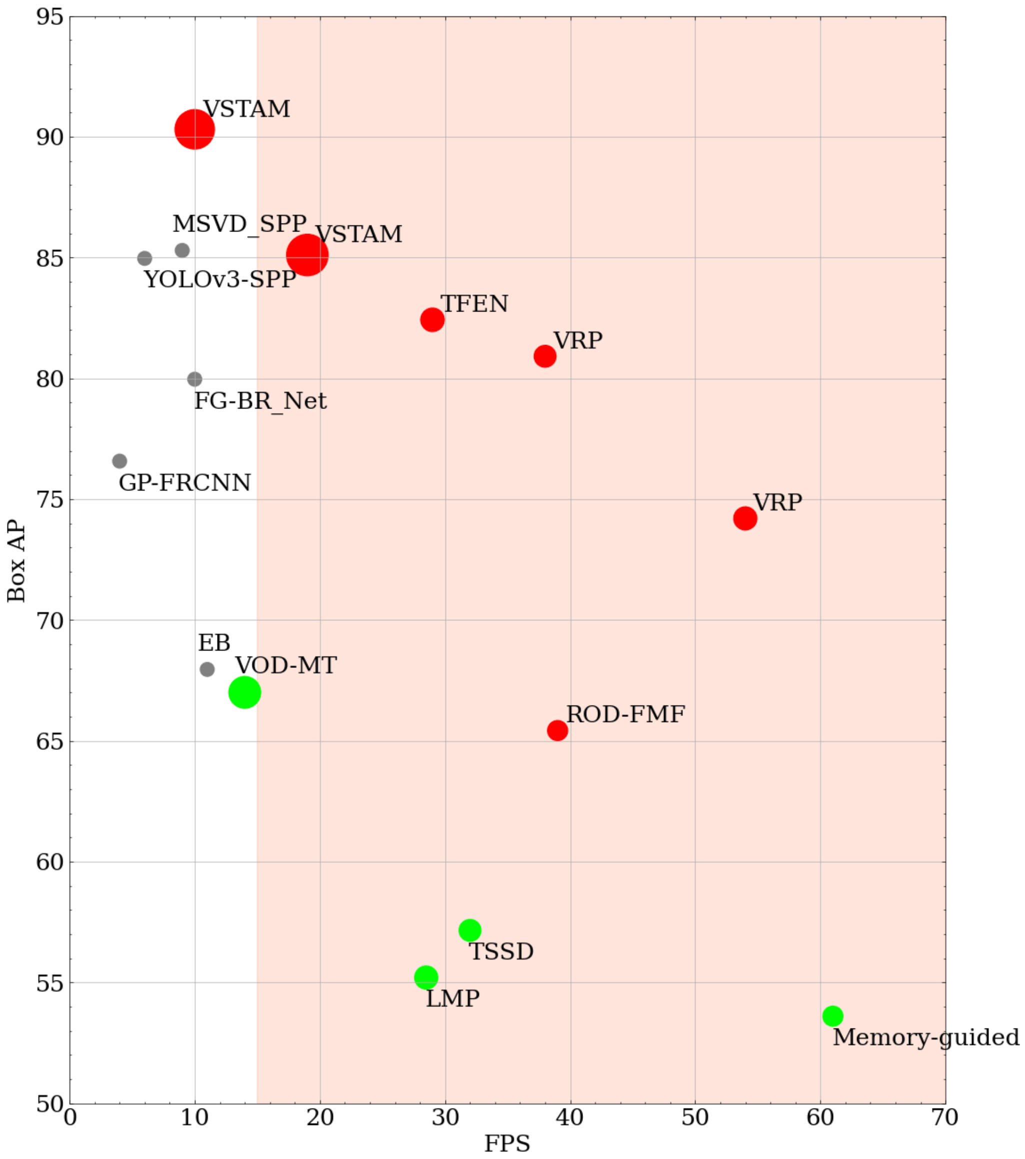


Figure 6.2: Accuracy-speed trade-off across various online detectors on UA-DETRAC `test` (our methods are plotted in red). Multiple points for our same-named method show the results when using different backbones. The size of the marker indicates the model size (GiB). Measured competitive methods are shown in green, and methods with unknown model sizes are shown in gray. The performance of TSSD [26] and VOD-MT [86] is measured by ourselves. The red area shows the general real-time processing performance.

- **Instantaneous class switching:** One issue that arises due to the apparent changes with time is the class false-positive detection, where the detected object's class is switched instantaneously. The element-level feature aggregation shows that it significantly reduces the number of false-negative and background false-positive detections. However, class false-positive detections still remain, caused by an instantaneous change to another class. The feature map enhancement method cannot guarantee the temporal consistency of the detection results. Therefore, it is necessary to incorporate tracking methods [25, 120] to keep the same class as the same object.

- **Discontinuous scene changes with time:** For object detection in more diverse videos, it is necessary to improve the robustness against discontinuous temporal changes in appearance. The current datasets for video object detection [127, 134, 176, 188] mainly consist of one continuous scene for each video clip. In other words, the content does not change within a video clip. Therefore, most video object detection methods have been proposed based on the assumption that the past and current frames are the same scenes. However, since videos in the real world are diverse and often switch to different scenes such as TV and movies, it is necessary to consider the discontinuity of scenes.

  It is challenging to cope with discontinuous scene changes by simply relying on past information. Figure 6.3 shows the accuracy change of scene switching when 300 consecutive frames are connected randomly as one video clip on the ImageNet VID dataset. When the accuracy of the previous scene is used as a reference, the accuracy of the TSSD, which propagates temporal information by a recurrent

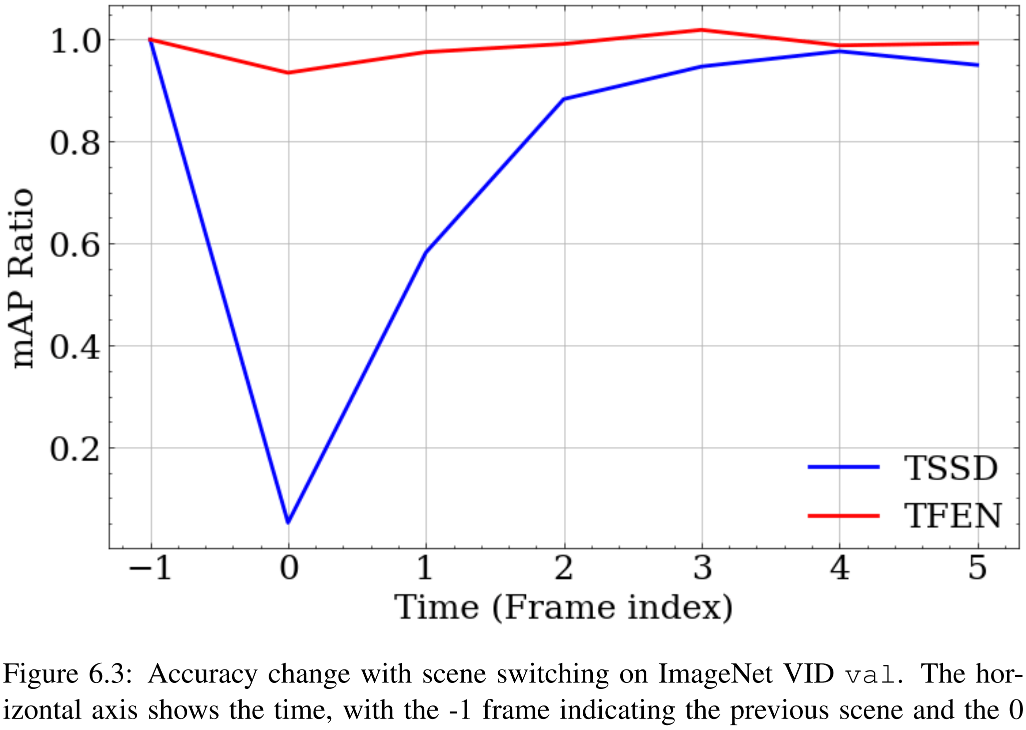


Figure 6.3: Accuracy change with scene switching on ImageNet VID `val`. The horizontal axis shows the time, with the -1 frame indicating the previous scene and the 0 to the fifth frame indicating the frames in the new scene. The vertical axis shows the accuracy ratio of each frame when the accuracy of the last scene is set to 1. We can see that the accuracy of TSSD, which does not take into account the decision to use past information, drops significantly when the scene switches to a new one.

neural network, is significantly decreased as a new scene starts. On the other hand, TFEN, which decides which frame to focus on sequentially for multiple past frames, is less affected. Thus, it is difficult to cope with discontinuous changes in a video by simply using past information.

To deal with discontinuous scene changes more flexibly, it is necessary to recognize the relationship between past and current frames. By recognizing the current and past scenes, we can decide whether to use the past temporal information or not. To consider the relation, it is advantageous to incorporate a video scene detection task [133], which divides a video into semantic scene clips. We believe that we can deal with discontinuous scene changes by recognizing the scene and switching the information used for detection.

References

[1] Sandra Aigner and Marco Körner. Futuregan: Anticipating the future frames of video sequences using spatio-temporal 3d convolutions in progressively growing gans. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4216:3–11, 2019.

[2] aim uofa. Adelaidet. `https://github.com/aim-uofa/AdelaiDet`.

[3] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. 33:9758–9770, 2020.

[4] Ali Athar, Sabarinath Mahadevan, Aljosa Osep, Laura Leal-Taixé, and Bastian Leibe. Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In *Proceedings of European Conference on Computer Vision*, pages 158–177. Springer, 2020.

[5] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. In *Proceedings of the International Conference on Learning Representations*, 2018.

[6] Nicolas Ballas, Li Yao, Christopher Joseph Pal, and Aaron C. Courville. Delving deeper into convolutional networks for learning video representations. In *Proceedings of the International Conference on Learning Representations*, 2016.

[7] Xinzhu Bei, Yanchao Yang, and Stefano Soatto. Learning semantic-aware dynamics for video prediction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 902–912, 2021.

[8] Gedas Bertasius and Lorenzo Torresani. Classifying, segmenting, and tracking object instances in video with mask propagation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 9739–9748, 2020.

[9] Gedas Bertasius, Lorenzo Torresani, and Jianbo Shi. Object detection in video with spatiotemporal sampling networks. In *Proceedings of European Conference on Computer Vision*, pages 331–346. Springer, 2018.

[10] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *Proceedings of IEEE International Conference on Image Processing*, pages 3464–3468, 2016.

[11] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *Proceedings of IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pages 1–6, 2017.

[12] Daniel Bolya, Sean Foley, James Hays, and Judy Hoffman. Tide: A general toolbox for identifying object detection errors. In *Proceedings of European Conference on Computer Vision*, pages 558–573. Springer, 2020.

[13] Wonmin Byeon, Qin Wang, Rupesh Kumar Srivastava, and Petros Koumoutsakos. Contextvp: Fully context-aware video prediction. In *Proceedings of European Conference on Computer Vision*, pages 753–769. Springer, 2018.

[14] Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *Proceedings of the IEEE International Conference on Computer Vision*, page 3361–3369, 2015.

[15] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 6154–6162, 2017.

[16] Jiale Cao, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. Sipmask: Spatial information preservation for fast image and video instance segmentation. In *Proceedings of European Conference on Computer Vision*, pages 1–18. Springer, 2020.

[17] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Workshops in Conjunction with IEEE International Conference on Computer Vision*, 2019.

[18] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[19] Lluis Castrejon, Nicolas Ballas, and Aaron Courville. Improved conditional vrnns for video prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7608–7617, 2019.

[20] Mohamed Chaabane, Ameni Trabelsi, Nathaniel Blanchard, and Ross Beveridge. Looking ahead: Anticipating pedestrians crossing with future frames prediction. In *Proceedings of IEEE Winter Conference on Applications of Computer Vision*, pages 2297–2306, 2020.

[21] Mohamed Chaabane, Peter Zhang, J Ross Beveridge, and Stephen O'Hara. Deft: Detection embeddings for tracking. *arXiv preprint arXiv:2102.02267*, 2021.

[22] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019.

[23] Kai Chen, Jiaqi Wang, Shuo Yang, Xingcheng Zhang, Yuanjun Xiong, Chen Change Loy, and Dahua Lin. Optimizing video object detection via a scale-time lattice. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 7814–7823, 2018.

[24] Long Chen, Haizhou Ai, Zijie Zhuang, and Chong Shang. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In *Proceedings of IEEE International Conference on Multimedia & Expo*, pages 1–6, 2018.

[25] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 8126–8135, 2021.

[26] Xingyu Chen, Zhengxing Wu, and Junzhi Yu. Tssd: Temporal single-shot detector based on attention and lstm. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–9, 2018.

[27] Xingyu Chen, Junzhi Yu, and Zhengxing Wu. Temporally identity-aware ssd with attentional lstm. *IEEE Transactions on Cybernetics*, 50(6):2674–2686, 2020.

[28] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[29] Yihong Chen, Yue Cao, Han Hu, and Liwei Wang. Memory enhanced global-local aggregation for video object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 10337–10346, 2020.

[30] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[31] Ting-Wu Chin, Ruizhou Ding, and Diana Marculescu. Adascale: Towards real-time video object detection using adaptive scaling. In *Proceedings of Machine Learning and Systems*, 2019.

[32] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3029–3037, 2015.

[33] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, 2020.

[34] Daniel Cores, Manuel Mucientes, and Vıctor M Brea. Roi feature propagation for video object detection. In *Proceedings of European Conference on Artificial Intelligence*, 2020.

[35] Nikolaus Correll, Kostas E. Bekris, Dmitry Berenson, Oliver Brock, Albert J. Causo, Kris K. Hauser, Kei Okada, Alberto Rodriguez, Joseph M. Romano, and Peter R. Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15:172–188, 2018.

[36] Yiming Cui, Liqi Yan, Zhiwen Cao, and Dongfang Liu. Tf-blender: Temporal feature blender for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8138–8147, 2021.

[37] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Proceedings of International Conference on Neural Information Processing Systems*, page 379–387, 2016.

[38] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 764–773, 2017.

[39] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[40] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of European Conference on Computer Vision*, pages 720–736. Springer, 2018.

[41] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision. *arXiv preprint arXiv:2006.13256*, 2020.

[42] Hanming Deng, Yang Hua, Tao Song, Zongpu Zhang, Zhengui Xue, Ruhui Ma, Neil Robertson, and Haibing Guan. Object guided external memory network for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6678–6687, 2019.

[43] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[44] Jiajun Deng, Yingwei Pan, Ting Yao, Wengang Zhou, Houqiang Li, and Tao Mei. Relation distillation networks for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7023–7032, 2019.

[45] Emily Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In *Proceedings of International Conference on Neural Information Processing Systems*, page 4414–4423, 2017.

[46] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[47] Piotr Dollár, Ron D. Appel, Serge J. Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, 2014.

[48] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, 2009.

[49] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

[50] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian. The unmanned aerial vehicle benchmark: Object detection and tracking. In *Proceedings of European Conference on Computer Vision*, pages 370–386. Springer, 2018.

[51] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6569–6578, 2019.

[52] Brendan Duke, Abdalla Ahmed, Christian Wolf, Parham Aarabi, and Graham W. Taylor. Sstvos: Sparse spatiotemporal transformers for video object segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5912–5921, 2021.

[53] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[54] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3057–3065, 2017.

[55] Pedro F Felzenszwalb, Ross B Girshick, and David McAllester. Cascade object detection with deformable part models. In *Proceedings of IEEE Computer society conference on computer vision and pattern recognition*, pages 2241–2248, 2010.

[56] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3636–3645, 2017.

[57] J. Ferryman and A. Shahrokni. Pets2009: Dataset and challenge. In *Proceedings of IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pages 1–6, 2009.

[58] Yang Fu, Linjie Yang, Ding Liu, Thomas S. Huang, and Humphrey Shi. Compfeat: Comprehensive feature aggregation for video instance segmentation. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2021.

[59] Zhihang Fu, Yaowu Chen, Hongwei Yong, Rongxin Jiang, Lei Zhang, and Xian-Sheng Hua. Foreground gating and background refining network for surveillance object detection. *IEEE Transactions on Image Processing*, 28(12):6077–6090, 2019.

[60] Masato Fujitake, Makito Inoue, and Takashi Yoshimi. Development of an automatic tracking camera system integrating image processing and machine learning. *Journal of Robotics and Mechatronics*, 33(6):1303–1314, 2021.

[61] Masato Fujitake and Akihiro Sugimoto. Temporal feature enhancement network with external memory for object detection in surveillance video. In *Proceedings of International Conference on Pattern Recognition*, pages 7684–7691, 2020.

[62] Masato Fujitake and Akihiro Sugimoto. Real-time object detection by feature map forecast for live streaming video. In *Proceedings of IEEE International Conference on Multimedia & Expo*, 2021.

[63] Masato Fujitake and Akihiro Sugimoto. Video representation learning through prediction for online object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision Workshops*, pages 530–539, 2022.

[64] Masato Fujitake and Takashi Yoshimi. Estimation system of construction equipment from field image by combination learning of its parts. In *Proceedings of Asian Control Conference*, pages 1672–1676, 2017.

[65] Michael Fulton, Jungseok Hong, Md Jahidul Islam, and Junaed Sattar. Robotic detection of marine litter using deep visual detection models. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 5752–5758, 2019.

[66] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Spatio-temporal closed-loop object detection. *IEEE Transactions on Image Processing*, 26(3):1253–1263, 2017.

[67] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[68] Qichuan Geng, Hong Zhang, Na Jiang, Xiaojuan Qi, Liangjun Zhang, and Zhong Zhou. Object-aware feature aggregation for video object detection. *arXiv preprint arXiv:2010.12573*, 2020.

[69] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 12046–12055, 2019.

[70] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, page 1440–1448, 2015.

[71] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, page 580–587, 2014.

[72] Tao Gong, Kai Chen, Xinjiang Wang, Qi Chu, Feng Zhu, Dahua Lin, Nenghai Yu, and Huamin Feng. Temporal roi align for video object recognition. In *Proceedings of AAAI Conference on Artificial Intelligence*, volume 35, pages 1442–1450, 2021.

[73] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[74] Chaoxu Guo, Bin Fan, Jie Gu, Qian Zhang, Shiming Xiang, Veronique Prinet, and Chunhong Pan. Progressive sparse local attention for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3909–3918, 2019.

[75] Liang Han, Pichao Wang, Zhaozheng Yin, Fan Wang, and Hao Li. Exploiting better feature aggregation for video object detection. In *Proceedings of ACM International Conference on Multimedia*, pages 1469–1477, 2020.

[76] Mingfei Han, Yali Wang, Xiaojun Chang, and Yu Qiao. Mining inter-video proposal relations for video object detection. In *Proceedings of European Conference on Computer Vision*, pages 431–446. Springer, 2020.

[77] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S. Huang. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*, 2016.

[78] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[79] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[80] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[81] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[82] Lu He, Qianyu Zhou, Xiangtai Li, Li Niu, Guangliang Cheng, Xiao Li, Wenxuan Liu, Yunhai Tong, Lizhuang Ma, and Liqing Zhang. End-to-end video object detection with

spatial-temporal transformers. In *Proceedings of ACM International Conference on Multimedia*, page 1507–1516, 2021.

[83] Congrui Hetang, Hongwei Qin, Shaohui Liu, and Junjie Yan. Impression network for video object detection. *arXiv preprint arXiv:1712.05896*, 2017.

[84] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[85] Hamid Izadinia, Imran Saleemi, Wenhui Li, and Mubarak" Shah. (mp)2t: Multiple people multiple parts tracker. In *Proceedings of European Conference on Computer Vision*, pages 100–114. Springer, 2012.

[86] Kim Jaekyum, Koh Junho, Lee Byeongwon, Yang Seungji, and Jun Won Choi. Video object detection using object ’ s motion context and spatio-temporal feature aggregation. In *Proceedings of International Conference on Pattern Recognition*, pages 1604–1610, 2020.

[87] Zhengkai Jiang, Peng Gao, Chaoxu Guo, Qian Zhang, Shiming Xiang, and Chunhong Pan. Video object detection with locally-weighted deformable neighbors. In *Proceedings of AAAI Conference on Artificial Intelligence*, volume 33, pages 8529–8536, 2019.

[88] Zhengkai Jiang, Yu Liu, Ceyuan Yang, Jihao Liu, Peng Gao, Qian Zhang, Shiming Xiang, and Chunhong Pan. Learning where to focus for efficient video object detection. In *Proceedings of European Conference on Computer Vision*, pages 18–34. Springer, 2020.

[89] Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. Object detection in videos with tubelet proposal networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 889–897, 2017.

[90] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, and Wanli Ouyang. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2896–2907, 2018.

[91] Kai Kang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Object detection from video tubelets with convolutional neural networks. pages 817–825, 2016.

[92] Kwang-Ju Kim, Pyong-Kun Kim, Yun-Su Chung, and Doo-Hyun Choi. Performance enhancement of yolov3 by adding prediction layers with spatial pyramid pooling for vehicle detection. In *Proceedings of IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pages 1–6, 2018.

[93] Kwang-Ju Kim, Pyong-Kun Kim, Yun-Su Chung, and Doo-Hyun Choi. Multi-scale detector for accurate vehicle detection in traffic surveillance data. *IEEE Access*, 7:78311–78319, 2019.

[94] Junho Koh, Jaekyum Kim, Younji Shin, Byeongwon Lee, Seungji Yang, and Jun Won Choi. Joint representation of temporal image sequences and object motion for video object detection. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2021.

[95] Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, and Boqing Gong. Movinets: Mobile video networks for efficient video recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 16020–16030, 2021.

[96] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Proceedings of International Conference on Neural Information Processing Systems*, 25:1097–1105, 2012.

[97] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of European Conference on Computer Vision*, pages 734–750. Springer, 2018.

[98] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese cnn for robust target association. In *Workshops in Conjunction with IEEE Conference on Computer Vision and Pattern Recognition*, pages 33–40, 2016.

[99] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

[100] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 667–676, 2017.

[101] Suichan Li and Feng Chen. 3d-detnet: a single stage video-based vehicle detector. In *Third International Workshop on Pattern Recognition*, volume 10828, pages 60 – 66. SPIE, 2018.

[102] Chung-Ching Lin, Ying Hung, Rogerio Feris, and Linglin He. Video instance segmentation tracking with a modified vae architecture. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 13147–13157, 2020.

[103] Lijian Lin, Haosheng Chen, Honglun Zhang, Jun Liang, Yu Li, Ying Shan, and Hanzi Wang. Dual semantic fusion network for video object detection. In *Proceedings of ACM International Conference on Multimedia*, pages 1855–1863, 2020.

[104] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *Proceedings of the International Conference on Learning Representations*, 2014.

[105] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 936–944, 2017.

[106] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[107] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[108] Dongfang Liu, Yiming Cui, Yingjie Chen, Jiyong Zhang, and Bin Fan. Video object detection for autonomous driving: Motion-aid feature calibration. *Neurocomputing*, 409:1–11, 2020.

[109] Dongfang Liu, Yiming Cui, Wenbo Tan, and Yingjie Chen. Sg-net: Spatial granularity network for one-stage video instance segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 9816–9825, 2021.

[110] Mason Liu and Menglong Zhu. Mobile video object detection with temporally-aware feature maps. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5686–5695, 2018.

[111] Mason Liu, Menglong Zhu, Marie White, Yinxiao Li, and Dmitry Kalenichenko. Looking fast and slow: Memory-guided mobile video object detection. *arXiv preprint arXiv:1903.10172*, 2019.

[112] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *Proceedings of European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[113] Wei Liu, Shengcai Liao, Weiqiang Ren, Weidong Hu, and Yinan Yu. High-level semantic feature detection: A new perspective for pedestrian detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5182–5191, 2019.

[114] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations*, 2018.

[115] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *Proceedings of the International Conference on Learning Representations*, 2017.

[116] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[117] Hao Luo, Wenxuan Xie, Xinggang Wang, and Wenjun Zeng. Detect or track: Towards cost-effective video object detection/tracking. In *Proceedings of AAAI Conference on Artificial Intelligence*, volume 33, pages 8803–8810, 2019.

[118] Hui Lv, Chuanwei Zhou, Zhen Cui, Chunyan Xu, Yong Li, and Jian Yang. Localizing anomalies from weakly-labeled videos. *IEEE Transactions on Image Processing*, 30:4505–4515, 2021.

[119] Siwei Lyu, Ming-Ching Chang, Dawei Du, Wenbo Li, Yi Wei, Marco Del Coco, Pierluigi Carcagnì, Arne Schumann, Bharti Munjal, Dinh-Quoc-Trung Dang, Doo-Hyun Choi, Erik Bochinski, Fabio Galasso, Filiz Bunyak, Guna Seetharaman, Jang-Woon Baek, Jong Taek Lee, Kannappan Palaniappan, Kil-Taek Lim, Kiyoung Moon, Kwang-Ju Kim, Lars Sommer, Meltem Brandlmaier, Min-Sung Kang, Moongu Jeon, Noor M. Al-Shakarji, Oliver Acatay, Pyong-Kun Kim, Sikandar Amin, Thomas Sikora, Tien Dinh, Tobias Senst, Vu-Gia-Hy Che, Young-Chul Lim, Young-min Song, and Yun-Su Chung. Ua-detrac 2018: Report of avss2018 & iwt4s challenge on advanced traffic monitoring. In *Proceedings of IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pages 1–6, 2018.

[120] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. *arXiv preprint arXiv:2101.02702*, 2021.

[121] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Proceedings of International Conference on Neural Information Processing Systems*, page 2863–2871, 2015.

[122] Jongchan Park, Sanghyun Woo, Joon-Young Lee, and In-So Kweon. Bam: Bottleneck attention module. In *Proceedings of British Machine Vision Conference*, 2018.

[123] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of*

International Conference on Neural Information Processing Systems, pages 8026–8037, 2019.

[124] Hughes Perreault, Guillaume-Alexandre Bilodeau, Nicolas Saunier, and Maguelonne Héritier. Spotnet: Self-attention multi-task network for object detection. In *Proceedings of Conference on Computer and Robot Vision*, pages 230–237, 2020.

[125] Hughes Perreault, Guillaume-Alexandre Bilodeau, Nicolas Saunier, and Maguelonne Héritier. Ffavod: Feature fusion architecture for video object detection. *Pattern Recognition Letters*, 151:294–301, 2021.

[126] Hughes Perreault, Maguelonne Héritier, Pierre Gravel, Guillaume-Alexandre Bilodeau, and Nicolas Saunier. Rn-vid: A feature fusion architecture for video object detection. In *International Conference on Image Analysis and Recognition*, pages 125–138, 2020.

[127] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5296–5305, 2017.

[128] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[129] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 6517–6525, 2017.

[130] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[131] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[132] Mikel Rodriguez, Saad Ali, and Takeo Kanade. Tracking in unstructured crowded scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1396, 2009.

[133] Daniel Rotman, Dror Porat, Gal Ashour, and Udi Barzelay. Optimally grouped deep features using normalized cost for video scene detection. In *Proceedings of ACM International Conference on Multimedia Retrieval*, pages 187–195, 2018.

[134] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[135] Edmund J Sadgrove, Greg Falzon, David Miron, and David W Lamb. Real-time object detection in agricultural/remote environments using the multiple-expert colour feature extreme learning machine (mec-elm). *Computers in Industry*, 98:183–191, 2018.

[136] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[137] Han Shi, Jiahui Gao, Xiaozhe Ren, Hang Xu, Xiaodan Liang, Zhenguo Li, and James T Kwok. Sparsebert: Rethinking the importance analysis in self-attention. In *Proceedings of International Conference on Machine Learning*. PMLR, 2021.

[138] Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Proceedings of International Conference on Neural Information Processing Systems*, pages 802–810, 2015.

[139] Mykhailo Shvets, Wei Liu, and Alexander C Berg. Leveraging long-range temporal relationships between proposals for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9756–9764, 2019.

[140] Amin Sikandar and Galasso Fabio. Geometric proposals for faster r-cnn. In *Proceedings of IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pages 1–6, 2017.

[141] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations*, 2015.

[142] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *Proceedings of International Conference on Machine Learning*, pages 843–852. PMLR, 2015.

[143] Guanxiong Sun, Yang Hua, Guosheng Hu, and Neil Robertson. Mamba: Multi-level aggregation via memory bank for video object detection. In *Proceedings of AAAI Conference on Artificial Intelligence*, volume 35, pages 2620–2627, 2021.

[144] ShiJie Sun, Naveed Akhtar, HuanSheng Song, Ajmal Mian, and Mubarak Shah. Deep affinity network for multiple object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):104–119, 2019.

[145] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9627–9636, 2019.

[146] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Herve Jegou. Fixing the train-test resolution discrepancy. In *Advances in Neural Information Processing Systems*, volume 32, pages 8252–8262, 2019.

[147] Ameni Trabelsi, Mohamed Chaabane, Nathaniel Blanchard, and Ross Beveridge. A pose proposal and refinement network for better 6d object pose estimation. In *Proceedings of IEEE Winter Conference on Applications of Computer Vision*, pages 2382–2391, 2021.

[148] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5552–5561, 2019.

[149] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1526–1535, 2018.

[150] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[151] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of International Conference on Neural Information Processing Systems*, pages 5998–6008, 2017.

[152] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *Proceedings of the International Conference on Learning Representations*, 2017.

[153] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. Feelvos: Fast end-to-end embedding learning for video object segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 9481–9490, 2019.

[154] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-object tracking and

segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 7942–7951, 2019.

[155] Jiangliu Wang, Jianbo Jiao, and Yun-Hui Liu. Self-supervised video representation learning by pace prediction. In *Proceedings of European Conference on Computer Vision*, pages 504–521. Springer, 2020.

[156] Li Wang, Yao Lu, Hong Wang, Yingbin Zheng, Hao Ye, and Xiangyang Xue. Evolving boxes for fast vehicle detection. In *Proceedings of IEEE International Conference on Multimedia & Expo*, pages 1135–1140, 2017.

[157] Limin Wang, Zhan Tong, Bin Ji, and Gangshan Wu. Tdn: Temporal difference networks for efficient action recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1895–1904, 2021.

[158] Shiyao Wang, Yucong Zhou, Junjie Yan, and Zhidong Deng. Fully motion-aware network for video object detection. In *Proceedings of European Conference on Computer Vision*, pages 542–557. Springer, 2018.

[159] Tao Wang, Ning Xu, Kean Chen, and Weiyao Lin. End-to-end video instance segmentation via spatial-temporal graph neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10797–10806, 2021.

[160] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[161] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 8741–8750, 2021.

[162] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In *Proceedings of European Conference on Computer Vision*, pages 107–122. Springer, 2020.

[163] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[164] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. UA-DETRAC: A new benchmark and

protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, 193:102907, 2020.

[165] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *Proceedings of IEEE International Conference on Image Processing*, pages 3645–3649, 2017.

[166] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of European Conference on Computer Vision*, pages 3–19. Springer, 2018.

[167] Haiping Wu, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Sequence level semantics aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9217–9225, 2019.

[168] Lingyun Wu, Zhiqiang Hu, Yuanfeng Ji, Ping Luo, and Shaoting Zhang. Multi-frame collaboration for effective endoscopic video polyp detection via spatial-temporal feature transformation. In *Proceedings of Medical Image Computing and Computer Assisted Intervention*, pages 302–312, 2021.

[169] Yue Wu, Rongrong Gao, Jaesik Park, and Qifeng Chen. Future video synthesis with object motion prediction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5539–5548, 2020.

[170] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[171] Fanyi Xiao and Yong Jae Lee. Video object detection with an aligned spatial-temporal memory. In *Proceedings of European Conference on Computer Vision*, pages 485–501. Springer, 2018.

[172] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.

[173] Mengde Xu, Zheng Zhang, Han Hu, Jianfeng Wang, Lijuan Wang, Fangyun Wei, Xiang Bai, and Zicheng Liu. End-to-end semi-supervised object detection with soft teacher. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3060–3069, 2021.

[174] Yihong Xu, Aljosa Osep, Yutong Ban, Radu Horaud, Laura Leal-Taixé, and Xavier Alameda-Pineda. How to train your deep multi-object tracker. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 6787–6796, 2020.

[175] Zhujun Xu, Emir Hrustic, and Damien Vivet. Centernet heatmap propagation for real-time video object detection. In *Proceedings of European Conference on Computer Vision*, pages 220–234. Springer, 2020.

[176] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5188–5197, 2019.

[177] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K Katsaggelos. Efficient video object segmentation via network modulation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 6499–6507, 2018.

[178] Shusheng Yang, Yuxin Fang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Crossover learning for fast online video instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8043–8052, 2021.

[179] Yanni Yang, Huansheng Song, Shijie Sun, Yan Chen, Xinyao Tang, and Qin Shi. A feature temporal attention based interleaved network for fast video object detection. *Journal of Ambient Intelligence and Humanized Computing*, 2021.

[180] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2636–2645, 2020.

[181] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *Proceedings of the International Conference on Learning Representations*, 2016.

[182] Wei Yu, Y. Lu, S. Easterbrook, and S. Fidler. Efficient and information-preserving future frame prediction and beyond. In *Proceedings of the International Conference on Learning Representations*, 2020.

[183] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In *Proceedings of International Conference on Neural Information Processing Systems*, 2020.

[184] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, pages 1–19, 2021.

[185] Guangxiang Zhao, Junyang Lin, Zhiyuan Zhang, Xuancheng Ren, Qi Su, and Xu Sun. Explicit sparse transformer: Concentrated attention through explicit selection. *arXiv preprint arXiv:1912.11637*, 2019.

[186] Qijie Zhao, Tao Sheng, Yongtao Wang, Feng Ni, and Ling Cai. Cfenet: An accurate and efficient single-shot object detector for autonomous driving. *arXiv preprint arXiv:1806.09790*, 2018.

[187] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *Proceedings of European Conference on Computer Vision*, pages 474–490. Springer, 2020.

[188] Pengfei Zhu, Dawei Du, Longyin Wen, Xiao Bian, Haibin Ling, Qinghua Hu, Tao Peng, Jiayu Zheng, Xinyao Wang, Yue Zhang, et al. Visdrone-vid2019: The vision meets drone object detection in video challenge results. In *Workshops in Conjunction with IEEE International Conference on Computer Vision*, pages 227–235, 2019.

[189] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. Towards high performance video object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 7210–7218, 2018.

[190] Xizhou Zhu, Jifeng Dai, Xingchi Zhu, Yichen Wei, and Lu Yuan. Towards high performance video object detection for mobiles. *arXiv preprint arXiv:1804.05830*, 2018.

[191] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417, 2017.

[192] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4141–4150, 2017.

[193] Zhifan Zhu and Zechao Li. Online video object detection via local and mid-range feature propagation. In *Proceedings of First International Workshop on Human-Centric Multimedia Analysis*, page 73–82, 2020.