REAL-TIME VIDEO OBJECT DETECTION WITH TEMPORAL FEATURE AGGREGATION

By

MEIHONG CHEN

Thesis Submitted to the University of Ottawa

in Partial Fulfillment of the Requirements for the Degree of

Master of Computer Science

with Concentration in Applied Artificial Intelligence

UNIVERSITY OF OTTAWA
School of Electrical Engineering and Computer Science

SEPTEMBER 2021

ABSTRACT

In recent years, various high-performance networks have been proposed for single-image object detection. An obvious choice is to design a video detection network based on state-of-the-art single-image detectors. However, video object detection is still challenging due to the lower quality of individual frames in a video, and hence the need to include temporal information for high-quality detection results. In this thesis, we design a novel interleaved architecture combining a 2D convolutional network and a 3D temporal network. We utilize Yolov3 as the base detector. To explore inter-frame information, we propose feature aggregation based on a temporal network. Our temporal network utilizes Appearance-preserving 3D convolution (AP3D) for extracting aligned features in the temporal dimension. Our multi-scale detector and multi-scale temporal network communicate at each scale and also across scales. The number of inputs of our temporal network can be either 4, 8, or 16 frames in this thesis and correspondingly we name our temporal network TemporalNet-4, TemporalNet-8 and TemporalNet-16. Our approach achieves 77.1% mAP (mean Average Precision) on ImageNet VID 2017 dataset with TemporalNet-4, where TemporalNet-16 achieves 80.9% mAP which is a competitive result on this video object detection benchmark. Our network is also real-time with a running time of 35ms/frame.

ACKNOWLEDGMENT

I would like to give my deepest gratitude to my supervisor Professor Jochen Lang for his patient guidance and continuous encouragement throughout the recent three years. He always give quich response to my questions and cares about my mood.

DECLARATION

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Ottawa's products or services.Internal or personal use of this material is permitted.If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to IEEE website to learn how to obtain a License from RightsLink.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AP3D	Appearance-Preserving 3D convolution
APM	Appearance-Preserving Module
C3D	Convolution 3D
ConvNet/CNN	Convolutional neural network
CV	Computer Vision
FC	Fully Connected
FCN	Fully Convolutional Networks
FP	False-Positive
FPN	Feature Pyramid Networks
FPS	Frames Per Second
GIoU	Generalized intersection over union
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoU	Intersection over Union
LSTM	Long Short-Term Memory
mAP	mean Average Precision
MHI	Motion History Image
NMS	Non-Maximum Suppression
PASCAL	Pattern Analysis, Statistical Modelling and Computational Learning
PR	precision/recall

PRU	Propagation and Refinement UnitPRU
R-C3D	Region Convolutional 3D Network
RCNN	Region-based CNN
RDN	Relation Distillation Networks
ReLU	Rectified linear unit
ResNet	Residual network
RoI	Region of Interest
RPN	Region proposal Network
SELSA	Sequence Level Semantics Aggregation
SGD	Stochastic gradient decent
SSD	Single Shot MultiBox Detector
STMM	Spatial-temporal memory module
SVM	Support Vector Machine
TP	True-Positive
VOC	Visual Object Classes
YOLO	You Only Look Once

Chapter 1

Introduction

Object detection [14] [15] refers to a task which consists of object classification and object localization. As a fundamental research topic in image analysis and understanding, object detection has a wide range of applications such as face recognition [16], violence prediction [17], and autonomous driving [18]. In recent years, increasingly elaborate Convolutional Neural Networks (CNNs) have been the state-of-the-art for object detection. There are mainly two branches: single image object detection and video object detection.

In recent years, some good single-image detectors emerged but video object detection is still challenging. Video detection is the basic task for a wide range of applications, such as pedestrian detection, intelligent surveillance, autonomous driving and video processing. Real-time performance is required in these applications as actions need to be triggered in a timely fashion. Especially, our thesis is motivated by video editing. Real-time semantic video editing requires a fast detection module in videos to label related content. Detection results contribute to understanding videos and further video processing such as applying visual effects and autonomous zoom-in. After investigation of these fields, we propose a real-time video object detection network.

1.1 Single Image Object Detection

The fundamental idea of an image object detector is to figure out the category and the position of specific objects in a picture. For example, the bounding boxes in Fig. 1.1 mark the locations of objects to be detected.

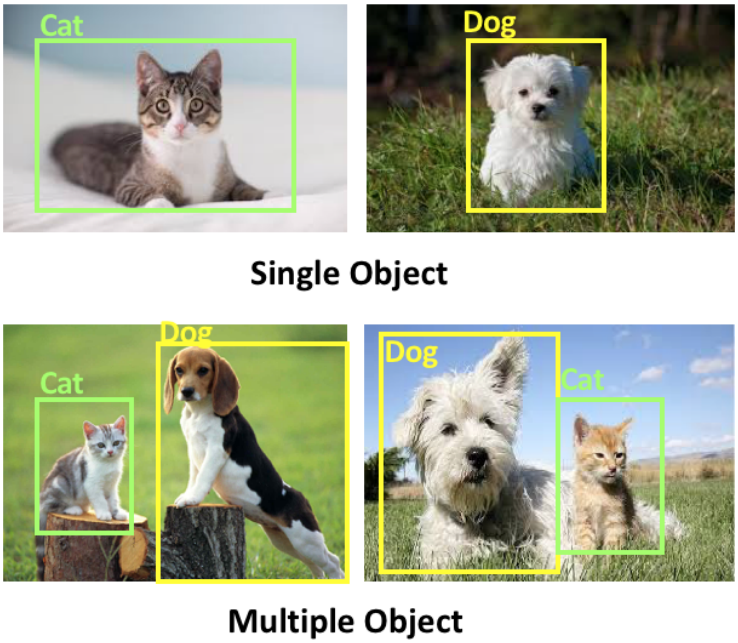


Fig. 1.1 Example of object detection.

The output of CNNs for object detection consists often of two branches [19]. One branch yields the classification result. Another branch yields the bounding boxes of objects to be detected. The conventional object detection method can usually be divided into three steps: first select candidate regions on a given image, then perform feature extraction on these regions, and finally train a classifier for classification. However, the accuracy of detection is decided by the choice of features. Manual feature extraction is a key limitation for conventional object detectors. The emergence of two-stage object detection network based on CNN replaces manual feature extraction with "machine learning". Fig. 1.2 shows in more detail the basic architecture of the two-stage object detector. Compared to manually-designed feature extraction, feature extraction using CNNs is adaptive to different detection tasks.

Conventional methods require effort to manually select features, require heuristic expertise and rely largely on manual feature selection. CNNs extract multiple features and calculate their combination for classification and regression, which is more efficient.

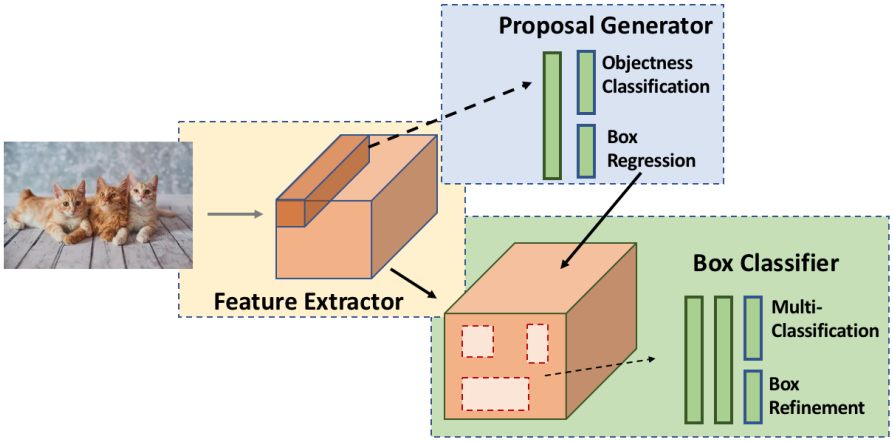


Fig. 1.2 Basic architecture of an object detection network.

Two-stage object detection is accurate but time-consuming. Therefore, fast one-stage detection networks have been proposed in recent years. There is a network for region proposals in two-stage detectors. For example, faster RCNN (Region-based CNN) [5] used RPN (Region Proposal Network) [5]. The subsequent convolution network optimizes the output of the proposal generator and runs a classifier on it. The two-stage detection networks is accurate because a network for RoI (Region of Interests) and another network for bounding boxes prediction are applied. While one-stage detection networks applies anchors for prediction of the corresponding cell. It calculate bounding boxes regression and simultaneously performs classification. One-stage networks can be fast as they complete localization and classification tasks in one network.

1.2 Video Object Detection

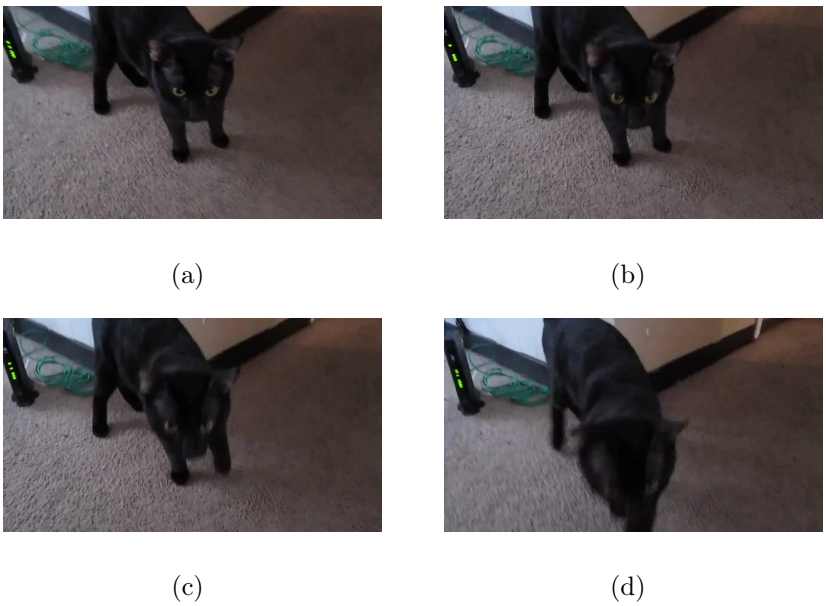
With the popularity of deep learning networks, a significant number of image detection networks are proposed with outstanding performance both in accuracy and speed. However,

video object detection is still challenging due to motion blur, out-of-focus image regions, rare poses, and jitters in the displayed video. In addition, data redundancy is another problem to consider when a detector is deployed on a video. In a video, each frame tells respective "words" and neighbor frames can be combined to tell "a story", which indicates frames are not separate but interconnected pieces. Fig. 1.3 shows consecutive frames in a video. With the information from frames (a)(b)(c), we can know the occluded shape in frame (d) is a cat. To be specific, frames in a video have a temporal order and are spatially related. The problem of feature extraction in a video is extracting spatial and temporal features from a image stream. Recent works [7, 20–23] utilize spatially and temporarily cross-frame features to reduce the negative influence of low-quality frames. Generally, video object detection algorithms are mainly based on feature aggregation between local and global frames; also, some of the detectors are based on object tracking [24].

In the past few years, obvious accuracy improvements can be witnessed with an increasing number of well-designed architectures. However, most of the architecture are too complicated to be fast enough for real-time processing of videos. Recently, Liu et al. [7, 25] proposed an efficient memory-guided network that runs with a lower computational budget and a smaller time cost. Apart from that, the concept "gist" [7] allows the neural network to 'watch' like a human. Minimal computation was reached by interleaving conventional feature extractor and lightweight 'gist' features. But the mAP (mean of Average Precision) of this small video detection network is only 61% on ImageNet VID dataset [23] for object detection tasks. Hence remains a challenge to design an efficient video detection network.

1.3 Thesis Approach

When watching a video, we can predict the next frame as neighbor frames are interrelated, and our predictions are often accurate. The prediction is based on our experience. For example, we know people will put lipstick on their mouth instead of their eyes. As deep



(a) (b)

(c) (d)

Fig. 1.3 Consecutive frames in a video with motion blur.

learning makes it possible for the machine to learn temporal relationships, it is also reasonable for the machine to make a prediction in video processing. In this thesis, the relationship between frames is exploited to enhance the features of the current frame based on our proposed feature aggregation method. Then bounding boxes are predicted by a state-of-the-art still-image detector. Our feature aggregation method is based on 3D convolution which is efficient in learning temporal information. By aggregating cross-frame features in the temporal dimension, a single-image object detection model can be ameliorated to accomplish video object detection tasks with competitive performance.

This thesis describes the details of our research exploring the application of machine learning methods to the challenge of object detection from videos. We utilize 2D convolution and 3D convolution networks in our work. The 3D convolution network is for feature aggregation in the temporal dimension and the 2D convolution network is for spatial features extraction. We utilize a multi-scale 3D network and multi-scale backbone. We allow our 3D network and 2D backbone to communicate at each scale. Specifically, we develop the feature aggregation policy to aggregate features extracted by 2D ConvNet (Convolution

Neural Network) using a 3D ConvNet. To decrease memory and computing requirements, we utilize one-stage Yolov3 [26] as the basic detector. To preserve appearance information in 3D convolution, we utilize AP3D (Appearance-Preserving 3D Convolution) as our basic 3D convolution module. We also apply octave convolution for speed and accuracy in our novel architecture. We conduct our evaluation on the ImageNet VID2017 dataset [23]. We achieve favorable results comparing with the single-image object detection baseline, and our performance is competitive compared to other online video detectors.

1.4 Research Questions

Accuracy is the primary concern in video detection, but computation time is also important because a real-time system is required in practical application. We address the following questions in this thesis:

1. We explore what design leads to an efficient temporal network for feature aggregation?
2. We aim to find an efficient combination of the temporal network with a 2D one-stage object detection network.
3. we investigate what design choices leads to an accurate real-time method.

1.4.1 Contributions

In seeking answers to the above research questions, we have arrived at the following contributions:

- We have designed an efficient multi-scale 3D convolution network based on AP3D for feature aggregation across frames. The temporal network aligns features of neighboring frames and enhances features of the current frame based on temporal information.
- We import channel attention into Octave convolution [11] and propose selective-frequency

octave convolution which is capable of adjusting the ratio of high-frequency and low-frequency features.

- We apply selective-frequency octave convolution to Darknet-53 (the backbone of Yolov3 [26]) and utilized the improved Darknet-53 as the backbone in our experiments.

- We design a multi-scale communication between our temporal network and the backbone. It allows multi-scale feature extraction and multi-scale feature aggregation.

1.5 Thesis Organization

This thesis consists of 6 chapters. We provide a brief summary below.

In Chapter 2, we first discuss related concepts such as the definition of and problems in object detection, and the basic concepts of detection networks. We provide an introduction to neural networks which are the basis of convolutional neural networks. We also describe online video detection. Then we review the most important and applicable CNN architectures in single-image and video object detection, respectively. We also describe the evaluation metrics used in this thesis.

In Chapter 3, we provide a detailed background of the building blocks of our network design. We provide an overview to Yolov3 and AP3D, respectively. We review octave convolution and 3D convolution as the fundamental knowledge for Chapter 4.

In Chapter 4, we introduce our model design by starting with the overview architecture of our video detection network. We provide the mathematical basis of our feature aggregation policy and selective-frequency octave convolution. We describe in detail each component of our work including the temporal network, feature extraction, and the detection module.

In Chapter 5, we describe our data set and experimental setup for training and evaluation as well as the experimental results. First, we describe our dataset, training procedure, hyper-parameters and loss function. Then, we provide a comparison of our proposed architecture

versus state-of-the-art video detection networks. We also provide a qualitative comparison of results between our work and other detection networks. We provide an ablation study to demonstrate the performance of the components that are proposed in this thesis.

Chapter 6 summarizes this thesis and states conclusions based on the results. Based on detection performance and computing time, we discuss the results and identify challenges and improvements. We discuss the limitation of our work and how the work could be improved and taken further in future work.

Chapter 2

Related Work

2.1 Introduction

In this chapter, we provide a basic overview of the theoretical concepts of object detection and CNN. We explain important terminology in object detection including methods of generating potential bounding boxes and optimizing predicted boxes. We describe convolutional networks which indicate the evolution of CNNs on the field of computer vision. Online video detection is defined in this chapter according to the definition of online action recognition. We introduce 3D convolution to show its ability in extracting temporal information from consecutive frames in a video. We review video detection networks and describe the four video detectors which are used for the comparison with our work.

2.2 Object Detection

In this section, we introduce important components of object detection and evaluation terminology.

2.2.1 Sliding Window and Selective Search

In object recognition and segmentation, the problem of locating possible objects is challenging. At present, different strategies are proposed to address it. RCNN architectures (RCNN [5], Fast RCNN [6], Faster RCNN [27]) are based on region proposals, YOLO [28] and SSD [29] are based on area division, and AttentionNet [30] is based on reinforcement learning. Today most object detection algorithms are based on region proposals. The basic method for region proposals can be a sliding window or selective search. We compare the selective search strategy with the conventional sliding window.

Sliding Window is a classic method that transforms the problem of detection into the problem of classification. The principle is to slide windows of different sizes and ratios (aspect ratio) across the entire picture at a certain step size and then utilize classifiers on the regions framed by these windows. A classifier scores the current region based on probability indicating the existence of specific targets. Finally, candidate boxes that have higher scores are marked. Obviously, this method has the disadvantage that an enormous amount of calculation is necessary. Due to target size uncertainty, different sized windows must be applied to generate candidate sub-regions, which requires a large number of calculations and high-quality hardware. Overall, the sliding window method is simple and easy to understand, but the global search of images with an enormous amount of windows results in low efficiency. Therefore, this method is difficult to adopt in detectors with real-time requirements.

The sliding window method is similar to exhaustive search, however, most of the sub-regions are meaningless background. Selective search improves computational efficiency by extracting sub-regions that are most likely to contain objects in a picture. The main point is to gradually merge sub-regions according to the similarity distribution in the image. First, the segmentation algorithm generates many small sub-regions. Secondly, based on the similarity between these sub-regions (the similarity criteria mainly include color, texture, size, etc.), related regions are merged, and this operation is iterative. In each iteration, the co-

ordinates of new sub-regions are marked. It can save time compared to sliding windows as the number of sub-regions decreases. Moreover, various sizes of candidate boxes are predicted because of the sub-regions merging strategy. It also outperforms the sliding window on accuracy as various features are calculated during iterations.

2.2.2 Anchor Boxes

The anchor boxes in object detection networks is widely used today. They indicate a set of pre-defined bounding boxes of different scales and at different positions, covering the regions of the image to be detected. Each anchor box is responsible for detecting objects whose intersection with the corresponding anchor box is over the threshold (usually set as 0.5 or 0.7). As depicted in Fig. 2.1, the anchor technology converts the question of "where is the objects" into "Is there object in this anchor box".

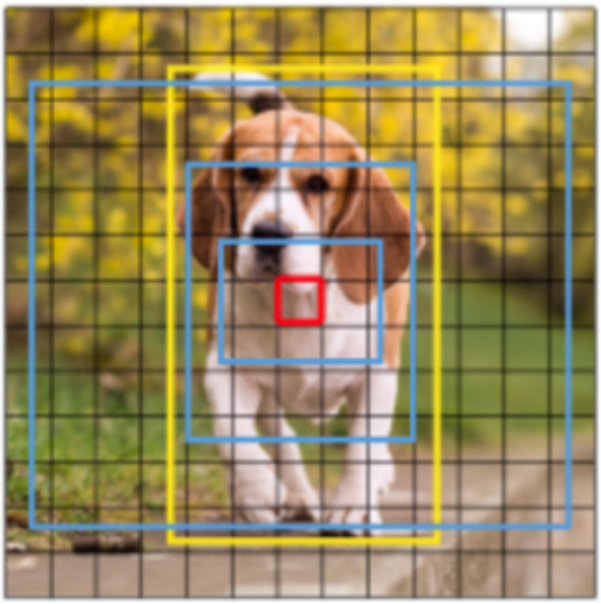


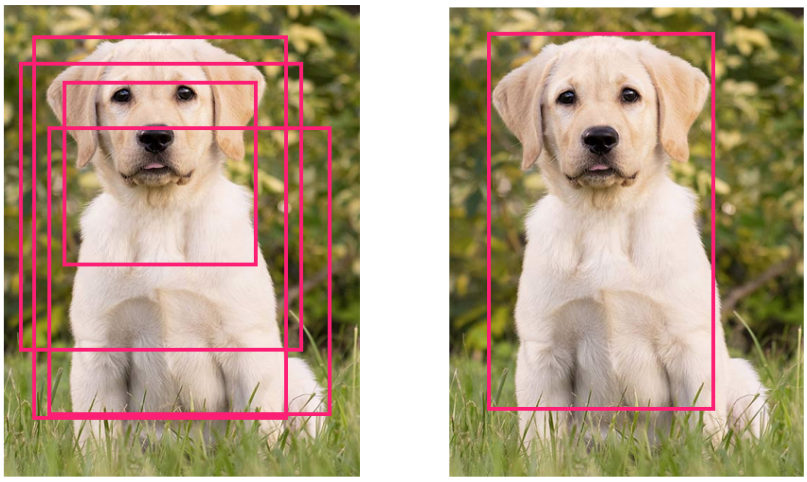
Fig. 2.1 Anchor boxes for the center cells. The yellow box is ground-truth; the blue boxes are prior anchor boxes; and the red box indicates the center point of the object (dog).

An anchor box can be defined by the aspect ratio and the scale. Therefore, anchor boxes can be easily generated according to pre-defined rules. The anchor box has first been

introduced in Fast R-CNN. Anchor boxes enable an element on the feature map to generate a certain number of boxes of different shapes and scales.

2.2.3 Non-Maximum Suppression

NMS (Non-Maximum Suppression) suppresses elements that are not of maximum value. It also can be understood as maximum local search. NMS can filter out non-optimal boxes in object detection. In experiments, detectors may predict more than one bounding boxes that contain the same object with a corresponding probability calculated for each box, which is depicted in Fig. 2.2.



(a) Bounding boxes before NMS (b) Bounding boxes after NMS

Fig. 2.2 Bounding boxes before and after Non-Maximum Suppression (NMS).

As depicted, a bunch of bounding boxes may be predicted for one target. Therefore, it is necessary to determine an optimal bounding box. It is the task of NMS. Assuming that four bounding boxes for the dog are predicted in the picture, then sort them in ascending order according to the calculated probability. Let them be A, B, C, and D in order of increasing probability, respectively. NMS can remove redundant boxes in the following steps. First, calculate the IoU (Intersection over Union) between D (the one of maximum probability

score) and A, B, C, respectively. If the overlap of B and D exceeds a threshold, B will be discarded, and bounding box D will be marked as the optimal one. Then run the first step iteratively on remaining boxes A and C. To be specific, the IoU between C (scored the highest probability now) and A should be calculated in the second iteration. Assuming the overlap is larger than a threshold again, then throw A away and mark C as another optimal box. Until there are no remaining unmarked bounding boxes, all the marked bounding boxes are optimal predictions.

2.2.4 Intersection over Union

IoU calculates the ratio of the intersection and union between the "predicted bounding box (A)" and the "ground-truth bounding box (B)", which is:

$$IoU = \frac{A \cap B}{A \cup B} \tag{2.1}$$

IoU is used to evaluate the performance of object detection networks. While it is simple to calculate the accuracy of classification, we also need to measure the positioning precision of detectors. As shown in Fig 2.3, the green frame indicates the predicted bounding box while the yellow one is the ground-truth. The evaluation of prediction is transferred into calculating the correlation between ground-truth and predicted value, and a higher correlation value denotes the more precise prediction.

2.2.5 Mean Average Precision (mAP)

The mAP for multiple-classes object detection is the mean of the interpolated AP over classes. The per-class AP is given by the area under the PR (precision/recall) curve . The metric AP_{50} we have selected is based on a set of three primary measures of the quality of a detection result: True positive (TP): correct positive inference with $IoU > 50\%$ between the predicted bounding box and ground-truth (the same ground-truth will be only calculated

Fig. 2.3 Example of Intersection over Union (IoU).

once). False positive (FP): incorrect positive inference with $IoU <= 50\%$ between the predicted bounding box and ground-truth. False negative (FN): incorrect negative inference indicating ground-truth that is not detected.

Precision and recall can be calculated based on TP, FP and FN as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

The PR curve is then given by plotting these recall-and-precision pairs as progressively lower-scored detection inferences are included [31]. To be specific, a confidence threshold is selected and instances of which scores are over the threshold are recognized as positive. We start from a large threshold and end with a small threshold based on which all instances are recognized as positive. We calculate these recall-and-precision pairs for each threshold and we can draw the PR curve. The true area under the interpolated PR curve is the AP value for the class [32]. mAP is the mean of AP over all classes in an experiment.

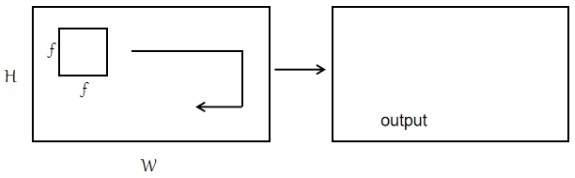
2.3 Online Video Detection

Video action recognition can be divided into two settings, offline and online. For the offline setting, the complete video is available for action recognition, which means all frames in the video can be used for a better understanding of the current frame. However, in the online setting, video action recognition networks read a video stream and the input is processed frame by frame. In our thesis, we define online video object detection as referring to the online setting of video action detection. As the following frames in a video are unavailable, real-time detection or prediction is necessary. We believe the online setting is closer to many practical application scenario. Due to the requirement of real-time processing, methods with high computational complexity are not applicable. In general, online video detection is more in line with surveillance, in which a real-time warning is required, such as in anomaly detection [33] while offline video detection is more in line with video search, such as highlight detection or preview generation used by YouTube.

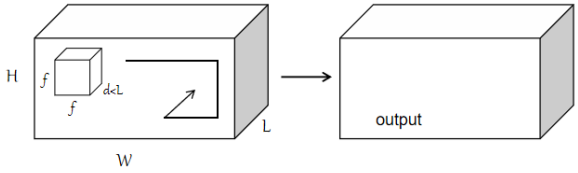
In our work, the input is always a video stream, and hence we address the task of online video object detection. Each frame of the video has contextual correspondence and similarity. Because of the relationship, the information of previous frames can be used to enhance the feature of the current frame. For a blurred frame, we may observe objects clearly in the neighboring frames of the current blurred frame. Due to the high similarity between neighboring frames, removing redundant information can help to speed up the detection in a video. Over recent years, keyframes and non-key frames are proposed to exploit the redundancy of local frames. In general, there are two methods for selecting keyframes: the selection of frames at a fixed interval and the selection based on adaptive learning methods. A complex detection algorithm is applied to keyframes. While for non-key frames, the detection results are usually calculated by referring to the results of the keyframes and the updated information between the non-key frames and the keyframes [7].

2.4 3D Convolution for Video Processing

In recent years, deep learning has developed rapidly, and CNNs have played a very important role in the current development in computer vision. For image processing, the advantage of using a CNN is that the convolution kernel can extract spatial features from images. Compared to 2D images, videos have another depth dimension that is also called the temporal dimension. Therefore, a 3D convolution kernel is needed to extract temporal features. The difference between 2D and 3D convolution is shown in Fig. 2.4.



(a) 2D Convolution



(b) 3D Convolution

Fig. 2.4 Comparison between 2D and 3D convolution

A video clip can be represented as a 4D tensor with the size of $c \times d \times h \times w$, where c, d, h and w denote the number of color channels, temporal length, height and width of each frame, respectively. A 3D convolution kernel can be formulated as a 3D tensor with the size of $d \times h \times w$(the channel dimension is ignored for simplicity), where d is the depth of kernel, while h and w denote the height and width, respectively. The 3D convolution encodes values by sliding along both the spatial and temporal dimensions of the video clip, e.g., the 3D kernels in Fig. 2.5 capture the features across three frames. Tran [1] concluded 3D ConvNets outperform 2D ConvNets in learning spatial-temporal features and proposed C3D (Convolutional 3D). However, a large-scale cascaded 3D ConvNet cannot meet our processing

speed requirements in the video object detection task. In our work, 3D convolution layers are applied to merge features from neighbor frames in our 3D backbone. With our module of dimension reduction, our algorithm can extract temporal information like 3D ConvNets at the cost of 2D ConvNets.

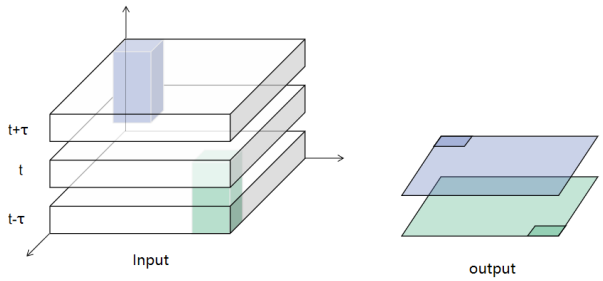


Fig. 2.5 Example of 3D convolution on a video.

2.5 Convolutional Neural Network Architecture

In recent years, advanced convolutional network architectures have emerged that serve as the basis for incremental changes in object detection. In this section, we provide a brief introduction to the concept of CNNs and related terminologies.

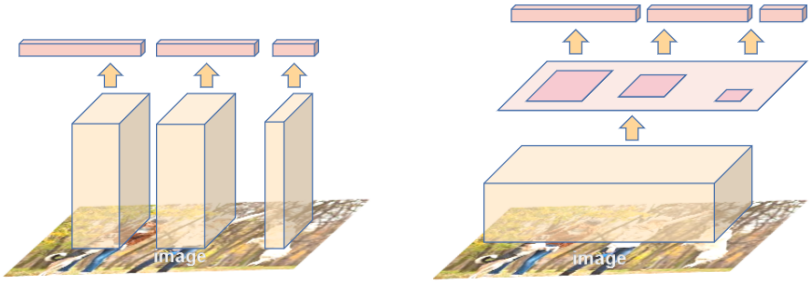
We provide a brief overview of four networks that could be considered the basis of most other convolutional networks today. Their work had a revolutionary impact on deep learning methods at the time.

ILSVRC2012 (ImageNet Large Scale Visual Recognition Challenge 2012) is the milestone in large-scale image classification as AlexNet [34] started the revolution of deep learning. AlexNet used dropout layers to randomly ignore some neurons during training to prevent model overfitting. Previously, average pooling has been frequently used. However, AlexNet replaced average pooling with max-pooling to avoid the blurring of average pooling. In pooling layers, the step size is smaller than the size of the pooling core, which is known as overlapping pooling.

The success of AlexNet in ILSVRC2012 not only affected the research of image classification but also attracted the attention of researchers in other fields of computer vision. Ross Girshick et al. [5] applied CNNs to object detection and proposed the RCNN model. The RCNN model utilized selective search for candidate regions proposal. RCNN then extracts features from each candidate region using a CNN. Finally, RCNN inputs these features into a linear support vector machine for classification. To make the bounding boxes more accurate, RCNN has also trained a linear regression model to correct the coordinates of the predicted boxes, which is called bounding box regression. This model outperformed traditional algorithms on the PASCAL VOC dataset which is a dataset for object detection and segmentation. The average accuracy rate is about 20% higher than conventional algorithms. Because the PASCAL VOC data set is smaller than the ImageNet data set, RCNN is first trained on ImageNet, then the pre-trained model is fine-tuned on PASCAL VOC to achieve better performance.

However, it can be time-consuming to extract features from each candidate region separately. The team of MSRA proposed SPP-Net [35] to improves the architecture of RCNN. In ILSVRC2014, SPP-Net won third place with an error rate of 8.1%. In SPP-Net, the pyramid pooling layer was imported after the last convolutional layer of RCNN. The pyramid pooling layer allowed the network to adapt to images of any size and also generate a fixed-size output feature vector. The difference between RCNN and SPP-Net is shown in Fig. 2.6. The biggest difference from RCNN is that SPP-Net only extracts features from the picture once and then find features of ROI in the feature maps. As SPP-Net only extracts the features of the entire picture once, the speed is greatly improved. From results on the PASCAL VOC dataset, SPP-Net achieved the same level of accuracy as RCNN with lower computation.

GoogleNet [36] won ILSVRC2014 with an error rate of 6.7% that was half of the best record at the time. Inspired by Hebb's rule [37], the Google team improved CNN based on the concept of "multi-scale" and sparse structure. The network is composed of twenty-two layers. GoogleNet imported the inception module based on Lin's et al. work [38] following



(a) RCNN (b) SSP-Net

Fig. 2.6 Difference between R-CNN and SPP-Net.

the traditional convolutional pooling layers. The structure of the inception module is shown in Fig. 2.7. The inception module utilized convolution kernels of size 1×1, 3×3, and 5×5. The 3×3 and 5×5 convolution kernels enabled diverse features to be extracted. The 1×1 convolution kernel achieved a low-cost feature transformation. In general, the inception module achieved dimension reduction, so it did not increase computing requirements for the addition of the width and depth in the network. Moreover, the inception module reduced the parameters that need to be trained, thereby prevented overfitting and improved the generalization of the model. Compared with AlexNet, the number of convolutional layers in GoogleNet has increased, the number of parameters decreased.

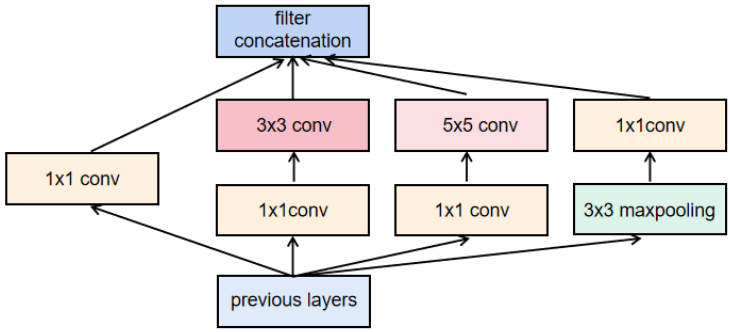


Fig. 2.7 Inception module.

2.6 3D CNNs for Video Processing

Due to the popularity of video processing using convolutional neural network, researches of 3D CNN (3D convolutional neural network) developed rapidly in recent years. Tran et al. [1] proposed C3D (Convolution 3D) to extract temporal and spatial features in a video. The work explored the use of 3D CNNs in large-scale supervised training data set UCF101. The modern 3D deep-learning architectures C3D obtains the best performance in the video analysis tasks. They conclude 3D ConvNets outperformed 2D ConvNets for learning spatio-temporal features and the convolution kernel of $3 \times 3 \times 3$ is optimal. The architececture of C3D is depicted in Fig. 2.8. C3D net consists of 8 convolution, 5 max-pooling, and 2 fully connected layers, followed by a softmax output layer.

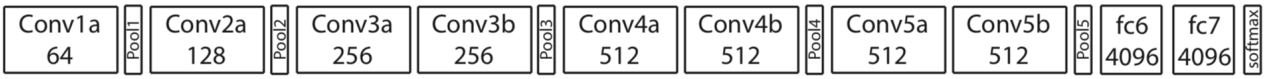


Fig. 2.8 Architecture of C3D [1]. © Copyright IEEE 2015.

Carreira et al. [2] proposed the Two-Stream Inflated 3D ConvNet (I3D), which can achieve extraordinary performance after being fully pre-trained on Kinetics which is a collection of large-scale, high-quality datasets for human action recognition [39]. The two-steam networks is composed of one I3D network trained on RGB inputs and another on flow inputs which carries optimized, smooth flow information. The two networks were trained separately. The RGB-input I3D network was implemented based on the successful image network architecture, of which the convolution and pooling kernels were expanded from 2D to 3D to finally get a very deep spatio-temporal classification network. The architecture of Inflated Inception-V1 is depicted in Fig. 2.9.

To be specific, a set of RGB pictures and the corresponding stack of calculated optical flow frames are input into the 3D CNN that is pretrained on ImageNet, then the final output was the average score of the two streams output. They also re-evaluate models on the dataset Kinetics and conlude that two-stream I3D could improve the accuracy of action recognition

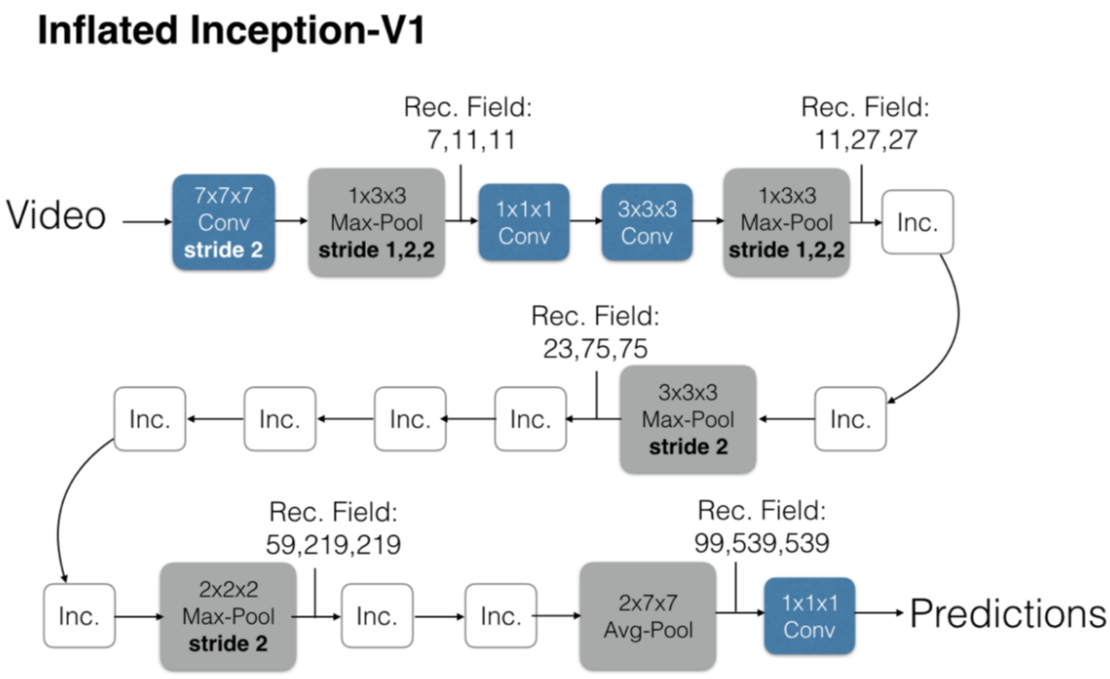


Fig. 2.9 Architecture of Inflated Inception-V1 [2]. © Copyright IEEE 2017

compared with only RGB-input 3D CNN.

D3D (Distilled 3D Network) [3] discards the two-stream architecture and provides competitive performance using a one-stream network. The author of D3D designed experiments to prove only inputting RGB frames into the 3D convolutional neural network loses motion information. In an experiment, a 3D convolutional neural network is designed to extract the 3D features of the input frames, and then decode the motion features to the optical flow. The architecture of the 3D convolutional neural network is depicted in Fig. 2.10. They apply decoders at the hidden layers as depicted in Fig. 2.10. The blue boxes with dashed lines represent convolution, and the blue boxes with solid lines represent inception blocks. The gray boxes indicate pooling layers. The architecture is based on I3D.

They compared EPE (EndPoint Error) loss of the decoded optical flow and the regular optical flow. They concluded that 3DCNN could capture much more motion information from optical flow than from RGB frames. Based on the conclusion, D3D was proposed and distillation was applied to improve motion representations. To be specific, the concept of

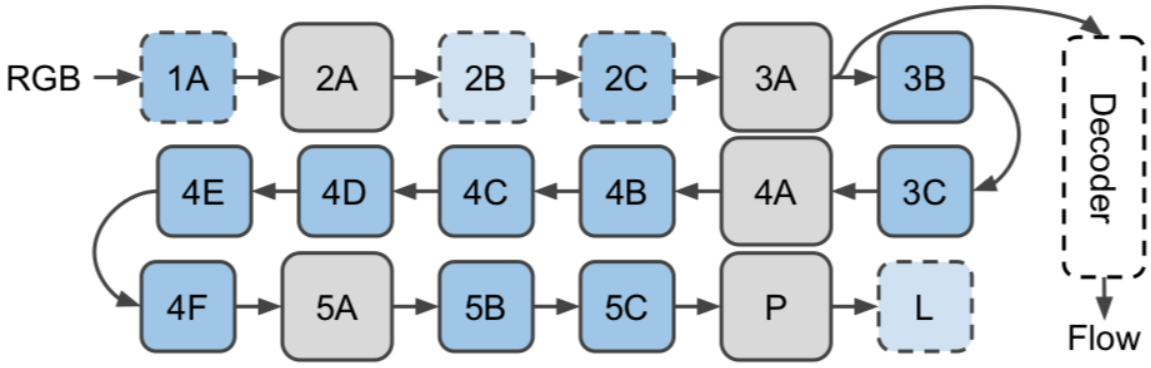


Fig. 2.10 The network used to predict optical flow from 3D CNN features [3]. ©Copyright IEEE/CVF 2020

D3D was to tune the spatial stream to mimic the temporal stream, which can effectively capture more motion information. For the loss function, D3D adopts cross entropy together with optical flow feature supervision loss.

Qiu et al. [4] propose P3D ResNet (Pseudo-3D Residual Net) based on the concept of residual blocks. They first decouple the 3D convolutions into 2D spatial convolutions to encode spatial information and 1D temporal convolutional filters for the temporal dimension. They designed various bottleneck building blocks as shown in Fig. 2.11. The author designed the final P3D ResNet using mixed P3D blocks as the strutural diversity improves the performance. The overview of P3D ResNet is depicted in Fig. 2.12. P3D ResNet utilizes the combination of three P3D blocks and gained the absolute improvement over P3D-A ResNet, P3D-B ResNet and P3D-C ResNet by 0.5%, 1.4% and 1.2% in accuracy, respectively.

2.7 Keyframe Selection in a Video

Key-frames always refer to frames in which the key action or object exists. In video object detection, key-frames are sparse. Considering that the redundancy of adjacent frames in a video is generally high, key-frames selection can reduce the number of frames for which complete detection has to be performed and it can improve the efficiency of detection in

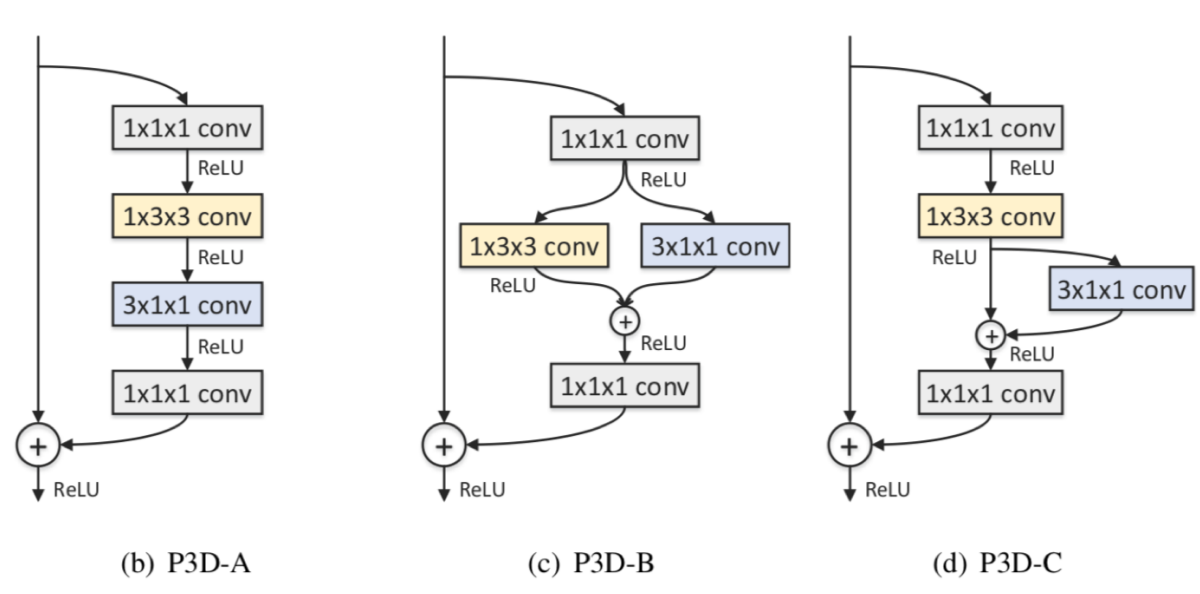


Fig. 2.11 Bottleneck building blocks of P3D [4]. © Copyright IEEE 2017

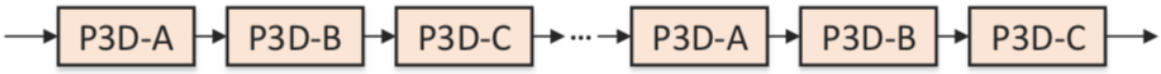


Fig. 2.12 The overview of P3D ResNet using the combination of P3D-A, P3D-B and P3D-C blocks [4]. © Copyright IEEE 2017.

videos. Therefore, the distribution of key-frames is a popular research topic.

There are a wide range of methods to select key-frames. In Flow-net [20] and the work of Zhu et al. [40], key frames are selected following a fixed interval. It is easy and fast but not good for video understanding. Therefore, key-frame selection policy based on machine learning were proposed. For example, key-frames are selected by a recurrent "attention model" in the work of B Mahasseni, et al [41].

2.8 Object Detection in Images

In recent years, object detection algorithms have made great progress. Detection algorithms can be divided into two categories. One is a Region Proposal based RCNN system algorithm (RCNN [5], Fast RCNN [6] and Faster RCNN [27]). They are two-stage detectors, which

means they need to first generate the candidate regions and then utilize the classification and regression methods on the proposed regions. R-CNN utilizes selective search for region proposal and a SVM (Support Vector Machine) for classification [5]. The overview of RCNN is depicted in Fig. 2.13.

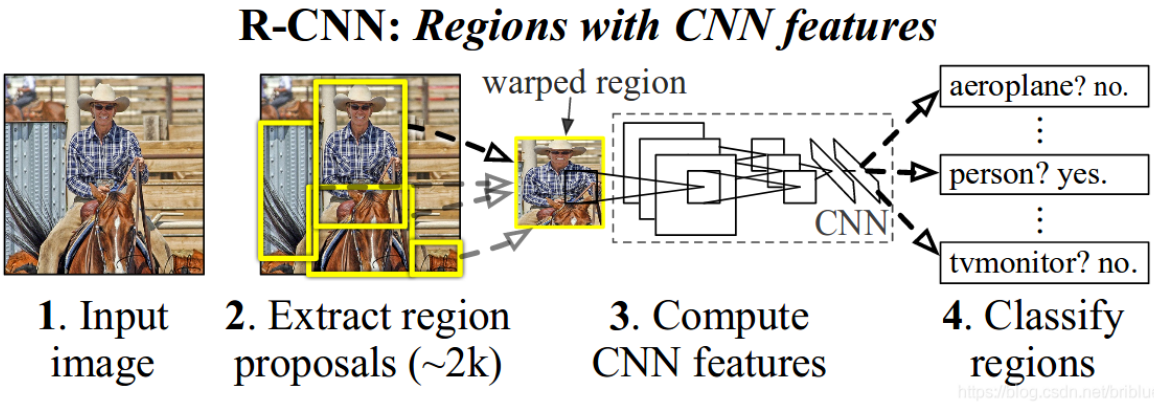


Fig. 2.13 The overview of RCNN [5]. © Copyright IEEE 2014.

However, additional space was required to store the extracted features used by the classifier. Convolution was applied to each region proposal, which can include a large number of redundant calculation. Time and memory consumption for RCNN were relatively large. Therefore, Fast R-CNN is proposed to speed up the detection. The structure of Fast R-CNN is shown in Fig. 2.14. An input image and multiple RoIs (regions of interests) were input to an fully convolutional network [6]. Fast RCNN propose the RoI pooling. RoI pooling layers extract a fixed-size feature map from the output feature map corresponding to region proposals of different sizes. The fixed-size feature map is prepared for the fully connected layer. Because RoI pooling is performed on the output features corresponding to proposed candidate regions, it is no longer necessary to perform convolution on the proposed regions. Additionally, Fast RCNN trains regressors together, in which each regressor corresponds to a class. Fast RCNN also uses softmax to replace the original SVM classifier. The loss function of Fast RCNN is composed of classification and regression loss. The performance of Fast RCNN on PASCAL VOC 2007 dataset was 70% mAP which outperformed RCNN (66%

mAP).

Faster RCNN [27] aggregates feature extraction, region proposal, bounding boxes regression and classification into a complete network. It proposes RPN (Region Proposal Networks) for generating proposal RoIs. From results on MS COCO dataset, Faster RCNN is more accurate than Fast RCNN.

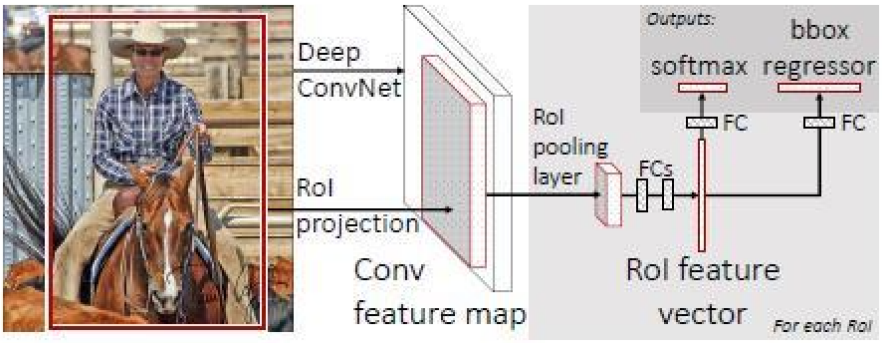


Fig. 2.14 Fast RCNN architecture [6]. © Copyright IEEE 2015.

Apart from two-stage detectors, there are one-stage detectors such as Yolo [28] and SSD [29]. One-stage detectors skip the region proposal and are faster than two-stage detectors. Yolo series and SSD utilize a single deep neural network and provide competitive performance in object detection. SSD predictions are based on multi-scale feature maps as it utilizes pyramidal feature hierarchy. It also uses convolutional predictors instead of fully connected layers for prediction. Therefore, the input images of SSD can be of various sizes. From the testing results on VOC2007, SSD512 achieves 80% mAP (Mean Average Precision) and runs at 19 FPS (Frames Per Second); SSD300 achieves 77% mAP and runs at 46 FPS; Faster RCNN achieves 73% mAP and runs at 7 FPS; Yolo achieves 66% mAP and runs at 21 FPS; overall, SSD provides best performance.

Usually, two-stage detector can be more accurate but slower. While the one-stage algorithm can be faster and less accurate. However, the gap between two-stage and one-stage detectors has been narrowed in recent years due to the emergence of SSD and Yolov3. For a more comprehensive review of object detection methods, the interested reader is referred

to the review of Zhao et al. [42] and in particular the recent analysis of the importance of sampling imbalance by Oksuz et al. [43].

2.9 Object Detection in Videos

2.9.1 Review of Recent Video detection networks

Video object detection is challenging. The main challenges are occlusion, blurring and rare poses. It is feasible to utilize a single-image detector for video detection. However, the performance of single-image detectors drops in video object detection compared to single-image detection. A main direction in video detection in recent years is to fuse spatial and temporal features from bounding-boxes, frames and feature maps. Recently, different solutions to the challenges have been presented.

Methods based on optical flow play an important role in video detection. Sparsely recursive feature aggregation for key-frames and spatially-adaptive partial feature updating for non-key frames [21] were proposed by Zhu et al. The architecture of the feature aggregation module in Zhu's work is based on Flow-net [20]. They apply sparsely recursive feature aggregation based on optical flow only on sparse key-frames and propagate features of key-frames to non-key frames. The feature temporal consistency $Q_{k\to i}$ is introduced to quantify the quality of propagated feature $F_{k\to i}$. If $Q_{k\to i} \leq \tau$, the propagated feature $F_{k\to i}$ is inconsistent with F_i (the real feature of the current frame). The feature of the non-key frame can be calculated through spatially-adaptive partial feature updating. The final feature of the non-key frame feature can be the sum of weighted partial updated features and propagated features. This effective network utilized R-FCN [44] for detection.

LSTM is widely used to extract short-term memory information. Liu et al [25] introduces an real-time online model for video object detection which runs at 40 FPS (frames per second). They propose an efficient Bottleneck-LSTM that significantly reduces computational

cost [25]. They import the convolutional Bottleneck-LSTM into fast single-image object detection to refine and propagate feature maps across frames. Liu et al. also integrate multiple feature extractors with Bottleneck-LSTM [7]. Different feature extractors can specifically extract different features and has various computing cost. The interleaved feature extractors model is depicted in Fig. 2.15. The ratio of lightweight and heavy feature extractors is 2. The Q-learning strategy is used to learn the order in which the feature extractor is utilized. The extracted features are stored and processed by the Bottleneck-LSTM module.

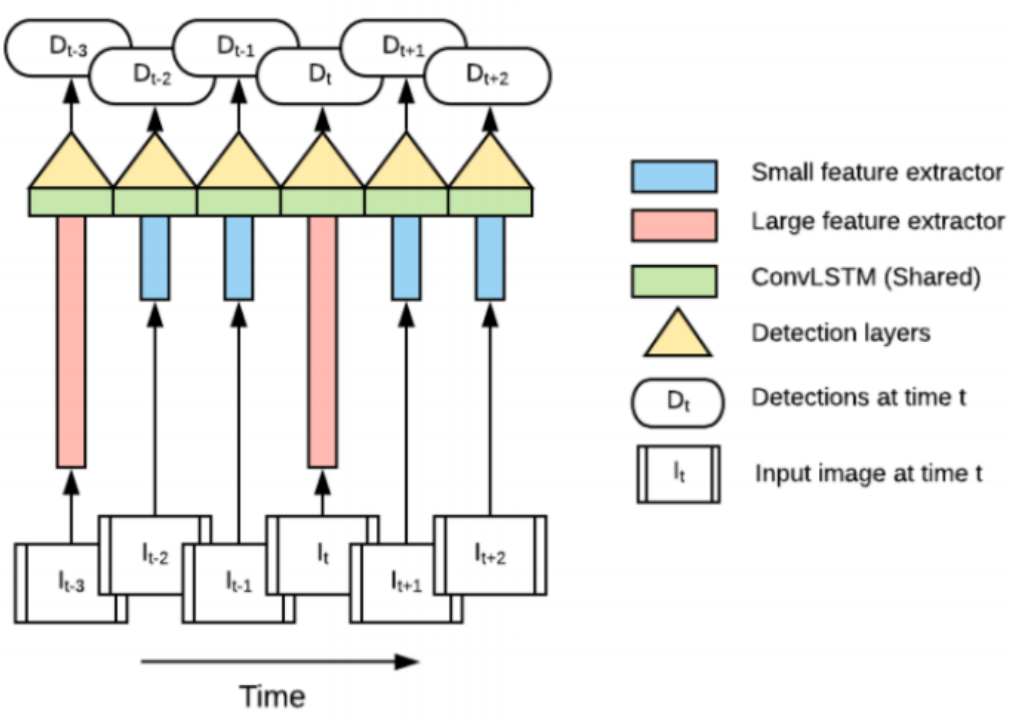


Fig. 2.15 Illustration of the interleaved feature extractors model [7]. © Copyright Liu et al.

To seek a good balance of computing cost and competitive performance, previous efforts usually focus on optimizing the model architectures. The work of Chen et al [45] explores an approach to reallocate the computation over a scale-time space. The Scale-Time Lattice was proposed in their work for non-key frame position correction from different scales and in the temporal dimension. The Scale-Time Lattice is illustrated in Fig. 2.16. PRU (Propagation and Refinement Unit) is for the feature propagation between key-frames and non-key frames.

The input of PRU are two adjacent key frames which is represented by red dots in Fig. 2.16. PRU is composed of a temporal propagation operator, a spatial refinement operator, and a simple rescaling operator. The temporal propagation operator takes motion information on the MHI [46] (Motion History Image) of two key-frames as input, propagate them to an intermediate frame. MHI expresses the object movement in the form of brightness by calculating the pixel value change in the time period. The propagation and refinement based on MHI is multi-scale. Compared with the calculation of optical-flow motion representations, the speed of calculating the MHI is fast. The spatial refinement network is similar to Fast R-CNN and it performs spatial bounding-box regression.

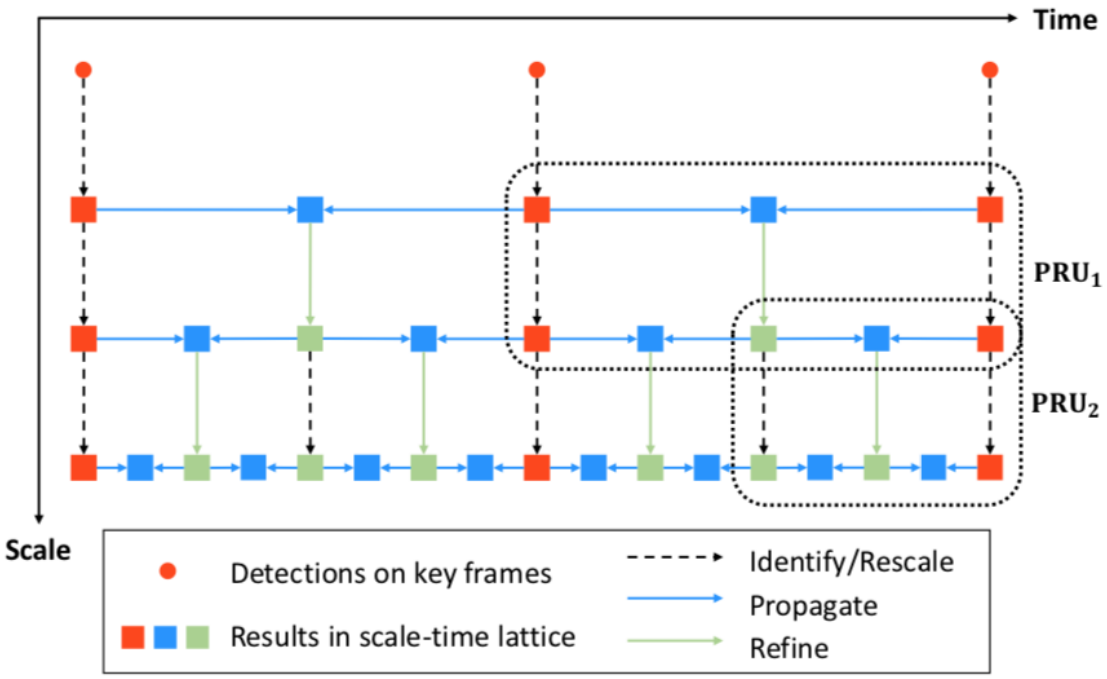


Fig. 2.16 Illustration of the Scale-Time lattice. © Copyright IEEE 2018.

Deng et al [8] proposes RDN (Relation Distillation Networks) that aggregated and propagated object relation to augment object features for detection. The overview of RDN is shown in Fig. 2.17. A RPN (Region Proposal Networks) is first employed to produce object proposals from reference frames and all support frames. R^r consists of the top-K object

proposals from reference frame and all the top-K object proposals from support frames are packed into the supportive pool R^s. Then RDN augments the feature of each reference proposal in R^r by aggregating its relation features over the supportive proposals in R^s for spatio-temporal context. The augmentation is multi-stage. In the basic stage, a set of refined reference proposals R^{r1} is output through relation modules. In the advanced stage, r% supportive proposals in R^s with high objectness scores are packed into the advanced supportive pool R^{sa}. R^{r2} is output from R^S, R^{sa} and R^{r1} from the advanced stage. Finally, the upgraded features of all reference proposals output from advanced stage is exploited for prediction.

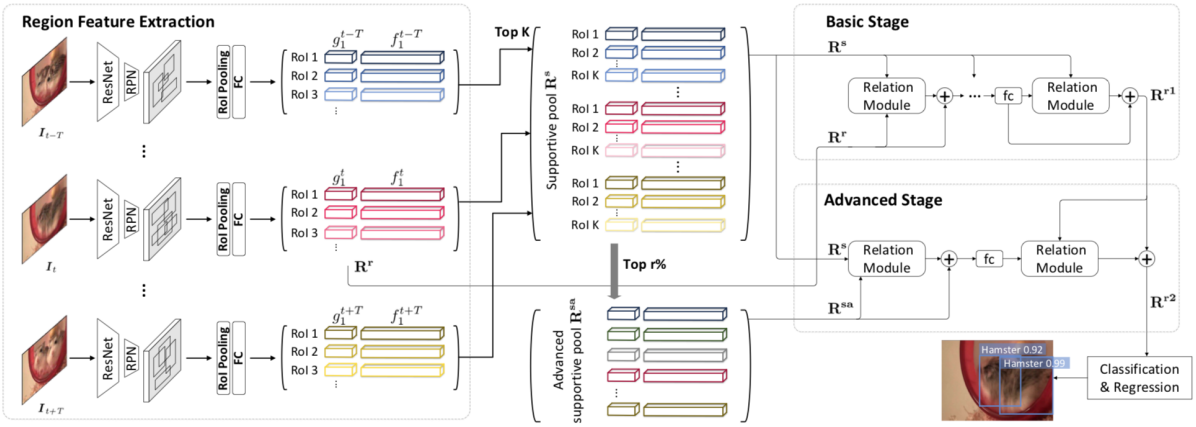


Fig. 2.17 The overview of RDN [8]. © Copyright IEEE/CVF 2019.

The work of Wu et al. [47] proposed SELSA (Sequence Level Semantics Aggregation) module for feature aggregation. They defined the similarity between proposals as the semantic similarity that served as guidance for aggregation between the reference proposal and features from other proposals. The similarity is calculated using cosine similarity.

Memory is useful for long-term relationship extraction. Deng, et al. [48] built the object guided external memory network which can store features of previous frames. They store long-term information in an addressable external data matrix. They apply hard-attention to selectively store valuable features. The feature propagation is achieved through a set of read/write operations from the external memory.

The above methods utilize feature aggregation between neighboring frames and the reference frame. MEGA (Memory Enhanced Global-Local Aggregation) imported global feature aggregation module into video object detection networks [49]. First, RPN is used to generate some candidate proposals from local frames (neighboring frames of the key-frame) and global frames (randomly selected from the video). Then the authors utilize the relation network to aggregate the relative features from the proposals. The aggregation is performed in two stages. The first stage is to aggregate features in those global frame into the features of key-frames (global aggregation stage). The second stage is for the feature aggregation between local frames and key-frames (local aggregation stage). They also design Long Range Memory (LRM) for the storage of local localization information and global semantic information.

There is also tracking-related work [9] that introduces object-tracking into the detection. The architecture of D&T (the Detect and Track) is depicted in Fig. 2.18. It utilized R-FCN [44] for detection. The correlation block is to extract the relation between frames. To be specific, the correlation maps for all positions and these features maps through RoI pooling are computed for track regression which is dash lines in Fig. 2.18. D&T achieves detection and tracking tasks in a network.

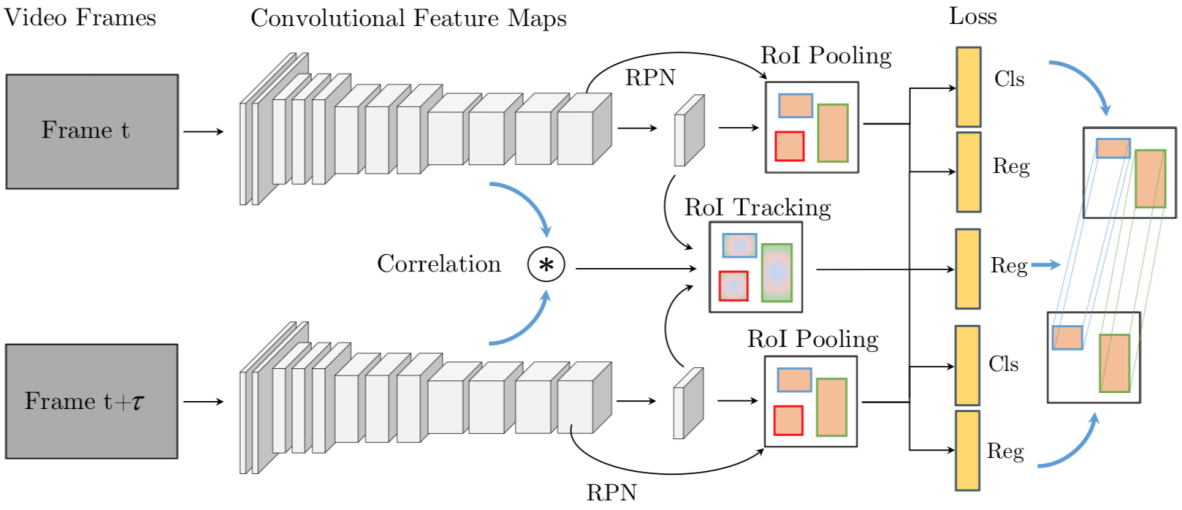


Fig. 2.18 Architecture of DT approach [9]. © Copyright IEEE 2017

Reviewing recent video detection networks provides different strategies for extracting cross-frames relationship on feature level or boxes level. Overall, video detection based on feature aggregation is more popular and provides better performance. The detection based on optical flow can be time-consuming and instead we can use 3D convolution to extract motion information. The interleaved architecture of Liu's et al. work inspires the communication between the 2D and 3D network. The multi-scale feature propagation [45] inspires our multi-scale temporal network. RDN, SELSA and MEGA aggregate features using relation networks or semantic similarity. However, the time cost of utilizing a specific module for exploring features of neighboring frames is high. We utilize AP3D to synchronously align features of neighboring frames and extract motion information, which is more efficient for feature aggregation.

2.10 Video Detectors for Comparison

We selected four efficient video detectors for the comparison with our work. They are FGFA [20], a-LSTM [10], STMN [50] and REPP+Yolov3 [51]. They utilized different feature aggregation methods or post-processing methods.

2.10.1 Flow-Guided Feature Aggregation (FGFA)

FGFA [20] is a representative algorithm for many video detection approaches. Similar to single-image detectors, the feature extraction network of FGFA extracts feature maps of a single frame. To enhance the features of the current frame, Flow-net is used to estimate the motion of adjacent frames and the current frame based on optical flow. It uses bilinear warping to fuse the features of the current frame and neighboring frames obtained by Flow-net. Then they calculate the weighted sum of the warped feature map and the extracted feature map of the current frame through an adaptive weight network. Finally the aggregated feature map is utilized for prediction by the detection network to output the detection result

of the current frame. They utilize Resnet50, Resnet101 [52] and aligned-Inception-Resnet [53] for detection.

2.10.2 Association Long Short-Term Memory (a-LSTM)

Lu et al. [10] proposes association LSTM to explore cross-frames relationship. Association LSTM not only performs regression and classification, it also associates extracted features of each objects. An architectural overview of the network is shown in Fig. 2.19. There are two outputs as shown in Fig. 2.19, including regression results and the novel association features output. The predictions of objects are obtained from the LSTM output hidden state in each frame while association features are computed between the output hidden states in two consecutive frames. Compared to LSTM, a-LSTM extract cross-frame features fundamentally. The loss for a-LSTM is composed of regression error and association error. The association error is optimized to provide features representations of associated pairs.

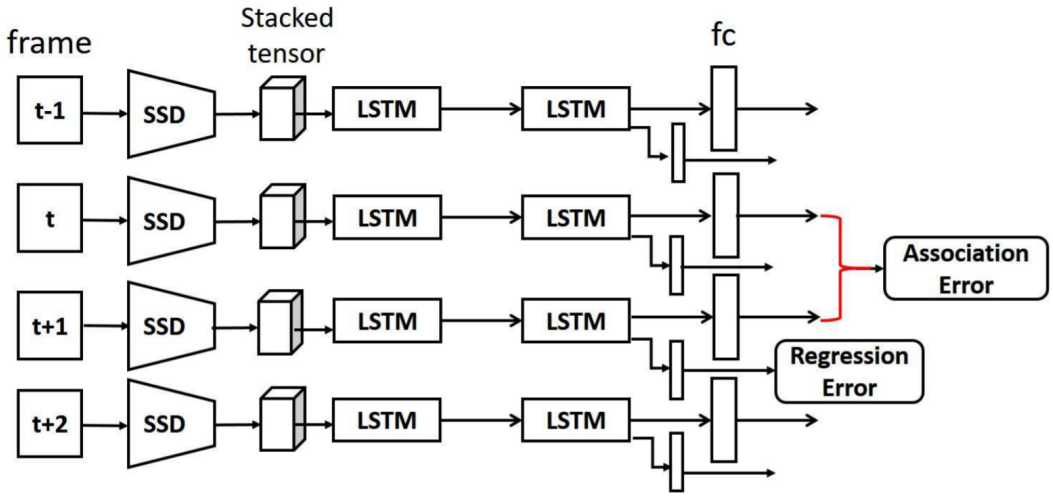


Fig. 2.19 Architecture overview of a-LSTM [10]. © Copyright IEEE 2017.

2.10.3 Spatial-Temporal Memory Network (STMN)

Feature aggregation based on a relation network and optical flow can be time-consuming. Xiao et al. [50] propose feature aggregation based on an aligned spatial-temporal memory. They design a STMM (Spatial-Temporal Memory module) for features to model long-term temporal appearance and motion dynamics. The concept is to perform feature alignment to features of neighboring frames and the reference frame. The feature alignment is based on deformable convolution [53].

2.10.4 Robust and Efficient Post-Processing (REPP)

REPP is a boxes-level method for video detection. All possible detected objects pairs from consecutive frames construct features based on their position, geometry, appearance and semantics. A link scoring model is utilized to calculate the similarity between frames. The link is established between consecutive frames, and the tubelet is generated between the first pair of frames. The tubelet will be expanded if a similar object is detected in the following frames. The link is utilized to correct bounding boxes. REPP has a small calculation cost. The performance of REPP using Yolov3 as the detection network outperforms a-LSTM. However, detection with REPP is offline as the next frame is used for the bounding boxes correction of the current frame.

2.11 Summary

In this chapter, we introduce the concepts of object detection including terminologies and evaluation metrics. We discuss the online video detection tasks. We also provide introduction of 3D convolution and key-frame selection. We review works in single-image and video object detection. We discuss different strategies, including the state-of-the-art methods for detection tasks.

Chapter 3

Background

3.1 Introduction

The purpose of this chapter is to introduce the concepts that are utilized in our work. We describe Yolov3 which is the baseline. Then we introduce octave convolution [11]. We also illustrate the CBAM [13] (Convolutional Block Attention Module) for the introduction of attention module.

3.2 You-Only-Look-Once version 3 (YoloV3)

One of the core problems in CV (Computer Vision) is object detection. Currently, Faster R-CNN [27] and Mask R-CNN [54] have achieved satisfactory results in tasks of instance segmentation, target recognition, and human key-point detection. However, they are time-consuming as they are two-stage detection networks. Yolo (You Only Look Once) [28] is proposed as a one-stage algorithm using a single end-to-end network for object detection. Instead of using RPN (Region Proposal Network), Yolo utilize anchor strategy for predicting bounding boxes. For example, the image to be detected is divided into $7\times7=49$ grids. And in each grid, two bounding boxes are allowed to be predicted. In total, there can be $49\times2=98$

bounding boxes which roughly cover the entire area of the picture. Then we need to fine-tune the candidate boxes to make them closer to the ground truth, which is known as bounding-boxes regression. The architecture of Yolo is a simple combination of convolution layers and two fully connected layers. In conclusion, the architecture of Yolo is not essentially different from other object classification networks based on CNN. The biggest difference is that a linear function is adopted as an activation function in the final output layer as both position and probability will be predicted together.

Compared with Yolo, the main improvements of YOLOv3 include the new feature extraction backbone Darknet-53, multi-scale prediction, and independent logistic classifiers. For better classification results, Darknet-53 is designed and trained on ImageNet. It learns from Residual Networks and sets up shortcut connections between certain layers. YOLOv3 uses the first 52 layers of darknet-53 (without a fully connected layer), as Fig. 3.1 shows.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 3.1 Darknet-53. © Copyright Redmon et al.

The size of the input is $256 \times 256 \times 3$, and the numbers 1, 2, 8 in the leftmost column of Fig. 2.5 indicate the repeating number of residual components. It is a fully convolutional network as the pooling layers are not capable of improving the performance of a big network [55]. Instead of using a pooling layer, a convolution layer with a step size of 2 is used for

downsampling. Moreover, YOLOv3 [26] adopts a multi-scale method that performs detection on the feature map of three scales(with downsampling rates of 32, 16 and 8. Similar to FPN, deep features will be fused with shallow ones through upsampling. Finally, features of the three scales are combined. As the size of the output feature maps changes, the scale of bounding box priors also needs to be adjusted accordingly. YOLOv3 adopts K-means clustering to calculate the size of the box prior to each scale. YOLOv3 adopts the nine bounding box priors trained on the COCO data set. They are (10x13), (16x30), (33x23), (30x61), (62x45), (59x119), (116x90), (156x198), (373x326). To be specific, the three largest box priors (116x90), (156x198), (373x326) are allocated to the smallest 13*13 feature map with the largest receptive field for detecting larger objects. The medium bounding box priors (30x61), (62x45), (59x119) are applied on the medium 26*26 feature map with the medium receptive field for detecting medium-sized objects. The largest 52*52 feature map with the smallest receptive field is adopted for the smallest box priors (10x13), (16x30), (33x23) for detecting smaller objects. For classification, the logistic output is calculated for predicting categories, which can support multi-label objects. To better understand the architecture of YOLOv3, Fig. 3.2 illustrates its input-output relationship on the COCO dataset.

As depicted in Fig. 3.2, three sizes of output are predicted, in which (13*13), (26*26), (52*52) are the grid sizes, and 255 is length of the output. The length of the output is calculated by:

$$Len = N_{pb} \times (N_{para} + N_{cls}), \tag{3.1}$$

where N_{pb} indicates the number of prior boxes in a cell which is Yolov3; N_{para} denotes the number of parameters $(x, y, w, h, confidence)$ for boxes which is 5. N_{cls} denotes the classes number which is 80 as COCO dataset is used.

With the integration of the above solutions into YOLO, the prediction accuracy is improved while maintaining the speed advantage. Especially the ability to recognize small objects is strengthened. Yolov3 outperforms SSD in accuracy and running time.

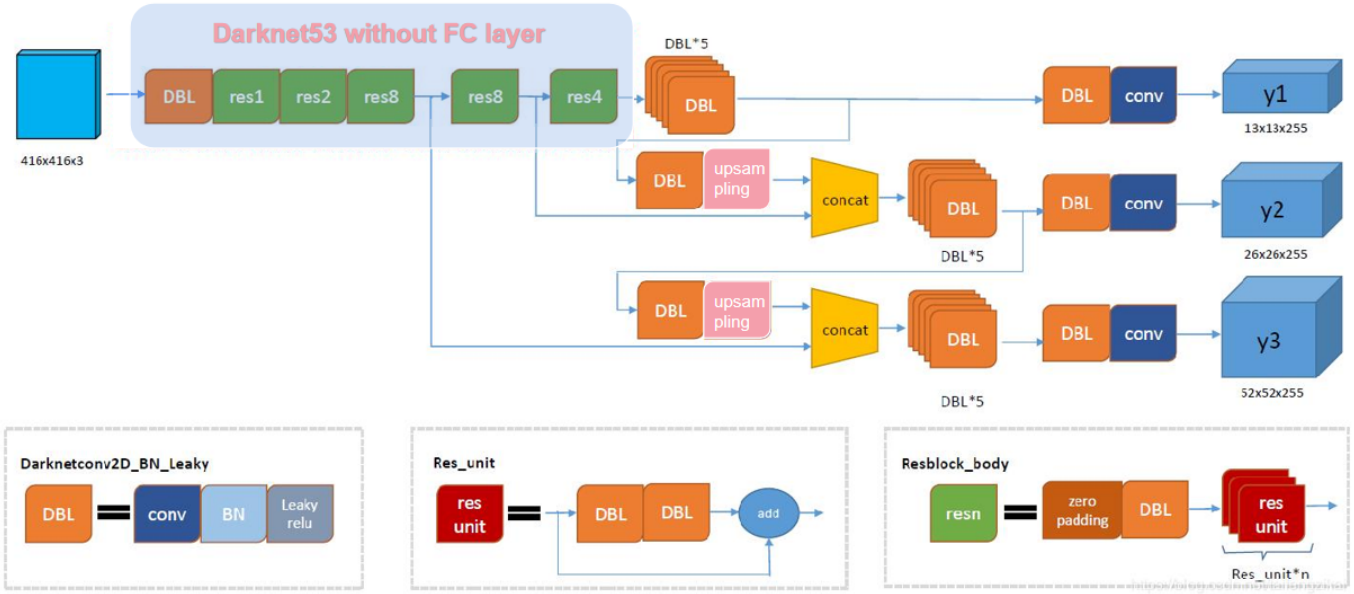


Fig. 3.2 The architecture of YOLOv3.

3.3 Octave Convolution

The efficiency of convolution layers in CNNs decides the practicability of the network, especially in real-time systems. However, the feature maps generated by CNNs always have a lot of redundancy in the spatial dimension. The feature descriptors of different locations are stored independently, ignoring the common information that can be processed together in adjacent locations. Chen et al. [11] state that the output feature map of the convolutional layer can be separated into features of different spatial frequencies. They propose a new multi-frequency feature representation method that stores the high-frequency and low-frequency feature maps in different groups, as shown in Fig. 3.3.

We can compare octave convolution with vanilla convolution. In vanilla convolution, we assume $\mathcal{W} \in \mathcal{R}^{c \times k \times k}$ denotes a k convolution kernel with c channels, F_{in}, F_{out} denote the input and output tensors, respectively. Each feature map $F_{out}(x, y) \in \mathcal{R}^c$ can be computed by

$$F_{out}(x, y) = \sum_{i,j \in N_k} \mathcal{W}_{i+\frac{k-1}{2}, j+\frac{k-1}{2}}{}^{\intercal} F_{in}(x + i, y + j), \tag{3.2}$$

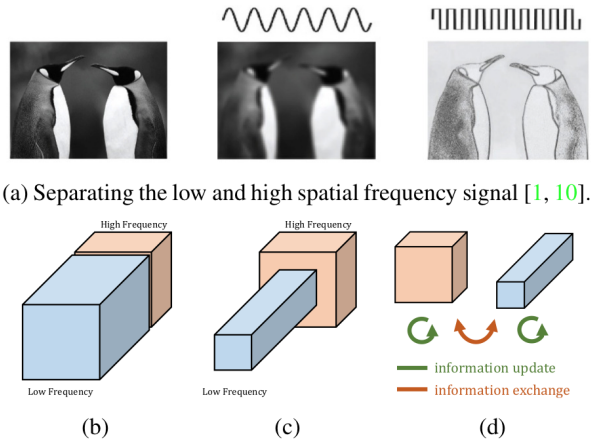


Fig. 3.3 Octave convolution (original image from [11]). © Copyright IEEE/CVF 2019.

where (x,y) denotes the location coordinate and $N_k = \{(i,j) : i = \{-\frac{k-1}{2}, ..., \frac{k-1}{2}\}, j = \{-\frac{k-1}{2}, ..., \frac{k-1}{2}\}\}$ defines a local neighborhood. For simplicity, we assume k is an odd number and $c_{in} = c_{out} = c$. Octave convolution is designed to effectively process the low and high frequency in their corresponding frequency tensor but also enable inter-frequency communication. Let F_{in}, F_{out} denote factorized input and output tensors. The output $F_{out} = \{F^H_{out}, F^L_{out}\}$, where $F^H_{out} = F^{H\to H}_{out} + F^{L\to H}_{out}$ and $F^L_{out} = F^{L\to L}_{out} + F^{H\to L}_{out}$. The detailed implementation of octave convolution is shown in Fig. 3.4.

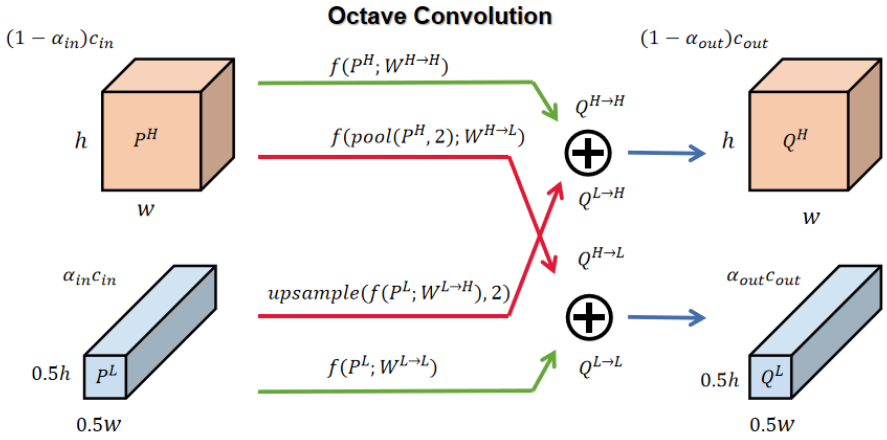


Fig. 3.4 Detailed design of the octave convolution [11] (α is the ratio of the low frequency to the overall features). © Copyright IEEE/CVF 2019.

To be specific, $F^{H\to H}_{out}$ and $F^{L\to L}_{out}$ indicate intra-frequency update, while $F^{L\to H}_{out}$ and $F^{H\to L}_{out}$

denote inter-frequency communication. The convolution kernel $\mathcal{W}$ is split into two components $\{\mathcal{W}^H, \mathcal{W}^L\}$ to calculate the corresponding frequency tensor. For the high-frequency feature map at (x,y), an intra-frequency update is computed by using a regular convolution. For the inter-frequency communication, the up-sampling is folded over the feature tensor F_{in}^L into the convolution. The computation of F_{out}^H, F_{out}^L is shown as follows:

$$\begin{aligned} F_{out}^H(x,y) &= F_{out}^{H\to H}(x,y) + F_{out}^{L\to H}(x,y) \\ &= \textstyle\sum_{i,j\in N_k} \mathcal{W}_{i+\frac{k-1}{2},j+\frac{k-1}{2}}^{H\to H\intercal} F_{in}^H(x+i, y+j) \\ &+ \textstyle\sum_{i,j\in N_k} \mathcal{W}_{i+\frac{k-1}{2},j+\frac{k-1}{2}}^{L\to H\intercal} F_{in}(\lfloor \tfrac{x}{2} \rfloor + i), (\lfloor \tfrac{y}{2} \rfloor + j)^L, \end{aligned} \tag{3.3}$$

where $\lfloor . \rfloor$ denotes the floor operation.

Similarly, we can compute inter-frequency update using the regular convolution as follows:

$$\begin{aligned} F_{out}^L(x,y) &= F_{out}^{L\to L}(x,y) + F_{out}^{H\to L}(x,y) \\ &= \textstyle\sum_{i,j\in N_k} \mathcal{W}_{i+\frac{k-1}{2},j+\frac{k-1}{2}}^{L\to L\intercal} F_{in}^L(x+i, y+j) \\ &+ \textstyle\sum_{i,j\in N_k} \mathcal{W}_{i+\frac{k-1}{2},j+\frac{k-1}{2}}^{H\to L\intercal} F_{in}^H(2\times x + 0.5 + i), (2\times y + 0.5 + j), \end{aligned} \tag{3.4}$$

For the output of last octave layer, we need to combine high-frequency and low-frequency features as:

$$F_{out} = Concat(F_{out}^H, Upsamle(F_{out}^L)). \tag{3.5}$$

In this thesis, we utilize channel attention module to calculate the weight of high-frequency channels and low-frequency channels in the last octave layer. The output feature F_{out} can be calculated as:

$$F_{out} = Concat((w_h \times F_{out}^H), Upsamle(w_l \times F_{out}^L)). \tag{3.6}$$

3.4 Appearance-Preserving 3D Convolution (AP3D)

Appearance-Preserving 3D convolution (AP3D) is proposed to solve the problem of apparent feature misalignment for video-based person re-identification. Person re-identification requires to match the same elements with different pose, different angles and different illuminations. The architecture of AP3D is shown in Fig, 3.5.

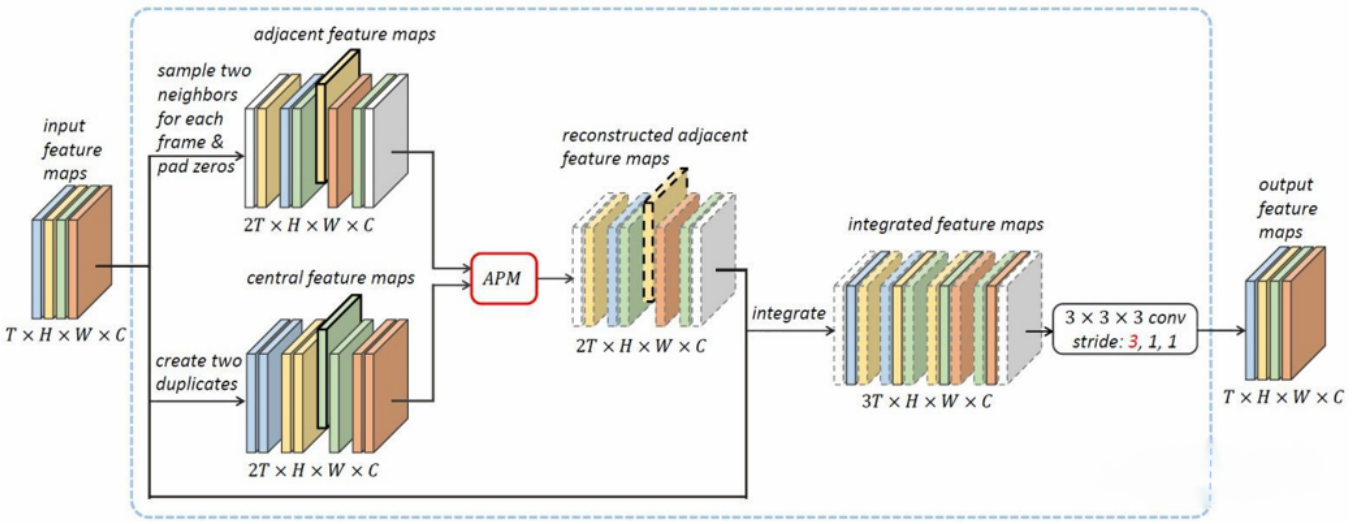


Fig. 3.5 The overall architecture of AP3D [12]. © Copyright Springer Nature Switzerland AG, 2020

As depicted in Fig. 3.5, the original input($C \times T \times H \times W$) is split into two branches($C \times 2T \times H \times W$) at the first step. The first branch is obtained by sampling the previous frame and the following frame of each input frame. The second branch is obtained by creating two duplicates of each input feature map. The two branches are aggregated through the Appearance-Preserving Module (APM) into reconstructed features maps of ($C \times 2T \times H \times W$). The reconstructed feature maps are aligned with the appearance information of each frame. Then the original input is inserted into the reconstructed feature maps to output the integrated feature maps of ($C \times 3T \times H \times W$). Finally, the output is obtained through the 3D convolution layer with the kernel size of ($3 \times 3 \times 3$).

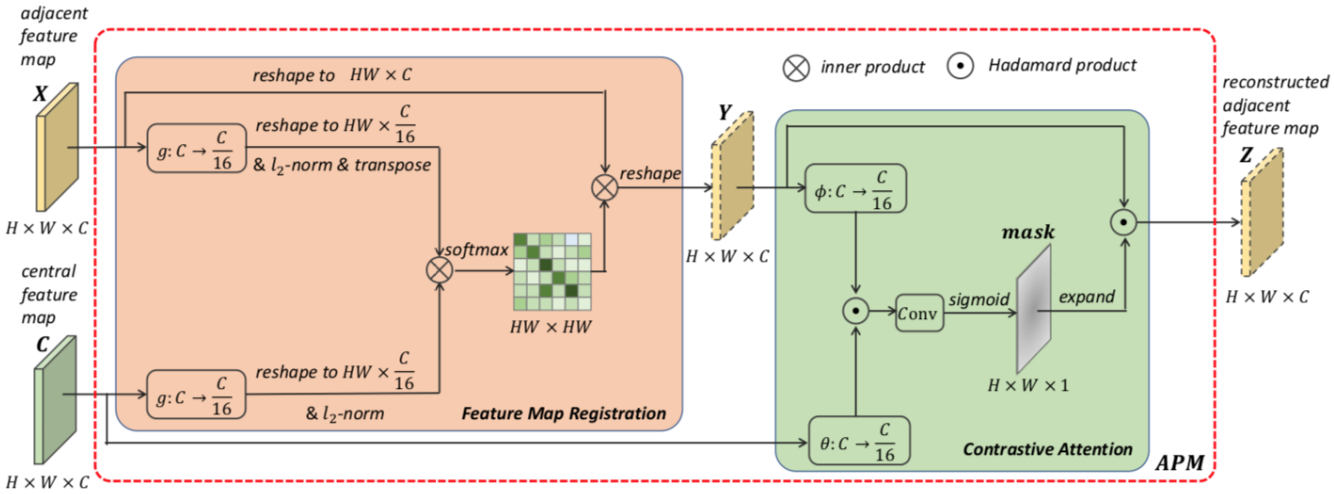


Fig. 3.6 The illustration of APM [12]. © Copyright Springer Nature Switzerland AG, 2020

The APM (Appearance-Preserving Module) is designed for the feature alignment across neighboring frames. As depicted in Fig. 3.6, APM is composed of feature map registration and contrastive attention. Feature map registration is to match central frame with neighboring frames by calculating the cross-pixel cosine similarity. A scaling factor s is multiplied by the similarity map and the softmax function is applied for the normalization. The normalized similarity map is prepared as the weight map for calculating the aligned feature maps. The weighted sum of features in the corresponding position is obtained after registration.

However, there may be no common body parts between the adjacent frames and the registration will be inaccurate in this condition. Contrastive attention is proposed to find the dissimilar regions of two adjacent frames. A mask based on semantic similarity is obtained, and then the mask is applied to the current frame for suppressing the feature expression of regions that exists only in neighboring frames. Finally the reconstructed feature map is generated through APM.

3.5 Convolutional Block Attention Module (CBAM)

Woo [13] proposed a lightweight attention module CBAM to perform attention in the channel and spatial dimensions. CBAM includes two independent sub-modules, CAM (the Channel Attention Module) and SAM (the Spatial Attention Module), which are responsible for channel and spatial attention respectively. An architectural overview of CBAM is shown in Fig. 3.7.

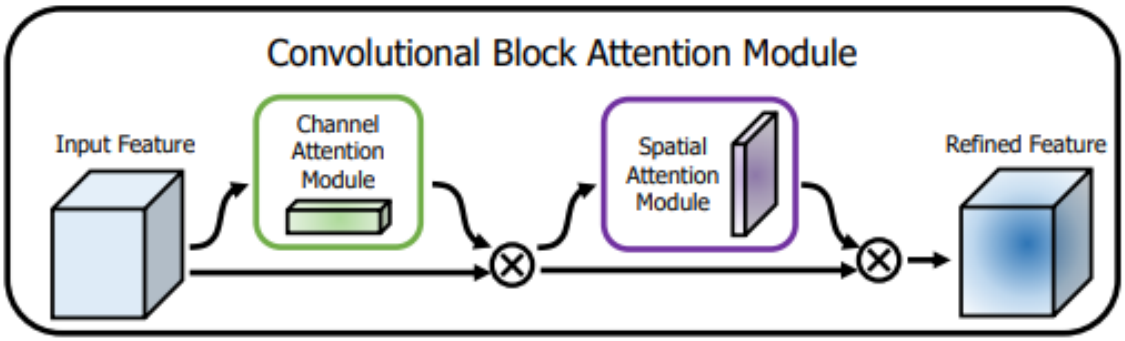


Fig. 3.7 Architecture overview of CBAM [13].

The input feature is refined through CAM and SAM. CAM is for calculating channel attention. The architecture of CAM is depicted in Fig. 3.8. Suppose the input feature map

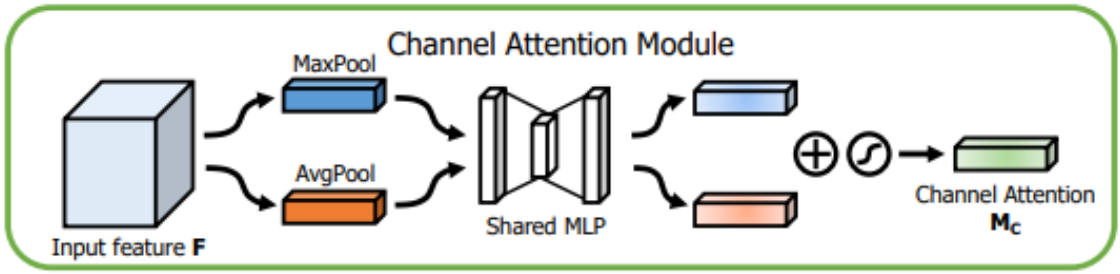


Fig. 3.8 Architecture of CAM [13].

F is of (c, h, w), where c indicates channels; h and w indicate height and width of F. Global max pooling and global average pooling are utilized to F to squeeze the spatial dimension of the input feature map, respectively. Two features of $(c, 1, 1)$ are obtained through pooling

layers. The two features are input into a MLP (multi-layer perceptron). The MLP in Fig. 3.8 is a two-layer neural network. The number of neurons in the first layer is $\frac{c}{r}$ (r is the reduction rate), and the number of neurons in the second layer is c. Then, the MLP output features are subjected to an element-wise addition operation, and the sigmoid activation operation is applied to generate the final channel attention feature of $(c, 1, 1)$. Finally, the final channel attention feature and the input feature F undergo an element-wise multiplication operation to generate the refined feature F_c.

SAM is designed for calculating spatial attention. The input of SAM is F_c of (c, h, w). The architecture of SAM is depicted in Fig. 3.9. First, we utilize channel-wise global max pooling and global average pooling to squeeze F_c in the channel dimensions, respectively. The two squeezed features are of $(1, h, w)$. Then, the two squeezed features are concatenated in the channel dimension. For channel dimension reduction, a 7×7 convolution layer is applied to reduce the channel number from 2 to 1. The output of the convolution layer is of $(1, h, w)$. Passing through a sigmoid layer, the spatial attention feature F_s is generated. Finally, $F_s \times F_c$ is obtained as the output of CBAM.

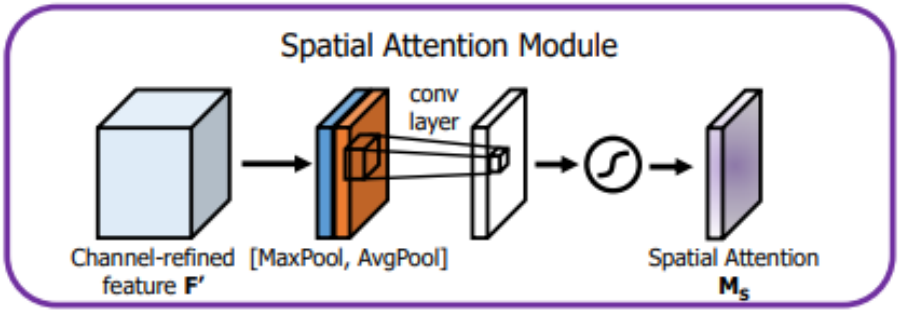


Fig. 3.9 Architecture of SAM [13].

3.6 Summary

In this chapter, we describes the methods that are utilized in this thesis, including Yolov3, octave convolution, AP3D, and CBAM. We discuss these algorithms in detail to set the stage for the development of our method in the next chapter.

Chapter 4

Video Object Detection Network

This chapter will describe the resources and methods utilized to design our network in this thesis. First, we introduce a baseline approach for video object detection, then we describe our feature aggregation policy in our work. We provide the architecture of our network and the data pipeline. We also describe the components of our network in detail, including our selective-frequency octave convolution and temporal network.

4.1 Baseline Network

Given the input video frames $f_i, i = 1, \ldots, l$, the task of video detection is to output object bounding boxes of all the frames, $b_i, i = 1, \ldots, l$. A baseline approach to video detection is to apply a single-image detector to each frame individually. In this thesis, we adopt Yolov3 as the baseline detection network. Yolov3 is fast due to the one-stage structure. Yolov3 also has competitive performance in the task of object detection in images. The image detector extracts features of the frame for detection. However, useful information is missing or hidden from feature extraction due to blurring or occlusion in some frames. A still-image object detector cannot perform well when encountering blurring or occlusions. It is necessary to enhance features of the frame to be detected by context. Applying a detector on each frame

is also time-consuming as a high degree of redundancy exists between neighboring frames. Our work proposes an accurate and efficient video detection network.

While people are watching a video, our eyes can capture the relationship between consecutive frames [56]. The relationship helps the prediction of the following frames. The prediction also accelerates our understanding of the next frame. For example, we can accurately predict the location of objects based on previous frames' motion information. For the input to be recognized, the human visual system will deduce and predict the visual content based on memory about previous frames. The concept of our thesis is to enhance the features of the current frame based on features from neighboring frames.

3D convolution can extract short-term cross-frame relationships that can be described as short-term memory. It has an extra dimension over 2D convolution in images and can process 3D relations. Temporal features of neighboring frames can be propagated to the key-frame (also known as a reference frame) using 3D CNN (Convolutional Neural Networks). Temporal information can be utilized for feature enhancement due to the similarity between local frames. We can utilize a 3D CNN to extract temporal features in our network. For online video detection, the frame rate is a critical evaluation criterion. The purpose of online object detection is to recognize the target at the speed of video streams, which is also a key difference in the requirements between online and offline video detection. Therefore, a fast detection algorithm needs to be adopted. We build a network based on 3D CNN for video object detection at a low cost.

4.2 Architectural Overview of Our Network

We present a network architecture that resulted from a series of experiments to combine temporal networks with state-of-the-art object detection networks. The experiments sought to improve the performance of image detection networks by using the feature aggregation module based on 3D convolution. In our thesis, we adopt Yolov3 as our basic detector.

The backbone of Yolov3 is composed of 53 convolution layers and is known as Darknet-53. Darknet-53 is fast and provides competitive performance with ResNet-152 [57] and ResNet-101 [52]. We improve Darknet-53 to allow better feature extraction performance for video detection. Moreover, Yolov3 is not capable of solving video detection. Occlusion, blurring and other short-term disturbances prevent image object detection networks from performing well in video detection. The solution to the challenges of video detection can be mainly based on feature aggregation and tracking.

In our thesis, we propose a temporal network to aggregate the features from neighboring frames. We adopt the multi-scale architecture of Yolov3 and design the multi-scale temporal network for multi-scale feature aggregation. The temporal network is based on AP3D [12]. An overview of our network is shown in Fig. 4.1.

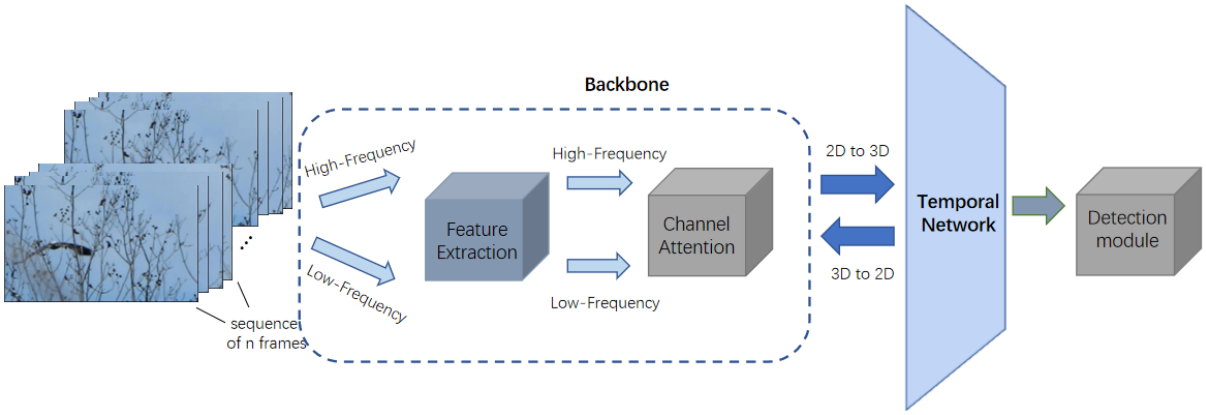


Fig. 4.1 Architecture of the network.

The input of our network is a sequence of n consecutive frames. The size of the input is (n, c, h, w), where c, h, w denotes the channel number, height, and width of source images in the sequence. Suppose we input consecutive frames $f_1, f_2, f_3, \ldots f_n$. Then we concatenate $f_1, f_2, f_3, \ldots f_n$ in the dimension of batch size. Given the batch size b, the input is reshaped into $(n \times b, c, h, w)$ for feature extraction in our backbone.

Our backbone is composed of octave convolution layers and the channel attention module. As depicted in Fig. 4.1, the input is split into high-frequency and low-frequency features.

The multi-frequency features can be calculated through octave convolution layers in the backbone. The fusion of high-frequency and low-frequency features is based on the channel attention module. The details are described in Chapter 4.3.

As depicted in Fig. 4.1, there is inter-communication between our temporal network and our backbone. Our backbone can calculate feature maps $F_1, F_2, F_3, \ldots, F_n$ of $f_1, f_2, f_3, \ldots, f_n$, respectively. Our temporal network is designed for feature aggregation between $F_1, F_2, F_3, \ldots, F_n$. and is for calculating 3D features while our backbone is for calculating 2D features. Therefore, we have to implement feature reconstruction for dimension expansion and dimension reduction. The size of the output of our backbone is $(n \times b, c, h', w')$, where $h' < h, w' < w$. We need to integrate the 2D tensor into the 3D tensor of (b, c, n, h', w') for our temporal network. Our backbone provides multi-scale output. After calculating the output feature maps of each scale, we use the temporal network for cross-frame feature aggregation.

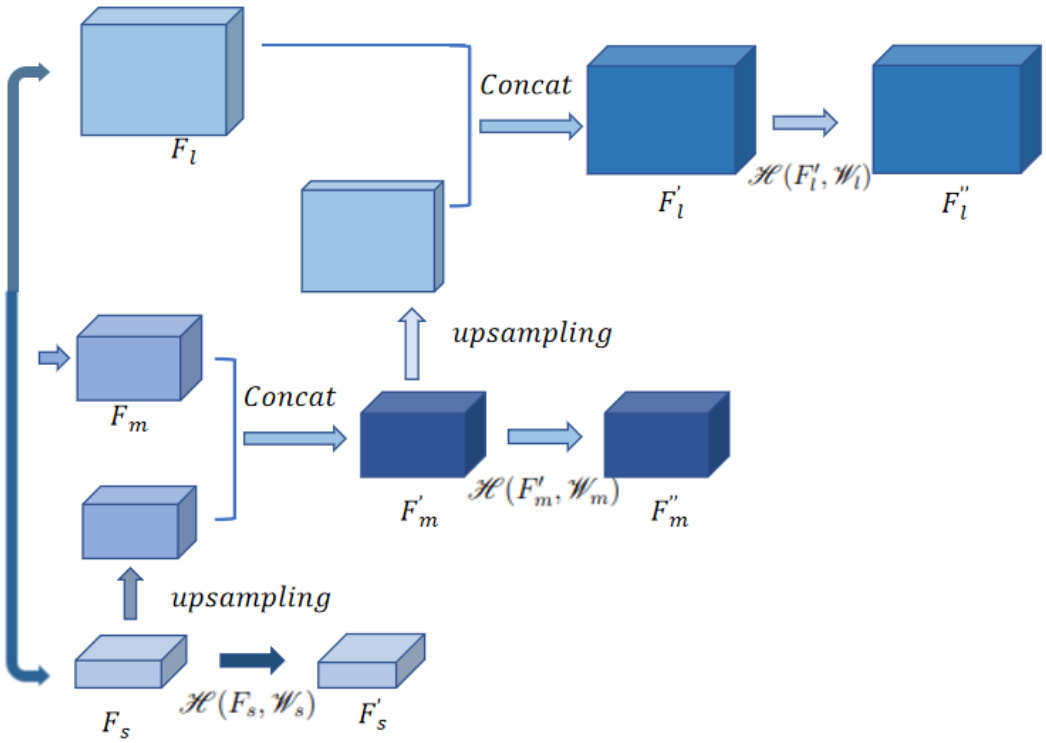


Fig. 4.2 Multi-scale feature in the detection module.

The multi-scale feature maps are output through our backbone and temporal network. The size of the feature maps of each scale is (b, c, n, h, w), where h and w indicate the height and width for corresponding feature maps. We integrate the large-scale and medium-scale feature maps into the tensor of $(b \times n, c, h, w)$. This tensor will be returned to our backbone for calculating the output of the next scale.

The detection module follows the temporal network for features calculation across scales. Suppose the small-scale, medium-scale, and large-scale features from the backbone are F_s, F_m, F_l, respectively. We utilize the inter-scale communication of Yolov3 to calculate the multi-scale output based on F_s, F_m, F_l. The inter-scale communication is shown in Fig. 4.2. The concept is to take feature maps from earlier in the backbone and merge it with the upsampled current-scale features using concatenation. This method allows us to get more meaningful semantic information from the upsampled features and finer-grained information from the earlier feature maps. The method is proposed in Yolov3 [26] and we implement it for the communication across scales in our work.

For the small-scale feature map F_s' , we apply convolution layers to F_s. The output channel depends on the number of classes for detection.

$$F_s' = \mathscr{H}(F_s, \mathscr{W}_s) \tag{4.1}$$

where $\mathscr{H}$ denote the convolution layer, $\mathscr{W}_s$ indicate the weights for the convolution. The kernel size of the convolution layer is (1, 1).

For the medium-scale output, we upsample the small-scale features and concatenate it with F_m.

$$F_m' = Concat(F_m, Upsample(F_s)) \tag{4.2}$$

Then we apply convolution layers to F_m' to calculate the medium-scale output.

$$F_m'' = \mathscr{H}(F_m', \mathscr{W}_m) \tag{4.3}$$

where the kernel size is (1, 1).

For the large-scale output, we upsample F and concatenate it with F.

$$F_l' = Concat(F_l, Upsample(F_m')) \tag{4.4}$$

Similarly, the large-scale output can be calculated through convolution layers.

$$F_l'' = \mathscr{H}(F_l', \mathscr{W}_l) \tag{4.5}$$

where the kernel size is (1, 1). The output channel of $\mathscr{H}$ is based on the number of classes for detection. The sizes of the multi-scale outputs are $(n, c_{out}, h_{out}, w_{out})$, $(n, c_{out}, \frac{h_{out}}{2}, \frac{w_{out}}{2})$, and $(n, c_{out}, \frac{h_{out}}{4}, \frac{w_{out}}{4})$. The convolution kernel of 1×1 enables communication between channels, and allows for changing the number of channels. Finally, we predict the bounding boxes based on the features of three scales.

To be specific, suppose the input of our video object detection network is composed of $f_{t-3}, f_{t-2}, f_{t-1}, f_t$. We first utilize our backbone for feature extraction. Then we use our temporal network for cross-frame features alignment based on appearance information. The temporal network can output multi-scale aligned feature maps for each input frame, respectively. We calculate the feature maps across scales and output the multi-scale feature maps for prediction. Finally, the bounding boxes can be predicted by passing through the Yolov3 detection module.

Our network also applies multi-scale feature aggregation based on AP3D to realize video detection based on the single-image detector. The data pipeline is constructed from the appearance feature extractor and our temporal network for feature aggregation based on appearance information. We enhance our features of the current frame based on the aligned features from neighboring frames. We can then detect objects based on enhanced features. It prevents blurring and occlusion. The combination of 3D CNN and 2D CNN are a trade-off between accuracy and speed.

4.3 Object Detection Backbone

Our backbone utilizes the structure based on the Darknet-53. We design a selective frequency octave convolution and apply it in the backbone of Darknet53 to improve the performance. We replace the convolution layers of Darknet53 with octave convolution [11]. An improvement of accuracy and speed is witnessed. We also design a channel attention strategy into our octave layer to reassign the weight of high-frequency and low-frequency feature maps. The data flow of the backbone is shown in Fig 4.3.

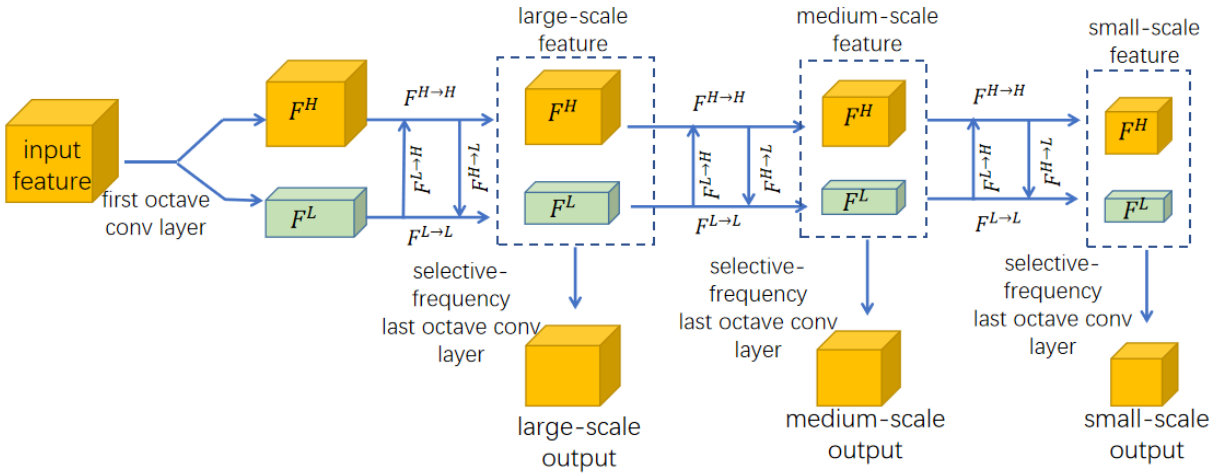


Fig. 4.3 The data flow of our object detection backbone.

As we mentioned in Chapter 2, octave convolution split feature maps into high-frequency features and low-frequency features. The low-frequency is stored at a down-sampled size. It decreases the redundancy of low-frequency features and the overall amount of calculation is decreased as the size of features are reduced. The parameter α that indicates the ratio of low-frequency features is 0.125 in this thesis because experimental results showed the performance of octave convolution with $w = 0.125$ is best [11]. We recombine high-frequency and low-frequency features using channel attention in which the weights for feature maps of different channels are adjusted according to the attention value. Our results (See Chapter 5) showed that the performance of the network is improved by utilizing channel attention. We train

the channel attention to use global information for selectively enhancing features containing useful information and suppressing useless features. Next, we describe the architecture of our backbone in detail.

4.3.1 Selective-frequency Octave Convolution

We utilize octave convolution layers to build our backbone. In the first step, we need to obtain a linear scale representation of the input channel (or image), called Octave feature representation. The high-frequency component describes the rapidly changing fine details; The low-frequency component describes the smoothly changing structure [11]. Low-frequency features have a wider receptive field and the spatial resolution of the low-frequency group can be safely reduced by sharing information between neighboring locations to reduce spatial redundancy [11]. The length and width of the low-frequency component is set to half of the length and width of the high-frequency component, respectively. Because the sizes of high-frequency and low-frequency features are different, vanilla convolution cannot be used to calculate features in octave format. Octave convolution can be used to solve the problem. To be specific, the feature map is explicitly decomposed into groups of high-frequency and low-frequency features as follows.

$$F = [F^H, F^L]. \tag{4.6}$$

The octave convolution updates high-frequency and low-frequency features with communication between them.

$$F^H = F^{H \to H} + F^{L \to H}, \tag{4.7}$$

$$F^L = F^{H \to L} + F^{L \to L}, \tag{4.8}$$

where $F^{H \to H}$ and $F^{L \to L}$ indicate the update of intra-frequency, while $F^{L \to H}$ and $F^{H \to L}$ represent update of inter-frequency. For implementation of above formulas, the weight $\mathscr{W}_{oct}$ for

convolution is split into $\mathscr{W}^H$ and $\mathscr{W}^L$, and they are responsible for F^H and F^L, respectively.

$$\mathscr{W}_{oct} = [\mathscr{W}^H, \mathscr{W}^L], \tag{4.9}$$

$$\mathscr{W}^H = [\mathscr{W}^{H\to H}, \mathscr{W}^{L\to H}], \tag{4.10}$$

$$\mathscr{W}^L = [\mathscr{W}^{H\to L}, \mathscr{W}^{L\to L}], \tag{4.11}$$

where $\mathscr{W}^{H\to H}, \mathscr{W}^{L\to L}$ calculate the update of intra-frequency. For the calculation of $\mathscr{W}^{L\to H}$, we first upsample the input features before convolution. By contrast, we first downsample the input channel while calculating $\mathscr{W}^{H\to L}$.

The implementation of octave convolution is composed of three octave-convolution blocks. The first block is for composing representations of high-frequency and low-frequency features. For the representation of the low-frequency feature, we utilize average pooling for down-sampling. The down-sampling process can be described as:

$$X^L = Avgpool(X), \tag{4.12}$$

where X and X^L denotes the input and the down-sampled input. Then we utilize a convolution layer to extract low-frequency features. The kernel of the convolution layer is (1, 1) as the kernel of (1, 1) can focus on channel communication. The input channel and output channel of the first block is $\mathscr{C}_{in}, \mathscr{C}_{out}$ and α indicates the ratio of low to all frequencies channel numbers. The output channel of low-frequency features c^L_{out} is equal to $\alpha \times \mathscr{C}_{out}$. The input channel is $\mathscr{C}_{in}$. The convolution can be described a

$$F^L = \mathscr{F}(X^L, \mathscr{W}^L). \tag{4.13}$$

Similarly, the high-frequency features can be calculated by applying a convolution layer with a kernel of (1, 1).

$$F^H = \mathscr{F}(X, \mathscr{W}^H), \tag{4.14}$$

where $c^H_{out} = (1-\alpha) \times \mathscr{C}_{out}$. Now F^L and F^H are ready for octave convolution through the first octave convolution block.

The second octave-convolution block is for updating F^L and F^H. We calculate $F^{H\to L}$, $F^{L\to L}$, $F^{H\to H}$ and $F^{L\to H}$ in the block. Suppose the input channels and output channels of the block are $\mathscr{N}_{in}$ and $\mathscr{N}_{out}$. X^H and X^L denote the input of the block. The activity of the block can be described as follow.

$$X^{H\to L} = Avgpool(X^H), \tag{4.15}$$

where high-frequency features are down-sampled. Then we apply a convolution layer for calculating $F^{H\to L}$.

$$F^{H\to L} = \mathscr{F}^{H\to L}(X^{H\to L}, \mathscr{W}^{H\to L}), \tag{4.16}$$

where $c_{in}^{H\to L}$ (the input channel of $\mathscr{F}^{H\to L}$)$= (1-\alpha)\times\mathscr{N}_{in}$, and $c_{out}^{H\to L}$ (the output channels of $\mathscr{F}^{H\to L}$)$= \alpha\times\mathscr{N}_{out}$. The kernel size of $\mathscr{F}$ is (3×3). Then we apply a convolution layer for the calculation of $F^{H\to H}$.

$$F^{H\to H} = \mathscr{F}^{H\to H}(X^H, \mathscr{W}^{H\to H}), \tag{4.17}$$

where $c_{in}^{H\to H} = (1-\alpha)\times\mathscr{N}_{in}$, and $c_{out}^{H\to H} = (1-\alpha)\times\mathscr{N}_{out}$. The kernel size of $\mathscr{F}^{H\to H}$ is (3×3). Similarly, the calculation of $F^{L\to H}$ and $F^{L\to L}$ can be described as:

$$X^{L\to H} = Upsample(X^L), \tag{4.18}$$

where *Upsample* is realized by Nearest Neighbor interpolation. Nearest method is to insert pixels that are equal to the closest pixel into the source image. The algorithm is simple and fast. The size of $X^{L\to H}$ is the same with the size of X^H.

$$F^{L\to H} = \mathscr{F}^{L\to H}(X^{L\to H}, \mathscr{W}^{L\to H}), \tag{4.19}$$

where $c_{in}^{L\to H} = \alpha\times\mathscr{N}_{in}$, and $c_{out}^{L\to H} = (1-\alpha)\times\mathscr{N}_{out}$. The kernel size of $\mathscr{F}^{H\to H}$ is (3×3).

$$F^{L\to L} = \mathscr{F}^{L\to L}(X^L, \mathscr{W}^{L\to L}), \tag{4.20}$$

where $c_{in}^{L\to L} = \alpha\times\mathscr{N}_{in}$, and $c_{out}^{L\to L} = \alpha\times\mathscr{N}_{out}$. The kernel size of $\mathscr{F}^{H\to H}$ is (3×3). We can calculate $F^H F^L$ according to Eq. 4.12 and Eq. 4.13 now. This block will replace the normal convolution layers in our backbone.

The last octave convolution layer is for the fusion of high-frequency and low-frequency features. The original implementation of the last octave convolution [11] is to concatenate features of multiple frequencies. However, we believe the weighted sum can better assign resources to information of different frequency. As depicted in Fig. 4.3, we import the channel attention module into the last octave convolution layer to build our selective-frequency last octave convolution layer. With the channel attention module shown in Fig. 4.4, different weights are learned in the channel dimension. The weight indicates the attention to the feature of the specific channel. There is redundancy between the features of each channel. With channel attention, our network can pay more attention to useful features and ignore redundant information. It improves the performance of our network.

First, we need to fuse the multi-frequency features. Suppose the input of the last octave block is X^H, X^L. The input channels of the block are equal to the output channels. Given the input channels are $\mathscr{C}$ (the input channels of high-frequency and low-frequency). We utilize a convolution layer to calculate F^H as follows.

$$F^H = \mathscr{F}^H(X^H, \mathscr{W}^H), \tag{4.21}$$

where the input channels of $\mathscr{F}^H$ are $c_{in}^H = (1-\alpha) \times \mathscr{C}$; the output channels $c_{out}^H = \mathscr{C}$. Then we use a convolution layer to calculate F^L as follows.

$$F^L = \mathscr{F}^L(X^L, \mathscr{W}^L), \tag{4.22}$$

where $c_{in}^L = \alpha \times \mathscr{C}$; the output channels $c_{out}^L = \mathscr{C}$. We can see the channel number of F^H and F^L are equal. However, the height and width of F^L are smaller than the respective height and width of F^H. We can upsample F^L to match the size of F^H.

$$F_{up}^L = Upsample(F^L). \tag{4.23}$$

The size of F_{up}^L and F^H are identical now. The multi-frequency features can be fused by calculating the sum of F^L, F^H.

$$F = F_{up}^L + F^H, \tag{4.24}$$

where the number of channels of F is $\mathscr{C}$.

As can be seen from Eq. 4.24, we fuse multi-frequency feature by calculating the sum of high-frequency and low-frequency features. However, there can be redundancy of information in different channels. We import the channel attention module for the fusion of selective frequencies as shown in Fig. 4.4. We first apply global max-pooling to feature maps of

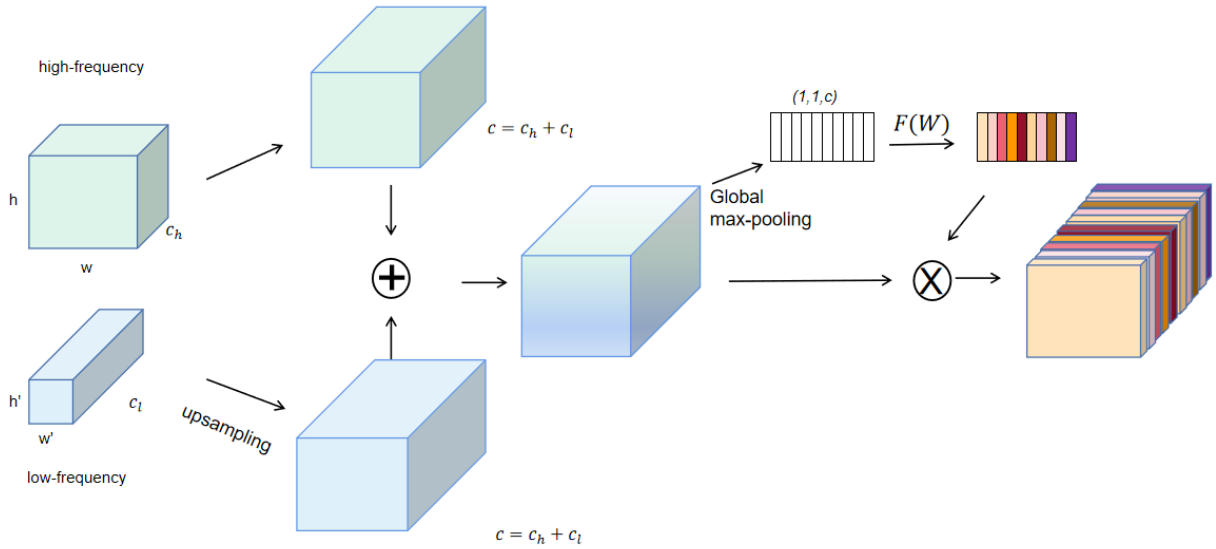


Fig. 4.4 Channel attention module.

$h \times w \times \mathscr{C}$. The feature map is transferred into a sequence of $1 \times 1 \times \mathscr{C}$. The global max-pooling is depicted in Fig. 4.5. Global max-pooling enables the elements in the sequence to have a global receptive field. It compresses the features along the spatial dimension, turning the two-dimensional features into a single real number.

$F(W)$ in Fig. 4.4 aims to extract channel dependencies. It must be able to learn non-linear interactions between channels. The output of the function is a sequence, of which the length is the number of channels. The element of the sequence indicates the respective weights for all pixels of a channel corresponding to the original feature map. Therefore, $F(W)$ denotes the calculation of the weight that reflects attention to each channel. We have reviewed recent channel attention modules [58] [59] [13] [60] and decided to use the structure

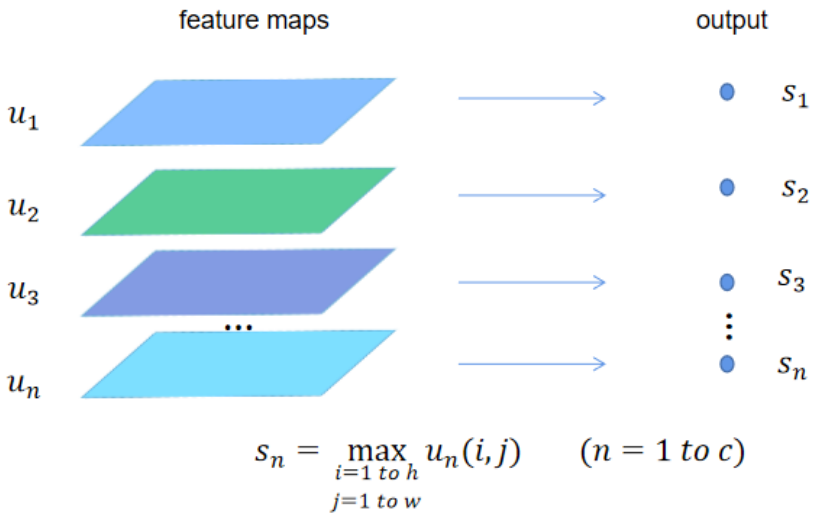


Fig. 4.5 Global max-pooling in channel attention module.

depicted in Fig. 4.6 for the implementation of $F(W)$.

We utilize two fully connected layers for the estimation of the correlation between channels. The input of the first fully connected layer is the output of the global max-pooling layer. Suppose the input is S, and the output of the first fully connected layer is S_1. The size of S_1 is $1 \times 1 \times \frac{c}{r}$. The first fully connected layer is followed by the activation function Relu. Relu is used to increase non-linearity.

$$S_2 = Relu(S_1), \tag{4.25}$$

where S_2 indicates the activated attention vector. Following the activation layer, we utilize another fully connected layer. The output of the layer S' is of $(1 \times 1 \times c)$. Using two FC layers allow more non-linearity and can better fit the complex correlation between channels; it also reduces the amount of parameters and calculations [59]. Finally, We utilize softmax for normalization of W'.

$$S_c = Softmax(S'), \tag{4.26}$$

where the sum of S_c is 1 (satisfying the properties of probability). We can understand the attention value as a probability of adopting the feature in the channel for the decision. The

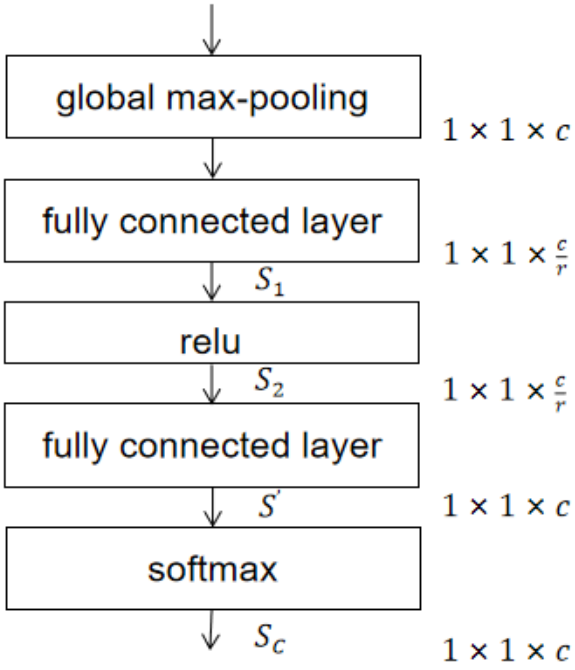
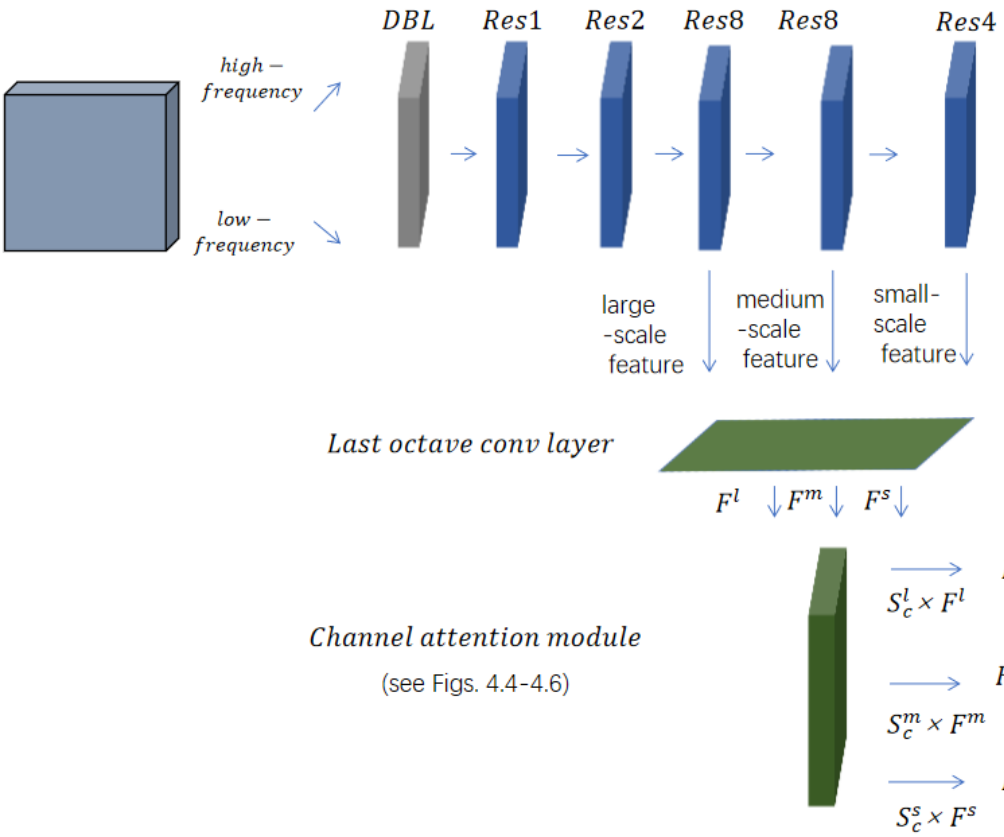


Fig. 4.6 $F(,w)$ processing blocks to extract channel attention.

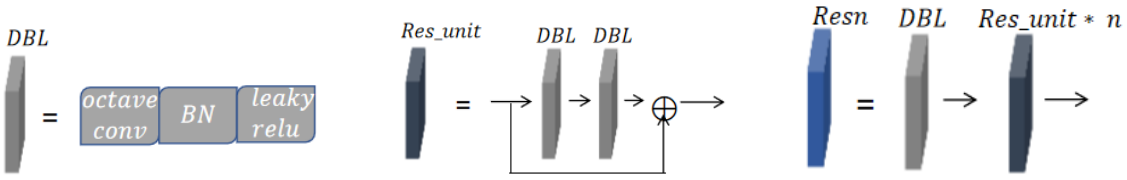
node with a larger weight can be assigned more attention. To calculate the feature map with channel attention, we apply the channel-wise multiply to S_c with F. The calculated feature map is the output of our backbone.

4.3.2 Architecture of the Backbone

We have illustrated the important components such as octave convolution layers and channel attention. Now we have a view of architectural overview of our backbone. The backbone can extract feature maps of three scales for detection. It allows the detection of the object of various sizes. The architecture of the backbone is depicted in Fig. 4.7. Our backbone is composed of five residual blocks (without counting the first octave convolution layer). The number of residual layers in each block is described in Fig. 4.7. Each residual layer is composed of two octave-convolution layers. DBL in Fig. 4.7 denotes the octave convolution. It is composed of an octave convolution layer, batch normalization and activation function.



(a) Backbone



(b) DBL (c) Residual unit (d) Resn in (a)

Fig. 4.7 Overall architecture of the object detection backbone.

We utilize leaky relu as our activation function in our backbone as Darknet-53 does.

Compared to Darknet-53 in Yolov3, our backbone is deeper with octave convolution and channel attention module. There are 216 convolution layers in our backbone compared with 53 convolution layers in Darkent-53. We widely use the residual block to reduce the negative effect of the deep network.

Suppose the size of the input for our backbone is (c, h, w), where c, h and w indicate the number of channels, height and width of a input frame. The large-scale features are output passing through one DBL, Res1, Res2, and Res8 in Fig. 4.7. The medium-scale features are output after one DBL, Res1, Res2, and two Res8 in Fig. 4.7. The small-scale features are output passing through one DBL, Res1, Res2, two Res8 and Res4 in Fig. 4.7. The high-frequency and low-frequency output feature maps of the three scales are stored separately. We apply the selective last octave convolution block to aggregate the multi-frequency feature maps. The output of our backbone is composed of feature maps of three scales. The size of large-scale feature maps is $(256, \frac{h}{8}, \frac{w}{8})$. The size of the medium-scale feature maps is $(512, \frac{h}{16}, \frac{w}{16})$. And the size of small-scale features map is $(1024, \frac{h}{32}, \frac{w}{32})$.

4.4 Feature Aggregation Based on Temporal Network

4.4.1 Feature Aggregation Policy

Given the current frame to be detected f_t and neighbor frames $f_1, \ldots, f_{t-1}$, the temporal feature of f_t are F_t, and t denotes the range of the neighboring frames. These neighbor frames can provide diverse information of the target instances (e.g., because of varied illuminations, object poses, resolution, or non-rigid deformations). The temporal feature F_t can be obtained as:

$$F_t = \mathscr{M}(f_1, ..., f_t), \tag{4.27}$$

where $\mathscr{M}$ indicates the temporal feature extractor. Feature maps of neighbor frames should be assigned weights. Usually, frames that are most similar to key-frames also have the closest connection with them. Unnecessary information should be discarded to reduce the calculation. The enhanced feature of the current frame should be the weighted sum of features at different time and locations. Specifically, generic temporal feature extraction can be defined as:

$$F_{j\to i} = \frac{1}{\mathcal{N}(x)} \sum_{\forall j} \mathcal{R}(F_i, F_j) g(F_j), \tag{4.28}$$

where i indicate the index of input while j indicates all possible temporal positions. F_i, F_j denotes the feature map of input at temporal order i and j, respectively. $F_{j\to i}$ is the output of frame at temporal location i. $\mathcal{R}(f_i, f_j)$ is the function to express the relationship between i and j. The function g computes a representation of the feature at the temporal position j. Then we normalize the response with $\mathcal{N}(x)$. When the relation between input at variant temporal order is computed, we can project features from neighbor frames to the frame to be detected.

Feature aggregation in our network indicates aggregation between different frames. As we have computed the projected features, feature aggregation can be achieved by calculating a weighted sum of them. Suppose different weights $w_{1\to t}, \ldots, w_{t\to t}$ are assigned to denote the feature similarity between the current frame and neighboring frames, respectively. Then the aggregated features at the current frame are the weighted sum of projected features

$$F_t = \sum_{j=1}^{t} F_{j\to t} w_{j\to t}. \tag{4.29}$$

As Yolov3 is one-stage, we cannot enhance our features based on region proposals [49]. To explore cross-frame connection, we need to propagate features that can enhance the current frame's feature. For example, if there is an occluded cat to be detected in the current frame. Then we can decide the bounding box of the cat using motion information and similarity between neighbor frames. In our work, we apply a temporal network based on 3D convolution to extract the cross-frame connection.

4.4.2 Architecture of the Temporal Network

Tran [1] demonstrates the ability of 3D convolution to extract temporal features. In this thesis, we design a 3D CNN based on AP3D (see Chapter 3) [12]. The temporal network aims to aggregate features between adjacent frames based on appearance information. The input is feature maps of K frames from the backbone, and the output is the aligned features maps. We have introduced AP3D in Chapter 3. We utilize the P3D block using a AP3D layer as the basic component of our temporal network. We provide the architecture of candidate AP-P3D blocks in Fig. 4.8.

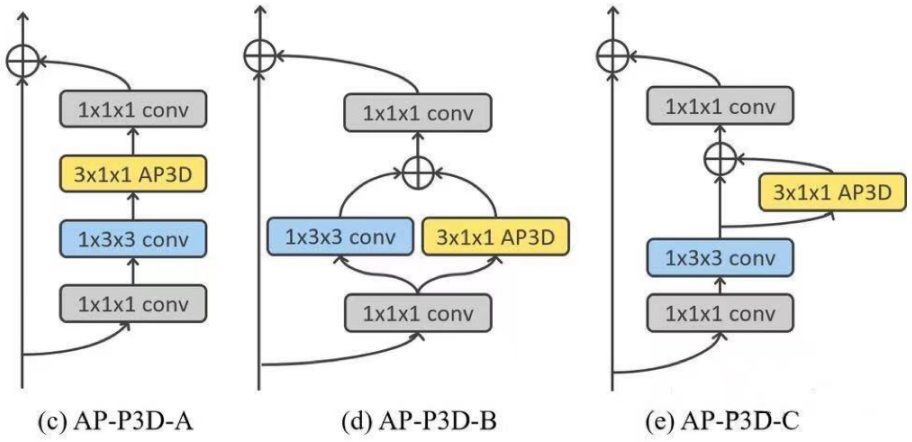
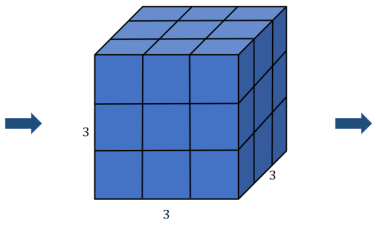
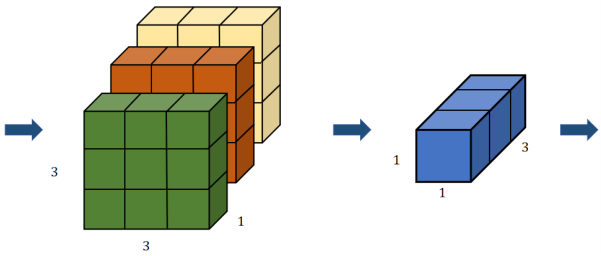


Fig. 4.8 AP-P3D blocks [12].© Copyright Springer.

We decide the parameters referring to C3D [1] and R-C3D (Region Convolutional 3D Network) [61]. The experimental results [1] showed the 3D kernel with the size of (3,3,3) has the best performance in learning spatial-temporal features. However, it is time-consuming to calculate a large number of parameters. Ye [62] propose the depth-wise separable convolution, which drastically reduces the number of parameters in 3D convolutions. The application of separable convolution allows our network to extract temporal information at a small calculation and time cost. AP-P3D-C in Fig. 4.8 is implemented using depth-wise 3D convolution. The difference between depth-wise 3D convolution and 3D convolution is depicted in the following figure.



(a) 3D Convolution



(b) Depth-wise separable 3D Convolution

Fig. 4.9 Comparison between 3D and depth-wise separable 3D Convolution

As depicted in Fig. 4.9, the 3D convolution kernel of size (3,3,3) is replaced with the kernel of size (1,3,3) and (3,1,1). The 3D convolution process can be described as:

$$Y = \mathscr{F}(X, \mathscr{W}) \tag{4.30}$$

The depth-wise separable 3D convolution is divided into two steps. The process can be described as:

$$Y_s = \mathscr{F}(X, \mathscr{W}_1) \tag{4.31}$$

$$Y = \mathscr{F}(Y_s, \mathscr{W}_2) \tag{4.32}$$

From the experimental results by Gu [12], AP-P3D-C showed the best performance. We utilize AP-P3D-C as our basic temporal network block. After deciding the implementation of our AP-P3D blocks, we need to figure out the combination of the blocks. The size of the input in our network is (b, D, c, h, w). D denotes the temporal dimension. The structure of our 3D CNN is decided by D. In our work, D can be 4, 8 or 16 given the hardware

requirements. The architecture of our temporal network for the input of $D = 4$ is depicted in Fig. 4.10. The temporal network for $D = 4$ is composed of 5 AP-P3D-C blocks. The 3D CNN for the input of $(b, 8, c, h, w)$ is composed of 8 AP-P3D-C blocks. Correspondingly, the temporal network architecture is depicted in Fig. 4.11. The 3D CNN for the input of $(b, 16, c, h, w)$is composed of 10 AP-P3D-C blocks. Correspondingly, the temporal network architecture is depicted in Fig. 4.12. We tested the performance of temporal networks of different design. The temporal network architecture shown in Fig. 4.12 achieves the best performance. However, we find there is no obvious improvement between architecture shown in Fig. 4.10 and Fig. 4.12 when $D = 4$, and between architecture shown in Fig. 4.11 and Fig. 4.12 when $D = 8$. Therefore, we utilize architecture in Fig. 4.10 when $D = 4$ and architecture in Fig. 4.11 when $D = 8$ as they are of smaller time cost. Finally, we utilize three temporal networks corresponding to D=4, 8 and 16, respectively.

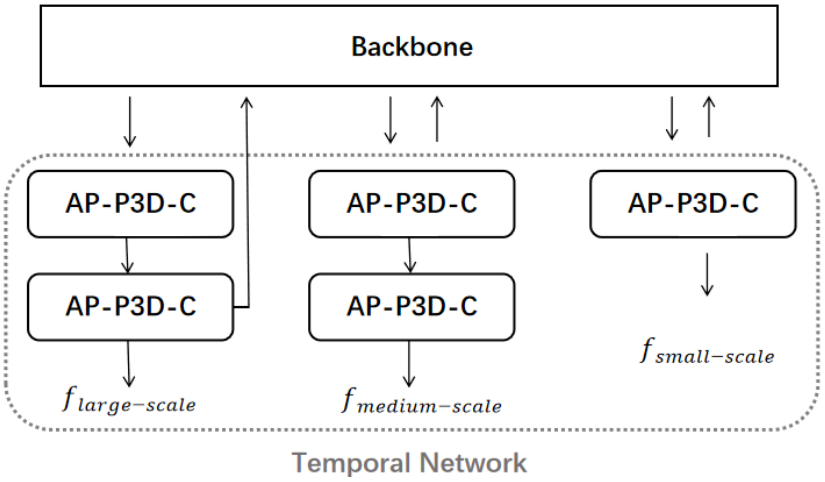


Fig. 4.10 The temporal network architecture for D=4.

As depicted in Fig. 4.10 through 4.12, our temporal networks expand as D increases. We choose the most efficient network corresponding to D. The channels of temporal network layers for outputting $f_{large-scale}$ are 256; The channels of temporal network layers for outputting $f_{medium-scale}$ are 512; The channels of temporal network layers for outputting $f_{small-scale}$ are 1024. The stride of all 3D convolutional layers is $(1, 1, 1)$.

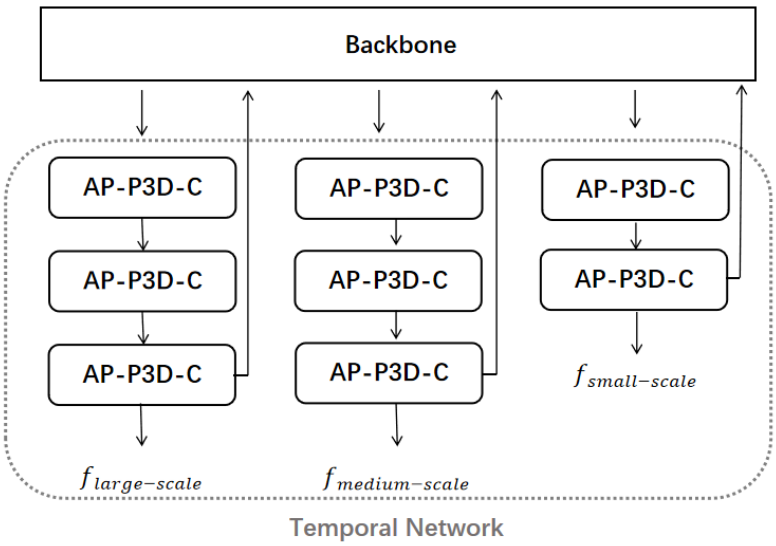


Fig. 4.11 The temporal network architecture for D=8.

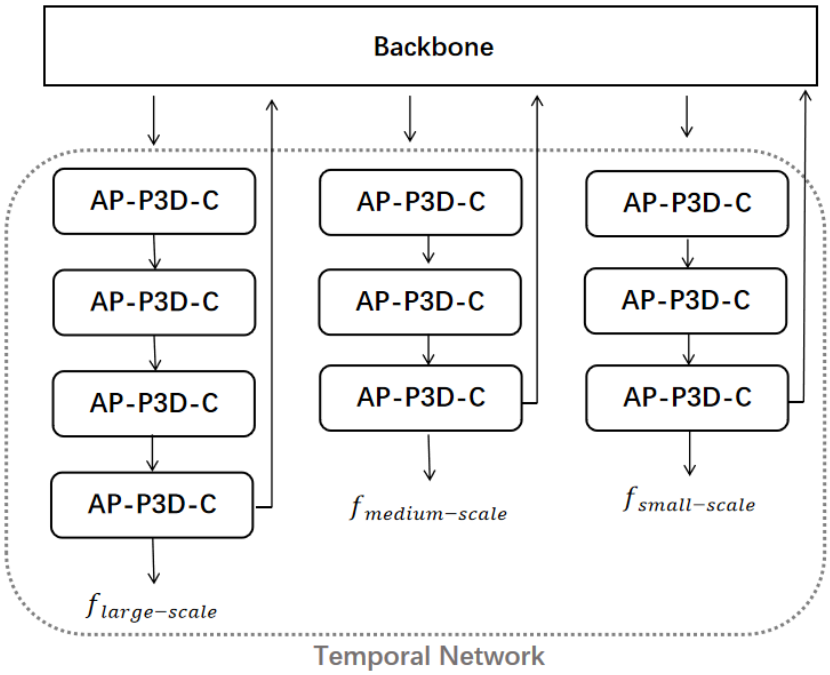


Fig. 4.12 The temporal network architecture for D=16.

4.4.3 The Implementation of Feature Aggregation

Our temporal network focus on the communication of related features across frames. We decide to put our temporal blocks after the last layer for each scale output in our backbone. Our backbone aims to extract rich features from 2D images. Our temporal blocks can aggregate extracted features from adjacent frames. We have illustrated the architecture of our temporal networks. We describe the data flow of feature aggregation in details in this section.

Our backbone outputs feature maps of three scales. The output feature maps of each scale is input into our temporal network for feature aggregation. The output feature maps is a 2D tensor (without the temporal dimension). We need to reconstruct it into a 3D tensor. Our temporal network will align features across frames while applying 3D convolution to feature maps. Suppose D=4, the data flow are as follows.

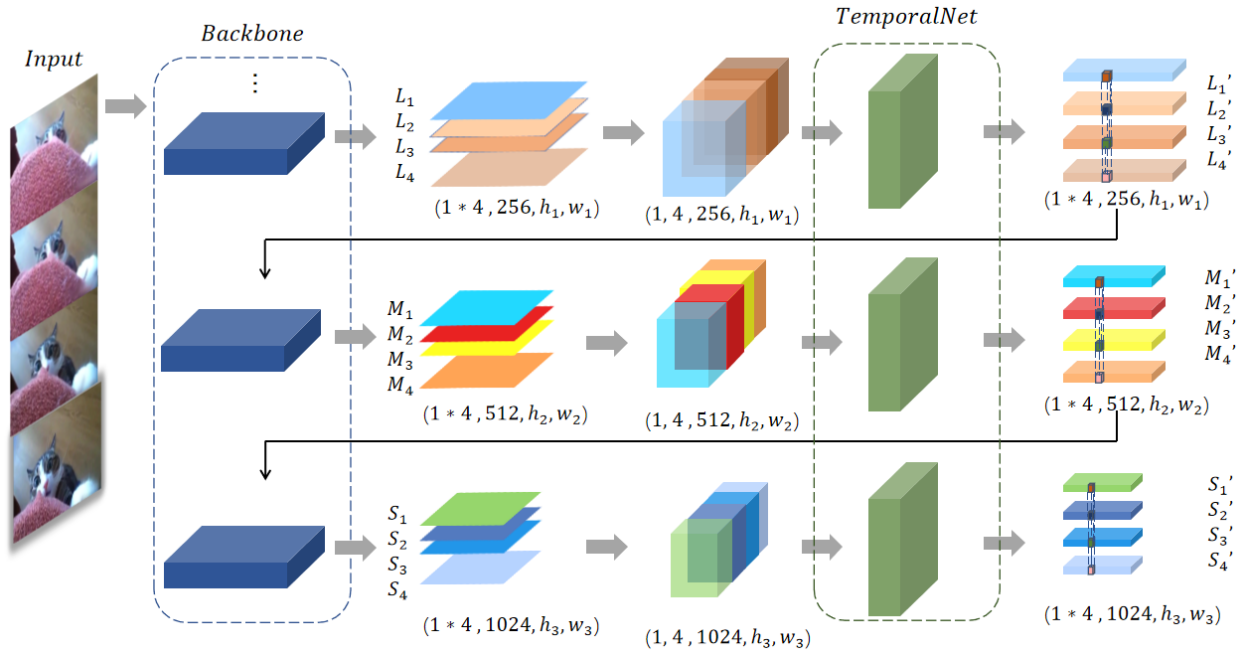


Fig. 4.13 The data flow for feature aggregation based on our TemporalNet (batch size=1).

We utilize our backbone for multi-scale feature extraction. We reshape the output feature maps of each scale from a 2D tensor to a 3D tensor as depicted in Fig. 4.13. The size of

features maps reverts from $(1 \times 4, c, h_1, w_1)$ to $(1, 4, c, h_1, w_1)$, where 1 is the batch size. Then we explore the temporal relationship across frames using our temporal network. Through our temporal network, feature alignment is accomplished based on appearance information across frames. We reshape the 3D output features into 2D as the output for our feature extraction of a certain scale.

Compared to the data flow of Yolov3, our network uses a specially designed temporal network and it utilizes the communication between the backbone and the TemporalNet for feature enhancement based on local feature aggregation. The local aggregation is illustrated in Fig. 4.14. $F_1, F_2, F_3, ..., F_K$ indicate the feature map of $f_1, f_2, f_3, \ldots, f_K$. AP3D can

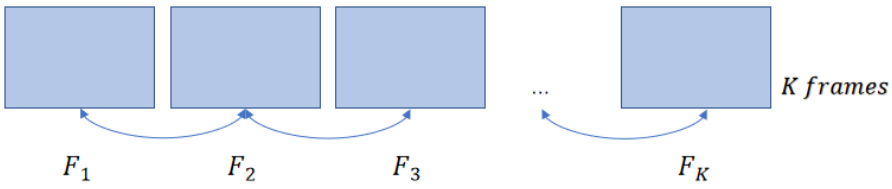


Fig. 4.14 The illustration of local feature aggregation between K adjacent frames.

propagate features across local frames. In this thesis, we aggregate features from 4, 8 or 16 neighboring frames. We can describe the feature aggregation of F_K^p as an example. Given F_n^p indicates the feature at position p of F_n, F_K^p can be calculated as:

$$F_K^p = \sum_{n=1}^{K} F_{n \to K}^p w_{n \to K}, \tag{4.33}$$

where $w_{n \to K}$ is assigned to denote the relationship between F_n and F_K. $F_{n \to K}^p$ denotes the feature aggregation between the pair of frames at position p. It can be calculated as follow.

$$F_{n \to K}^p = \sum_{\forall j} F_n^j w_{j \to p}, \tag{4.34}$$

where j indicates all possible positions in F_n; $w_{j \to p}$ is calculated as normalized cosine similarity between position p of F_K and position j of F_n.

4.5 Summary

In this chapter we presented the major technical components which cooperate to provide a complete description of the work conducted in this thesis. We have described the selective-frequency octave convolution and our feature aggregation policy. We also introduce the data flow and architecture of our backbone and temporal network. We conclude with summarizing our design contributions as follows.

We improved octave convolution using channel attention as selective-frequency octave convolution. We incorporated selective-frequency octave convolution into Darkent-53 as our backbone. We designed a multi-scale temporal network based on AP3D. We combine our backbone and our temporal network through a multi-scale communication between them.

Chapter 5

Experiments

In this chapter we will describe the experiments that were conducted to determine the major technical components of this thesis. We evaluate the proposed component on a public dataset to indicate the impact. We separate the experiments into two major areas: training and evaluation. In the section on training procedures, we describe the data augmentation, pretraining and hyper-parameters. In the evaluation section, we describe the evaluation metrics and compare the results of comparators with our proposed network. Finally, we investigate the effect of the individual components of our proposed network in ablation studies.

5.1 Dataset and Data Augmentation

5.1.1 Video Dataset

The dataset used in this evaluation is the ImageNet VID2017 dataset [23]. It consists of fully-annotated 759 video snippets for training and fully-annotated 138 snippets for evaluation and fully-annotated 139 snippets for testing. The number of auto-extracted frames of each snippet ranges from 63 to 1060. We apply data augmentation while training. Data augmentation creates new samples by flipping, cropping and mixup [63] to provide a large

number of similar data for training.

The dataset includes 30 basic-level categories for video detection. All classes are fully labeled for each clip. For each video clip, there are a set of annotations including the frame number f_i, class labels c_i, bounding boxes b_i and the not occluded flag. There are 259079 occluded objects and 260953 not occluded objects in the dataset. Correspondingly, we generate a .xml file as the annotation for each frame.

For training, we split the frames of each snippet into groups of a fixed number of consecutive frames (The number will be 4, 8 or 16). The groups are the input for our video detection network.

5.1.2 Data Augmentation

Data augmentation is widely utilized in single-image detection. It can improve the accuracy and accelerate the optimization of models. Therefore, we apply data augmentation methods to our training data for improving the robustness of our model. In experiments, there can be online and offline augmentation. Offline augmentation is performed for preparing the training data to expand the size of the dataset. Online augmentation is performed in a mini-batch before the data being sent for training. The cost of performing offline augmentation on a large dataset can be prohibitive. We apply online augmentation while training.

To be specific, our data augmentation is designed for video clips. While data augmentation for frames can be implemented individually, data augmentation for video clips needs to respect temporal information. We apply data augmentation to frames in the input sequence synchronously. The data augmentation in our thesis includes resizing, horizontal-flipping, cropping, affine transformation, and temporal-mixup [64]. Data augmentation allows more efficient training and better performance of a machine learning model [65]. To evaluate the performance of data augmentation, we train our model in a small dataset including 10000 frames for one class with and without data augmentation. In our experiment, 3% mAP drop

is observed without mixup.

Suppose the input video clip is composed of four frames which are of the same size. The source images are shown in Fig. 5.1:

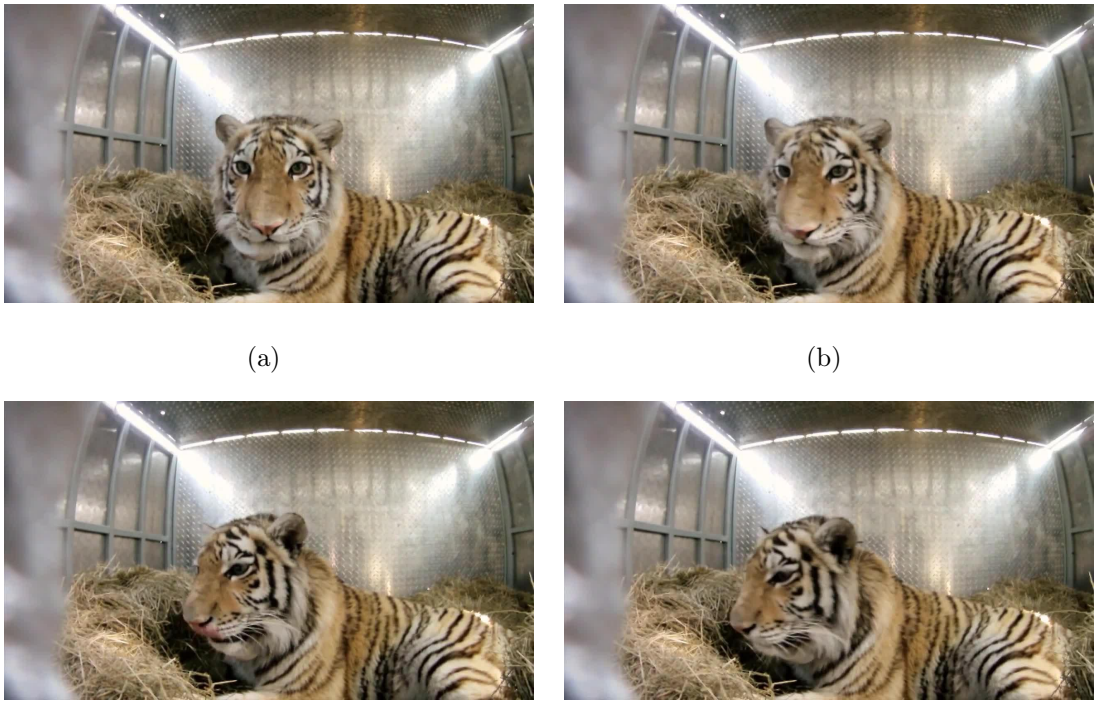
(a) (b)

(c) (d)

Fig. 5.1 Source images in training data.

Horizontal-Flipping: Horizontal-Flipping is widely used in data augmentation. It never changes the class and the shape of the object but it can generate similar samples. Our model is to detect the target ignoring the pose and position. Horizontal-flipping helps the generalization of our model. To be specific, we randomly implement horizontal-flip to the input frames. We set the threshold $p = 0.5$. We randomly generate a value r_1 and compare it with p. If $r_1 > p$, we apply horizontal-flipping to the input frames.

Cropping: We utilize random cropping as another data augmentation method in our work. Random cropping is not a perfect data enhancement method, as it is possible that the target is not completely retained. Therefore, our cropping algorithm reserves the ground-

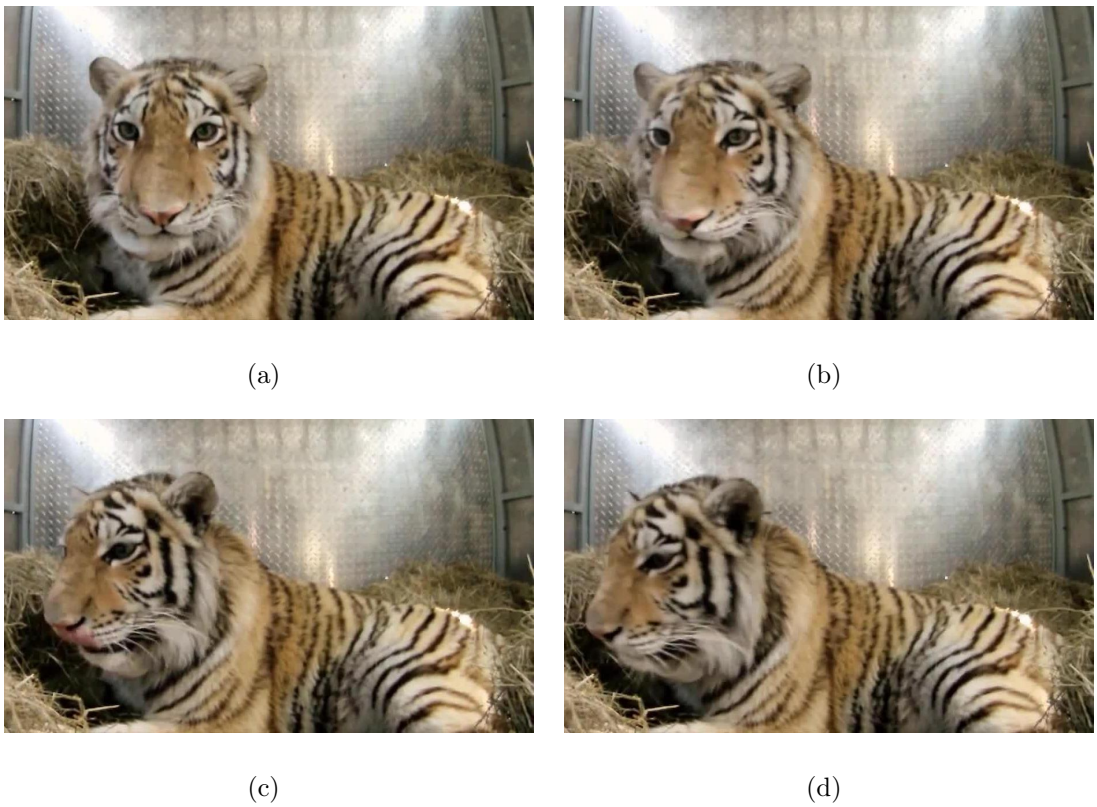


(a) (b)

(c) (d)

Fig. 5.2 Horizontal-flip for data augmentation.

truth bounding boxes. We calculate the max bounding box based on coordinates of ground-truh bounding boxes in the input video clip. Then we implement random cropping that maintains the content within the max bounding box. For example, we crop the images in Fig. 5.2 and the results are shown in Fig. 5.3. The only object (a tiger) is preserved while background areas are cropped. Similar to random horizontal-flipping, we compare a random value r_2 with the threshold p and apply cropping if $r_2 > p$. The original frame will be cropped into random size while preserving the max bounding box. Cropping enables more appropriate training for the dataset [66].



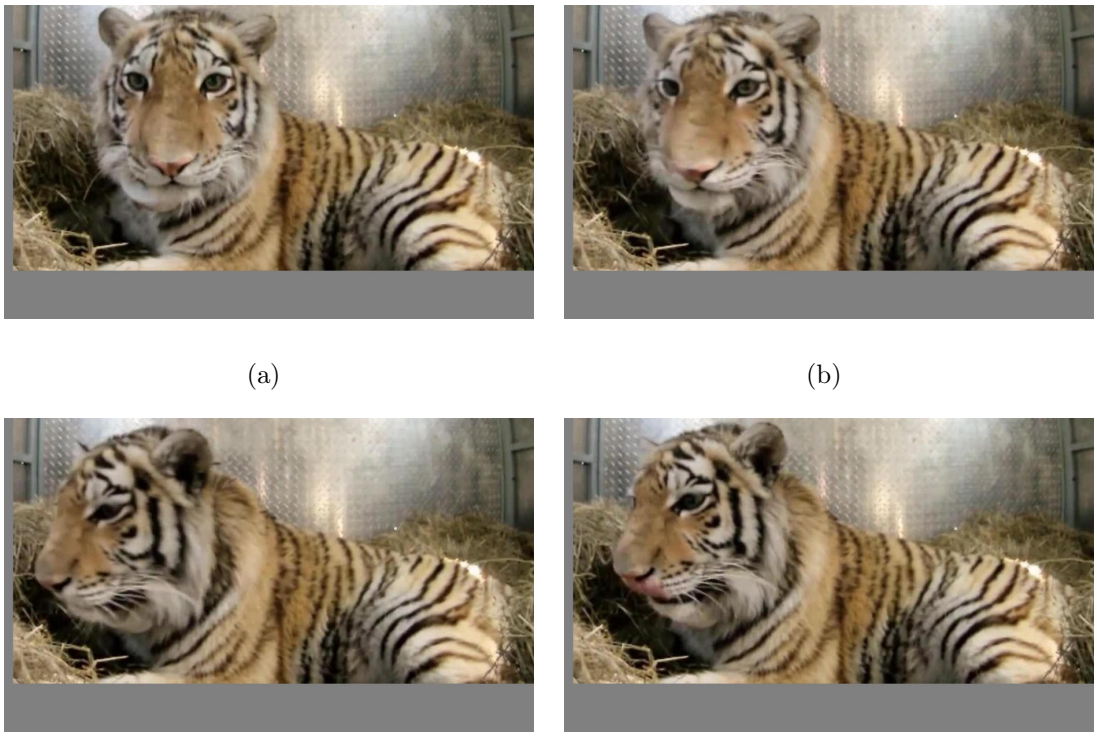
(a) (b)

(c) (d)

Fig. 5.3 Random cropping for data augmentation.

Affine transformation: Generally, affine transformation [67] includes rotation, translation, shear, and scale. In our work, we utilize translation and scale. We first calculate the matrix corresponding to the random translation and scale transformation. Similarly, we calculate the max bounding box to prevent the loss of the target and the translation will

preserve the max bounding box. We apply constant padding to keep the size of frames fixed. Constant padding is simple and effective in image data augmentation [63]. It preserves the spatial dimensions of the image post-augmentation. The color of padding is defined as (128, 128, 128). The result of affine transformation are depicted in Fig. 5.4.



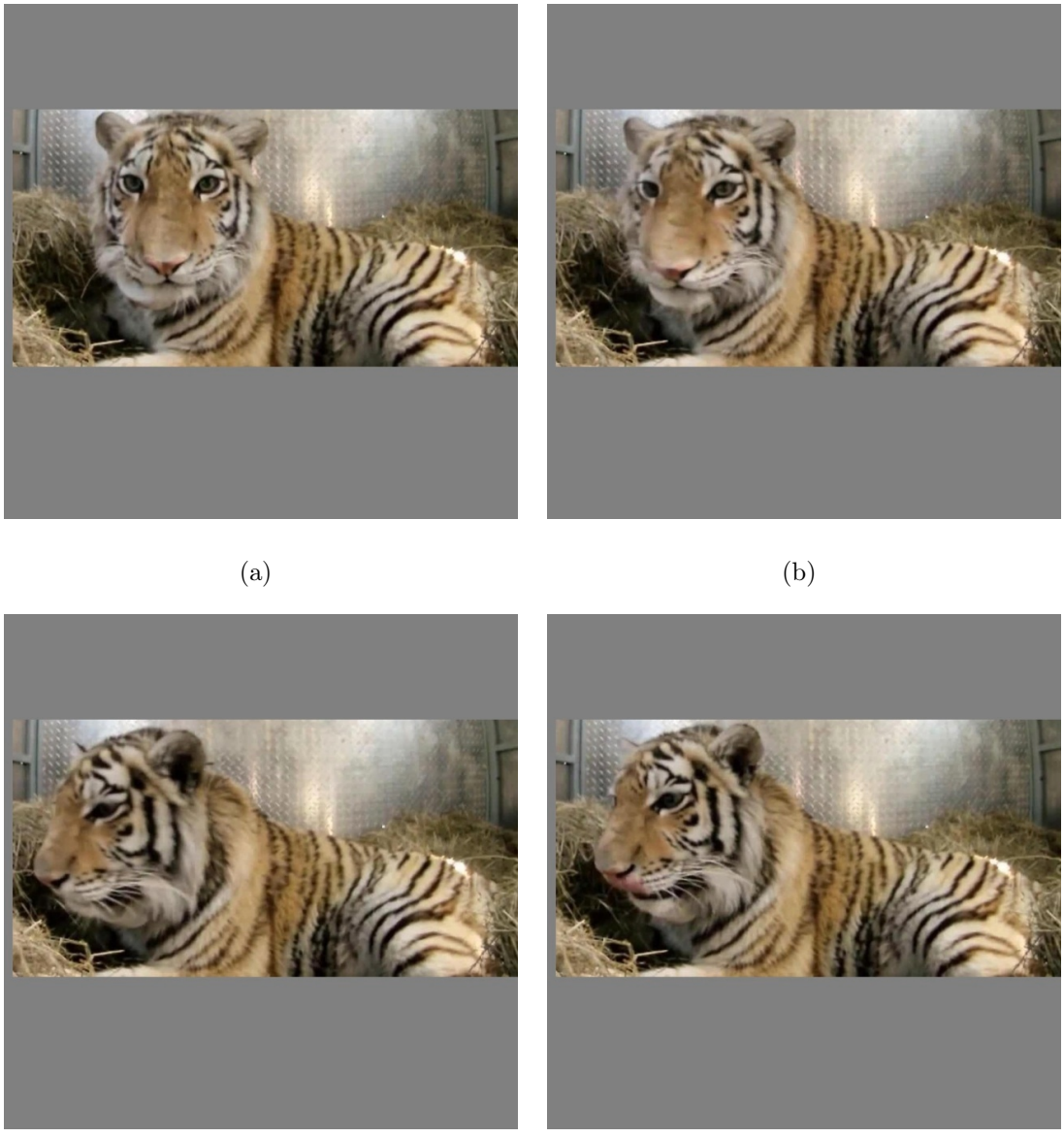
(a) (b)

(c) (d)

Fig. 5.4 Random affine transformation for data augmentation.

Resizing: The size of the frames in the input video clip should be uniform. We choose the minimum of the height to width for resizing. Then we resize the input frames. The expected shape of the frame in the input is square. We apply padding to fill the resized image. The padding color is also (128, 128, 128). In the thesis, padding color in data augmentation is identical to prevent various padding color. The frames are normalized at this step.

Mixup: Mixup has been proposed as a simple solution to improve the robustness of a model [64]. From experimental results [64], mixup improved the performance of current state-



(a) (b)

(c) (d)

Fig. 5.5 Resizing images for data augmentation.

of-the-art neural networks. Essentially, mixup trains neural networks on convex combinations of paired samples and their labels. Therefore, mixup enhances the linear expression between training samples in neural networks. To be specific, mixup can be described as:

$$\tilde{f} = \lambda f_i + (1 - \lambda) f_j, \tag{5.1}$$

$$label_{\tilde{f}} = \lambda label_i + (1 - \lambda) label_j, \tag{5.2}$$

where $\lambda \sim Beta(\gamma, \beta)$, and $\gamma = \beta = 1.5$ in our work. f_i and f_j are images for mixup. $label_i$ and $label_j$ indicate the one-hot label encodings for the corresponding image.

For the implementation of mixup in our experiment, we randomly select another video clip for the input. We apply mixup to each pair of the two video clips. For example, suppose the two video clips are $(f_{t-3}, f_{t-2}, f_{t-1}, f_t)$ and $(m_{t-3}, m_{t-2}, m_{t-1}, m_t)$. The mixup is applied to four pairs that are (f_{t-3}, m_{t-3}), (f_{t-2}, m_{t-2}), (f_{t-1}, m_{t-1}) and (f_t, m_t). The mixup results is depicted in Fig. 5.6. We retain the temporal relationship of the two video clips in the mixed video clips. The mix-up in our experiment can be referred as temporal mix-up.

5.2 Training Procedure

Experimental models were trained over 100 epochs on the same dataset. The training procedure consists of two phases. First, we pretrain the backbone Darknet-53 with octave convolution layers on the Imagenet2017 VID dataset in order to obtain a good initialization of the backbone weights. It is more efficient to train our network with pretrained backbone weights as fewer overall epochs are required. The pretrained weights are produced from 50 epoches training on the dataset. The input to our detector is shuffled frames of video clips. We increase the size of the training images to (512,512) in data augmentation.

For training of our network, we have to reshape the video clips from a 3D tensor to a 2D tensor. The size of our video clips is (n, h, w), where n is the number of frames in the video clip. We split the video clip into n tensors of (h, w). Then we concatenate them in

(a) (b)

(c) (d)

Fig. 5.6 Temporal Mix-up. f_i is an image of a tiger; f_j is an image of a horse. The dark red boxes indicate ground-truth of the tiger; the light green boxes indicate ground-truth of the horse. The label for each dark red boxes is $P_{tiger} = \lambda$; the label for each light greeen boxes is $P_{horse} = 1 - \lambda$; $\lambda \sim Beta(1.5, 1.5)$ in this thesis.

the dimension of the batch. Now the size of the input is $(batchsize \times n, c, h, w)$, where c indicates the channel number.

All experiments and subsequent results were conducted using a NVIDIA GTX 1080ti GPU and deep learing platform FloydHub GPU cloud (Tesla K80 12GB and Tesla V100 16 GB). Models were trained using the PyTorch open source machine learning library [68].

5.2.1 Hyper-parametery Settings

We trained our networks using SGD (Stochastic gradient descent) with momentum [69] for reducing training time. We first transferred the hyper-parameters from Yolov3 [26] to check the feasibility of our concept. Then we empirically tuned the hyper-parameters in Table 5.1. Our batch size is limited by the hardware requirements and may not be optimal. The final settings used are:

Table 5.1 Hyper-parameters settings used for primary training.

Hyper-parameter	**Setting**
Learning Rate	0.0001
Learning Rate Decay	cosine to 0.000001
batch size	2
Training epoches	100
Momentum	0.9
Weight decay	0.0005

For updating parameters of our model, we use SGD with momentum as the optimizer. The momentum is 0.9 and weight decay is 0.0005. The learning rate starts from 0.0001 and ends with 0.000001. For learning rate decay, we utilize cosine decay for adjusting learning rate while training.

5.2.2 Loss Function

The loss in object detection tasks is composed of the losses for classification and regression. To be specific, there are class loss, confidence loss and location loss for training. In Yolov3 [26], GIoU (Generalized intersection over union) [70] is utilized for loss calculation; in Yolov4 [71], CIoU (Complete intersection of union) [72] is used for loss calculation. In this thesis, we utilize CIoU for calculating the location loss as it can better reflect the loss of the center of predicted boxes.

IoU (Intersection over union)

We first describe the IoU loss (L_{IoU}) as it is the bass for GIoU and CIoU.

$$IoU = \frac{P \cap G}{P \cup G}, \tag{5.3}$$

where P indicates predicted bounding boxes; G indicates ground-truth bounding boxes. IoU calculates the ratio of the intersection over union of the two boxes. L_{IoU} can be calculated based on IoU as follows:

$$L_{IoU} = 1 - IoU. \tag{5.4}$$

However, when the IoU of P and G is 0, L_{IoU} will not be updated as it is always $L_{IoU} = 1$. Therefore, the performance of IoU loss in regression tasks is not good.

GIoU [70]

GIoU loss is proposed to solve the problem that L_{IoU} is always 1 when P and G have no intersection. GIoU is defined as:

$$GIoU = 1 - IoU + \frac{A - A^u}{A}. \tag{5.5}$$

where A indicate the area of the smallest enclosing convex for P and G; A^u indicate the area of the union of P and G. Even though A and B do not intersect, GIoU will tend to -1 as the distance between them increases. GIoU loss can be calculated as:

$$L_{GIoU} = 1 - GIoU. \tag{5.6}$$

CIoU [72]

IoU loss is unable to optimize two non-overlapping boxes, and GIoU cannot reflect the distance between the centers of the predicted box and the ground-truth box and their aspect ratio. CIoU was proposed to calculate the loss considering all elements.

CIoU loss provide multi-measurement on the basis of IoU loss, which includes the IoU, the distance of the center points between predicted and ground-truth boxes, and the aspect ratio of the boxes. The CIoU loss (L_{CIoU}) is calculated as:

$$L_{CIoU} = 1 - IoU + R(P,G) = 1 - IoU + \frac{\rho^2(p,g)}{c^2} + \alpha\upsilon, \tag{5.7}$$

$$\upsilon = \frac{4}{\pi^2}(arctan\frac{w^g}{h^g} - arctan\frac{w^p}{h^p})^2, \tag{5.8}$$

$$\alpha = \frac{\upsilon}{(1 - IoU) + \upsilon}, \tag{5.9}$$

where p,g represent the centers of P and G, respectively; ρ represent the Euclidean distance between the two centers p and g; c indicates the diagonal distance of the smallest enclosing convex for P and G; w^g and h^g indicate the width and height of ground-truth bounding boxes; w^p and h^g indicate the width and height of predicted bounding boxes.

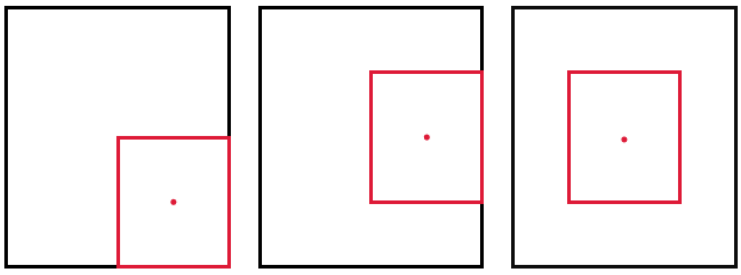


Fig. 5.7 Comparsion of loss function: L_{IoU}, L_{GIoU} and L_{CIoU}.

As depicted in Fig. 5.7, L_{IoU} and L_{GIoU} is equal for the three pairs of predicted boxes and ground-truth. Obviously, L_{IoU} and L_{GIoU} cannot provide the relative position relationship

between the prediction and ground truth. Compared with L_{IoU} and L_{GIoU}, L_{CIoU} allows more accurate learning for bounding boxes regression.

Class Loss

We use binary cross-entropy loss to calculate the loss for each class and calculate the sum. The binary cross-entropy loss can be calculated as:

$$L_{bce}(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}). \tag{5.10}$$

For each class, we calculate the sum of L_{bce} for each sample in training data as the loss for the class.

$$L_{cls} = \sum_{i=1}^{m} L_{bce}(\hat{x_i}, x), \tag{5.11}$$

where m is the sample.

Confidence Loss

Confidence loss is to calculate the loss for the prediction of objects. Confidence loss is divided into two parts, confidence loss with target and confidence loss without target. We calculate confidence loss based on binary cross-entropy as follows:

$$L_{conf} = \sum_{i=0}^{s^2} \sum_{j=0}^{B} \ell_{ij}^{obj} L_{bce}(\hat{C_i^j}, C_i^j) + \lambda_{noobj} \ell_i^{noobj} L_{bce}(\hat{C_i^j}, C_i^j), \tag{5.12}$$

where s^2 is the number of grid cells (See Chapter 2); B indicate the predicted bounding boxes; ℓ_{ij}^{obj} represents the object mask (denotes if object appears in grid i); ℓ_{ij}^{noobj} can be calculated as $(1 - \ell_{ij}^{obj})$. C_i^j is the confidence score; $\hat{C_i^j}$ is decided by the intersection of the predicted bounding box and ground truth. If the predicted bounding box has the maximum IoU comparing with other predicted boxes of the grid cell, $\hat{C_i^j} = 1$; Or else $\hat{C_i^j} = 0$ When the target is not detected, there is the lowest confidence prediction penalty λ_{noobj}.

Overall Loss

We utilize the overall loss in Yolov4 [71] in this thesis. The overall loss is the sum of L_{CIoU}, L_{cls} and L_{conf}. L can be calculated as:

$$L = L_{CIoU} + L_{cls} + L_{conf}. \tag{5.13}$$

L considers the classification loss L_{cls}, the location loss L_{CIoU} and prediction loss L_{conf}. Without L_{conf}, L shows no difference between corrent object prediction and wrong object prediction when P and G have no intersection.

5.3 Quantitative Results

We train our model in ImageNet VID2017 dataset for 100 epochs using pretrained weights. At inference, an NMS of 0.5 IoU threshold is adopted to suppress duplicate detection boxes. We utilize mAP(mean Average Precision%) as our evaluation metric to measure the performance of the proposed network architecture. We also provide the inference time for processing each frame to illustrate the network latency.

We also compare our network with four distinct networks which are Yolov3 [26], a_LSTM [10], REPP+Yolov3 [51], FGFA [20] and MEGA [49] and STMN [50]. All networks were fine-tuned using publicly available pretrained weights. Yolov3 is for object detection; the a-LSTM model is for online video detection; REPP+Yolov3, FGFA, STMN and MEGA are for offline video detection. FGFA is a fundamental approach for recent feature aggregation policies. FGFA proposed flow-guided feature aggregation. The MEGA model is a state-of-the-art method in offline video detection tasks. MEGA is based on global and local feature aggregation. STMN propagates features across frames and calculates the long-term spatial-temporal memory, which also utilize feature alignment. MEGA, FGFA, a-LSTM, STMN and our model are all based on feature aggregation. The results are therefore indicative of the abilities of the proposed feature aggregation module. REPP evaluates the similarity of prediction between frames, and corrects the prediction of the current frame. REPP is the only post-processing method for video detection in our comparison. As REPP+YOLOV3 and our work utilize Yolov3 as the basic detector, the comparison between them can show the performance gap between REPP and our feature aggregation module. The purpose of these experiments is to evaluate our designs against popular networks.

For each network, we present performance results including mAP(%) and inference time. Table 5.2 shows the result comparison between models without any post-processing. Among all methods, MEGA [49] achieves the best performance. With a ResNet-101 backbone, MEGA can achieve 82.6% mAP. The mAP of STMN [50] is also over 80%. FGFA also achieves high performance using aggregation based on optical flow. Among all competitors, only a-LSTM and our method are designed for online video detection. Our method outperforms a-LSTM, REPP+Yolov3, and Yolov3 with 75.8% mAP. By replacing the backbone feature extractor from Darknet-53 based on octave convolution with Darknet-53 based on an improved octave convolution, our method achieves better performance of 76.8% mAP. With spatial attention module, our method can achieve 77.1% mAP.

The methods for video detection in Table 5.2 are mostly based on multiple feature aggregation method. From the results, we can conclude our feature aggregation based on the temporal network outperforms a-LSTM module. Compared to the a-LSTM module, 3D convolution can learn richer spatial-temporal features. STMN and FGFA calculate the feature of the current frame based on 11 neighbor frames or even more, while our work is only based on 4 neighbor frames. Therefore, the mAP gap between our method and them is from the information gap. Our method can also achieve a mAP over 80(%) (section 5.5.3) when we utilize TemporalNet-16 with a larger memory usage. However, our work aims to be real-time and at limited hardware requirements. The mAP gap is affordable for the accuracy/speed/memory usage trade-off.

We utilize Yolov3 as our basic detector in the thesis. It is a representative method in one-stage object detection. The frame rate is 37 FPS (Frames Per Second) with the resolution of 512×512 i.e. it is real-time. Our model is also real-time with running at 29 FPS. MEGA and FGFA may suffer from the latency of a two-stage detector and the complex architecture which lead to extra time cost. Our model is fast and also competitive with offline video detectors. Other methods utilize 2D convolution networks for feature extraction. Our model is the only one that uses 3D convolution networks for feature extraction in video clips.

Table 5.2 Performance comparison with state-of-the-art video object detection models on ImageNet VID validation set. TemporalNet-4 is the temporal network using 4 frames.

Methods	**Online**	**Backbone**	**mAP(%)**	**Inference time(ms)**
Yolov3 [26]	✓	Darknet-53	68.5	27
FGFA [20]	–	ResNet-101	76.9	733
MEGA [49]	–	ResNet-101	82.6	115
STMN [50]	–	ResNet-101	80.6	80
a-LSTM [10]	✓	SSD	72.3	83
REPP+Yolov3 [51]	–	Darknet-53	74.8	30
TemporalNet-4	✓	Darkenet-53 +Octave conv	75.8	35
	✓	Darkenet-53 +Selective-frequecy Octave conv	76.8	35
	✓	Darkenet-53 +Selective-frequecy Octave conv +spatial attention [13]	77.1	35.5

5.4 Qualitative Results

Here we present our final qualitative results of our network, which is trained for a total of 100 epochs. We selected three pairs of results for the comparison with Yolov3 (the baseline), STMN and a-LSTM. Each pair of results are predicted on 4 consecutive frames. We visualize the prediction by drawing bounding boxes. In Fig. 5.8 through Fig. 5.9 we include the example detection results of different methods. We can compare the performance qualitatively. As the parameters of our comparators may not be perfect, the performance may be not optimal.

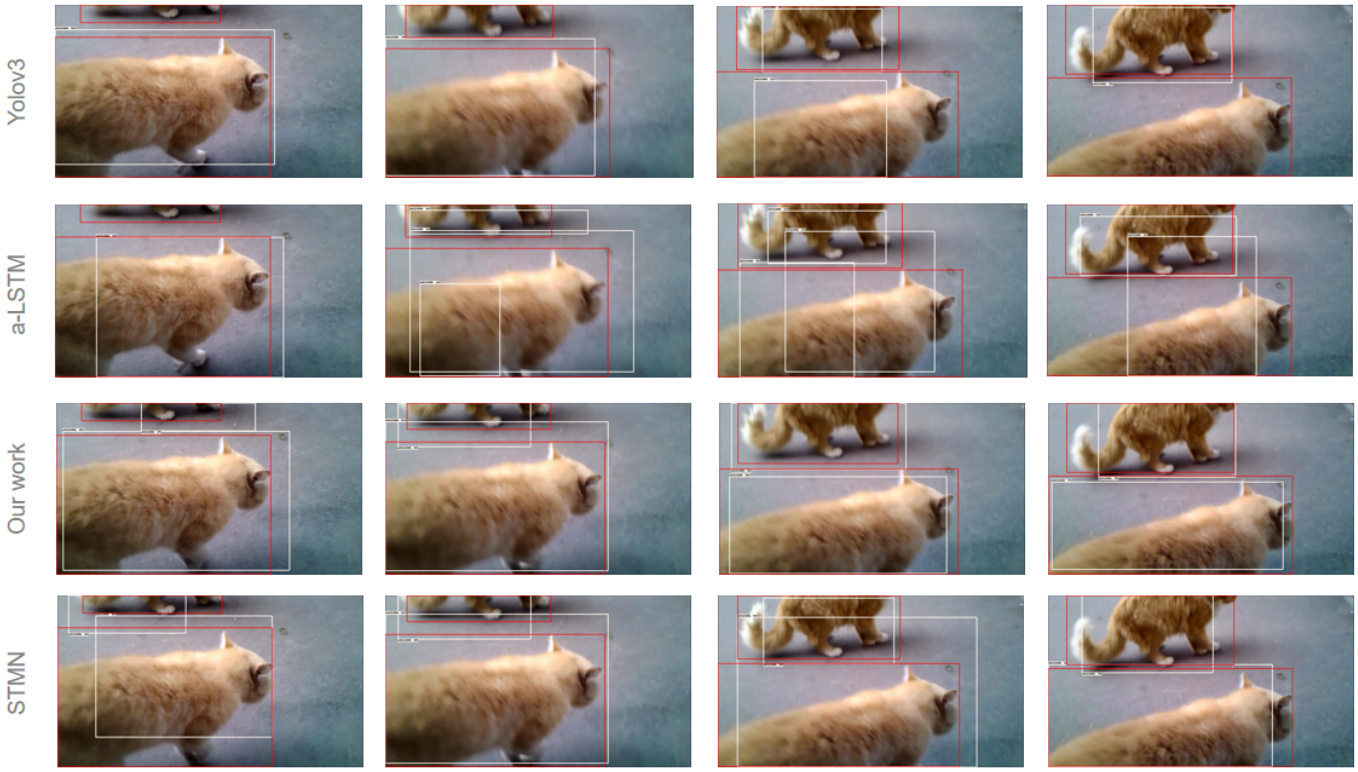


Fig. 5.8 Example detection results of networks (red boxes indicate ground-truth).

As shown in Fig. 5.8, a-LSTM, STMN and our work detected the partially visible cat while Yolov3 fails to detect the cat when not enough of the cat is visible. The feature aggregation of neighboring frames are responsible for the difference in video detectors and single-frame detectors. Compared to a-LSTM, the quality of bounding boxes predicted by

our work is better. There is no obvious bounding-box quality gap between our work and STMN in this sequence.

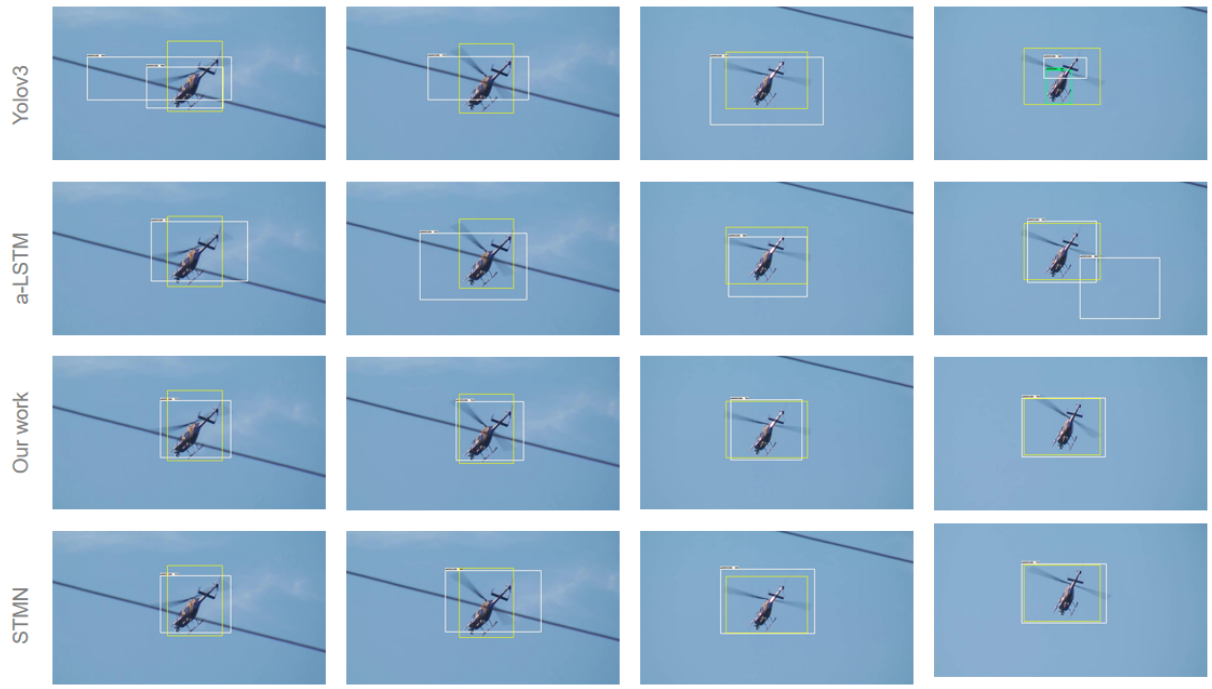


Fig. 5.9 Example detection results of networks (yellow boxes indicate ground-truth).

As shown in Fig. 5.9, the bounding-box quality of Yolov3 is not satisfactory due to misdetections. The incorrect box of a-LSTM may result from the non-optimal parameters. The performance of our work is the best in this example.

In Fig. 5.10, we present the prediction results along with previous frames to compare the detection quality. The picture on the far right is the frame to be detected. There is an occluded bird and the bird appears on the left three pictures. Yolov3 fails to detect it as no feature aggregation method is applied. a-LSTM fails to detect the occluded bird, while our work and STMN detecteted it. As Yolov3 is a state-of-the-art detector, the failure of it means detection based on the current frame only is challenging. The misdetection of a-LSTM means the memory of previous frames is still not capable of predictions in occlusion. The feature aggregation policy of our work is capable of aggregating corresponding features across frames as well as STMN. Our method and STMN utilize feature alignment. Therefore, it

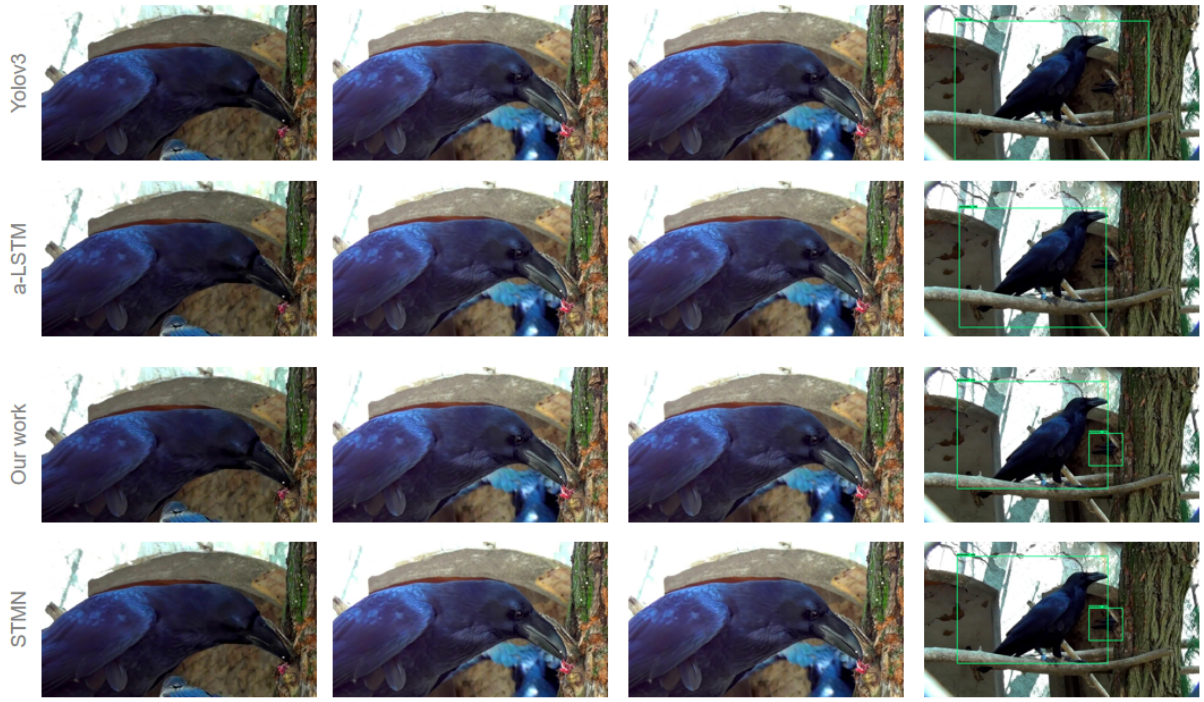


Fig. 5.10 Example detection results of different feature aggregation methods.

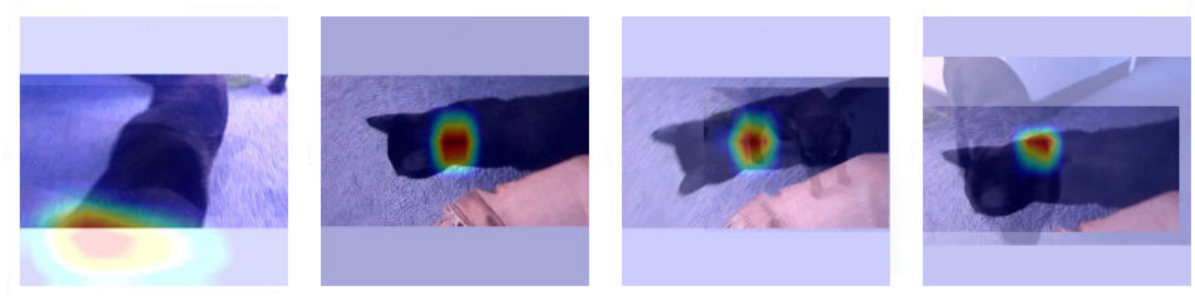
demonstrates the ability of feature alignment in the respective feature aggregation module.

We expect our model to extract features based on the similarity distribution of neighboring frames. It can help with occlusions and blurring which are challenging in video detection. Fig. 5.11 (a) and (b) show our temporal network can locate the corresponding region in neighboring frames for feature alignment. Fig. 5.11 (c) shows the performance of our model in blurred frames. The prediction is not influenced by the quality of the current frame.

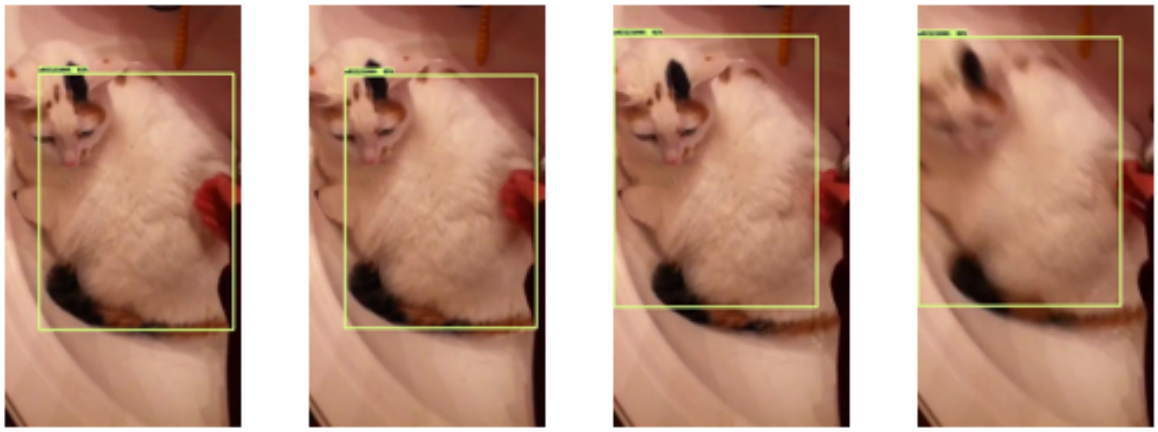
5.5 Ablation Study

This section is composed of component analysis and parameter comparison. To examine the impact of the key components and architecture-related parameters in our model, we conduct extensive experiments to study how they contribute to our final performance. We utilize the same training procedure and dataset for the ablation study.

(a) Visulisation of similarity distribution of a pair of consecutive frames. Feature alignment is applied in our temporal network. Our temporal network finds similar areas across frames based on AP3D. As shown in (a), we locate areas that have a high response to the center area of the first frame in the following frames.



(b) Visulisation of similarity distribution of another pair of consecutive frames. We locate the similar area of the black cat across frames in (b).



(c) Example detection results. The last picture is blurred. However, our method detect it correctly based on the appearance information from the three left frames.

Fig. 5.11 Example performance of our work.

5.5.1 Backbone Improvement due to Octave Convolution

We utilize Darknet-53 as the base of our backbone. We replace convolution layers in Darknet-53 with octave convolutions layers in our backbones. Octave convolution processes high-frequency and low-frequency features separately. It stores low-frequency features in a down-sampling size to decrease the memory usage. We expect our backbones based on octave convolution to be faster and more efficient. Table 5.3 presents the results of models based on Darknet-53 and on our backbone (Darkent-53+Octave conv).

Table 5.3 Ablation study on Octave convolution in our backbone.

Backbone	**TemporalNet**	**mAP(%)**	**Inference time**
Darkent-53	TemporalNet-4	75.4	34
Darknet-53+Octave conv	TemporalNet-4	75.8	35

From Table 5.3, a gap of 0.4% mAP exists between the two models. Our backbone is more accurate than the base backbone. However, the time cost of our backbone is higher than Darknet-53, which is not as we had expected. After the inspection, we believe the the first octave convolution layer and last octave convolution followed by and following the communication between the 3D convolution layer and the 2D convolution layer is responsible for the extra time cost.

Based on the channel attention module (See Chapter 4), we propose the selective-frequency octave convolution layers. In our experiments, the parameter α of octave convolution is 0.125 for separating high-frequency and low-frequency features. However, the parameter should be adaptive to the input. We design a selective-frequency layer for the dynamic fusion of high-frequency and low-frequency features. From the results in Table 5.2, a gap of 1.0% mAP exists between our backbone based on the octave convolution and the improved octave convolution. There is little extra time cost. We can also observe the 0.4% mAP improvement from the spatial attention. We can conclude that the attention module

can help our network to focus on useful information.

5.5.2 Effect of Temporal Aggregation

We propose the TemporalNet to extract temporal information from consecutive frames in our work. It can enhance the features of the current frame referring to features of previous frames. Table 5.4 shows the results of our models.

Table 5.4 Ablation study on our temporal network.

Backbone	**TemporalNet**	**mAP(%)**	**Inference time(ms)**
Darkent53+improved Octave conv +spatial attention	–	68.7	26
	TemporalNet-4	77.1	35.5

As shown in Table 5.4, our temporal network provides a large improvement (8.4% mAP) on the performance of our model. The time cost of temporal network is mainly from 3D convolution layers. The accuracy/speed trade-off is also a major concern in designing our real-time online model. Compared to the state-of-the-art models, our proposed temporal network is accurate and fast. Our model processes 4 frames in 140ms. The extra time cost is $(140 \div 4) - 26 = 9$ms for aggregating features across frames using our temporal network.

Our proposed temporal network aggregates features in the temporal dimension which leads to the large improvement from the Yolov3 single-frame base detector. The result shows the ability of our temporal network. The temporal network based on 3D convolution learns features in the temporal dimension. Our backbone based on 2D convolution can extract spatial features. Our model is accurate and fast as we aggregate appearance information in the temporal dimension for enhancing the feature of the current frame.

5.5.3 Local Feature Aggregation

Local feature aggregation indicates the feature aggregation across neighboring frames in this thesis. The local span of our aggregation policy decides the architecture of our model and the performance. The local span can be 4, 8 and 16 referring to C3D [1] in this thesis. Correspondingly, the temporal networks are the TemporalNet-4, TemporalNet-8 and TemporalNet-16 (See section 4.4). We provide the performance comparision between them in Table 5.5.

Table 5.5 Ablation study on local feature aggregation. Number after "TemporalNet-" indicates the local span.

TemporalNet	**Backbone**	mAP(%)	Inference time
TemporalNet-4	**Darkent53+improved Octave conv +spatial attention**	77.1	35.5
TemporalNet-8		79.2	47
TemporalNet-16		80.9	57

As shown in Table 5.5, the performance of the TemporalNet-16 is best in terms of mAP. And the performance of our temporal network drops as the local span narrows. Obviously, the wide local span provide more information for feature aggregation. The wide local span can provide global information, which may be the reason for the mAP gap between our temporal networks. Similar to STMN [50], we utilize feature alignment for local feature aggregation. From the results in Table 5.2 and Table 5.5, our method based on TemporalNet-16 can achieve better mAP (80.9%) than STMN(80.6%). Our method is also faster.

Accuracy, speed and memory usage trade-off

We aim to design a real-time video detector in our research. We need to balance speed requirement and accuracy. The TemporalNet-16 achieves the best performance with the highest time cost. From the experimental results on the 1080Ti, our model based on TemporalNet-16 runs at 15 FPS, which is slow for real-time processing. The TemporalNet-8

achieves high-quality results with an improvement of 2.1% mAP over TemporalNet-4. It runs at 21 FPS in the experiment. TemporalNet-4 can run 28 FPS and the mAP is better than another online video detector (a-LSTM).

Moreover, the memory usage is a limitation of our method. We utilize the appearance-preserving 3D convolution for the feature alignment in our temporal networks. However, the memory usage is increasing exponentially while calculating the offset map. For training the model using TemporalNet-4, the memory usage is 4.47 GB. The training of the TemporalNet-8 and TemporalNet-16 requires memory of 9.11 and 18.25 GB. We aim to design a network that can be trained on a single GPU PC, and therefore we decided to use the TemporalNet-4 in our model.

Table 5.6 Memory requirement for training (batch size is 1)

TemporalNet	Memory Requirment(GB)
TemporalNet-4	>4.47
TemporalNet-8	>9.11
TemporalNet-16	>18.25

5.6 Summary

In this chapter we have described the details of our data processing and training process. We included the loss function and the hyper-parameter selection. We evaluated our methods qualitively and quantitatively. We compared our model with state-of-the-art networks based on commonly-used metrics in video detection. We also designed an ablation study for analysing our main components and parameters for our proposed network. We discussed the trade-off among accuracy, speed and memory usage. We presented the improvements and challenges observed in our experiments.

From results, our online video detection network using TemporalNet-4 outperforms another online video detector (a-LSTM) and offline REPP+Yolov3 in accuracy. Our network with TemporalNet-16 outperforms state-of-the-art offline video detectors (FGFA, STMN and REPP+Yolov3). Our network with TemporalNet-8 can be competitive with mentioned offline video detectors. Our network with TemporalNet-4 is a real-time online video detector which runs at 28 FPS.

From the ablation study, the improvements from our selective-frequency octave convolution and local feature aggregation based on the temporal network are witnessed. Our method achieves high-quality feature extraction and feature aggregation.

The size of a original image in the dataset is 1280×720, which is also the size of a video frame in practical use. However, we utilize images of 448×448 as input frames in our thesis. On average, the time cost of resizing 4 frames is 12ms.

Chapter 6

Conclusion

The research in this thesis arose after a review of the deep learning literature in video detection led to the realization that little work had been done based on 3D convolution and one-stage detectors. Today, real time detection methods cannot provide competitive detection accuracy in videos; while state-of-the-art video detectors have high run-time costs. Practical use of video detection networks often requires online and (close to) real-time performance. We explored if we can utilize a one-stage detector to replace the widely-used two-stage detection networks for higher speed. We also exploit the use of aligned 3D convolution in the field of video detection. We take the opportunity to combine state-of-the-art components in the areas of single-image object detection, in learning temporal information and in convolution layers for a high-performance real time video detection network.

6.1 Thesis Summary

We build a deep learning model capable of real-time online detection in videos.

In Chapter 2, we introduce related concept and provide a review of related work in object detection. We also described online video detection to set the stage for the specific work that must be done in our thesis. And we also provided a discussion of 3D convolution,

interpreting how 3D convolution outperforms 2D convolution in video processing. Then we provided a fast-paced overview of convolutional networks. We review 3D CNNs that provide inspiration for our temporal network. We also reviewed single-image and video object detection networks, respectively. We introduce the four video detectors for comparison in this thesis. We also provide the introduction of keyframe selection as it is key to the approach of some competitors.

In Chapter 3, we describe the components that are utilized in this thesis. We introduce the architecture of Yolov3 and AP3D, which are essential components of detection module and temporal network in this thesis. We also discuss CBAM (Convolutional Block Attention Module) as we utilize its spatial attention module.

In Chapter 4, we presented our major contributions starting with a description of the baseline. Then we provided the overview architecture of our network, describing the data flow. It illustrates how 3D and 2D components cooperate for detection in videos. We described elements of our workflow in details. First, we introduced our backbone based on selective-frequency octave convolution. We imported the improved octave convolution into Darknet-53 and provide a mathematical description of the selective-frequency octave convolution layer. Then we provided the mathematical description of our feature aggregation policy. We also introduced the architecture of our temporal network for feature aggregation.

Finally in Chapter 5, we conducted an experimental evaluation of our work and discuss the results. We started by describing the dataset for our experiments. Then we outlined data augmentation methods used to improve the training performance. We described the training of our model, and illustrated the hyper-parameter setting and loss function. We also presented a comparison of quantitative performance between our approach and other detectors. We also presented the qualitative results comparison between our work and four other methods. We visualized the cross-frame similarity distribution to interpret our local feature aggregation. To illustrate the performance of key components of our work, we provided ablation studies of our improved octave convolution, temporal network and local

feature aggregation. Finally, we described the accuracy/speed/memory usage trade-off to introduce the decision of the parameters for our model.

6.2 Contributions

In this thesis we have developed real-time online video detection network based on an interleaved 3D-2D convolution network. We utilized successful components in an aligned 3D convolution network and one-stage detection network in our work. We designed a multi-scale interleaved architecture for integrating the components. We also imported improved state-of-the-art octave convolution into our network. We used various techniques to improve the training performance. These techniques included horizontal-flipping, cropping, affine transformation and mixup for data augmentation. Our network outperforms YOLOv3 [26] (online), a-LSTM [10] (online), REPP+YOLOv3 [51] (offline) and FGFA [20] (offline) with TemporalNet-4. Our network outperforms STMN [50] (offline) with TemporalNet-16.

For video detection, the two main challenges were feature extraction and feature aggregation. We designed an efficient backbone for feature extraction and a temporal network for feature aggregation. Previously, research on video detection was mainly based on two-stage detectors. Our work used 3D ConvNet to work with an one-stage detector and provided competitive performance. This success answers our primary research inquiry which sought to determine if it was possible to improve still-image detection networks for detection in videos. We also demonstrate that the inference time for video clips can be practical to use in online tasks by one-stage detection. In developing our network we have made the following contributions:

1. We have improved the existing backbone for more efficient feature extraction based on selective-frequency octave convolution.

2. We designed the temporal network for feature aggregation across neighboring frames based on AP3D.

3. We developed a multi-scale interleaved communication between our backbone and temporal network for feature aggregation between neighboring frames.

4. We replaced the backbone of Yolov3 with our backbone and enabled the communication between our backbone and our temporal network. We replaced the loss function of Yolov3 with CIoU, improving the training performance.

6.3 Limitations

Despite these encouraging results there are a number of limitations to this work in its current state. Here we describe the major limitations and weaknesses:

Network Design

Our network performs detection based on Yolov3. Our thesis focused on improving one-stage detector YOLOv3 for detection in videos. It limited our network's design with respect to architecture of our basic detector. We designed the temporal network that obviously is useful for feature aggregation across frames. However, the potential of our detector is not fully explored by not comparing different base detectors.

We pretrained our basic detector with the improved octave convolution and accommodate the weight file for the training of our whole network. The pretraining allows for significantly less training time and enable attention to the optimization of our temporal network. Undoubtedly, the quality of our detector can decide the performance of our feature aggregation and prediction. While designing our network, we haven't changed the fundamental architecture of our basic detector for better performance. We expect a higher-quality detector to bring improvements to the overall accuracy.

Training

As we mentioned in Chapter 5, most experiments were conducted using a single NVIDIA GTX1080ti GPU with 8GB of memory. The batch size for training is 2 which is likely too small. As a result, additional experiments are required to find an optimal performance using

a GPU with more memory.

Memory Usage

Our video detection network can be fast but the memory usage for training should be reduced. By inspection, memory usage is mainly from calculation of AP3D. AP3D includes 3D convolution and feature alignment, which leads to high consumption of memory. Research into an efficient feature alignment method to be applied in 3D convolution would greatly benefit our method.

6.4 Future Work

We expect machines to eventually perform better than humans in video detection tasks. However, it may be still a long-term research goal. We can start by going beyond the existing limitations. Our network can be faster than other video object detection networks with similar accuracy. However, the accuracy of our work is not as high as state-of-the-art offline methods. There are also a number of near term goals that merit further exploration:

• Our network only aggregates features from neighboring frames. However, the method may benefit from the information in frames in previous neighborhoods. A global feature aggregation method may lead to better performance. To our knowledge, there are only a few video detection networks that use global information and they achieve state-of-the-art accuracy [49, 73]. Global feature aggregation mimics the long-term memory of the human visual system. It can possibly lead to big performance improvement.

• Our work utilizes YOLOv3 as basic detector for predicting bounding boxes. However, newer one-stage single-image detection networks are performing even better. Using a better detector in our work can be the next step in the future work for optimized performance. For example, YOLOv4 [71] and YOLOv5 (no published paper is available but the code is availble on Github) have been proposed very recently.

• For most work in video detection, the utilization of a key-frame strategy enables an

efficient resource allocation. These works apply accurate detection methods to key-frames, and the predictions on non-key frames are produced from key-frame results. The strategy saves time and lowers compute demands. It is possible to implement our feature aggregation on adjacent key-frames. Our feature aggregation is based on a temporal network that requires most of the computing resources. Using this method, the calculations for our temporal network can be reduced. Due to the redundancy between adjacent frames, key-frame feature aggregation may provide competitive results with the results of our work.

- As mentioned in Chapter 5, our temporal network has a high memory consumption. By inspection, the feature alignment method of AP3D is responsible for a large number of calculations. We like to research a more efficient feature alignment method to be applied in 3D convolution or improve the existing feature alignment method.

References

[1] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4489–4497, 2015.

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[3] Jonathan Stroud, David Ross, Chen Sun, Jia Deng, and Rahul Sukthankar. D3d: Distilled 3d networks for video action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 625–634, 2020.

[4] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

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

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

[7] 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. (Last accessed: 11.05.2020).

[8] 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/CVF International Conference on Computer Vision*, pages 7023–7032, 2019.

[9] 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 3038–3046, 2017.

[10] Yongyi Lu, Cewu Lu, and Chi-Keung Tang. Online video object detection using association lstm. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2344–2352, 2017.

[11] Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3435–3444, 2019.

[12] Xinqian Gu, Hong Chang, Bingpeng Ma, Hongkai Zhang, and Xilin Chen. Appearance-preserving 3d convolution for video-based person re-identification. In *European Conference on Computer Vision*, pages 228–243. Springer, 2020.

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

[14] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

[15] Farhana Sultana, Abu Sufian, and Paramartha Dutta. A review of object detection models based on convolutional neural network. *arXiv preprint arXiv:1905.01614*, 2019.

[16] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458, 2003.

[17] G Sreenu and MA Saleem Durai. Intelligent video surveillance: A review through deep learning techniques for crowd analysis. *Journal of Big Data*, 6(1):1–27, 2019.

[18] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37, 2020.

[19] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

[20] 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.

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

[22] 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. (Last accessed: 15.04.2020).

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

[24] Chen Huang, Simon Lucey, and Deva Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 105–114, 2017.

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

[26] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. (Last accessed: 19.03.2020).

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. (Last accessed: 01.11.2019).

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

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

[30] Donggeun Yoo, Sunggyun Park, Joon-Young Lee, Anthony S Paek, and In So Kweon. Attentionnet: Aggregating weak directions for accurate object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2659–2667, 2015.

[31] Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. In *Asian Conference on Computer Vision*, pages 198–213. Springer, 2016.

[32] 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.

[33] G Sreenu and MA Saleem Durai. Intelligent video surveillance: A review through deep learning techniques for crowd analysis. *Journal of Big Data*, 6(1):1–27, 2019.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[37] Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

[38] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. (Last accessed: 13.11.2019).

[39] Lucas Smaira, João Carreira, Eric Noland, Ellen Clancy, Amy Wu, and Andrew Zisserman. A short note on the kinetics-700-2020 human action dataset. *CoRR*, abs/2010.10864, 2020, (Last accessed: 05.03.2021).

[40] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[41] Behrooz Mahasseni, Sinisa Todorovic, and Alan Fern. Budget-aware deep semantic video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1029–1038, 2017.

[42] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

[43] K Oksuz, BC Cam, S Kalkan, and E Akbas. Imbalance problems in object detection: A review. arxiv e-prints p. *arXiv preprint arXiv:1909.00169*, 2019. (Last accessed: 01.07.2021).

[44] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *arXiv preprint arXiv:1605.06409*, 2016. (Last accessed: 27.06.2020).

[45] 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 the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7814–7823, 2018.

[46] Md Atiqur Rahman Ahad, Joo Kooi Tan, Hyoungseop Kim, and Seiji Ishikawa. Motion history image: its variants and applications. *Machine Vision and Applications*, 23(2):255–281, 2012.

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

[48] 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/CVF International Conference on Computer Vision*, pages 6678–6687, 2019.

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

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

[51] Alberto Sabater, Luis Montesano, and Ana C Murillo. Robust and efficient post-processing for video object detection. *arXiv preprint arXiv:2009.11050*, 2020. (Last accessed: 30.04.2021).

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

[53] 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.

[54] 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.

[55] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. (Last accessed: 15.10.2020).

[56] Robert Desimone. Neural mechanisms for visual memory and their role in attention. *Proceedings of the National Academy of Sciences*, 93(24):13494–13499, 1996.

[57] Congrui Hetang, Hongwei Qin, Shaohui Liu, and Junjie Yan. Impression network for video object detection. *arXiv preprint arXiv:1712.05896*, 2017. (Last accessed: 18.05.2020).

[58] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5659–5667, 2017.

[59] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[60] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

[61] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5783–5792, 2017.

[62] Rongtian Ye, Fangyu Liu, and Liqiang Zhang. 3d depthwise convolution: Reducing model parameters in 3d vision tasks. In *Canadian Conference on Artificial Intelligence*, pages 186–199. Springer, 2019.

[63] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[64] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. (Last accessed: 03.11.2019).

[65] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017. (Last accessed: 01.03.2020).

[66] Luke Taylor and Geoff Nitschke. Improving deep learning with generic data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1542–1547. IEEE, 2018.

[67] Eric W Weisstein. Affine transformation. *https://mathworld. wolfram. com/*, 2004, (Last accessed: 12.12.2019).

[68] 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 *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[69] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234. PMLR, 2016.

[70] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019.

[71] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. (Last accessed: 15.01.2021).

[72] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12993–13000, 2020.

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