Deep Learning-Based Low Complexity and High Efficiency Moving Object Detection Methods

Bingxin Hou

Santa Clara University

Department of Computer Science and Engineering

Date: Mar 1, 2022

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISIOR BY

Bingxin Hou

ENTITLED

Deep Learning-Based Low Complexity and High Efficiency Moving Object Detection Methods

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING

N. Ling N. Ling (Mar 3, 2022 09:15 PST) Thesis Advisor	Tokunbo Ogunfunmi Thesis Reader
Ying Liu Ying Liu (Mar 4, 2022 17:42 PST) Thesis Co-Advisor	Yuhong Liu Thesis Reader
N. Ling N. Ling (Mar 3, 2022 09:15 PST) Department Chair	[illegible] Xiang Li (Mar 3, 2022 08:09 PST) Thesis Reader

Deep Learning-Based Low Complexity and High Efficiency Moving Object Detection Methods

By

Bingxin Hou

Dissertation

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Science and Engineering
in the School of Engineering at
Santa Clara University, 2022

Santa Clara, California

Dedicated to the loving memory of my grandpa

Mr. Yunpeng Liu

Acknowledgements

I would like to express my sincere thanks to my esteemed supervisor – Dr. Nam Ling, for his invaluable supervision, support, and tutelage during the course of this doctorate degree. I'm very grateful for his financial help in funding my tuition and for his navigation through the whole process to finish my degree successfully.

I would like to thank my beautiful co-advisor – Dr. Ying Liu for her excellent advice, guidance and her treasured support which was influential in shaping my experiment methods and critiquing my results. I'm very grateful for her patience, constant help, and continuous support during my studies.

I would also like to thank Dr. Tokunbo Ogunfunmi, Dr. Yuhong Liu, and Dr. Xiang Li for being my committee member and for their valuable suggestions and assistance.

I would like to express gratitude to Kwai. Inc Dr. Yongxiong Ren, Dr. Lingzhi Liu, and Dr. MingKai Hsu for the funding help and their mentorship.

My gratitude extends to the department faculty for the Teaching Assistantship and Scholarship funding opportunities to undertake my studies at the Department of Computer Science and Engineering, Santa Clara University.

I would like to thank my friends, lab mates, colleagues–for a cherished time spent together in the lab, and in social settings. My appreciation also goes out to my parents and my family for their understanding, encouragement, and support all through my studies.

Deep Learning-Based Low Complexity and High Efficiency Moving Object Detection Methods

Bingxin Hou

Department of Computer Science and Engineering
Santa Clara University
Santa Clara, California
2022

ABSTRACT

Moving object detection (MOD) is the process of extracting dynamic foreground content from the video frames, such as moving vehicles or pedestrians, while discarding the non-moving background. It plays an essential role in computer vision field. The traditional methods meet difficulties when applied in complex scenarios, such as videos with illumination changes, shadows, night scenes,and dynamic backgrounds. Deep learning methods have been actively applied to moving object detection in recent years and demonstrated impressive results. However, many existing models render superior detection accuracy at the cost of high computational complexity and slow inference speed. This fact has hindered the development of such models in mobile and embedded vision tasks, which need to be carried out in a timely fashion on a computationally limited platform.

The current research aims to use the technique of separable convolution in both 2D and 3D CNN together with our proposed multi-input multi-output strategy and two-branch structure to devise new deep network models that significantly improve inference speed, yet require smaller model size and achieve reduction in floating-point operations as compared to existing deep learning models with competitive detection accuracy.

This research devised three deep neural network models, addressing the following main problems in the area of moving object detection:

1. *Improving Detection Accuracy by extracting both spatial and temporal information*: To improve detection accuracy, the proposed models adopt 3D convolution which is more suitable to extract both spatial and temporal information in video data than 2D convolution. We also put this 3D convolution into two-branch network that extracts both high-level global features and low-level detailed features can further increase the accuracy.

2. *Reduce model size and computational complexity by changing network structure*: The standard 2D and 3D convolution are further decomposed into depthwise and pointwise convolutions. While existing 3D separable CNN all addressed other problems such as gesture recognition, force prediction, 3D object classification or reconstruction, our work applied it to the moving object detection task for the first time in the literature.

3. *Increasing inference speed by changing the input-output relationship*: We proposed a multi-input multi-output (MIMO) strategy to increase inference speed, which can take multiple frames as the network input and output multiple frames of detection results. This MIMO embedded in 3Dseparable CNN can further increase model inference speed significantly and maintain high detection accuracy.

Compared to state-of-the-art approaches, our proposed methods significantly increases the inference speed, reduces the model size, meanwhile achieving the highest detection accuracy in the scene dependent evaluation (SDE) setup and maintaining a competitive detection accuracy in the scene independent evaluation (SIE) setup. The SDE setup is widely used to tune and test the model on a specific video as the training and test sets are from the same video. The SIE setup is designed to assess the generalization capability of the model on completely unseen videos.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Overview of Moving Object Detection

With the increasing amount of network cameras, produced visual data and Internet users, it becomes quite challenging and crucial to process a large amount of video data at a fast speed. Moving object detection (MOD) is the process of extracting dynamic foreground content from the video frames, such as moving vehicles or pedestrians, while discarding the non-moving background. It plays an essential role in many real-world applications [1], such as intelligent video surveillance [2], medical diagnostics [3], anomaly detection[4], human tracking and action recognition [5, 6].

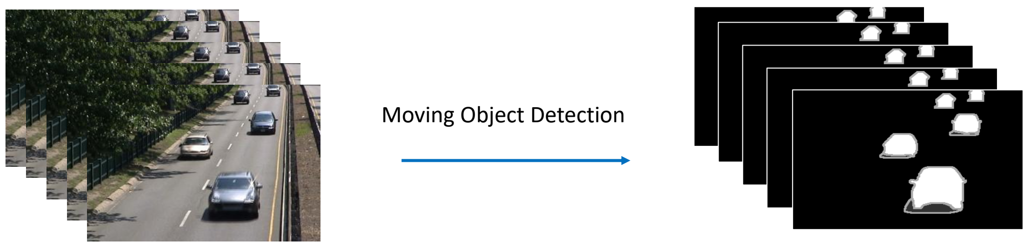


Figure 1.1: Moving Object Detection (Background Subtraction).

Traditional methods [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29] are unsupervised which do not require labeled ground truth for algorithm development. They usually include two steps: background modeling and pixel classification. However, these traditional methods meet difficulties when applied in complex scenarios, such as videos with illumination changes, shadows, night scenes, and dynamic backgrounds.

With the availability of a huge amount of data and the development of powerful computational infrastructure, deep neural networks (DNNs) [30, 31, 32, 33, 34] have shown remarkable improvements in MOD problems and are developed to replace either background modeling or pixel classification in traditional methods or to combine these two steps into an end-to-end network. Existing DNN models are mostly supervised approaches based on 2D convolutional neural networks (CNNs)[35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52], 3D CNNs [53, 54, 55, 56, 57, 58], 2D separable CNNs [59], or generative adversarial networks (GANs) [60, 61, 62, 63, 64, 65]. Besides, unsupervised GANs [66, 67] and semi-supervised networks are also proposed [68, 69, 70, 71, 72, 73, 74, 75]. It demonstrates that the DNNs can automatically extract spatial low-, mid-, and high-level features as well as temporal features, which turn out to be very helpful in MOD problems. Recently, 3-dimensional convolutional neural network (3D CNN) was also proposed to learn the spatial and temporal features simultaneously, which are more suitable and effective in video-related tasks [54, 55, 57, 76, 77, 78].

1.2 Purpose of the Research

While existing DNN models offer superior moving object detection accuracy, they suffer from computationally expensive and memory-intensive issues. In particular, big model size and high computational complexity make it challenging to apply these models to real-world scenarios, such as robotics, self-driving cars, and augmented reality. These tasks are usually deployed on mobile and embedded devices, which have limited memory and computing resources. Besides, these tasks are delay-sensitive and need to be carried out in a timely manner, which cannot be achieved by high-complexity deep learning models. Thus, we aim to design a deep moving object detection model suitable for mobile and embedded environment, that can achieve faster inference speed and smaller model size while maintaining high detection accuracy.

In this research, we propose three deep learning models tailored for computation-resource-limited and delay-sensitive applications : (1) 2D separable CNN - based efficient neural network; (2) An efficient 3D separable convolutional neural network with a multi-input multi-output strategy called "3DS_MM"; (3) A fast two-branch 3D separable CNN called "F3DsCNN".

According to the analysis for supervised methods in [52], we can further categorize experimental setups into scene dependent evaluation (SDE) setup and scene independent evaluation (SIE) setup. The difference on data-division strategies between SDE and SIE setup is shown in Table 5.5 in Chapter 5. In SDE, frames in training and testing sets are from the same video or video sequences, whereas, in SIE, completely unseen videos are used for testing. In our experiments, we use both SDE and SIE setup.

1.3 Outline of the Dissertation

The organization of this dissertation is as follows:

In Chapter 2, we introduce existing algorithms for moving object detection and issues of these existing methods.

In Chapter 3, we explain the principles of the 2D separable convolution and 3D separable convolution which lay the foundation for our proposed methods. We also introduced the performance evaluation metrics used in this research.

In Chapter 4, we elaborate on our proposed 2D separable CNN - based network in detail. In this chapter, we devised a new 2D separable CNN -based moving object detection approach with a simple network structure and simplified convolution operations. The proposed method does not require explicit background modeling and maintenance. It significantly accelerates inference speed and still achieves high detection accuracy.

In Chapter 5, we discussed our proposed 3D separable CNN with multi-input multi-output strategy network “3DS_MM” in detail. In this chapter, we propose an efficient 3D separable convolutional neural network with a multi-input multi-output strategy called “3DS_MM”. This model is tailored for computation-resource-limited and delay-sensitive applications. Experimental results show the superior performance of this proposed method.

In Chapter 6, we discussed another proposed 3D separable CNN in two-branch network “F3DsCNN” in detail. In this chapter, we devise a fast two-branch 3D separable CNN that extracts both high-level global features and low-level detailed features for moving object detection in computation-resource-limited and delay-sensitive scenarios with high accuracy.

Chapter 7 summarizes the major contributions of this research and suggestions for future work.

CHAPTER 2

Review of Moving Object Detection Methods

2.1 Traditional Methods

The methods for MOD problems have been extensively studied and improved over the years. These methods can be broadly categorized into: (1) traditional methods (unsupervised learning), and (2) deep learning methods (supervised and semi-supervised learning).

Traditional methods [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29] are unsupervised which do not require labeled ground truth. They basically consist of two components: (1) background modeling which initializes the background scene and updates it over time, and (2) classification which classifies each pixel to be foreground or background. There are many background modeling schemes, such as the temporal or adaptive filters being applied to build the background like running average background [10], temporal median filtering [11], and Kalman filtering [12]. Another way for background modeling is to statistically represent the background using parametric probability density functions such as a single Gaussian or a mixture of Gaussians [13]. On the other hand, non-parametric methods directly rely on observed data to model the background such as IUTIS-5 [14], WeSamBE [15], SemanticBGS [16], and kernel density estimation [17]. Sample consensus is another non-parametric strategy used in PAWCS [18], ViBe [19] and SuBSENSE [20]. In particular, SuBSENSE uses a feedback system to automatically adjust the background model based on the local binary similarity pattern (LBSP) features and pixel intensities [21]. Eigen-background based on principal-component analysis (PCA) [22, 23, 24] is also used in background modeling. Further, background subtraction based on robust principal-component analysis (RPCA) [25, 26, 27, 28, 29] solves camera motion and reduces the curse of dimensionality and scale. However, it is quite difficult for traditional methods to perform object detection in complex scenarios, such as videos with illumination changes, shadows, night scenes, and dynamic backgrounds.

2.2 Deep Learning-based Methods

2.2.1 2D CNN-based methods

Deep learning-based methods are mostly supervised and have been recently proposed for MOD problems [32, 33, 34, 44, 46]. The first work based on CNNs is ConvNet-GT [35], which replaces the pixel classification component with a well-defined network structure. The background is estimated by a temporal median filter, then the estimated backgrounds are stacked with the original video frames to form the input of the CNN that outputs the binary masks of detected objects. DeepBS [42] utilizes SuBSENSE [20] algorithm to generate background image and multiple layers CNN for segmentation. Also, a spatial-median filter is used for post-processing to perform smoothing. Wang et al. [36] proposed a multi-scale patch-wise method with a cascade CNN architecture called MSCNN+Cascade [36]. Although it achieves good detection performance, the patch-wise processing is very time consuming. Other multi-scale feature learning-based models such as Guided Multi-scale CNN [37], MCSCNN [38], MsEDNet [39] and VGG-16 [79] based networks FgSegNet_M [40] and FgSegNet_v2 [41] were also proposed. FgSegNet_S [40] is a 2D CNN that takes each video frame at its original resolution scale as the input, while its extended version FgSegNet_M [40] takes each video frame at three different resolution scales in parallel as the input of the encoding network. FgSegNet_v2 is the best-performing FgSegNet model in CDnet2014 [80] challenge. Another example, MSFgNet [43], has a motion-saliency network (MSNet) that estimates the background and subtracts it from the original frames, followed by a foreground extraction network (FgNet) that detects the moving objects.

In VGG-PSL-CRF [44], pixel-level semantic features are extracted, then a novel attention long short-term memory (Attention ConvLSTM) is used to model pixel-wise changes over time. In Deep Pixel Distribution Learning (DPDL) [45] and Dynamic Deep Pixel Distribution Learning (D-DPDL) [46], a pixel-based feature called the Random Permutation of Temporal Pixels (RPoTP) is used to represent the distribution of past observations for a particular pixel, followed by using a CNN to determine whether each pixel is foreground or background. In D-DPDL, a dynamic training strategy is used as a compensation, in which the entries of RPoTP features are randomly repermutated for every training epoch.

Variational Auto-Encoder (VAE) architecture can also be used in video segmentation [81], which modified VAE architecture built on top of Mask R-CNN for instance-level video segmentation and tracking was proposed. The method builds a shared encoder and three parallel decoders, yielding three disjoint branches for predictions of future frames, object detection boxes, and instance segmentation masks.

2.2.2 3D CNN-based methods

3D convolution is applied to MOD problems to utilize spatial-temporal information in visual data. In [54], 3D CNN and a fully connected layer are adopted in a patch-wise method. 3D-CNN-BGS [55] uses 3D convolution to track temporal changes in video sequences. This approach performs 3D convolution on 10 consecutive frames of the video, and upsamples the low-, mid-, and high-level feature layers of the network in a multi-scale approach to enhance segmentation accuracy. 3DAtrous [56] captures long-term temporal information in the video data. It is trained based on a long short-term memory (LSTM) network with focal loss to tackle the class imbalance problem commonly seen in background subtraction. Another LSTM-based example is the autoencoder-based 3D CNN-LSTM [57] combining 3D CNNs and LSTM networks. In this work, time-varying video sequences are handled by 3D convolution to capture short temporal motions, while the long short-term temporal motions are captured by 2D LSTMs. Although these 3D convolution-based methods offer accurate detection results, they have high computational complexity.

2.2.3 GAN-based and other methods

Recently, the concept of generative adversarial networks (GAN) is adopted in MOD problems, such as BScGAN [60], BSGAN [61], BSPVGAN [62], FgGAN [63], BSls-GAN [64], and RMS-GAN [65]. BScGAN is based on conditional generative adversarial network (cGAN) that consists of two networks: generator and discriminator. BSGAN [61] and BSPVGAN [62] are based on Bayesian GANs. They use median filter for background modeling and Bayesian GANs for pixel classification. The use of Bayesian GANs can address the issues of sudden and slow illumination changes, non-stationary background, and ghost. In addition, BSPVGAN [62] exploits parallel vision to improve results in complex scenes. In [66, 67], adversarial learning is proposed to generate dynamic background information in an unsupervised manner.

2.3 Issues with Existing Methods

However, the performance of all the aforementioned deep learning-based moving object detection methods comes at a high computational cost and a slow inference speed due to complex network structures and intense convolution operations. To reduce the amount of calculation, our previous work [59] proposed to use 2D Separable CNN which splits the standard 2D convolution into a depthwise convolution and a pointwise convolution. It dramatically increases the inference speed and maintains high detection accuracy. However, this 2D separable CNN-based network does not exploit the temporal information in the video input.

While existing DNN models offer superior moving object detection accuracy, they suffer from computationally expensive and memory-intensive issues. In particular, the architecture change in 3D CNNs leads to a huge increase in model size and computational complexity compared to 2D CNNs, making it challenging to apply those models to real-world scenarios, such as robotics, self-driving cars, and augmented reality. These tasks are usually deployed on mobile and embedded devices, which have limited memory and computing resources. Besides, these tasks are delay-sensitive and need to be carried out in a timely manner, which cannot be achieved by high-complexity deep learning models. Thus, we aim to design a deep moving object detection model suitable for mobile and embedded environment, that can achieve faster inference speed and smaller model size while maintaining high detection accuracy.

Another factor that limits the inference speed is the input-output relationship. The input-output relationship of existing moving object detection networks has two types: (1) single-input single-output (SISO), which is widely exploited in 2D CNNs such as FgSegNet_S [40] and 2D separable CNN [59]; and (2) multi-input single-output (MISO) which can be found in 3D CNNs such as 3D-CNN-BGS [55], 3DAtrous [56], and DMFC3D [53]. The disadvantage of SISO and MISO is that they result in a slow inference speed because only one frame output is predicted in every forward pass. Recently, the X-Net [82] adopts a two-input two-output network structure, which takes two adjacent video frames as the network input and generates the corresponding two binary masks. Although it can track temporal changes, the network structure is inflexible and the temporal correlation it utilizes is limited. In this research, we propose a multi-input multi-output (MIMO) strategy, which can take multiple input frames and output multiple frames of binary masks in each sample. It explores temporal correlations on a larger time span and significantly increases the inference speed when embedded in 3D separable CNN.

Another issue for supervised methods is the generalization capability of the trained models on completely unseen videos. Several moving object detection models were designed and evaluated over completely unseen videos, such as BMN-BSN [49], BSUV-Net [50], BSUV-Net 2.0 [51], BSUV-Net+SemBGS [50], ChangeDet [52], and 3DCD [58]. Besides, semi-supervised networks were also designed to be extended to unseen videos. For example, GraphBGS [68] and GraphBGS-TV [69] are based on the reconstruction of graph signals and semi-supervised learning algorithm, MSK [70] is based on a combination of offline and online learning strategies, and HEGNet [73] combines propagation-based and matching-based methods for semi-supervised video moving object detection.

These issues above motivate us to design models that can achieve faster inference speed, smaller model size, less computational complexity while maintaining high detection accuracy.

CHAPTER 3

Methodology to Improve Network Efficiency

The approaches of building small and efficient neural networks in recent years can be generally categorized into either compressing pretrained networks or training small networks directly [83].

In our research, we try to train small networks directly. Depthwise separable convolutions as a factorization was initially introduced in [84] and subsequently used in Inception models [85] to reduce the computation. Factorized Networks [86], the Xception network [87] also uses factorization methods. Squeezenet [88] uses a bottleneck approach to design a very small network. Other approaches for obtaining small networks have shrinking, compression based on product quantization, hashing, and pruning, vector quantization and Huffman coding, distillation and low bit networks.

In this chapter, we elaborate on the rationale of the 2D separable convolution and 3D separable convolution operation, which are the building blocks of our proposed methods in the following chapters. In the following sections, we use the default data format "NLHWC" in Tensorflow to represent data, which denotes the batch size N, the temporal length L, the height of the image H, the width of the image W, and the number of channels C.

3.1 2D Convolution vs. 2D Separable Convolution

As shown in Fig. 3.2(a), the standard 2D convolutional layer is parameterized by a convolution filter of size $K \times K \times C_i$, where $K \times K$ is the spatial dimension of the filter and C_i is the number of input channels. The computational complexity of the standard 2D convolution measured by the number of floating-point multiplications is

$$K \times K \times C_i \times H_o \times W_o \times C_o. \tag{3.1}$$

While such convolution effectively extracts features using the 3D filter, it also requires intensive computation. The separable 2D convolution, on the other hand, splits this into a depthwise convolution and a pointwise convolution, which drastically reduces computation and model size. Fig 3.3 shows the filters in standard 2D convolution and filters in depthwise convolution and pointwise convolution.

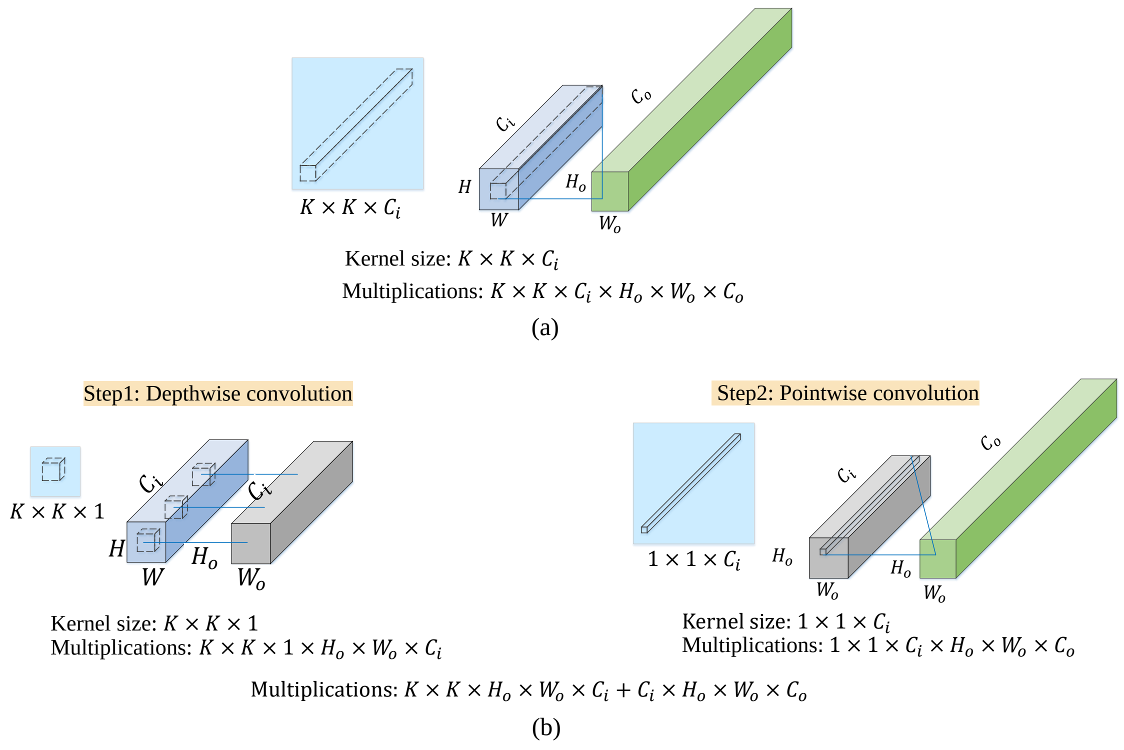
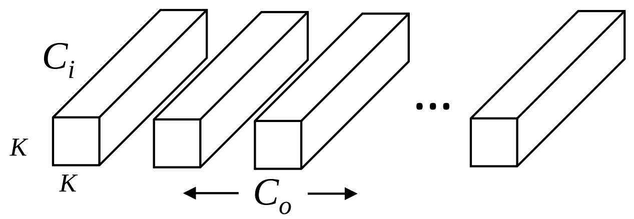
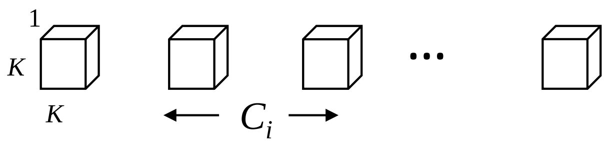

$K \times K \times C_i$
C_i
C_o
H
W
H_o
W_o
Kernel size: $K \times K \times C_i$
Multiplications: $K \times K \times C_i \times H_o \times W_o \times C_o$
(a)
Step1: Depthwise convolution
$K \times K \times 1$
C_i
C_i
H
W
H_o
W_o
Kernel size: $K \times K \times 1$
Multiplications: $K \times K \times 1 \times H_o \times W_o \times C_i$
Step2: Pointwise convolution
$1 \times 1 \times C_i$
C_i
C_o
H_o
W_o
H_o
W_o
Kernel size: $1 \times 1 \times C_i$
Multiplications: $1 \times 1 \times C_i \times H_o \times W_o \times C_o$
Multiplications: $K \times K \times H_o \times W_o \times C_i + C_i \times H_o \times W_o \times C_o$
(b)

Figure 3.2: Illustration of (a) the standard 2D convolution and (b) the 2D separable convolution.

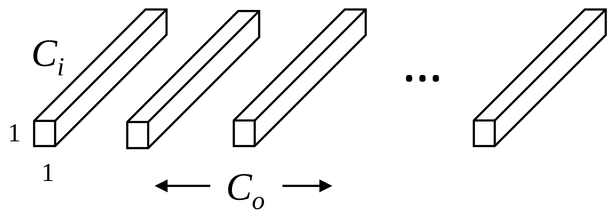
As shown in Fig. 3.2(b) Step 1, depthwise convolution performs an independent convolution on each input channel with a filter of size $K \times K \times 1$ without interactions among



(a) Standard 2D Convolution Filters



(b) Depthwise Convolution Filters



(c) Pointwise convolution Filters

Figure 3.3: Illustration of (a) the standard 2D convolution filter and (b) depthwise convolution filter (c) pointwise convolution filter.

channels. The required multiplications of the 2D depthwise convolution is

$$K \times K \times 1 \times H_o \times W_o \times C_i. \tag{3.2}$$

Following depthwise convolution is the pointwise convolution, as shown in Fig. 3.2(b) Step 2. It performs a 1D convolution on each depth column[1], using a filter of size $1 \times 1 \times C_i$. This creates a linear projection of the stack of feature maps. If C_o filters are used, then the required multiplications of this 1D pointwise convolution is

$$1 \times 1 \times C_i \times H_o \times W_o \times C_o. \tag{3.3}$$

By decomposing the standard 2D convolution into two separate steps, we achieve a

[1]A depth-column is formed by the voxels at the same spatial location (y, x) across all channels.

computation reduction of

$$\begin{aligned}\textbf{ratio} &= \frac{\textit{2D separable convolution}}{\textit{2D convolution}} \\ &= \frac{K \times K \times H_o \times W_o \times C_i + C_i \times H_o \times W_o \times C_o}{K \times K \times C_i \times H_o \times W_o \times C_o} \\ &= \frac{1}{C_o} + \frac{1}{K^2}.\end{aligned} \tag{3.4}$$

When the output channels C_o is a large number, the first term $\frac{1}{C_o}$ is negligible. For instance, if $K = 3$, then the 2D separable convolution can achieve roughly 9 times less computation than the standard 2D convolution.

2D separable convolution splits traditional 2D convolution into a depthwise convolution and a pointwise convolution, which drastically reduces computational complexity [59, 83, 89, 90].

3.2 2D Convolution vs. 3D Convolution

As shown in Fig. 3.4(a)[91], an ordinary 2D convolution takes a 3D tensor of size $H \times W \times C_i$ as the input, where H and W are the height and width of feature maps, and C_i is the number of input channels. In this case, the filter is a 3D filter in a shape of $K \times K \times C_i$ moving in two directions (y, x) to calculate a 2D convolution. The output is a 2D matrix of size $H_o \times W_o$. If the filter number is C_o, the output shape will be $H_o \times W_o \times C_o$. The mathematical expression of such 2D convolution is given by

$$Out[h, w] = \sum_{j=0}^{K-1} \sum_{i=0}^{K-1} \sum_{c=0}^{C_i-1} f[j, i, c] \times In[h - j, w - i, c] \tag{3.5}$$

where In represents the 3D input to be convolved with the 3D filter f to result in a 2D output feature map Out. Here, h, w and c are the height, width, and channel coordinates of the 3D input, while j, i and c are those of the 3D filter.

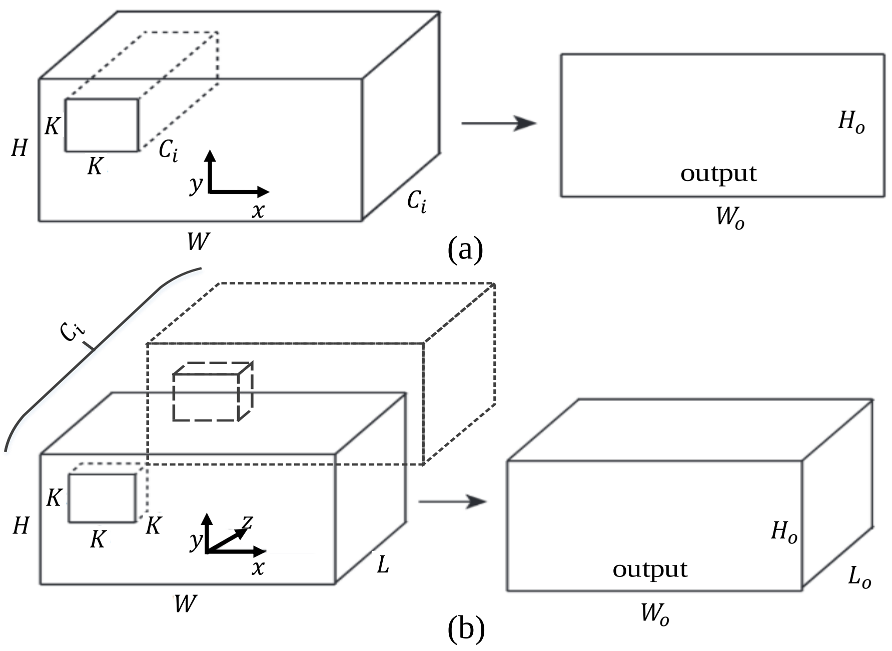


Figure 3.4: Illustration of (a) the 2D convolution with 3D input and (b) the 3D convolution with 4D input.

However, for video signal the 2D convolution in Fig. 3.4(a) does not leverage the temporal information among adjacent frames. 3D convolution addresses this issue using 4D convolutional filters with 3D convolution operation, as illustrated in Fig. 3.4(b). In a 3D convolution, the "input" becomes C_i channels of 3D tensors of size $L \times H \times W$, where L is the temporal length (i.e. the number of successive video frames). Hence, the input is 4D and is of size $L \times H \times W \times C_i$. A 4D convolutional filter of size $K \times K \times K \times C_i$ moves in 3 directions (z, y, x) to calculate convolutions, where z, y, and x align with the temporal length, height, and width axes of the 4D input. The output shape is $L_o \times H_o \times W_o$. If the filter number is C_o, the output shape will be $L_o \times H_o \times W_o \times C_o$. The mathematical expression of the 3D convolution with a 4D input is given by

$$Out[l, h, w] = \begin{array}{c} \sum_{k=0}^{K-1} \sum_{j=0}^{K-1} \sum_{i=0}^{K-1} \sum_{c=0}^{C_i-1} \\ f[k, j, i, c] \times In[l-k, h-j, w-i, c] \end{array} \tag{3.6}$$

where In represents the 4D input to be convolved with the 4D filter f to result in a 3D output *Out*. Here, l, h, w, and c are the temporal length, height, width, and channel coordinates of the 4D input, while k, j, i and c are those of the 4D filter. If the size of the filter is $K \times K \times K \times C_i$, then the indices k, j, i range from 0 to $K-1$, and c ranges from 0 to $C_i - 1$.

The ability to leverage the temporal context improves moving object detection accuracy. However, 3D CNN is rarely used in practice because it suffers from a high computational cost due to the increased amount of computation used by 3D convolutions, especially when the dataset scale goes larger and the neural network model goes deeper. Thus, in order to make use of the temporal features, a low-complexity 3D CNN must be developed.

As shown in Fig. 3.4(a)[91], an ordinary 2D convolution takes a 3D tensor of size $H \times W \times C_i$ as the input, where H and W are the height and width of feature maps, and C_i is the number of input channels. In this case, the filter is a 3D filter in a shape of $K \times K \times C_i$ moving in two directions (y, x) to calculate a 2D convolution. The output is a 2D matrix of size $H_o \times W_o$. If the filter number is C_o, the output shape will be $H_o \times W_o \times C_o$. The mathematical expression of such 2D convolution is given by

$$Out[h, w] = \sum_{j=0}^{K-1} \sum_{i=0}^{K-1} \sum_{c=0}^{C_i-1} f[j, i, c] \times In[h-j, w-i, c] \tag{3.7}$$

where In represents the 3D input to be convolved with the 3D filter f to result in a 2D output feature map *Out*. Here, h, w and c are the height, width, and channel coordinates of the 3D input, while j, i and c are those of the 3D filter.

However, for video signal the 2D convolution in Fig. 3.4(a) does not leverage the temporal information among adjacent frames. 3D convolution addresses this issue using 4D convolutional filters with 3D convolution operation, as illustrated in Fig. 3.4(b). In a 3D convolution, the "input" becomes C_i channels of 3D tensors of size $L \times H \times W$, where L is the temporal length (i.e. the number of successive video frames). Hence, the input is 4D and is of size $L \times H \times W \times C_i$. A 4D convolutional filter of size $K \times K \times K \times C_i$ moves in 3 directions (z, y, x) to calculate convolutions, where z, y, and x align with the temporal length, height, and width axes of the 4D input. The output shape is $L_o \times H_o \times W_o$. If the filter number is C_o, the output shape will be $L_o \times H_o \times W_o \times C_o$. The mathematical expression of the 3D convolution with a 4D input is given by

$$Out[l, h, w] = \begin{array}{c} \sum_{k=0}^{K-1} \sum_{j=0}^{K-1} \sum_{i=0}^{K-1} \sum_{c=0}^{C_i-1} \\ f[k, j, i, c] \times In[l-k, h-j, w-i, c] \end{array} \tag{3.8}$$

where In represents the 4D input to be convolved with the 4D filter f to result in a 3D output Out. Here, l, h, w, and c are the temporal length, height, width, and channel coordinates of the 4D input, while k, j, i and c are those of the 4D filter. If the size of the filter is $K \times K \times K \times C_i$, then the indices k, j, i range from 0 to $K - 1$, and c ranges from 0 to $C_i - 1$.

The ability to leverage the temporal context improves moving object detection accuracy. However, 3D CNN is rarely used in practice because it suffers from a high computational cost due to the increased amount of computation used by 3D convolutions, especially when the dataset scale goes larger and the neural network model goes deeper. Thus, in order to make use of the temporal features, a low-complexity 3D CNN must be developed.

3.3 3D Convolution vs. 3D Separable Convolution

2D separable convolution splits traditional 2D convolution into a depthwise convolution and a pointwise convolution, which drastically reduces computational complexity [59, 83, 89, 90].

In order to utilize temporal features in video data, the idea of separable convolution can be applied to the standard 3D convolution. As shown in Fig. 3.5 (a), in the standard

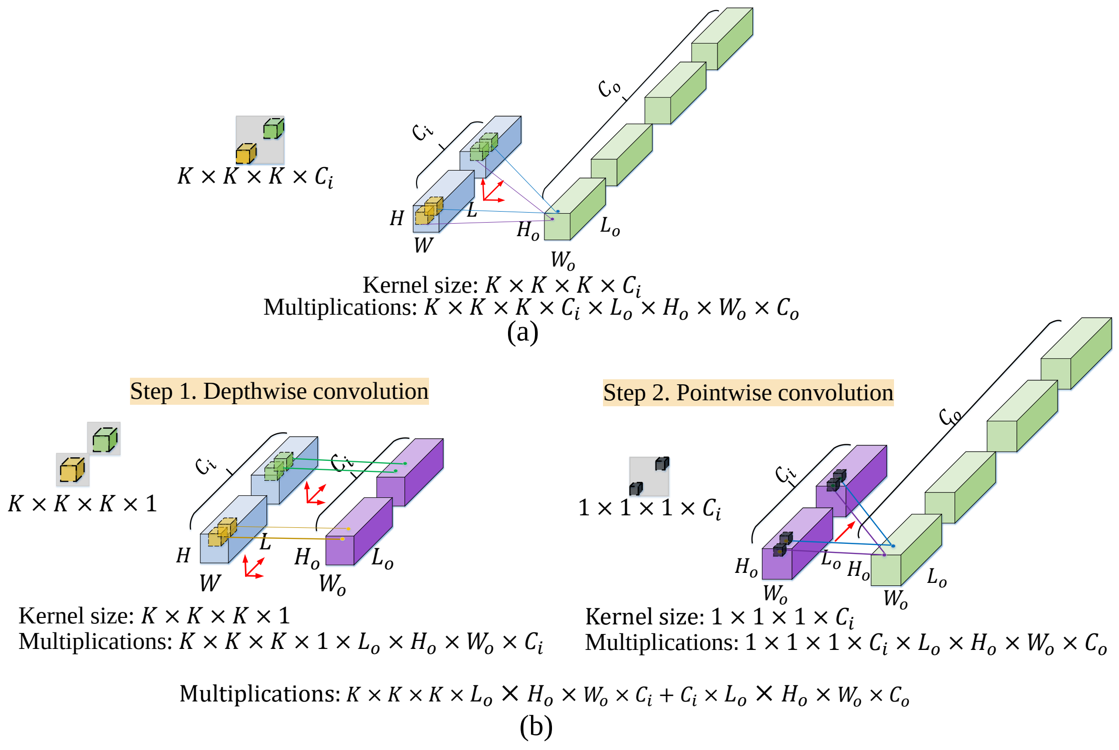

$K \times K \times K \times C_i$
C_i
C_o
H
W
L
H_o
W_o
L_o
Kernel size: $K \times K \times K \times C_i$
Multiplications: $K \times K \times K \times C_i \times L_o \times H_o \times W_o \times C_o$
(a)
Step 1. Depthwise convolution
$K \times K \times K \times 1$
Kernel size: $K \times K \times K \times 1$
Multiplications: $K \times K \times K \times 1 \times L_o \times H_o \times W_o \times C_i$
Step 2. Pointwise convolution
$1 \times 1 \times 1 \times C_i$
Kernel size: $1 \times 1 \times 1 \times C_i$
Multiplications: $1 \times 1 \times 1 \times C_i \times L_o \times H_o \times W_o \times C_o$
Multiplications: $K \times K \times K \times L_o \times H_o \times W_o \times C_i + C_i \times L_o \times H_o \times W_o \times C_o$
(b)

Figure 3.5: Illustration of (a) the standard 3D convolution and (b) the 3D separable convolution. Red arrows point to effective directions of the convolution calculation of the 3D filters.

3D convolution, the 4D input of size $L \times H \times W \times C_i$, is convolved with C_o filters of size $K \times K \times K \times C_i$, resulting in a 4D output of size $L_o \times H_o \times W_o \times C_o$. The filters calculate the 3D convolution by moving in the directions of length, height, and width as shown by the red arrows. The computational complexity of such standard 3D convolution is $K \times K \times K \times C_i \times L_o \times H_o \times W_o \times C_o$.

To simplify the 3D convolution, we decompose it into a 3D depthwise convolution and a 1D pointwise convolution. As shown in Fig. 3.5 (b) Step 1, the 3D depthwise convolution adopts C_i independent filters of size $K \times K \times K \times 1$ to perform a 3D convolution on each input channel. This procedure is described in (3.9). The required multiplications of such

3D depthwise convolution is $K \times K \times K \times 1 \times L_o \times H_o \times W_o \times C_i$.

$$Out[l,h,w,c] = \sum_{k=0}^{K-1}\sum_{j=0}^{K-1}\sum_{i=0}^{K-1} f[k,j,i,c] \times In[l-k, h-j, w-i, c], c = 1, 2, ..., C_i. \quad (3.9)$$

Afterwards, the output of Fig. 3.5 (b) Step 1 is used as the input of Fig. 3.5 (b) Step 2, where the pointwise convolution adopts a filter of size $1 \times 1 \times 1 \times C_i$, performs a linear projection along the channel axis as shown by the red arrow, and outputs a 3D tensor of size $L_o \times H_o \times W_o$. This procedure is described in (3.10). Using C_o such filters outputs C_o 3D tensors. The required multiplications of such 1D pointwise convolution is $1 \times 1 \times 1 \times C_i \times L_o \times H_o \times W_o \times C_o$.

$$Out[l,h,w] = \sum_{s=0}^{C_i-1} f[s] \times In[l,h,w,c-s]. \quad (3.10)$$

The combination of the 3D depthwise convolution and the 1D pointwise convolution, called 3D separable convolution, achieves a reduction in computational complexity of

$$\begin{aligned} \textbf{ratio} &= \frac{\textit{3D separable convolution}}{\textit{3D convolution}} \\ &= \frac{K \times K \times K \times L_o \times H_o \times W_o \times C_i + C_i \times L_o \times H_o \times W_o \times C_o}{K \times K \times K \times C_i \times L_o \times H_o \times W_o \times C_o} \\ &= \frac{1}{C_o} + \frac{1}{K^3}. \end{aligned} \quad (3.11)$$

With $K = 3$ and a large C_o, the computational complexity can be reduced by roughly 27 times compared to the standard 3D convolution.

This research adopts such 2D separable convolution and 3D separable convolution in the designed moving object detection networks for the first time. Together with other proposed techniques such as multi-input multi-output (MIMO) and the structure of two branches network, they substantially reduce the amount of computation, meanwhile the 3D separable convolution extracts temporal features in the video sequence.

3.4 Performance Evaluation Metrics

In our research, we evaluate the performance in both efficiency and detection accuracy.

Efficiency

To evaluate the efficiency of our proposed model, the inference speed is measured in frames per second (fps), the model size is measured in megabytes (MB), the number of trainable parameters is measured in millions (M), and the computational complexity is measured in floating point operations (FLOPs).

Detection Accuracy

To measure the detection accuracy, we adopt four metrics: the region-based F-measure, the structure measure (S-measure) [92], the enhanced alignment measure (E-measure) [93], and the mean absolute error (MAE) [94].

The F-measure is defined as:

$$F\text{-}measure = \frac{2 \times precision \times recall}{precision + recall} \tag{3.12}$$

where $precision = \frac{TP}{TP+FP}$, $recall = \frac{TP}{TP+FN}$, given the true positive (TP), false positive (FP), true negative (TN), and false negative (FN).

The S-measure [92] combines the region-aware structural similarity S_r and object-aware structural similarity S_o, which is more sensitive to structures in scenes:

$$S\text{-}measure = \alpha \times S_o + (1 - \alpha) \times S_r, \tag{3.13}$$

where $\alpha = 0.5$ is the balance parameter.

The E-measure is recently proposed [93] based on cognitive vision studies and combines local pixel values with the image-level mean value in one term, jointly capturing image-level statistics and local pixel matching information.

We also evaluate the MAE [94] between the predicted output and the binary ground-truth mask as:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |Pred_i - GT_i| \,, \tag{3.14}$$

where $Pred_i$ is the predicted value of the i-th pixel, GT_i is the ground-truth binary label of the i-th pixel, and N is the total number of pixels.

CHAPTER 4

Proposed Method and Experiments - Using 2D Separable CNN

4.1 Introduction

With the rise of the Internet of Things, the spread of machine vision, and the large amount of video data, it is challenging and crucial to process video data at a fast speed. Efficiently reducing redundant information in videos such as the background and extracting meaningful foreground information like moving vehicles or pedestrians is a crucial step for video surveillance systems. Recently, deep learning-based moving object detection algorithms demonstrated superior detection accuracy as compared with traditional methods. However, existing deep models are computationally expensive and memory-intensive.

In this chapter, we demonstrate a new light-weight deep network model using 2D separable CNN introduced in Section 3.1 that achieved high detection accuracy, while it significantly accelerates the detection speed compared with current state-of-the-art deep models. This chapter is organized as follows. In Section 4.2, we introduce existing algorithms used for moving object detection. In Section 4.3, we elaborate on our proposed model in detail. Section 4.4 describes our experimental setup and results compared with the current state-of-the-art models on the CDnet2014 dataset [80]. Section 4.4 concludes the chapter.

4.2 Proposed 2D Separable CNN-based Deep Model

The proposed network involves an encoder and a decoder shown in Fig 4.6. We elaborate on the details of our approach in the encoder network and the decoder network.

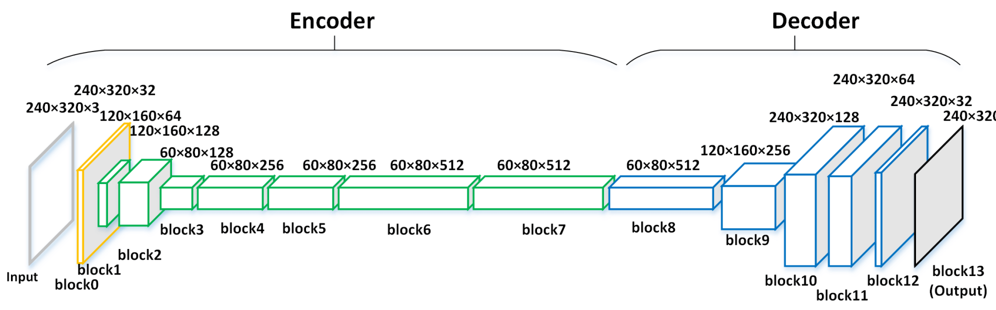
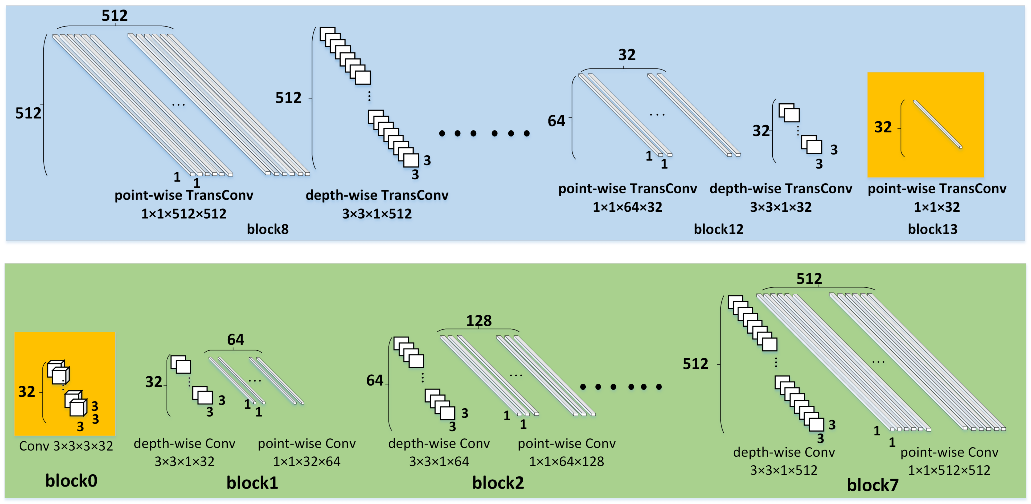


Figure 4.6: The architecture of the proposed network Illustration of the filters used in (a) Encoder network (b) Decoder network



512
512
point-wise TransConv 1×1×512×512
512
depth-wise TransConv 3×3×1×512
block8
32
64
point-wise TransConv 1×1×64×32
32
depth-wise TransConv 3×3×1×32
block12
32
point-wise TransConv 1×1×32
block13
32
Conv 3×3×3×32
block0
32
64
depth-wise Conv 3×3×1×32
point-wise Conv 1×1×32×64
block1
64
128
depth-wise Conv 3×3×1×64
point-wise Conv 1×1×64×128
block2
512
512
depth-wise Conv 3×3×1×512
point-wise Conv 1×1×512×512
block7

Figure 4.7: The architecture of the proposed network Illustration of the filters used in (a) Encoder network (b) Decoder network

As can been seen in Fig 4.6, we adopt a regular Auto-Encoder (AE) architecture, not other architectures such as Variational Auto-Encoder (VAE). Usually Auto-encoders work for data visualization via dimensionality reduction, data denoising, and data anomaly detection. The most common use of Variational Auto-encoders is for generating new image or text data. It's good to use VAE to generate synthetic data that is based on existing data. In VAE, encoder describes a probability distribution for each latent attribute, and latent encoding variables are assumed to be identically and independently distributed across both latent dimensions and samples, which is not realistic in many problems with high dimensional inter and intra-data correlation [95]. For example, in a video sequence, it is reasonable to expect that the frames would exhibit similar latent representations, so VAE should allow correlation among latent variables.

In our case, we are more focusing on looking for the small differences between frames to be able to detect moving objects, so dimensionality reduction and anomaly detection techniques are more suitable for our jobs. AE model is already enough for the task, we will not choose VAE. Using VAE would bring more complexity than AE to our job and in some cases and VAE tend to result in more blurry images in our case.

4.2.1 The encoder network

The encoder network extracts features from the RGB 3-channel input frames, so-called feature map 'encoding' by convolutions with filters, where different filters can capture different features. The encoder network consists of 8 blocks as shown in Fig 4.7(a). Iteratively tuning the configuration during training helps to choose 8 to be the least required number of blocks to be able to achieve a high prediction accuracy where each block detects specific features present in the input data. In block 0, the regular convolution with 3D filters is adopted. 3×3×3×32 represents that the filter is 3×3 spatially, length-3 in the depth dimension, and the number of filters is 32, resulting in a layer with 32 channels. From block 1 to block 7, each block contains one depthwise convolution, one batch normalization (BN) [85] and one pointwise convolution. For example, in block 1, the depthwise convolution (depthwise Conv) adopts 32 filters of dimension 3×3×1 and the pointwise convolution (pointwise Conv) adopts 64 filters of dimension 1×1×32. The filters of block 2 to block 7 are similarly defined. The notions of depthwise convolution and pointwise convolution were first proposed in MobileNet [83]. The depth separable convolution replaces traditional convolution by applying an independent 2D filter for each input channel followed by pointwise operation using a 1×1×C$_i$ (C_i is the number of input channels) convolution on each depth-column of the outputs from the depthwise convolution [83]. The effect of this separation is to greatly reduce the amount of computation and model size [83]. The computational cost can be reduced to $1/C_o + 1/K^2$ of a regular convolution, where C_o is the number of output channels and K is the spatial dimension of the $K \times K$ kernel [83]. In this chapter, for the first time in the literature, we adopt such separated depthwise and pointwise convolution in a moving object detection network.

4.2.2 The decoder network

Fig 4.7(b) shows our proposed decoder network that expands the encoder output and generates a binary mask for the detected moving objects. The decoder has 6 blocks. We propose from block 8 to block 12 each of them consists of one pointwise transposed convolution and one depthwise transposed convolution. Transposed convolution, informally called deconvolution, is a backward stride convolution for up-sampling optimally. In block 8, the pointwise transposed convolution (pointwise TransConv) adopts 512 filters of dimension 1×1×512, and the depthwise transposed convolution (depthwise TransConv) adopts 512 filters of dimension 3×3×1. The filters of block 9 to block 12 are similarly defined. In block 13, the pointwise transposed convolution adopts one filter of dimension 1×1×32. We propose to use a regular transposed convolution with 1×1×c (c is similarly defined as in the encoder network) filter and stride 2 for up-sampling as the pointwise transposed convolution, followed by a regular depthwise convolution to perform as a transposed convolution in depthwise. Finally, a sigmoid activation function is appended to output a 0/1 binary mask (0 is background, 1 is foreground) with the same spatial dimension as the input images. For the first time in the literature, we propose the pointwise and depthwise deconvolution in the decoder network symmetric to the encoder network for computational efficiency. Table 4.1 shows the details of the input and output shape and configuration in each layer.

Table 4.1: Proposed network configuration. Encoder consists of Blocks 0 to 7. Decoder consists of Blocks 8 to 13.

		Layer Type / Stride	**Filter Shape**	**Output Shape**	**Parameters #**
Encoder	**block 0**			240*320*3 **(Input Image)**	
		Conv / s=1	3*3*3*32	240*320*32	896
	block 1	Conv dw / s=2	3*3*32 dw	120×160×128	352
		BN		120×160×128	64
		Conv pw/ s=1	1*1*32*64 pw	120*160*64	2048
	block 2	Conv dw / s=1	3*3*64 dw	120*160*64	704
		BN		120*160*64	128
		Conv pw/ s=1	1*1*64*128 pw	120*160*128	8192
	block 3	Conv dw / s=2	3*3*128 dw	60*80*128	1408
		BN		60*80*128	256
		Conv pw/ s=1	1*1*128*128 pw	60*80*128	16384
	block 4	Conv dw / s=1	3*3*128 dw	60*80*128	1408
		BN		60*80*128	256
		Conv pw/ s=1	1*1*128*256 pw	60*80*256	32768
	block 5	Conv dw / s=1	3*3*256 dw	60*80*256	2816
		BN		60*80*256	512
		Conv pw/ s=1	1*1*256*256 pw	60*80*256	65536
	block 6	Conv dw / s=1	3*3*256 dw	60*80*256	2816
		BN		60*80*256	512
		Conv pw/ s=1	1*1*256*512 pw	60*80*512	131072
	block 7	Conv dw / s=1	3*3*512 dw	60*80*512	5632
		BN		60*80*512	1024
		Conv pw/ s=1	1*1*512*512 pw	60*80*512	262144
Decoder	**block 8**	TransposeConv / s=1	1*1*512*512 pw	60*80*512	262656
		Conv dw / s=1	3*3*512 dw	60*80*512	4608
	block 9	TransposeConv / s=2	1*1*512*256 pw	120*160*256	131328
		Conv dw / s=1	3*3*256 dw	120*160*256	2304
	block 10	TransposeConv / s=2	1*1*256*128 pw	240*320*128	32896
		Conv dw / s=1	3*3*128 dw	240*320*128	1152
	block 11	TransposeConv / s=1	1*1*128*64 pw	240*320*64	8256
		Conv dw / s=1	3*3*64 dw	240*320*64	576
	block 12	TransposeConv / s=1	1*1*64*32 pw	240*320*32	2080
		Conv dw / s=1	3*3*32 dw	240*320*32	288
	block 13	TransposeConv / s=1	1*1*32 pw	240*320*1	33
		Activation		240*320*1 **(Binary Mask)**	
		Total			983105

4.3 Experiments and Analysis

In this section, we evaluate the performance of our proposed deep moving object detection network on the CDnet2014 dataset [80]. It contains 11 categories of video sequences (such as baseline, bad weather, dynamic background, camera jitter, etc.), and each category contains 4 to 6 video sequences (such as highway, office, pedestrians, PETS2006 in the baseline category). In our experiments, 15 sequences covering 6 categories were selected for the training and testing. For each video sequence, 200 frames were selected randomly. The training was performed for every single video sequence using an Intel Xeon 8-core 3GHz CPU processor with an Nvidia Titan RTX 24G GPU. In the training phase, we use RMSprop optimizer and cross-entropy loss function at a learning rate $\alpha = 10^{-4}$ over 50 epochs in a batch size of 1. In the testing phase, we test more than $1,000$ frames for each sequence with training frames excluded.

To evaluate the model performance, we calculate the F-measure between the predicted binary masks and the ground truth binary masks provided in the CDnet2014 dataset [80].

Table 4.2 compares the F-measure scores and the inference speed of the proposed model, with the state-of-the-art FgSegNet (FgSegNet 1-scale [40], FgSegNet 3-scale) described in Section 4.2, and the MobileNet+UNet model which is a MobileNet based Unet [3], recently proposed for semantic segmentation. We observe that our proposed model achieves an inference speed of 149.81 fps, more than twice faster than those of the MobileNet+UNet and the FgSegNet 3-scale, and 1.8 times faster than that of the FgSegNet 1-scale. In terms of detection accuracy, our proposed model achieves an average F-measure of 0.9718, significantly higher than that of MobileNet+UNet, and only slightly lower than those of the two FgSegNet models.

Table 4.2: Comparison of F-measure and inference speed with other models on different categories and sequences.

Category	F-Measure				Inference Speed (fps) on PC			
	MobileNet +UNet	*FgSegNet (3-scale)*	*FgSegNet (1-scale)*	*Proposed model*	*MobileNet +UNet*	*FgSegNet (3-scale)*	*FgSegNet (1-scale)*	*Proposed model*
No.1	0.9698	0.997	0.9975	0.9931	56.02	70.42	80.19	143.35
No.1	0.9737	0.9934	0.9936	0.9839	62.19	66.42	86.28	158.58
No.1	0.0373	0.9743	0.9778	0.9596	51.36	67.28	86.73	153
No.1	0.0796	0.9948	0.9958	0.9714	69.74	72.73	85.98	154.08
No.2	0.9525	0.993	0.9941	0.9839	66.62	66.12	78.99	139.08
No.2	0.6424	0.9945	0.8241	0.9545	63.45	70.57	82.37	152.44
No.2	0.1638	0.982	0.929	0.9619	46.06	70.98	78.19	155.04
No.2	0.8563	0.9733	0.9669	0.9384	56.75	70.64	85.76	147.15
No.3	0.9797	0.9982	0.9982	0.9945	60.42	65.99	84.53	161.29
No.3	0.9606	0.9991	0.9994	0.9964	63.65	67.35	79.94	137.82
No.4	0.7515	0.9911	0.9928	0.9803	49.55	65.22	86.06	161.55
No.4	0.9536	0.9959	0.9969	0.9912	56.95	66.16	81.1	143.88
No.5	0.9215	0.9957	0.9958	0.9834	64.43	72.34	79.05	135.5
No.6	0.8841	0.9645	0.9561	0.9403	51.81	71.35	80.78	160.36
No.6	0.9389	0.9856	0.9887	0.9444	50.43	71.72	78.65	144.09
Average	0.7377	0.9888	0.9738	**0.9718**	57.96	69.02	82.31	**149.81**

Table 4.3: Summary of comparison between the proposed model and other models.

	F-Measure	Parameters #	Flops(millions/s)	Model Size (bytes)	Inference Speed (fps）
FgSegNet 3-scale	0.9888	15,857,665	16.2	60MB	69.02
FgSegNet 1-scale	0.9738	9,358,593	9.63	36MB	82.31
MobileNet+UNet	0.7377	4,217,397	4.18	16.8MB	57.96
Proposed Model	**0.9718**	**983,105**	**1.42**	**3.8MB**	**149.81**

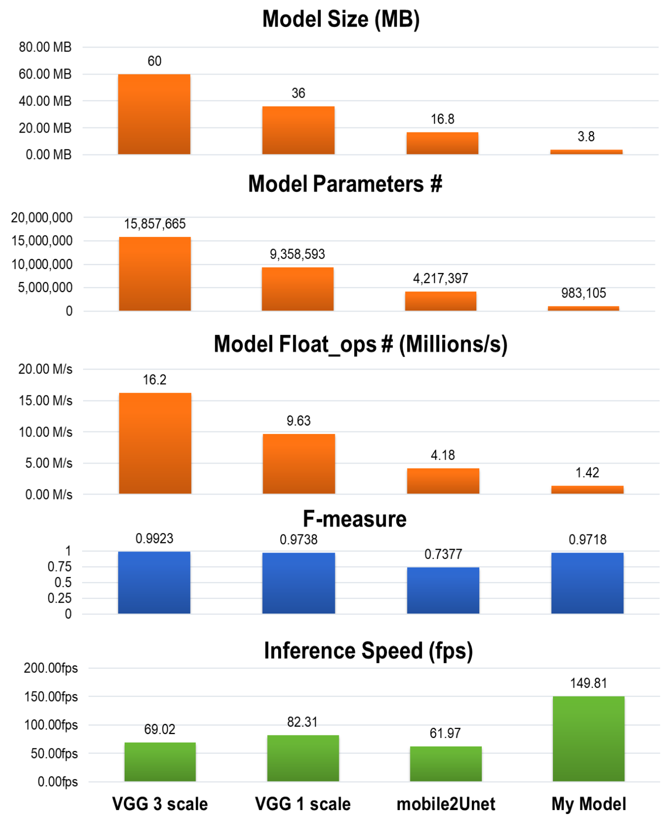


Figure 4.8: Summary of Comparison Between the Proposed Model and other Models.

Table 4.3 and Fig 4.8 show the overall comparisons on F-measure, the number of model parameters, floating-point operations (FLOPs in millions/second), the model space (the storage space size of weights in megabytes (MB)), and the inference speed. The proposed model requires the least number of model parameters, the least number of floating-point operations, the smallest model storage space size, while it offers the fastest average inference speed and a competitive moving object detection accuracy in terms of the F-measure.

Fig 4.9 shows the visual results for three sequences (from 15 sequences in the experiment) in three categories: baseline, low frame rate, and dynamic background. Our method has detected clear backgrounds and refined boundaries in the foreground regions, which is competitive with the results of FgSegNet 3-scale and FgSegNet 1-scale, and much better than the result of MobileNet+UNet.

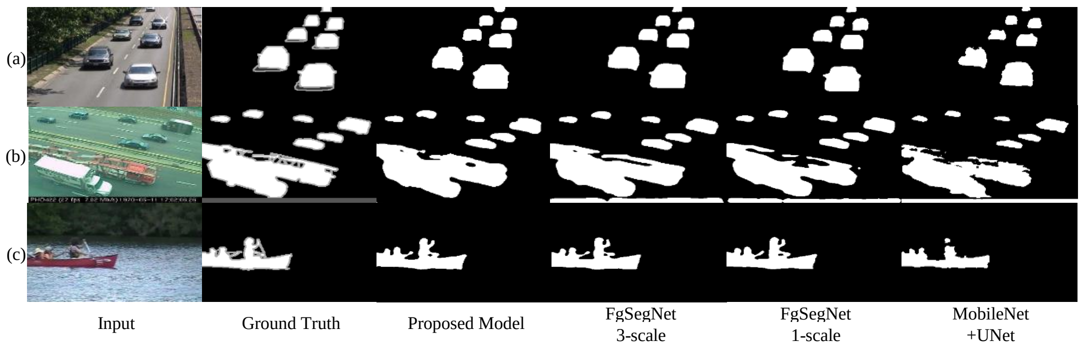


Figure 4.9: Comparison results on 3 out of 15 sequences (a) baseline: highway (b) lowFramerate: turnpike_0_5fps (c) dynamicbackground: canoe . The first column is input image, the second column is ground truth, third-sixth columns are the proposed model, FgSegNet 3-scale, FgSegNet 1-scale, MobileNet+UNet respectively.

4.4 Conclusion

In this chapter, we devise an efficient deep network model based on depthwise and pointwise convolution and deconvolution for moving object detection in surveillance video. The proposed model uses a simple network structure with simplified convolution operations. Experimental studies demonstrated the effectiveness of our proposed model.

CHAPTER 5

Proposed Method and Experiments - 3DS_MM

5.1 Introduction

As introduced in Chapter 3.3, 3D separable Convolution [96] has advantages in extracting temporal information to help further increase detection accuracy better than 2D separable Convolution.

In this chapter, we propose an efficient 3D separable convolutional neural network with a multi-input multi-output strategy called "3DS_MM". This model is tailored for computation-resource-limited and delay-sensitive applications. Compared to state-of-the-art models, it significantly increases inference speed and reduces model size, meanwhile increasing detection accuracy or maintaining a competitive detection accuracy. Our key contributions are as follows:

- We propose a new 3D separable CNN for moving object detection. The proposed network adopts 3D convolution to explore spatio-temporal information in the video data and to improve detection accuracy. To reduce computational complexity and model size, the 3D convolution is decomposed into a depthwise convolution and a pointwise convolution. While existing 3D separable CNN schemes all addressed other problems such as gesture recognition, force prediction, 3D object classification or reconstruction, our work applied it to the moving object detection task for the first time in the literature.
- We propose a multi-input multi-output (MIMO) strategy. While existing networks

are single-input single-output, multi-input single output, or two-input two-output, our MIMO network can take multiple input frames and output multiple binary masks using temporal-dimension in each sample. This MIMO embedded in 3D separable CNN can further increase model inference speed significantly and maintain high detection accuracy. To the best of our knowledge, this is the first time in the literature that such kind of MIMO scheme is used in the MOD task.

- We demonstrate that the proposed 3DS_MM offers overwhelmingly high inference speed in frames per second (154 fps) and extremely small model size (1.45 MB), while achieving the best detection accuracy in terms of F-measure, S-measure, E-measure, and MAE among all models in scene dependent evaluation (SDE) setup and achieving the best detection accuracy among the models with inference speeds exceeding 65 fps in scene independent evaluation (SIE) setup. The SDE setup is widely used to tune and test the model on a specific video as the training and test sets are from the same video. The SIE setup originally raised in [52] is specifically designed to assess the generalization capability of the model on completely unseen videos.

The rest of the chapter is organized as follows. In Section 5.2, we elaborate on our proposed network in detail. Section 5.3 explains the training and evaluation setup of the experiments. Section 5.4 describes our experimental results compared to the state-of-the-art models. Section 5.5 concludes the chapter.

In this chapter, we extend the 2D separable CNN to a 3D separable CNN, which reduces the computational complexity compared to standard 3D CNN. Although some existing works [97, 98, 99, 100] adopt 3D separable CNN to extract high-dimensional features, none of them applied it to the problem of moving object detection. For example, the 3D separable CNN in [97] is for hand-gesture recognition, in which the last two layers of the network are fully connected layers that output class labels. The 3D separable CNN in [98] is used for two tasks: 3D object classification and reconstruction. Neither task utilizes temporal data, hence no temporal convolution is involved. The 3D separable CNN in [99] is to predict interactive force between two objects, hence its network output is a scalar representing the predicted force value. This problem essentially is a regression problem. Besides, the way that the 3D convolution is separated in [99, 100] is different from our proposed method. It first conducts channel-wise 2D convolution for each independent frame and channel, then conducts joint temporal-channel-wise convolution. In contrast, our proposed 3D separable CNN performs spatial-temporal convolution first, then performs pointwise convolution along the channel direction.

As introduced in Section 2.3, another factor that limits the inference speed is the input-output relationship. The input-output relationship of existing moving object detection networks has two types: (1) single-input single-output (SISO), which is widely exploited in 2D CNNs such as FgSegNet_S [40] and 2D separable CNN [59]; and (2) multi-input single-output (MISO) which can be found in 3D CNNs such as 3D-CNN-BGS [55], 3DAtrous [56], and DMFC3D [53]. The disadvantage of SISO and MISO is that they result in a slow inference speed because only one frame output is predicted in every forward pass. In this chapter, we propose a multi-input multi-output (MIMO) strategy, which can take multiple input frames and output multiple frames of binary masks in each sample. It explores temporal correlations on a larger time span and significantly increases the inference speed when embedded in 3D separable CNN.

As introduced in Section 2.3, another issue for supervised methods is the generalization capability of the trained models on completely unseen videos.

In this chapter, we devise a new lightweight 3D separable CNN specifically for moving object detection in computation-resource-limited and delay-sensitive scenarios. It has an efficient end-to-end encoder-decoder structure with a multi-input multi-output (MIMO) strategy, named as the “3DS_MM”. The proposed 3DS_MM does not require explicit background modeling. We evaluate the model over CDnet2014 [80] dataset in an SDE framework with other state-of-the-art models, and we also assess the generalization capability of the model over CDnet2014 and DAVIS2016 [101] datasets in SIE setups over completely unseen videos.

The proposed 3DS_MM significantly increases the inference speed, reduces the trainable parameters, computational complexity and model size, meanwhile achieving the highest detection accuracy in SDE setup and maintaining a competitive detection accuracy in SIE setup.

5.2 Proposed 3DS_MM Network

The proposed deep moving object detection network shown in Fig. 5.10 is based on two major designs: (1) the encoder-decoder-based 3D separable CNN and (2) the multi-input multi-output (MIMO) strategy. This section describes the proposed approach in detail.

Table 5.4: Proposed network configuration. The encoder consists of blocks 0 to 5, and the decoder consists of blocks 6 to 8.

		Layer Type / Stride	(Filter Shape) × Filters	Output Shape
Encoder	**block 0**			$9 \times H \times W \times 3$
		Conv3D / s=[1,1,1]	$(3 \times 3 \times 3 \times 3) \times 32$	$9 \times H \times W \times 32$
	block 1	Conv3D dw / s=[1,2,2]	$(3 \times 3 \times 3 \times 1) \times 32$	$9 \times \frac{H}{2} \times \frac{W}{2} \times 32$
		Conv3D pw / s=[1,1,1]	$(1 \times 1 \times 1 \times 32) \times 64$	$9 \times \frac{H}{2} \times \frac{W}{2} \times 64$
	block 2	Conv3D dw / s=[2,1,1]	$(3 \times 3 \times 3 \times 1) \times 64$	$5 \times \frac{H}{2} \times \frac{W}{2} \times 64$
		Conv3D pw / s=[1,1,1]	$(1 \times 1 \times 1 \times 64) \times 128$	$5 \times \frac{H}{2} \times \frac{W}{2} \times 128$
	block 3	Conv3D dw / s=[1,2,2]	$(3 \times 3 \times 3 \times 1) \times 128$	$5 \times \frac{H}{4} \times \frac{W}{4} \times 128$
		Conv3D pw / s=[1,1,1]	$(1 \times 1 \times 1 \times 128) \times 128$	$5 \times \frac{H}{4} \times \frac{W}{4} \times 128$
	block 4	Conv3D dw / s=[2,1,1]	$(3 \times 3 \times 3 \times 1) \times 128$	$3 \times \frac{H}{4} \times \frac{W}{4} \times 128$
		Conv3D pw / s=[1,1,1]	$(1 \times 1 \times 1 \times 128) \times 256$	$3 \times \frac{H}{4} \times \frac{W}{4} \times 256$
	block 5	Conv3D dw / s=[2,1,1]	$(3 \times 3 \times 3 \times 1) \times 256$	$2 \times \frac{H}{4} \times \frac{W}{4} \times 256$
		Conv3D pw / s=[1,1,1]	$(1 \times 1 \times 1 \times 256) \times 512$	$2 \times \frac{H}{4} \times \frac{W}{4} \times 512$
Decoder	**block 6**	Conv3DTrans pw/s=[3,2,2]	$(1 \times 1 \times 1 \times 512) \times 256$	$6 \times \frac{H}{2} \times \frac{W}{2} \times 256$
		Conv3D dw/s=[1,1,1]	$(3 \times 3 \times 3 \times 1) \times 256$	$6 \times \frac{H}{2} \times \frac{W}{2} \times 256$
	block 7	Conv3DTrans pw/s=[1,2,2]	$(1 \times 1 \times 1 \times 256) \times 64$	$6 \times H \times W \times 64$
		Conv3D dw / s=[1,1,1]	$(3 \times 3 \times 3 \times 1) \times 64$	$6 \times H \times W \times 64$
	block 8	Conv3DTrans pw/s=[1,1,1]	$(1 \times 1 \times 1 \times 64) \times 1$	$6 \times H \times W \times 1$
		Sigmoid Activation		$6 \times H \times W \times 1$

The output shape is in data format "LHWC", where L is the temporal length, H is the height, W is the width, C is the number of channels, dw represents "depthwise convolution", pw represents "pointwise convolution", and s represents the strides in temporal length, height, and width.

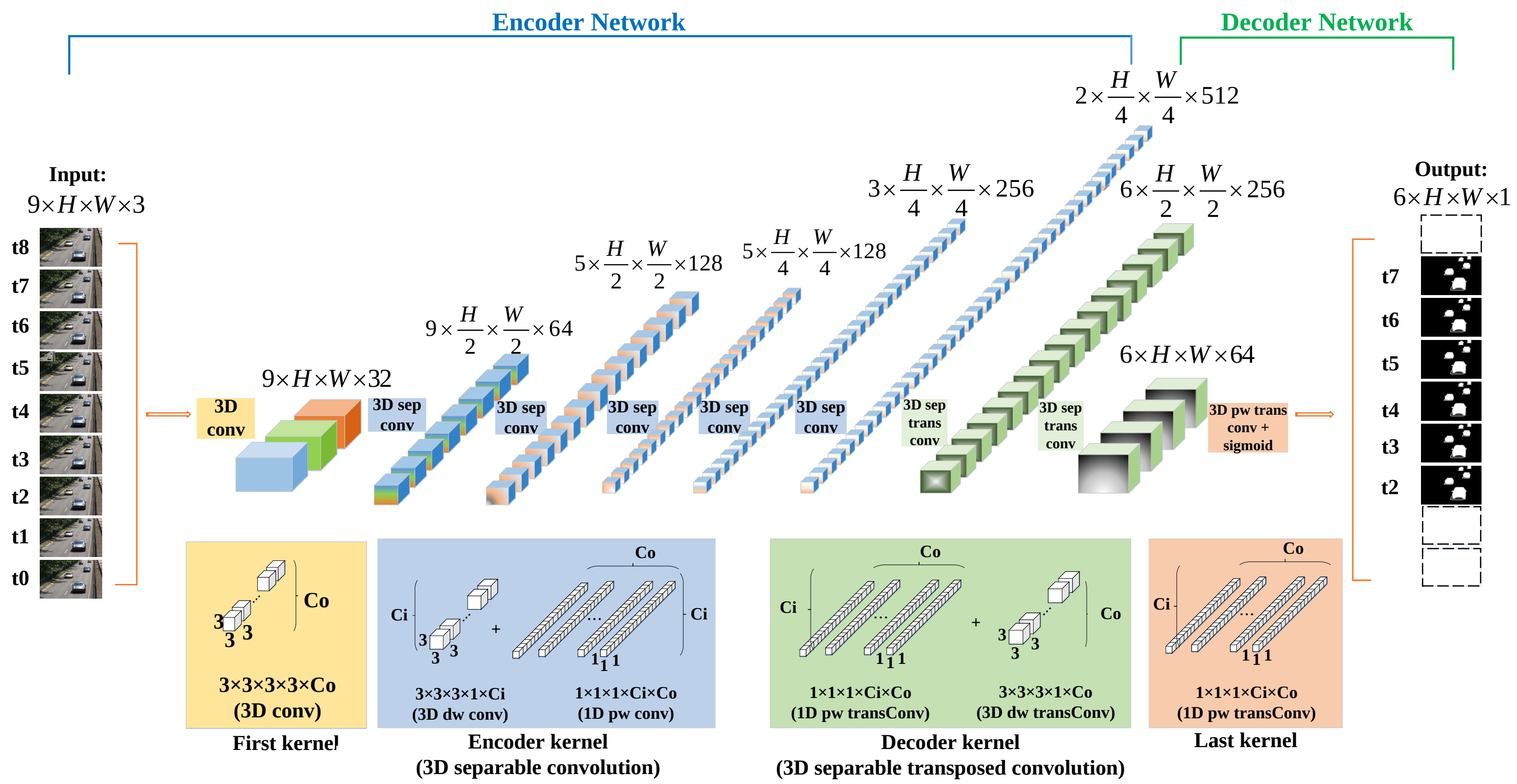


Figure 5.10: The architecture of the proposed 3DS_MM.

5.2.1 Encoder-decoder-based 3D separable CNN

As shown in Fig. 5.10, the proposed network is an encoder-decoder-based CNN utilizing the 3D separable convolution as described in Section 3.3. The network involves six blocks in the encoder network and three blocks in the decoder network. These block numbers are selected to provide a good trade-off between the inference speed and the detection accuracy empirically. Table 5.4 shows the details of the network and the shape of the input and output in each layer.

The Encoder Network

For each training sample, the input to the encoder network is a set of video frames in a 4D shape of $9 \times H \times W \times 3$ without background frame needed, where 9 is the number of video frames, H and W are the height and width of the video frames, and 3 is the RGB color channels. In Fig. 5.10, t_0, t_1, t_2, t_3, t_4... represent different time slots. In the first step, the standard 3D convolution described in Fig. 3.5(a) is adopted with 32 filters of size $3 \times 3 \times 3 \times 3$ to calculate the convolution on nine input frames. The input video frames are transformed to 32 feature maps in a shape of $9 \times H \times W \times 32$ at the output. In the following blocks, each of the output feature maps of each layer is convolved with an independent filter of size $3 \times 3 \times 3 \times 1$ with strides $[1, 2, 2]$ (in the direction of temporal length, height, width) for depthwise convolution, and then convolved with C_o filters of size $1 \times 1 \times 1 \times C_i$ with strides $[1, 1, 1]$ for pointwise convolution.

The Decoder Network

The output of the encoder network is fed to the decoder network for decoding to produce the binary masks of the moving objects.

Each layer of the decoder network adopts a transposed convolution, which spatially up-samples the encoded features and finally generates the binary masks at the same resolution as the input video frames.

The standard transposed convolution is split into a 1D pointwise transposed convolution and a 3D depthwise transposed convolution. These operations are defined similarly to the 1D pointwise convolution and the 3D depthwise convolution in the encoder network. In block 6 shown in Table 5.4, the encoder output of size $2 \times \frac{H}{4} \times \frac{W}{4} \times 512$ is converted to a tensor of size $6 \times \frac{H}{2} \times \frac{W}{2} \times 256$ using the 1D pointwise transposed convolution with 256 filters of size $1 \times 1 \times 1 \times 512$.

By setting strides to be $[3, 2, 2]$ for the temporal length, height and width in the pointwise transposed convolution, the feature maps are up-scaled by 3 times from 2 to 6 in the temporal length and enlarged by 2 times in height and width. Then followed by a 3D depthwise transposed convolution with 256 filters of size $3 \times 3 \times 3 \times 1$ and strides $[1, 1, 1]$, the feature maps are projected to a tensor of size $6 \times \frac{H}{2} \times \frac{W}{2} \times 256$ at the output of block 6. Block 7 is similarly defined. In the final block, the feature maps are projected to a 4D output of size $6 \times H \times W \times 1$, and a sigmoid activation function is appended to generate the probability masks for 6 successive frames. A threshold of 0.5 is applied to convert the probability masks to binary masks that indicate the detected moving objects.

5.2.2 MIMO strategy

Fig. 5.11 illustrates our proposed MIMO strategy and how it is different from SISO and MISO. The temporal-dimension L of a 4D input or output of size $L \times H \times W \times C$ is redefined as the number of input frames L_i and the number of output masks L_o. By applying different padding and stride values in the convolutions in the neural network, different number of output masks L_o can be predicted. In our study, we set L_i as 9 and L_o as 6. As shown in

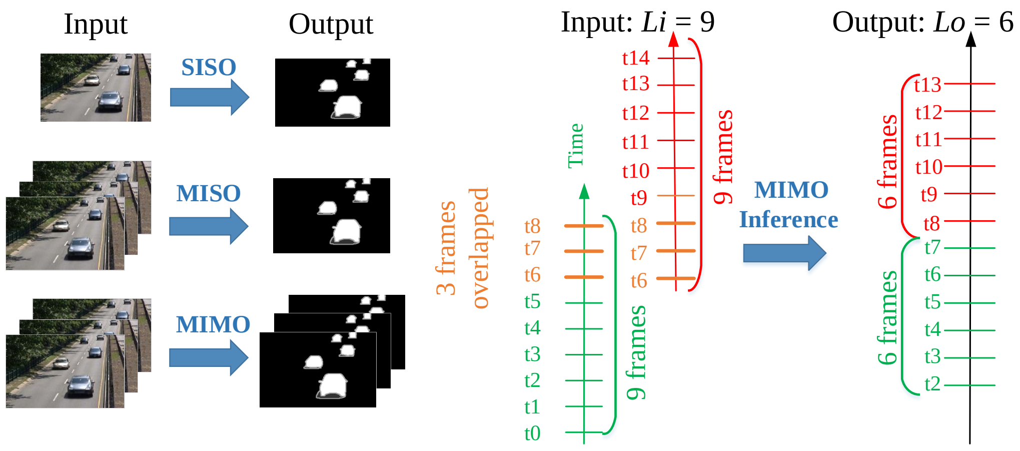


Figure 5.11: Left: Difference between Single-Input Single-Output (SISO), Multi-Input Single-Output (MISO), and Multi-Input Multi-Output (MIMO). Right: The proposed MIMO strategy used in the inference process.

Fig. 5.11 (above), in the inference process, two groups of 9 input frames with 3 frames overlapped can output two successive groups of 6 binary masks.

We also analyze how computational complexity can be reduced from MISO to this MIMO scheme. Let us consider our proposed network in Table 5.4. With the proposed MIMO scheme, the output layer in block 8 is of size $L_o \times H_o \times W_o \times (C_o = 1)$. Since block 8 mainly requires a pointwise convolution, the multiplications required to generate such output layer is $1 \times 1 \times 1 \times C_i \times L_o \times H_o \times W_o \times (C_o = 1) = C_i \times L_o \times H_o \times W_o$. Denote the total multiplications from block 0 to block 7 as M_{0-7}, then the overall complexity of generating L_o binary masks is

$$M_{0-7} + C_i \times L_o \times H_o \times W_o. \tag{5.15}$$

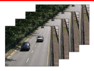
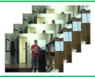
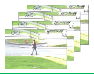
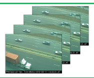
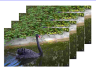
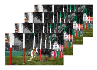
With the same network structure, if we adopt a MISO scheme, then the output layer is of size $(L_o = 1) \times H_o \times W_o \times (C_o = 1)$. The multiplications involved in block 8 to generate such output layer is $1 \times 1 \times 1 \times C_i \times (L_o = 1) \times H_o \times W_o \times (C_o = 1) = C_i \times H_o \times W_o$. To generate L_o output binary masks, the overall complexity is

$$(M_{0-7} + C_i \times H_o \times W_o) \times L_o = M_{0-7} \times L_o + C_i \times L_o \times H_o \times W_o. \tag{5.16}$$

Therefore, to output the same number of binary masks, MISO requires (5.16) − (5.15) = $(L_o - 1) \times M_{0-7}$ more multiplications than MIMO.

5.3 Training and Evaluation Setup

Table 5.5: Different data division schemes of scene dependent evaluation (SDE) and scene independent evaluation (SIE).

	Training and evaluation methodology		In this paper	Train	Test
SDE (Scene Dependent Evaluation)	Video-optimized		√	Video1:	
				Video2:	
SIE (Scene Independent Evaluation)	Video-agnostic	Category-wise	√	Category1: 	
				Category2:	
		Complete-wise	√	 	

To analyze how the proposed model performs, we conducted three experiments illustrated in Table 5.5: (1) video-optimized SDE setup on CDnet2014 dataset, (2) category-wise SIE setup on CDnet2014 dataset, and (3) complete-wise SIE setup on DAVIS2016 dataset. In SDE [52], frames in training and test sets were from the same video, whereas, in SIE [52], completely unseen videos were used for testing. Further, in category-wise SIE, the training and testing were done per category over CDnet2014, whereas, in complete-wise SIE, training and testing were done over the complete DAVIS2016 dataset.

All the experiments were carried out on an Intel Xeon with an 8-core 3GHz CPU and an Nvidia Titan RTX 24G GPU. The following sections present the details of the training and evaluation processes and performance evaluation metrics.

5.3.1 Video-optimized SDE setup on CDnet2014 dataset

The CDnet2014 dataset [80] was used in the experiment. It contains 11 video categories: baseline, badWeather, shadow, and so on. Each category has four to six videos, resulting in a total of 53 videos (e.g., the baseline category has sequences highway, office, pedestrians, and PETS2006). A video contains 900 to $7,000$ frames. The spatial resolution of the video frames varies from 240×320 to 576×720 pixels. In our experiments, we excluded the PTZ (pan–tilt–zoom) category since the camera has excessive motion.

We trained deep learning-based methods DeepBS [42], MSFgNet [43], VGG-PSL-CRF[44], BSPVGAN [62], RMS-GAN [65], MSCNN+Cascade [36], MsEDNet [39], FgSegNet_S [40], FgSegNet_M [40], FgSegNet_v2 [41], 2D_Separable CNN [59] and our proposed 3DS_MM in the same video-optimized SDE setup, in which a specific model was trained for each video.

From each video, we selected the first 50% of frames as the training set and the last 50% of frames as the test set. The SISO-based networks and the proposed MIMO-based 3DS_MM were using exactly the same frames for training. Suppose that one video contained 100 frames, then for the SISO-based networks, the first 50 frames $t_0 \sim t_{49}$ were used for training, and the last 50 frames $t_{50} \sim t_{99}$ were used for testing. For our proposed 3DS_MM, a 9-frame window slid over the same first 50% of frames, such as $t_0 \sim t_8$, $t_1 \sim t_9$, $t_2 \sim t_{10}, \ldots, t_{41} \sim t_{49}$ to form the training set if the stride was 1, and $t_{50} \sim t_{99}$ frames were for testing. In this way, all the deep-learning-based models were using the same frames for training. The only difference was that for the proposed network, the first 50% of frames were repeatedly utilized through the sliding operation. The traditional unsupervised methods WeSamBE [15], SemanticBGS [16], PAWCS [18], and SuBSENSE [20] were also tested on the same last 50% frames for performance comparison.

We used the RMSprop optimizer with binary cross-entropy loss function and trained each model for 30 epochs with batch size 1. The learning rate was initialized at 1×10^{-3} and was reduced by a factor of 10 if the validation loss did not decrease for 5 successive epochs.

5.3.2 Category-wise SIE setup on CDnet2014 dataset

In order to evaluate the generalization capability of the proposed 3DS_MM, we also run experiments for the SIE setup. Compared to SDE, in SIE the training and test sets contain a completely different set of videos. In the category-wise SIE setup, the training and evaluation were conducted per category. A leave-one-video-out (LOVO) strategy originally raised in [52] was applied to divide videos in each category into training and test sets for CDnet2014 dataset. For example, the baseline category contains four videos, then three videos (highway, office, PETS2006) were used for training, and the 4th video (pedestrians) was for testing. This SIE setup was carried out on seven categories, so for each method in comparison, seven models were trained totally from scratch.

The traditional unsupervised methods WeSamBE [15], PAWCS [18], and SuBSENSE [20] were compared in the category-wise SIE setup. We also compared our proposed 3DS_MM with the other DNN-based networks such as BMN-BSN [49], BSUV-Net [50], BSUV-Net 2.0 [51], and ChangeDet [52] which were demonstrated to have great performance on unseen videos.

We used the RMSprop optimizer with binary cross-entropy loss function and trained the model for 30 epochs with batch size 5. The learning rate was initialized at 1×10^{-3}

and was reduced by a factor of 10 if the validation loss did not decrease for five successive epochs.

5.3.3 Complete-wise SIE setup on DAVIS2016 dataset

We also conducted an experiment in complete-wise SIE setup on DAVIS2016 dataset. Different from the category-wise setup on CDnet2014, the complete-wise setup on DAVIS2016 refers to the training and evaluation on the whole dataset. In our experiment, 30 videos in DAVIS2016 dataset were used in training, and 10 completely unseen videos were used for testing. For each method in comparison, only one unified model was trained from scratch without using any pre-trained model data.

Semi-supervised deep learning-based methods such as MSK [70], CTN [71], SIAMMASK [72], PLM [75], and HEGNet [73], as well as FgSegNet_S [40], FgSegNet_M [40], FgSegNet_v2 [41], and 2D_Separable CNN [59] were trained and tested in the same SIE setup as our proposed 3DS_MM. We used the same training configuration parameters (optimizer, loss function, epochs, batch size, learning rate, etc.) as those in Section 5.4.2.

5.4 Experimental Results and Discussion

5.4.1 Ablation study

We first investigated the influence of different components of our proposed 3DS_MM through ablation experiments. In order to quantify the effect of two components "3D separable CNN" and "MIMO" in 3DS_MM, we conducted four experiments over 10 categories of CDnet2014 dataset in SDE setup. The results are shown in Table 5.6. We began with the standard 3D CNN and a MISO strategy, namely "3D CNN + MISO". It has an F-measure of 0.9532, a very low inference speed of 26 fps, approximately 9.13 M trainable parameters, and a computational complexity of 693.31 GFLOPs, which generates 1 output binary mask. To generate 6 output masks, the GFLOPs need to be multiplied by 6 (×6). We then replaced the standard 3D CNN by the 3D separable CNN, while the MISO strategy was retained. For a fair comparison, the 3D CNN and the 3D separable CNN structures adopted the same number of network layers, and their intermediate layers have the same output sizes. The resultant "3D separable CNN + MISO" method has a slightly reduced F-measure, but the inference speed increased from 26 fps to 31 fps. More importantly, the parameters and FLOPs were drastically reduced, due to the separable convolution operations. On the other hand, we retained the standard 3D CNN but replaced MISO by MIMO. In particular, we kept the front part of the network the same and only modify the last layer to output 6 binary masks instead of a single mask. The resultant method "3D CNN + MIMO" significantly increased the inference speed (144 fps) compared to "3D CNN + MISO".

Finally, the proposed "3D separable CNN + MIMO" method has a superior inference speed (154 fps) due to the MIMO strategy, as well as the fewest trainable parameters (~0.36 M) and FLOPs (~28.43 G) due to 3D separable convolutions. The above results have justified the effectiveness of our proposed model design.

Table 5.6: Ablation study of the proposed 3DS_MM.

Methods	Accuracy ↑ (F-measure)	Inference Speed ↑ (fps)	# Param ↓ (M)	FLOPs↓ (G)
3D CNN + MISO	0.9532	26	~9.13	~693.31 (×6)
3D separable CNN + MISO	0.9521	31	~0.36	~28.40 (×6)
3D CNN + MIMO	0.9522	144	~9.13	~693.97
3D separable CNN + MIMO	**0.9517**	**154**	**~0.36**	**~28.43**

#Param: Number of trainable parameters; M: millions; FLOPs: floating point operations, G: gigaflops; (×6): six times the FLOPs in order to generate the same number of output masks as the 'MIMO' strategy.

5.4.2 Objective performance evaluation

Objective Results in Video-Optimized SDE Setup on CDnet2014

The accuracy comparison of various methods in SDE setup in each video category is shown in Table 5.7. Each row lists the inference speed, F-measure, S-measure, E-measure and MAE values for a specific method, each column lists the algorithm category, learning type (supervised or unsupervised learning), input-output relationship (SISO, MISO or MIMO), inference speed, GPU type, and F-measure values averaged on test frames from a certain video category, while the last four columns show the average F-measure, S-measure, E-measure and MAE values across all video categories. The first four classical methods are traditional non-deep learning-based methods. These traditional models are tested on the same last 50% of frames as the other compared models. In the subsequent rows, the results of deep learning-based models, including our proposed model are obtained by training and testing in exactly the same SDE setup as introduced in Section 5.4.4. In Table 5.7, we highlight the best value in each column in bold. We observe that our proposed 3DS_MM model achieves the highest inference speed at 154 fps, and performs best in BDW-badWeather, DBG-dynamicBackground, IOM-intermittentObjectMotion, LFR-lowFramerate, and Turbulance categories in F-measure. It improved the average F-measure by 1.1% and 1.4% compared to methods with the second and third highest average F-measure values in Table 5.7. It also offers the highest average S-measure, E-measure, and the lowest average MAE values among all methods.

Objective Results in Category-Wise SIE Setup on CDnet2014

Table 5.8 lists the comparison results in category-wise SIE setup. Each column lists the inference speed and accuracy metrics values calculated on the unseen video being left out from each category for testing in the LOVO strategy. The models FgSegNet_S [40], FgSegNet_M [40], FgSegNet_v2 [41], BMN-BSN [49], BSUV-Net [50], BSUV-Net 2.0 [51], and ChangeDet [52] were trained and evaluated in the same SIE setup introduced in Section 5.4.2 as our proposed 3DS_MM. Our proposed 3DS_MM (with an inference speed at 154 fps, an F-measure of 0.8499, an S-measure of 0.8632, an E-measure of 0.9445, and an MAE of 0.0545) outperforms all the other listed methods in inference speed, while maintaining high detection accuracy by outperforming FgSegNet_S, FgSegNet_M, FgSegNet_v2, BMN-BSN, BSUV-Net, and BSUV-Net 2.0 by 26.6%, 34.8%, 24.9%, 7.2%, 2.7%, and 3.9% in F-measure, respectively. It achieves similar superiority in terms of S-measure, E-measure and MAE as well. Although ChangeDet [52] offers relatively better detection accuracy than our model, the inference speed of our model is 2.6 times that of ChangeDet.

Objective Results in Complete-Wise SIE Setup on DAVIS2016

All the models listed in Table 5.9 were trained and evaluated in the same complete-wise SIE setup as described in Section 5.4.2. It is more challenging for a model to perform well in such SIE setup on DAVIS2016 dataset, because (1) the complete-wise SIE setup mixes 30 different kinds of videos from the real-world together for training, and (2) the content complexity of DAVIS2016 dataset is high. We compared our proposed model 3DS_MM (with an inference speed at 154 fps and an average F-measure of 0.7317, S-measure of 0.7492, E-measure of 0.8024 and MAE of 0.2089 over 10 test videos) to the state-of-the-art semi-supervised deep learning-based models MSK [70], CTN [71], SIAMMASK [72], HEGNet [73], and PLM [75]. It turns out that our proposed model is superior over these models in inference speed. Besides, our model improved F-measure by 2.5%, 9.6% and 6.5% compared to CTN, PLM and SIAMMASK, respectively, and its F-measure is on par with HEGNet. Although MSK offers 1.5% higher F-measure than ours, its inference speed is extremely low. Our proposed model also outperforms the supervised learning-based models FgSegNet_S [40], FgSegNet_M [40], FgSegNet_v2 [41], and 2D_Separable CNN [59] in F-measure by 10.3%, 11.7%, 10.6%, and 16.5%, respectively. Our proposed method demonstrates a similar superiority in S-measure, E-measure and MAE values. Although there are other models in DAVIS Challenge with higher accuracy than our model, those models are far less efficient and their inference speed is too slow to be applied in delay-sensitive scenarios.

Table 5.7: Comparative F-measure, S-measure, E-measure and MAE performance in video-optimized SDE setup on CDnet2014 dataset.

Method	**Algorithms (Learning type, Input-Output)**	**Inference Speed ↑ (fps)**	**GPU**	**Accuracy**													
				F-measure ↑											**S-measure ↑**	**E-measure ↑**	**MAE ↓**
				BDW	BSL	CJT	DBG	IOM	NVD	LFR	SHD	THM	TBL	**Avg**	**Avg**	**Avg**	**Avg**
WeSamBE [15]	Traditional Methods (unSV)	2	CPU i5	0.8530	0.9293	0.7830	0.7274	0.7256	0.5801	0.6532	0.8492	0.7768	0.7667	0.7644	0.7835	0.8536	0.1423
SemanticBGS [16]		7	Titan	0.8190	0.9488	0.8332	0.9326	0.7742	0.4886	0.7818	0.9050	0.8025	0.6851	0.7971	0.8094	0.8935	0.1002
PAWCS [18]		27	CPU i5	0.8072	0.9277	0.7996	0.8772	0.7628	0.4024	0.6518	0.8719	0.8130	0.6350	0.7549	0.7644	0.8478	0.1453
SuBSENSE [20]		30	CPU i5	0.8539	0.9383	0.8006	0.8011	0.6433	0.5471	0.6375	0.8797	0.7977	0.7722	0.7671	0.7790	0.8598	0.1394
VGG-PSL-CRF [42]	Deep CNNs (SV, MISO)	4.9	Titan	0.8869	0.9474	0.9276	0.7190	0.7405	0.7398	0.6105	0.8890	0.8352	0.9137	0.8210	0.8398	0.9157	0.0801
DeepBS [40]		10	Titan	0.8221	0.9460	0.8844	0.8593	0.5962	0.5777	0.5932	0.9116	0.7389	0.8385	0.7768	0.7956	0.8712	0.1184
MSFgNet [41]		83.8	Titan	0.8424	0.9091	0.8167	0.8348	0.7669	0.7973	0.8352	0.9151	0.7822	0.8572	0.8357	0.8545	0.9266	0.0613
BSPVGAN[60]	GANs (SV, SISO)	10	Titan	0.9564	**0.9717**	**0.9747**	0.9683	0.9230	**0.8873**	0.8448	**0.9732**	**0.9570**	0.9240	0.9380	0.9466	0.9856	0.0123
RMS-GAN [63]	(SV, MISO)	50	Titan	0.9490	0.9658	0.9624	0.9612	0.9342	0.8812	0.9333	0.9262	0.9510	0.9434	0.9407	0.9490	0.9825	0.0155
MsEDNet [37]	Multiscale CNNs (SV, MISO)	13.6	Titan	0.8975	0.9248	0.9027	0.8902	0.8051	-	-	0.9002	0.8621	-	0.8832	0.8897	0.9766	0.0204
MSCNN+Cascade [34]		50	Titan	0.9351	0.9666	0.9612	0.9492	0.8358	0.8837	0.8312	0.9227	0.8764	0.9038	0.9066	0.9190	0.9568	0.0413
FgSegNet_M [38]		69	Titan	0.9307	0.9528	0.9403	0.9136	0.8943	0.8830	0.8897	0.9153	0.9160	0.7964	0.9032	0.9166	0.9789	0.0224
FgSegNet_S [38]	Deep CNNs (SV, SISO)	82	Titan	0.9331	0.9608	0.9407	0.9233	0.9045	0.8871	0.9123	0.9197	0.9152	0.7980	0.9095	0.9236	0.9758	0.0241
FgSegNet_v2 [39]		89	Titan	0.9396	0.9680	0.9475	0.9143	0.8985	0.8736	0.9247	0.9152	0.9196	0.8179	0.9119	0.9184	0.9876	0.0112
2D_Separable CNN [57]	2D Separable (SV, SISO)	149	Titan	0.9165	0.9552	0.9401	0.9324	0.9352	0.8459	0.9255	0.9030	0.9067	0.8936	0.9154	0.9304	0.9858	0.0123
Proposed 3DS_MM	3D Separable (SV, MIMO)	**154**	**Titan**	**0.9571**	0.9704	0.9417	**0.9686**	**0.9637**	0.8848	**0.9736**	0.9432	0.9516	**0.9621**	**0.9517**	**0.9687**	**0.9945**	**0.0067**

unSV: unsupervised learning, SV: supervised learning, SISO: single-input single-output, MISO: multi-input single-output, MIMO: multi-input multi-output. The best value in each column is highlighted in bold. ↑ Larger value of the metric denotes better performance. ↓ Smaller value of the metric denotes better performance.

Table 5.8: Comparative F-measure, S-measure, E-measure and MAE performance in category-wise SIE setup for unseen videos on CDnet2014 dataset.

Method	**Learning Type, Input-Output**	**Inference Speed ↑ (fps)**	**GPU**	**Accuracy**										
				F-measure ↑								**S-measure ↑**	**E-measure ↑**	**MAE ↓**
				blizzard-BDW	pedestrians-BSL	boats-DBG	turnpike5fps-LFR	winterStreet-NVD	busStation-SHD	corridor-THM	**Avg**	**Avg**	**Avg**	**Avg**
WeSamBE [15]	unSV	2	CPU i5	0.8584	0.9569	0.6401	0.9130	0.5900	0.8628	0.8944	0.8165	0.8198	0.9112	0.0723
PAWCS [18]	unSV	27	CPU i5	0.6612	0.9511	0.8820	0.9072	0.4610	0.8583	0.6489	0.7671	0.7746	0.8003	0.1789
SuBSENSE [20]	unSV	30	CPU i5	0.8501	0.9500	0.6893	0.8531	0.4469	0.8577	**0.9129**	0.7943	0.7990	0.8432	0.1432
BSUV-Net [48]	SV, MISO	6	Titan	0.8195	**0.9765**	0.9004	0.6802	0.6100	**0.9398**	0.8350	0.8231	0.8342	0.9109	0.0691
BSUV-Net 2.0 [49]	SV, MISO	29	Titan	0.8310	0.9630	0.8750	0.7077	0.6170	0.8012	0.8743	0.8100	0.8301	0.9032	0.0910
BMN-BSN [47]	SV, MISO	48	Titan	0.8401	0.9523	0.6400	0.6893	0.6122	0.9211	0.7933	0.7783	0.7894	0.8712	0.1213
ChangeDet [50]	SV, MISO	58.8	Titan	**0.9484**	0.9490	**0.9182**	0.8492	0.7699	0.7801	0.8350	**0.8643**	**0.8798**	**0.9484**	**0.0466**
FgSegNet_M [38]	SV, MISO	69	Titan	0.5511	0.7209	0.6857	0.2233	0.4200	0.6051	0.3104	0.5024	0.5232	0.6043	0.3812
FgSegNet_S [38]	SV, SISO	82	Titan	0.7412	0.6478	0.4045	0.5767	0.4500	0.5244	0.7435	0.5840	0.5987	0.6543	0.3778
FgSegNet_v2 [39]	SV, SISO	89	Titan	0.6990	0.6310	0.6189	0.5290	0.4300	0.5415	0.7590	0.6012	0.6281	0.7223	0.2712
Proposed 3DS_MM	SV, MIMO	**154**	**Titan**	0.8942	0.9165	0.7998	**0.9147**	**0.7856**	0.7978	0.8409	**0.8499**	**0.8632**	**0.9445**	**0.0545**

unSV: unsupervised learning, SV: supervised learning, SISO: single-input single-output, MISO: multi-input single-output, MIMO: multi-input multi-output. The best value in each column is highlighted in bold. The second best average accuracy values are also highlighted. ↑ Larger value of the metric denotes better performance. ↓ Smaller value of the metric denotes better performance.)

Table 5.9: Comparative F-measure, S-measure, E-measure and MAE performance in complete-wise SIE setup for unseen videos on DAVIS2016 dataset.

Method	**Learning Type, Input-Output**	**Inference Speed ↑ (fps)**	**GPU**	**Accuracy**													
				F-measure ↑											**S-measure ↑**	**E-measure ↑**	**MAE ↓**
				camel	car-roundab	car-shadow	cows	goat	horsejump-high	kite-surf	paragliding-launch	parkour	soapbox	**Avg**	**Avg**	**Avg**	**Avg**
MSK [68]	semi-SV, MISO	0.5	Titan	0.7350	**0.9260**	0.9480	**0.8120**	**0.8140**	0.8510	0.4380	0.2290	0.8740	**0.8420**	**0.7469**	**0.7598**	**0.8068**	**0.1900**
CTN [69]	semi-SV, MISO	4.5	Titan	0.7250	0.7750	0.8670	0.7750	0.7460	**0.8660**	0.4600	0.2270	**0.8820**	0.7440	0.7067	0.7123	0.7855	0.2102
PLM [73]	semi-SV, MISO	9.5	Titan	0.6130	0.7140	0.7310	0.7410	0.6940	0.7860	0.4560	0.1810	0.8120	0.6300	0.6358	0.6436	0.6975	0.2890
HEGNet [71]	semi-SV, MISO	12.5	Titan	0.7490	0.7892	0.7798	0.7792	0.7312	0.7402	0.6843	0.7392	0.8029	0.6500	0.7304	0.7489	0.7837	0.2110
SIAMMASK [70]	semi-SV, MISO	78	Titan	0.7480	0.8720	**0.9780**	0.7720	0.7210	0.6880	0.3260	0.1910	0.8290	0.5470	0.6672	0.6703	0.7182	0.2701
FgSegNet_M [38]	SV, MISO	69	Titan	0.6047	0.4892	0.8704	0.5620	0.4009	0.6199	0.6308	0.8639	0.5190	0.5835	0.6144	0.6265	0.7034	0.2803
FgSegNet_S [38]	SV, SISO	82	Titan	0.6163	0.5194	0.8940	0.5356	0.4063	0.6273	0.6904	0.8738	0.5345	0.5902	0.6288	0.6398	0.7134	0.2511
FgSegNet_v2 [39]	SV, SISO	89	Titan	0.6201	0.5120	0.8744	0.5309	0.4509	0.5940	0.6820	0.8729	0.5029	0.6194	0.6260	0.6379	0.7201	0.2710
2D_Separable CNN [57]	SV, SISO	149	Titan	0.5235	0.5286	0.8304	0.5387	0.4701	0.3815	0.4729	0.8163	0.4818	0.6209	0.5665	0.5934	0.6235	0.3723
Proposed 3DS_MM	SV, MIMO	**154**	**Titan**	**0.7495**	0.7103	0.7849	0.7039	0.7290	0.6103	**0.7012**	**0.8749**	0.7693	0.6835	**0.7317**	**0.7492**	**0.8024**	**0.2089**

semi-SV: semi-supervised learning, SV: supervised learning. SISO: single-input single-output, MISO: multi-input single-output, MIMO: multi-input multi-output. The best value in each column is highlighted in bold. The second best average accuracy values are also highlighted. ↑ Larger value of the metric denotes better performance. ↓ Smaller value of the metric denotes better performance.)

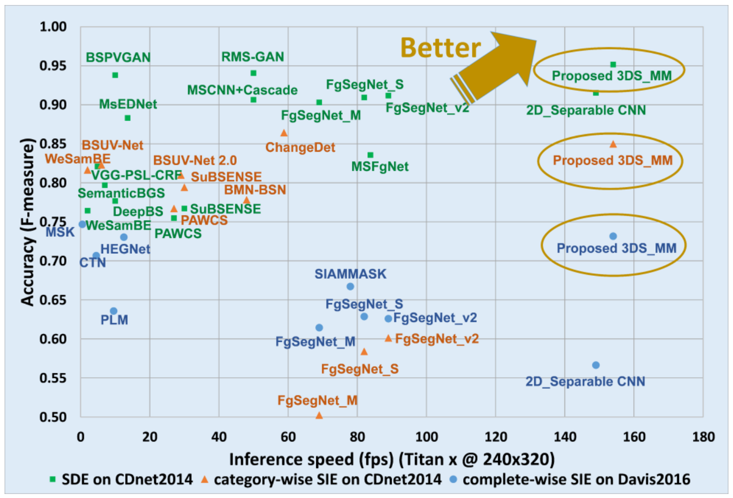
5.4.3 Accuracy, speed, memory, and computational complexity

Fig. 5.12 displays the detection accuracy metrics in F-measure, S-measure, E-measure and MAE versus the inference speed of all the compared models in the SDE setup, category-wise SIE setup, and complete-wise SIE setup. Since we aim at delay-sensitive applications, we expect our proposed 3DS_MM to offer overwhelmingly high inference speed, and a superior detection accuracy among models with high inference speeds. In Fig. 5.12, we observe that our proposed 3DS_MM surpasses all the other schemes in inference speed in all three experiment setups. In terms of the F-measure, S-measure, E-measure and MAE, in the SDE setup our method is the best among all models, while in both the category-wise and complete-wise SIE setups our method is the best among all models with an inference speed above 65 fps.

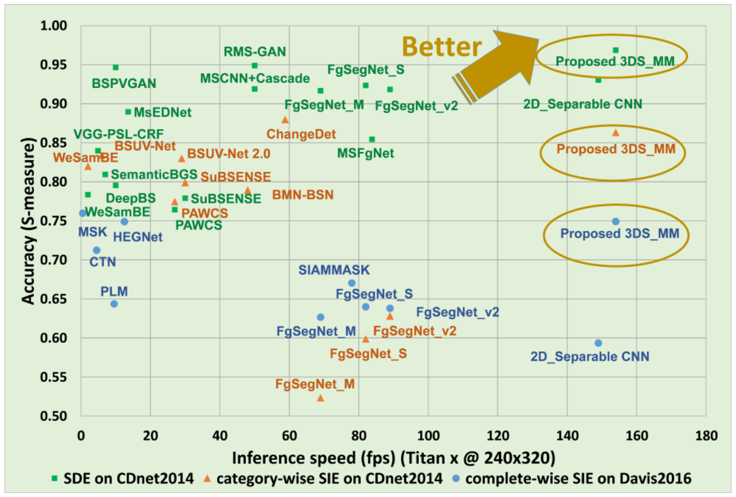
Table 5.10: The comparison between our proposed method and other deep learning-based methods for speed, trainable parameters, computational complexity, model size, and accuracy metrics values. The Table is sorted in ascending order of the inference speed.

Method	**Inference Speed ↑ (fps)**	**# Param ↓ (M)**	**FLOPs ↓ (G)**	**Model Size ↓ (MB)**	**SDE**				**SIE (category-wise)**				**SIE (complete-wise)**			
					F ↑	**S↑**	**E ↑**	**MAE ↓**	**F ↑**	**S↑**	**E ↑**	**MAE ↓**	**F ↑**	**S↑**	**E ↑**	**MAE ↓**
MSK [68]	0.5	-	-	-	-	-	-	-	-	-	-	-	**0.7469**	**0.7598**	**0.8068**	**0.1900**
CTN [69]	4.5	-	-	-	-	-	-	-	-	-	-	-	0.7067	0.7123	0.7855	0.2102
VGG-PSL-CRF [42]	4.9	~ 48.72	~3270 G	127	0.8210	0.8398	0.9157	0.0801	-	-	-	-	-	-	-	-
BSUV-Net [48]	6.0	-	-	116	-	-	-	-	0.8231	0.8342	0.9109	0.0691	-	-	-	-
PLM [73]	9.5	-	-	-	-	-	-	-	-	-	-	-	0.6358	0.6436	0.6975	0.2890
DeepBS [40]	10.0	~ 3.15	~1750 G	28.46	0.7768	0.7956	0.8712	0.1184	-	-	-	-	-	-	-	-
BSPVGAN [60]	10.0	-	-	-	0.9380	0.9466	0.9856	0.0123	-	-	-	-	-	-	-	-
HEGNet [71]	12.5	-	-	-	-	-	-	-	-	-	-	-	0.7304	0.7489	0.7837	0.2110
MsEDNet [37]	13.6	~ 23.29	~1120 G	95	0.8832	0.8897	0.9766	0.0204	-	-	-	-	-	-	-	-
BSUV-Net 2.0 [49]	29.0	~15.90	~540 G	110	-	-	-	-	0.8100	0.8301	0.9032	0.0910	-	-	-	-
BMN-BSN [47]	48.0	-	-	-	-	-	-	-	0.7783	0.7894	0.8712	0.1213	-	-	-	-
MSCNN+Cascade [34]	50.0	~ 10.30	~318 G	76.35	0.9066	0.9190	0.9568	0.0413	-	-	-	-	-	-	-	-
RMS-GAN [63]	50.0	-	-	-	0.9407	0.9490	0.9825	0.0155	-	-	-	-	-	-	-	-
ChangeDet [50]	58.8	**~ 0.13**	~262 G	1.59	-	-	-	-	**0.8643**	**0.8798**	**0.9484**	**0.0466**	-	-	-	-
FgSegNet_M [38]	69.0	~ 15.83	~220 G	60.40	0.9032	0.9166	0.9789	0.0224	0.5024	0.5232	0.6043	0.3812	0.6144	0.6265	0.7034	0.2803
SIAMMASK [70]	78.0	-	-	-	-	-	-	-	-	-	-	-	0.6672	0.6703	0.7182	0.2701
FgSegNet_S [38]	82.0	~ 8.16	~199 G	31.20	0.9095	0.9236	0.9758	0.0241	0.5840	0.5987	0.6543	0.3778	0.6288	0.6398	0.7134	0.2511
MSFgNet [41]	83.8	~ 0.29	~193 G	1.48	0.8357	0.8545	0.9266	0.0613	-	-	-	-	-	-	-	-
FgSegNet_v2 [39]	89.0	~ 7.49	~181 G	29.80	0.9119	0.9184	0.9876	0.0112	0.6012	0.6281	0.7223	0.2712	0.6260	0.6379	0.7201	0.2710
Proposed 3DS_MM	**154.0**	~ 0.36	**~28.43 G**	**1.45**	**0.9517**	**0.9687**	**0.9945**	**0.0067**	**0.8499**	**0.8632**	**0.9445**	**0.0545**	**0.7317**	**0.7492**	**0.8024**	**0.2089**

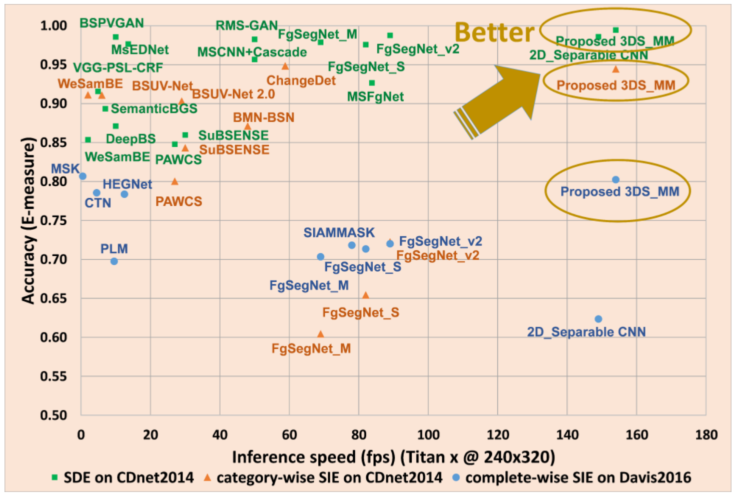
#Param: Number of trainable parameters; M: millions; G: gigaflops; F: F-measure; S: S-measure; E: E-measure; MAE: mean absolute error. The best value in each column is highlighted in bold. The second best accuracy values are also highlighted. ↑ Larger value of the metric denotes better performance. ↓ Smaller value of the metric denotes better performance.



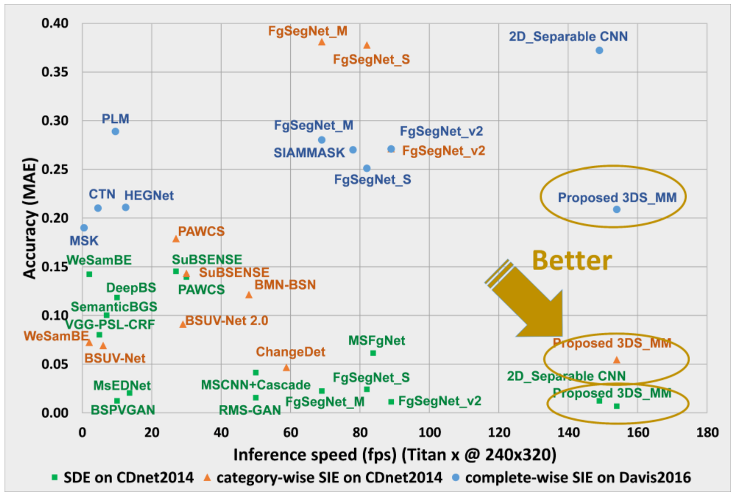
(a) F-measure vs. Inference speed



(b) S-measure vs. Inference speed



(c) E-measure vs. Inference speed



(d) MAE vs. Inference speed

Figure 5.12: Accuracy vs. inference speed (in fps) on an NVIDIA Titan GPU of our proposed model and other compared models in the three experiments (in SDE, category-wise SIE, and complete-wise SIE setup).

In Table 5.10, we summarize the overall performance including inference speed, trainable parameters, computational complexity, model size, and detection accuracy of our proposed 3DS_MM and other methods. The table is sorted in an ascending order of the inference speed. It is evident that the proposed 3DS_MM outperforms all the other listed methods with the highest inference speed at 154 fps, which is increased by 1.7 times and 1.8 times respectively, compared to the second and third fastest methods in Table 5.10. The computational complexity and the model size of our proposed method are 28.43 GFLOPs and 1.45 MB, smaller than all the other models in Table 5.10, due to our proposed 3D separable convolution.

In terms of detection accuracy (F-measure, S-measure, E-measure, and MAE), our proposed model outperforms all other models in SDE setup. In category-wise SIE setup, our proposed method offers the second best accuracy scores. Although it is slightly worse than changeDet [52], its inference speed (154 fps) is 2.6 times that of ChangeDet (58.8 fps). In complete-wise SIE setup, although our model offers slightly worse accuracy scores than MSK [70], it offers overwhelming superiority in terms of inference speed. The extremely low inference speed of MSK (0.5 fps) hinders the practical use of this model for delay-sensitive applications.

The number of trainable parameters of our proposed model (~0.36 million) is much less than most of the models in comparison. The reason that ChangeDet [52] (~0.13 million) and MSFgNet [43] (~0.29 million) have fewer trainable parameters than ours is because they use 2D filters and they are shallower networks with fewer convolutional layers, while our proposed 3DS_MM uses 3D filter and a deeper network. Nevertheless, the inference speeds of ChangeDet and MSFgNet are much slower than ours since they are both MISO networks. In contrast, our 3DS_MM is able to significantly increase the inference speed due to the proposed MIMO strategy and 3D separable convolution.

5.4.4 Subjective performance evaluation

In addition to objective performance, we also provide visual quality comparison as shown in Fig. 5.13[2], Fig. 5.14, and Fig. 5.15.

Subjective Results in Video-Optimized SDE Setup on CDnet2014

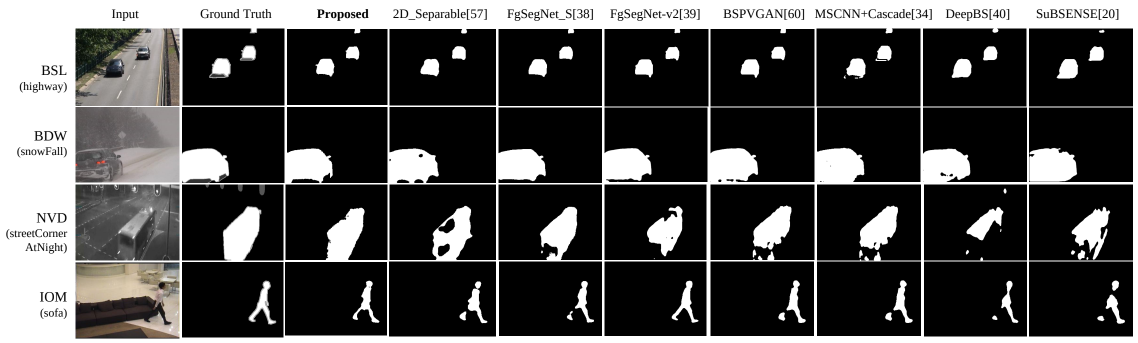


Figure 5.13: Visual comparison of sample results from CDnet2014 dataset in video-optimized SDE setup. BSL: baseline, BDW: badWeather, NVD: nightVideo, IOM: intermittentObjectMotion.

In Fig. 5.13, we randomly picked a sample test frame from categories BSL-baseline, BDW-badWeather, NVD-nightVideos, and IOM-intermittentObjectMotion. We observe that (1) the proposed 3DS_MM provides more details and clearer edges in the detected foreground objects, such as the car mirrors in "BSL" and "BDW", and (2) the proposed method detects more contiguous objects such as the bus in "NVD" and the walking man in "IOM". In contrast, the detected binary masks of other methods in comparison have either blurry edges or missing parts.

[2]There are some non-ROI (non-region-of-interest) areas shown as gray color regions in the ground truth images, which were not considered in the training.

Subjective Results in Category-Wise SIE Setup on CDnet2014

In Fig. 5.14, we randomly select a sample frame from each of the four categories (BSL-baseline, BDW-badWeather, LFR-lowFramerate, SHD-shadow) of CDnet2014 test results to show the visual quality of the models in Category-Wise SIE setup. Our proposed model has a better generalization capability compared to other models. It shows that our proposed model detects clearer shapes of the persons in BSL and SHD, and detects more details of person legs in SHD. The results of other methods, however, are either noisy, blurry, or have missing parts. In addition, the proposed model performs better in BDW and LFR categories with clear and correct shapes, while other models detect excessive or non-contiguous content.

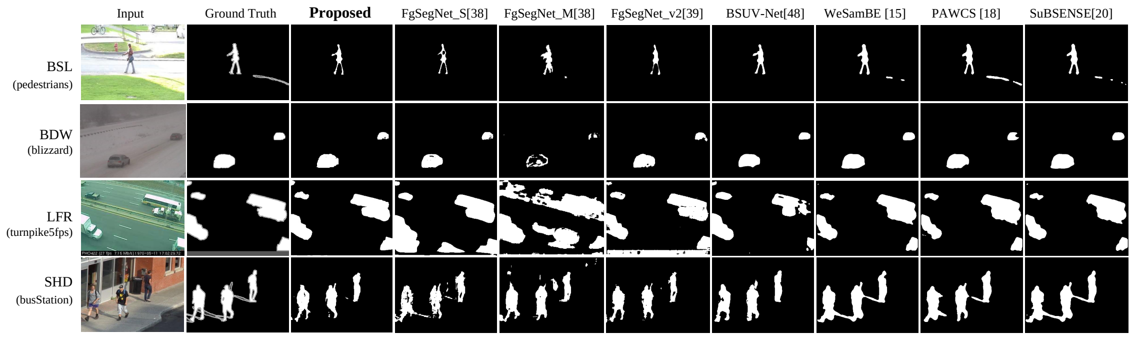


Figure 5.14: Visual comparison of unseen sample results from CDnet2014 dataset in category-wise SIE setup. BSL: baseline, BDW: badWeather, LFR: lowFramerate, SHD: shadow.

Subjective Results in Complete-Wise SIE Setup on DAVIS2016

In Fig. 5.15, we randomly select four videos (camel, horsejump-high, paragliding-launch, and kite-surf) from the results of DAVIS2016. Our proposed model detects the shapes of objects consistently well for all four videos, while the detection results of 2D_Separable [59], FgSegNet_S [40], FgSegNet_v2 [41], and SIAMMASK [72] are either

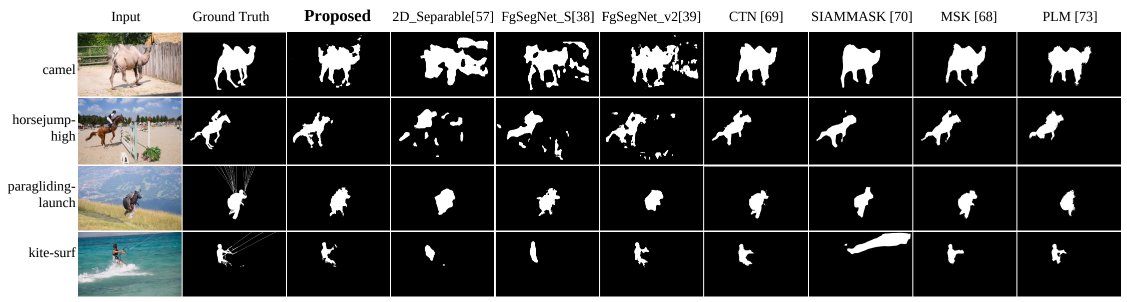


Figure 5.15: Visual comparison of unseen sample results from DAVIS2016 dataset in complete-wise SIE setup.

noisy or incomplete. Besides, the detection results of CTN [71], MSK [70], and PLM [75] for the kite-surf video are less accurate than the proposed model.

5.5 Conclusion

In this chapter, we propose the 3DS_MM model for moving object detection. Our model is designed specifically for memory- and computation-resource-limited environments and for delay-sensitive tasks. Our model utilizes spatial-temporal information in the video data via 3D convolution. The proposed 3D depthwise and pointwise convolutions with the MIMO strategy effectively reduce computational complexity and significantly enhance the inference speed. In addition, the 3D separable convolution leads to very few trainable parameters and a small model size. Finally, the defined SDE and SIE experiments demonstrate that our proposed model achieves superior detection accuracy among all compared models with high inference speeds suitable for low-latency vision applications.

CHAPTER 6

Proposed Method and Experiments - F3DsCNN

6.1 Introduction

In this chapter, we extend the way of using 3D separable convolution. Instead of one-branch Encode-decoder structure in ”3DS_MM” in Chapter 5, we propose a fast MOD algorithm called “F3DsCNN” based on a two-branch network architecture and the 3D separable convolution. The network extracts both high-level global features and low-level detailed features. It achieves a fast inference speed of 120 frames per second, suitable for tasks that need to be carried out in a timely manner on a computationally limited platform with high accuracy.

The rest of the chapter is organized as follows. In Section 6.2, we elaborate on our proposed network in detail. Section 6.3 describes our experimental results compared to the state-of-the-art models. Section 6.4 concludes the chapter.

6.2 Proposed F3DsCNN Network

The proposed deep moving object detection network [102] shown in Fig. 6.16 is based on a two-branch structure that captures global context and detailed information. While existing two-branch models [103, 104, 105, 106] adopt 2D convolutions, we adopt 3D convolution to explore spatial-temporal information. Further, to reduce complexity, we replace the standard 3D convolution by the 3D separable convolution.

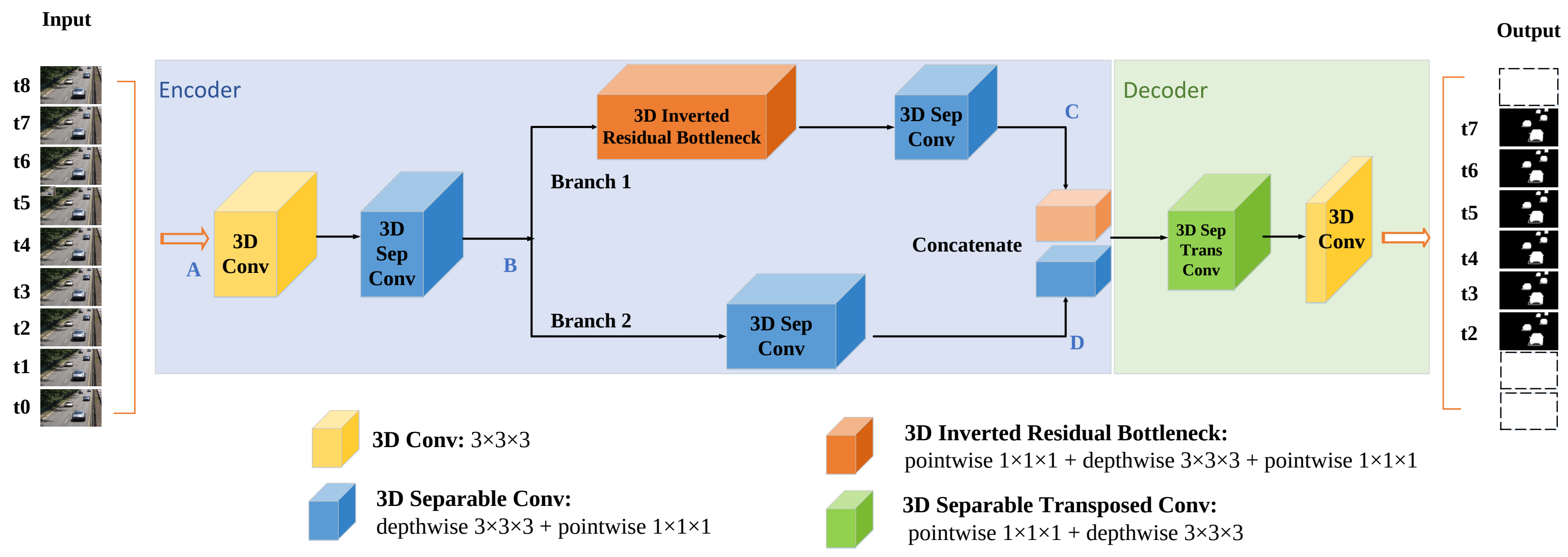

Input
t8
t7
t6
t5
t4
t3
t2
t1
t0
Encoder
A
3D Conv
3D Sep Conv
B
Branch 1
3D Inverted Residual Bottleneck
3D Sep Conv
C
Branch 2
3D Sep Conv
D
Concatenate
Decoder
3D Sep Trans Conv
3D Conv
Output
t7
t6
t5
t4
t3
t2
3D Conv: 3×3×3
3D Separable Conv:
depthwise 3×3×3 + pointwise 1×1×1
3D Inverted Residual Bottleneck:
pointwise 1×1×1 + depthwise 3×3×3 + pointwise 1×1×1
3D Separable Transposed Conv:
pointwise 1×1×1 + depthwise 3×3×3

Figure 6.16: The architecture of the proposed network F3DsCNN.

6.2.1 Design of proposed network

Two-branch Network

In order to increase detection accuracy, we design a two-branch network for feature extraction. Traditional two-branch methods [104, 105, 106] extract global features from low-resolution images with deeper neural network, and extract spatial details from full-resolution images with shallow neural network structures. To reduce computational complexity, the two branches can share the first few layers [103]. Our proposed method is shown in Fig.6.16. Branch 1 adopts deep inverted residual bottleneck layers to extract global features, and branch 2 adopts a shallower network to extract lower-level features. The two branches share layers from point A to B. The advantage of such layer sharing is that it learns a low-resolution representation from a full-resolution image, which is then used as the input of global feature extraction in branch 1, and simultaneously this low-resolution representation encodes the full-resolution image for detailed spatial feature extraction in branch 2. Finally, these two types of features are concatenated and fed into the decoder.

Inverted Residual BottleNeck with 3D Separable Convolution

In branch 1, we adopt the inverted residual bottleneck module that was originally proposed in MobileNet-V2 [107]. In an inverted residual bottleneck module [107], the input features with C_l channels are first expanded to a high-dimensional space with $C_h > C_l$ channels using a pointwise convolution. Subsequently, a 2D depthwise convolution with nonlinear activations is performed on each of these C_h channels. Afterwards, another pointwise convolution with linear activatons projects the features back onto a low-dimensional space with C_l channels. The reason for such operations is that it is better to apply nonlinear activations in a high-dimensional space than in a low-dimensional space to prevent information loss. To utilize spatio-temporal information in video data and to increase detection accuracy, we propose to replace such 2D separable convolutions in the inverted residual bottleneck [107] by 3D separable convolutions. The redesigned 3D inverted residual bottleneck first expands the 4D input of size $C_l \times L \times H \times W$ to a high-dimensional space by a pointwise convolution with C_h filters of size $C_l \times 1 \times 1 \times 1$ (C_l is the low-dimensional input channel, C_h is the high-dimensional output channel, $C_h > C_l$). Subsequently, a 3D depthwise convolution with a filter of $3 \times 3 \times 3$ (time $\times$ height $\times$ width) is performed on each of the C_h channels, and the output is then projected back to the low-dimensional space using another pointwise convolution with C_l filters of size $C_h \times 1 \times 1 \times 1$ with linear activations.

6.2.2 Configuration of proposed network

We use the format of "CLHW" to represent data, which denotes the number of channels C, the temporal length L, the height of the image H, and the width of the image W.

Table 6.11: The configuration of proposed network F3DsCNN.

Block		Input	Output	Layers
Encoder	3D Conv	3×9×240×320	32×9×240×320	1
	3D SepConv	32×9×240×320	64×5×30×40	9
Branch1	3D InvertedBottleNeck	64×5×30×40	256×2×15×20	24
Branch2	3D SepConv	64×5×30×40	256×2×15×20	8
Decoder	3D SepTransposedConv	512×2×15×20	1×6×120×160	4
	3D Conv	1×6×120×160	1×6×240×320	1

Input/Output data format (CLHW): C: Channel, L: Temporal length, H: Image height, W: Image width.

In Table 6.11, for each training sample, the input to the encoder network is a set of consecutive video frames in a 4D shape of $3 \times 9 \times 240 \times 320$, where 3 is the RGB color channels, 9 is the number of video frames, and 240 and 320 are the height and width of the video frames. In Fig. 6.16, t_0, t_1, t_2, t_3, t_4... represent different time slots. In the first step, standard 3D convolution is adopted with 32 filters of size $3 \times 3 \times 3 \times 3$ to calculate the convolution on nine input frames. The input video frames are transformed to 32 feature maps in a shape of $32 \times 9 \times 240 \times 320$ at the output. In the following blocks, the feature maps are down-sampled by 9 layers of 3D separable CNN and then separately go through the 24 layers of 8 consecutive 3D inverted residual bottleneck modules in branch 1 for deep global feature extraction, and through 8 layers of 3D separable CNN in branch 2 for shallower feature extraction, and then the outputs of the two branches are concatenated together to be fed into the decoder. In the decoder, we employ 4 layers of 3D separable transposed convolution and 1 layer of standard 3D convolution. A sigmoid activation function is appended at the end to generate the probability masks for 6 successive frames in a shape of $1 \times 6 \times 240 \times 320$.

6.3 Experimental Results and Analysis

To analyze how the proposed model performs, we conducted two experiments: (A) Training and evaluation on seen videos of CDnet2014 dataset [80], and (B) Training and evaluation on unseen videos of DAVIS2016 dataset [101]. In Experiment (A), frames in training and test sets were non-overlapped, but from the same video, whereas, in Experiment (B), videos completely unseen in the training set were used for testing.

To evaluate the performance of our proposed model, the inference speed is measured in frames per second (fps), and the detection accuracy is measured by F-measure.

We used the RMSprop optimizer with binary cross-entropy loss function and trained the model for 30 epochs with batch size 5. The learning rate was initialized at 1×10^{-3} and was reduced by a factor of 10 if the validation loss did not decrease for five successive epochs. Both experiments were carried out on an Intel Xeon with an 8-core 3GHz CPU and an Nvidia Titan RTX 24GB GPU. The number of trainable parameters for the proposed model is 4.34 millions.

6.3.1 Training and evaluation on seen videos (CDnet2014)

The CDnet2014 dataset has 11 video categories which include a total of 53 video sequences. In Experiment (A), we excluded the PTZ (pan-tilt-zoom) category since the camera has excessive motion. The proposed model was trained on the first 50% frames in each of the 49 videos, and test on the last 50% frames from the same videos.

All the other nine compared deep learning-based methods such as VGG-PSL-CRF [44], DeepBS [42], BSPVGAN [62], MsEDNet [39], MSCNN+Cascade [36], MS-FgNet [43], as well as FgSegNet_S [40], FgSegNet_M [40], and FgSegNet_v2 [41] were trained and tested in the same setup as our proposed model F3DsCNN.

Table 6.12 shows the objective performance. Each column lists the inference speed in fps and detection accuracy in F-measure values averaged on test frames from a certain video category, while the last column shows the average F-measure values across all the 10 video categories. We found that our proposed model outperforms the other nine deep-learning methods by 8.3% on average in F-measure and achieves the highest inference speed at 120 fps. Fig. 6.13 (a)shows the visual subjective performance of our proposed model in Experiment (A) on CDnet2014 dataset. We randomly picked a sample test frame from categories BSL-baseline, LFR-lowFramerate, and NVD-nightVideos. We observe that the proposed F3DsCNN provides more details and clearer edges in the detected foreground objects, such as the car mirrors in “BSL” and the truck in ”LFR”. Moreover, the proposed method detects more accurate and contiguous objects such as the bus in “NVD”. In contrast, the detected binary masks of other methods in comparison have either blurry edges or missing parts.

Table 6.12: Comparative F-measure performance for seen videos on CDnet2014 dataset.

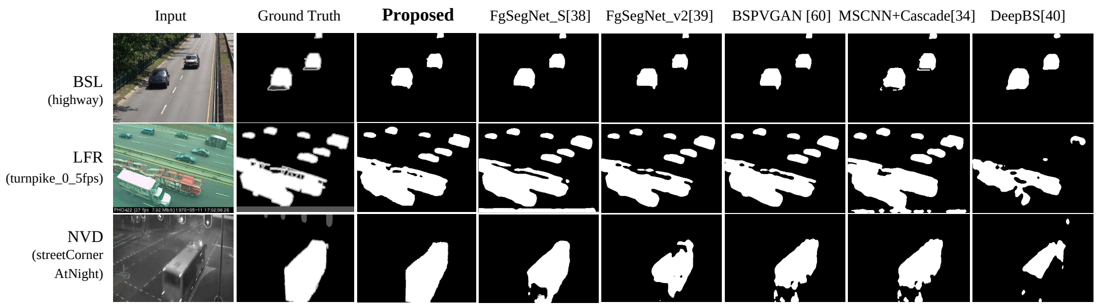
Method	Inference Speed (fps)	F-measure										
		BDW	BSL	CJT	DBG	IOM	NVD	LFR	SHD	THM	TBL	Avg
VGG-PSL-CRF [42]	4.9	0.8869	0.9474	0.9276	0.7190	0.7405	0.7398	0.6105	0.8890	0.8352	0.9137	0.8210
DeepBS [40]	10	0.8221	0.9460	0.8844	0.8593	0.5962	0.5777	0.5932	0.9116	0.7389	0.8385	0.7768
BSPVGAN[60]	10	0.9564	0.9717	0.9747	0.9683	0.9230	0.8873	0.8448	**0.9732**	0.9570	0.9240	0.9380
MsEDNet [37]	13.6	0.8975	0.9248	0.9027	0.8902	0.8051	-	-	0.9002	0.8621	-	0.8832
MSCNN+Cascade [34]	50	0.9351	0.9666	0.9612	0.9492	0.8358	0.8837	0.8312	0.9227	0.8764	0.9038	0.9066
FgSegNet_M [38]	69	0.9307	0.9528	0.9403	0.9136	0.8943	0.8830	0.8897	0.9153	0.9160	0.7964	0.9032
FgSegNet_S [38]	82	0.9331	0.9608	0.9407	0.9233	0.9045	0.8871	0.9123	0.9197	0.9152	0.7980	0.9095
MSFgNet [33]	83.8	0.8424	0.9091	0.8167	0.8348	0.7669	0.7973	0.8352	0.9151	0.7822	0.8572	0.8357
FgSegNet_v2 [39]	89	0.9396	0.9680	0.9475	0.9143	0.8985	0.8736	0.9247	0.9152	0.9196	0.8179	0.9119
Proposed F3DsCNN	**120**	**0.9712**	**0.9784**	**0.9755**	**0.9721**	**0.9737**	**0.8878**	**0.9718**	0.9432	**0.9576**	**0.9581**	**0.9589**

Table 6.13: Comparative F-measure performance for unseen videos on DAVIS2016 dataset.

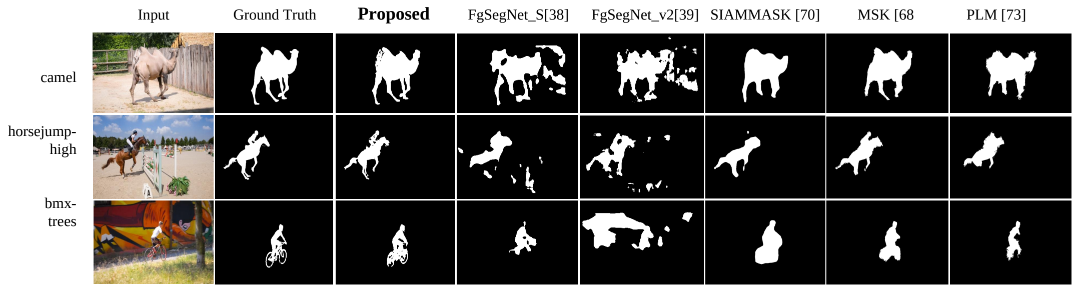
Method	Inference Speed (fps)	F-measure										
		camel	car-roundabout	car-shadow	cows	goat	horsejump-high	kite-surf	bmx-trees	parkour	soapbox	Avg
MSK [68]	0.5	0.7350	**0.9260**	0.9480	0.8120	0.8140	0.8510	0.4380	0.7360	0.8740	**0.8420**	0.7976
CTN [69]	4.5	0.7250	0.7750	0.8670	0.7750	0.7460	0.8660	0.4600	0.4800	**0.8820**	0.7440	0.7320
PLM [73]	9.5	0.6130	0.7140	0.7310	0.7410	0.6940	0.7860	0.4560	0.6840	0.8120	0.6300	0.6861
SIAMMASK [70]	78	0.7480	0.8720	**0.9780**	0.7720	0.7210	0.6880	0.3260	0.6590	0.8290	0.5470	0.7140
FgSegNet_M [38]	69	0.6047	0.4892	0.8704	0.5620	0.4009	0.6199	0.6308	0.5895	0.5190	0.5835	0.5870
FgSegNet_S [38]	82	0.6163	0.5194	0.8940	0.5356	0.4063	0.6273	0.6904	0.6948	0.5345	0.5902	0.6109
FgSegNet_v2 [39]	89	0.6201	0.5120	0.8744	0.5309	0.4509	0.5940	0.6820	0.5498	0.5029	0.6194	0.5936
Proposed F3DsCNN	**120**	**0.8144**	0.8155	0.8456	**0.8162**	**0.8213**	**0.8721**	**0.7020**	**0.8860**	0.7060	0.7271	**0.8006**

6.3.2 Training and evaluation on unseen videos (DAVIS2016)

To evaluate the generalization ability of the proposed model, we also conducted Experiment (B) on unseen videos of DAVIS2016 dataset. In this experiment, 30 videos in DAVIS2016 dataset were used in training, and 10 completely unseen videos were used for testing. Table 6.13 shows the comparison between the proposed model and publicly published deep learning-based methods such as MSK [70], CTN [71], SIAMMASK [72], and PLM [75] from the benchmark DAVIS2016 challenge [108], as well as FgSegNet_S [40], FgSegNet_M [40], and FgSegNet_v2 [41] which were trained and tested in the same setup as our proposed model F3DsCNN.



(a)



(b)

Figure 6.17: Visual comparison results (a) Experiment (A) on seen sample of CD-net2014 dataset, (b) Experiment (B) on unseen samples of DAVIS2016 dataset.

Table 6.13 shows the objective performance. We found that our proposed model offers overwhelmingly faster inference speed at 120 fps. Although the proposed method only offers slightly higher F-measure of 0.8006 than that of MSK [70], its advantage in inference speed at 120 fps is more suitable for mobile and embedded devices. Compared to the remaining existing methods, the proposed method F3DsCNN enhanced the F-measure by 12.6% on average.

In Fig. 6.13 (b), we randomly selected three videos (camel, horsejump-high, and bmx-trees) for comparison illustration. Our proposed model accurately and clearly detects the shapes and details of objects such as the bike and person in ”bmx-trees”, the camel and the horse in ”camel” and ”horsejump-high”, while the other models hardly detect correct shapes of objects or can only detect blurry, noisy or incomplete objects.

6.4 Conclusion

In this chapter, we propose the F3DsCNN model for moving object detection. Our model is designed specifically for environments with limited computing resources and for delay-sensitive tasks. Our model increases detection accuracy by utilizing the spatial-temporal information in the video data via 3D convolution, and also by feature fusion via the two-branch structure. Our model improves the efficiency of the model via 3D separable convolution and 3D inverted residual bottleneck module. Moreover, the two experiments conducted on seen videos and unseen videos demonstrate that our proposed model achieves superior detection accuracy among all compared models with high inference speeds suitable for low-latency vision applications.

CHAPTER 7

Conclusion

7.1 Summary of Major Contributions

Moving object detection (MOD) is an essential step of a video processing pipeline which extracts dynamic foreground content from the video frames, while discarding the non-moving background. It plays an important role in many real-world applications. Deep learning methods have been actively applied in moving object detection and achieved great performance. However, many existing models render superior detection accuracy at the cost of high computational complexity and slow inference speed, which hindered the application on mobile and embedded devices with limited computing resources. We have proposed three deep learning models, 2D separable CNN, 3D separable CNN with "MIMO", and 3D separable CNN in two-branch. These models are real-time models tailored for computation-resource-limited and delay-sensitive applications.

Our key contributions are:

- We propose a new 2D separable CNN for moving object detection. The proposed network adopts 2D separable convolution to reduce computational complexity and model size, the 2D convolution is decomposed into a depthwise convolution and a pointwise convolution. Our work applied it to the moving object detection task for the first time in the literature.
- We propose a new 3D separable CNN for moving object detection. The proposed network adopts 3D convolution to explore spatio-temporal information in the video

data and to improve detection accuracy. To reduce computational complexity and model size, the 3D convolution is decomposed into a depthwise convolution and a pointwise convolution. While existing 3D separable CNN schemes all addressed other problems such as gesture recognition, force prediction, 3D object classification or reconstruction, our work applied it to the moving object detection task for the first time in the literature.

- We propose a multi-input multi-output (MIMO) strategy. While existing networks are single-input single-output, multi-input single output, or two-input two-output, our MIMO network can take multiple input frames and output multiple binary masks using temporal-dimension in each sample. This MIMO embedded in 3D separable CNN can further increase model inference speed significantly and maintain high detection accuracy. To the best of our knowledge, this is the first time in the literature that such kind of MIMO scheme is used in the MOD task.
- We propose a new 3D separable CNN model in two-branch structure with Inverted Residual BottleNeck with 3D Separable Convolution. Branch 1 adopts deep inverted residual bottleneck layers to extract global features, and branch 2 adopts a shallower network to extract lower-level features. We propose to replace the 2D separable convolutions in the inverted residual bottleneck [107] by 3D separable convolutions. The redesigned 3D inverted residual bottleneck first expands the 4D input to a high-dimensional space by a pointwise convolution. Subsequently, a 3D depthwise convolution is performed on each of the channels, and the output is then projected back to the low-dimensional space using another pointwise convolution.
- We demonstrate our proposed 2D separable CNN achieves an inference speed of 149.81 fps, more than twice faster than those of the MobileNet+UNet and the FgSegNet 3-scale, and 1.8 times faster than that of the FgSegNet 1-scale. In terms of

detection accuracy, our proposed model achieves an average F-measure of 0.9718, significantly higher than that of MobileNet+UNet, and only slightly lower than those of the two FgSegNet models.

- We demonstrate that the proposed 3DS_MM offers overwhelmingly high inference speed in frames per second (154 fps) and extremely small model size (1.45 MB), while achieving the best detection accuracy 0.9517 in terms of F-measure, S-measure, E-measure, and MAE among all models in scene dependent evaluation (SDE) setup and achieving the best detection accuracy F-measure 0.7317 among the models with inference speeds exceeding 65 fps in scene independent evaluation (SIE) setup with Davis dataset. Note: the data selection strategy is different from 2D separable CNN experiment, so the accuracy is not comparable.
- We demonstrate that the proposed F3DsCNN achieves a fast inference speed of 120 frames per second, with high detection accuracy F-measure 0.9589 in scene dependent evaluation (SDE) and 0.8006 in scene independent evaluation (SIE) setup with Davis dataset. Compared with "3DS_MM", the inference speed is a bit lower, but the accuracy is higher. It's still suitable for tasks that need to be carried out in a timely manner on a computationally limited platform with high accuracy.

7.2 Suggestions for Future Research

We have proposed 2D separable convolution and 3D separable convolution based schemes for developing the deep learning models in the application of moving object detection. We have demonstrated that such models and in conjunction with other designed algorithms such as MIMO or network structures such as two-branch give an improved performance over traditional based methods and other deep learning based methods.

As we mentioned in Chapter 1, moving object detection would be the beginning step in the pipeline of a bigger computer vision task. It provides important cues for numerous applications in computer vision, for example surveillance tracking or human pose estimation. So in order to make a bigger computer vision task running in real-time, each module of the task should be able to achieve beyond real-time. So it's quite valuable to keep improving the speed of moving object detection module.

The accuracy of the model could be improved. During our initial research work, the accuracy performs well for CDnet 2014 dataset, but the accuracy is not the highest in Davis 2016 dataset. One possible direction to explore is to use transformer-based network. The original transformer network can achieve a decent level of accuracy, but the disadvantage of transformer-based network is the computation complexity is quite high, and the training and inference are too slow. But if we use transformer-based network in conjunction with separable convolution modules, the complexity may be reduced. This could be further explored.

The methods that are used here would also be applicable to other areas such as low-level vision. Low-level vision is the pre-processing step for high-level vision. Based on low-level image processing, low-level vision tasks could be preformed, such as image matching, optical flow computation and motion analysis. In the Appendix, we provide some successful experiments results of applying 2D separable CNN based models in low-level computer vision task called ”demosaicing” which is transferring RAW images to RGB images.

Another area for further research is to extend this work to video coding. Video Coding for Machine (VCM), arising from the emerging MPEG standardization efforts1 [109]. Towards collaborative compression and intelligent analytics, VCM attempts to bridge the gap between feature coding for machine vision and video coding for human vision. A possible next generation of video compression standards is machine vision oriented video compression and hybrid human and machine vision oriented video compression. Some proposals exist in literature for a unified frame work for content understanding and compression. And our research could be as one kind of the content understanding. So our model could be combined into the VCM model to detect moving objects and meanwhile achieving video compression. This area can be explored further.

APPENDIX A

Other Experiments

Here are some experiments results from participating workshop challenges using the separable convolution methods in low-level vision tasks:

1. AIM 2019 Challenge on RAW to RGB Mapping [89]

This challenge is to mapping RAW image to RGB image, simulation of the "demosiacing" step in camera pipeline. We proposed a simple pointwise convolution based network to convert 4 input channels to 3 output channels. The model consists of 7 pointwise convolutional layers followed by non-linear activation functions and contains in total 362, 307 parameters. The inference speed is 0.008s/image on Titan GPU.

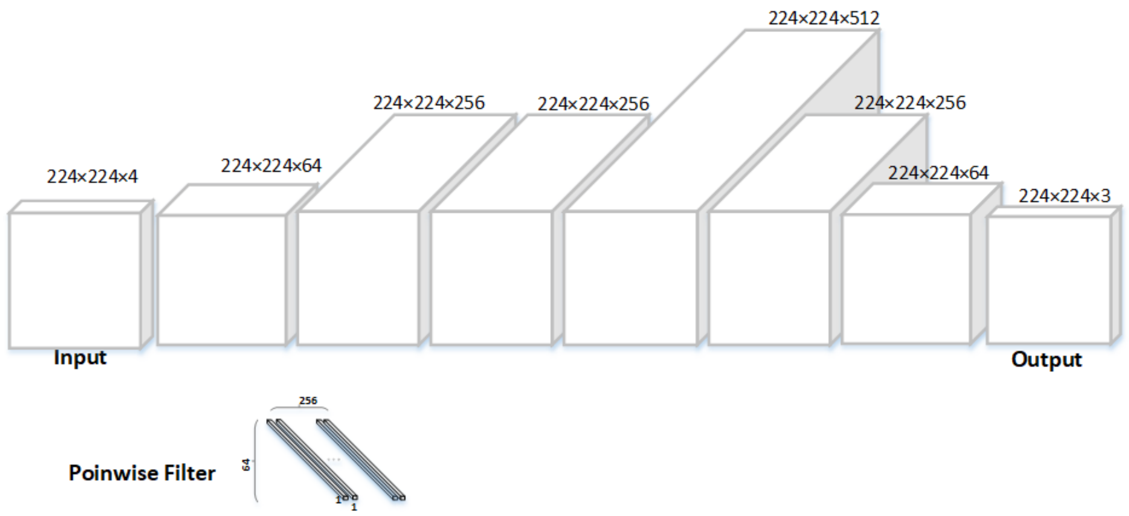


Figure A.1: Proposed pointwise based neural network.

For 224×224 images: model input $224 \times 224 \times 4$ images, model output $224 \times 224 \times 3$; For full resolution images like Fig A.2, the image is divided into patches and do prediction.

Since the training dataset is so big and duplicated, so it’s better to select some valuable data before training. We suggest use color checker colors as centers and choose those data which are around the centers.

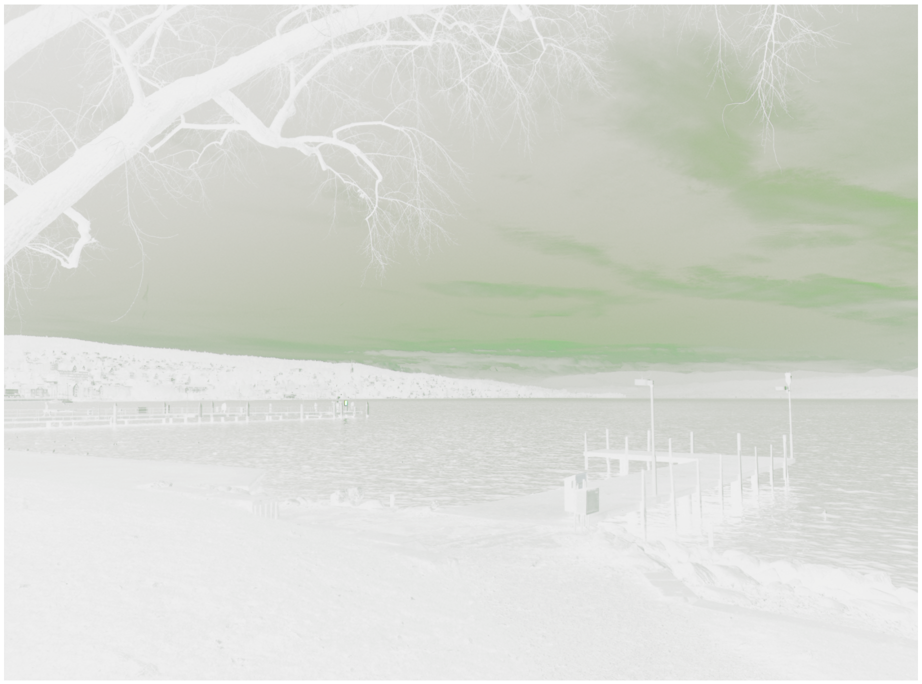


Figure A.2: Input of the model-RAW image.

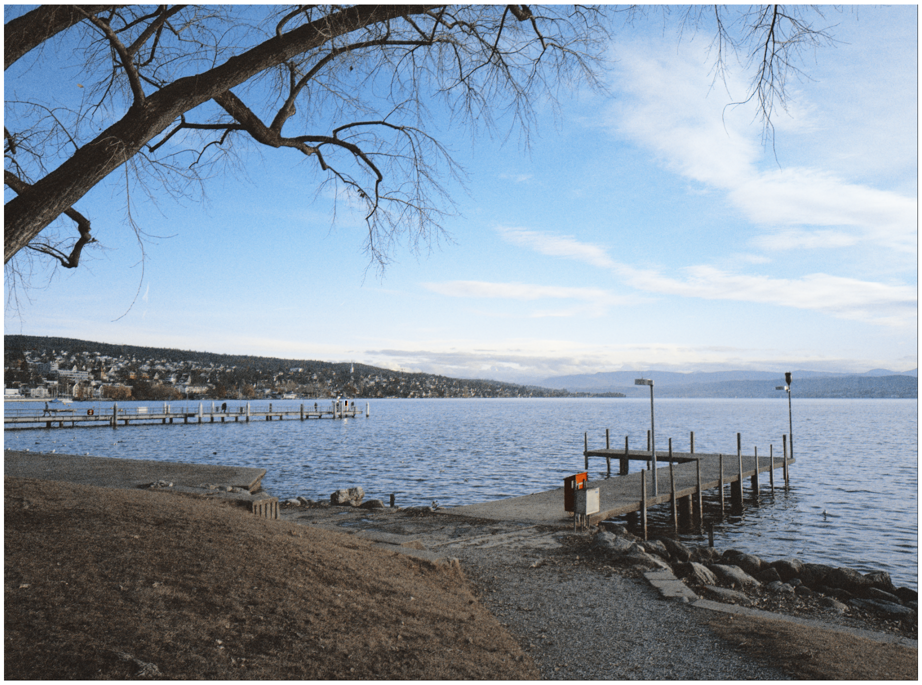


Figure A.3: Output of the model-RGB image.

2. AIM 2020 Challenge on Learned Image Signal [90]

This challenge is also mapping RAW image to RGB image, different from AIM 2019, the input images are with higher resolution and the dataset is more imbalanced.

We proposed a Pixel-Wise Color Distance (PWCD) model illustrated in Fig A.4. The model performed pointwise convolution in the LAB color space instead of the RGB one, and was trained to minimize the CIELAB color difference between the predicted and target images. This PWCD model is the upgraded version from the method in AIM2019 by using LAB space rather than RGB space with point-wise convolution.

To balance the input dataset, we use 24 color patches as centers to select 500 images closest to the centers as training images. Total training images are 24*500*(8/10) = 9600. Input: 448×448 RAW images, output $448 \times 448 \times 3$; Then, transfer all the data to LAB space rather than RGB space. Loss function is used to minimize CIELAB color difference values between predicted and ground truth color. Inference speed is 0.04s/image on 24G

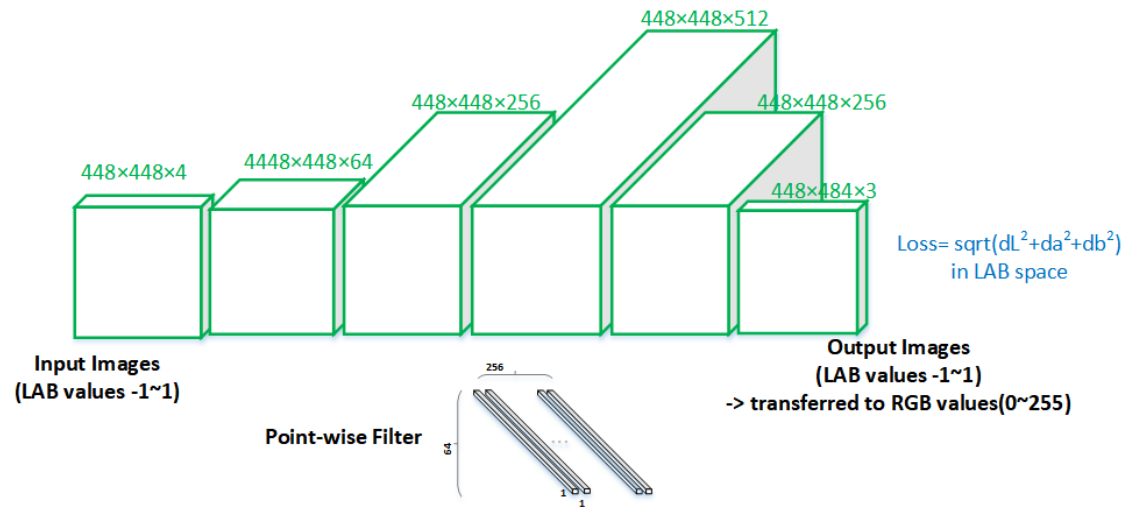


Figure A.4: Output of the model-RGB image.

Titan GPU.

BIBLIOGRAPHY

[1] T. Bouwmans and B. García, "Background Subtraction in Real Applications: Challenges, Current Models and Future Directions," *Comput. Sci. Rev.*, vol. 35, p. 100204, 2020.

[2] S. Brutzer, B. Höferlin, and G. Heidemann, "Evaluation of Background Subtraction Techniques for Video Surveillance," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1937–1944.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, ser. LNCS, vol. 9351. Springer, 2015, pp. 234–241.

[4] A. Basharat, A. Gritai, and M. Shah, "Learning Object Motion Patterns for Anomaly Detection and Improved Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[5] F. Porikli and O. Tuzel, "Human Body Tracking by Adaptive Background Models and Mean-Shift Analysis," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, Mar 2003, pp. 1–9.

[6] S. Zhu and L. Xia, "Human Action Recognition Based on Fusion Features Extraction of Adaptive Background Subtraction and Optical Flow Model," *Mathematical Problems in Engineering*, Aug 2015.

[7] T. Bouwmans, "Recent Advanced Statistical Background Modeling for Foreground

Detection: A Systematic Survey," *Recent Patents on Computer Science*, vol. 4, pp. 147–176, 09 2011.

[8] S. Javed, P. Narayanamurthy, T. Bouwmans, and N. Vaswani, "Robust PCA and Robust Subspace Tracking: A Comparative Evaluation," in *IEEE Statistical Signal Processing Workshop (SSP)*, 2018, pp. 836–840.

[9] S. Bianco, G. Ciocca, and R. Schettini, "How Far Can You Get by Combining Change Detection Algorithms?" in *Image Analysis and Processing (ICIAP)*. Springer, 2017, pp. 96–107.

[10] Y. Zheng and L. Fan, "Moving Object Detection based on Running average Background and Temporal Difference," in *IEEE International Conference on Intelligent Systems and Knowledge Engineering*, 2010, pp. 270–272.

[11] Q. Zhou and J. Aggarwal, "Tracking and Classifying Moving Objects using Single or Multiple Cameras," in *Handbook of Pattern Recognition and Computer Vision*, Jan 2005, pp. 499–524.

[12] S. C. Sen-ching and K. Chandrika, "Robust Techniques for Background Subtraction in Urban Traffic Video," in *Visual Communications and Image Processing 2004*, vol. 5308, 2004, pp. 881–892.

[13] T. S. F. Haines and T. Xiang, "Background Subtraction with Dirichlet Process Mixture Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 670–683, 2014.

[14] S. Bianco, G. Ciocca, and R. Schettini, "Combination of Video Change Detection Algorithms by Genetic Programming," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 914–928, 2017.

[15] S. Jiang and X. Lu, “WeSamBE: A Weight-Sample-based Method for Background Subtraction,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 9, pp. 2105–2115, 2018.

[16] M. Braham, S. Piérard, and M. Van Droogenbroeck, “Semantic Background Subtraction,” in *IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 4552–4556.

[17] A. Elgammal, D. Harwood, and L. Davis, “Non-Parametric Model for Background Subtraction,” in *Proceedings of the 6th European Conference on Computer Vision-Part II*, ser. ECCV ’00, D. Vernon, Ed., pp. 751–767.

[18] P. St-Charles, G. Bilodeau, and R. Bergevin, “A Self-Adjusting Approach to Change Detection Based on Background Word Consensus,” in *IEEE Winter Conference on Applications of Computer Vision*, 2015, pp. 990–997.

[19] O. Barnich and M. Van Droogenbroeck, “ViBe: A Universal Background Subtraction Algorithm for Video Sequences,” *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1709–1724, 2011.

[20] P. St-Charles, G. Bilodeau, and R. Bergevin, “SuBSENSE: A Universal Change Detection Method with Local Adaptive Sensitivity,” *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 359–373, 2015.

[21] G. Bilodeau, J. Jodoin, and N. Saunier, “Change Detection in Feature Space using Local Binary Similarity Patterns,” in *International Conference on Computer and Robot Vision*, 2013, pp. 106–112.

[22] Y. Liu and D. A. Pados, “Compressed-Sensed-Domain L1-PCA Video Surveillance,” *IEEE Transactions on Multimedia*, vol. 18, no. 3, pp. 351–363, 2016.

[23] Y. Liu, Z. Bellay, P. Bradsky, G. Chandler, and B. Craig, "Edge-to-Fog Computing for Color-Assisted Moving Object Detection," in *Big Data: Learning, Analytics, and Applications*, F. Ahmad, Ed., vol. 10989, International Society for Optics and Photonics. SPIE, 2019, pp. 9–17.

[24] Y. Pei, Y. Liu, N. Ling, L. Liu, and Y. Ren, "Class-Specific Neural Network for Video Compressed Sensing," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.

[25] S. E. Ebadi, V. G. Ones, and E. Izquierdo, "Dynamic Tree-Structured Sparse RPCA via Column Subset Selection for Background Modeling and Foreground Detection," in *IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3972–3976.

[26] S. Javed, A. Mahmood, S. Al-Maadeed, T. Bouwmans, and S. K. Jung, "Moving Object Detection in Complex Scene Using Spatiotemporal Structured-Sparse RPCA," *IEEE Transactions on Image Processing*, vol. 28, no. 2, pp. 1007–1022, 2019.

[27] S. Javed, T. Bouwmans, and S. K. Jung, "Improving OR-PCA via Smoothed Spatially-Consistent Low-rank Modeling for Background Subtraction," in *Proceedings of the Symposium on Applied Computing*, 04 2017.

[28] P. Narayanamurthy and N. Vaswani, "A Fast and Memory-Efficient Algorithm for Robust PCA (MEROP)," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4684–4688.

[29] P. Rodríguez and B. Wohlberg, "Translational and Rotational Jitter Invariant Incremental Principal Component Pursuit for Video Background Modeling," in *IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 537–541.

[30] R. Berns and B. Hou, "RIT-DuPont Supra-Threshold Color-Tolerance Individual Color-Difference Pair Dataset," *Color Research and Application*, vol. 35, pp. 274–283, 2009.

[31] B. Hou, "Extending the RIT-DuPont Suprathreshold Data Set: Weighted Individual Discrimination Pair Data and New Chroma Dependency Visual Data," Master's thesis, Rochester Institute of Technology, 2010.

[32] T. Minematsu, A. Shimada, H. Uchiyama, and R. Taniguchi, "Analytics of Deep Neural Network-based Background Subtraction," *Journal of Imaging*, vol. 4, p. 78, 06 2018.

[33] T. Minematsu, A. Shimada, and R. Taniguchi, "Rethinking Background and Foreground in Deep Neural Network-based Background Subtraction," in *IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 3229–3233.

[34] T. Bouwmans, S. Javed, M. Sultana, and S. K. Jung, "Deep Neural Network Concepts for Background Subtraction: A Systematic Review and Comparative Evaluation," *Neural Networks*, vol. 117, pp. 8 – 66, 2019.

[35] M. Braham and M. Van Droogenbroeck, "Deep Background Subtraction with Scene-Specific Convolutional Neural Networks," in *International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2016, pp. 1–4.

[36] Y. Wang, Z. Luo, and P.-M. Jodoin, "Interactive Deep Learning Method for Segmenting Moving Objects," *Pattern Recognition Letters*, vol. 96, pp. 66–75, 2017.

[37] X. Liang, S. Liao, X. Wang, W. Liu, Y. Chen, and S. Z. Li, "Deep Background Subtraction with Guided Learning," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6.

[38] J. Liao, G. Guo, Y. Yan, and H. Wang, *Multiscale Cascaded Scene-Specific Convolutional Neural Networks for Background Subtraction: 19th Pacific-Rim Conference on Multimedia, Hefei, China, September 21-22, 2018, Proceedings, Part I*, 09 2018, pp. 524–533.

[39] P. W. Patil, S. Murala, A. Dhall, and S. Chaudhary, "MsEDNet: Multi-Scale Deep Saliency Learning for Moving Object Detection," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 1670–1675.

[40] L. A. Lim and H. Yalim Keles, "Foreground Segmentation using Convolutional Neural Networks for Multiscale Feature Encoding," *Pattern Recognition Letters*, vol. 112, pp. 256–262, 2018.

[41] L. A. Lim and H. Yalim Keles, "Learning Multi-Scale Features for Foreground Segmentation," *Pattern Analysis and Applications*, vol. 23, no. 3, pp. 1369–1380, Aug 2019.

[42] M. Babaee, D. T. Dinh, and G. Rigoll, "A Deep Convolutional Neural Network for Background Subtraction," *Pattern Recognition*, Sep 2017.

[43] P. W. Patil and S. Murala, "MSFgNet: A Novel Compact End-to-End Deep Network for Moving Object Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 11, pp. 4066–4077, 2019.

[44] Y. Chen, J. Wang, B. Zhu, M. Tang, and H. Lu, "Pixelwise Deep Sequence Learning for Moving Object Detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 9, pp. 2567–2579, 2019.

[45] C. Zhao, T. Cham, X. Ren, J. Cai, and H. Zhu, "Background Subtraction based on

Deep Pixel Distribution Learning," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6.

[46] C. Zhao and A. Basu, "Dynamic Deep Pixel Distribution Learning for Background Subtraction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 11, pp. 4192–4206, 2020.

[47] K. Lim, W. Jang, and C. Kim, "Background Subtraction using Encoder-Decoder Structured Convolutional Neural Network," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6.

[48] S. Choo, W. Seo, D. Jeong, and N. I. Cho, "Multi-Scale Recurrent Encoder-Decoder Network for Dense Temporal Classification," in *24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 103–108.

[49] V. Mondéjar-Guerra, J. Rouco, J. Novo, and M. Ortega, "An End-to-End Deep Learning Approach for Simultaneous Background Modeling and Subtraction," in *British Machine Vision Conference*, 2019, p. 266.

[50] M. O. Tezcan, P. Ishwar, and J. Konrad, "BSUV-Net: A Fully-Convolutional Neural Network for Background Subtraction of Unseen Videos," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 2763–2772.

[51] M. O. Tezcan, P. Ishwar, and J. Konrad, "BSUV-Net 2.0: Spatio-Temporal Data Augmentations for Video-Agnostic Supervised Background Subtraction," *IEEE Access*, vol. 9, pp. 53 849–53 860, 2021.

[52] M. Mandal and S. K. Vipparthi, "Scene Independency Matters: An Empirical Study of scene dependent and scene Independent Evaluation for CNN-Based Change Detection," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2020.

[53] Y. Wang, Z. Yu, and L. Zhu, “Foreground Detection with Deeply Learned Multi-Scale Spatial-Temporal Features,” *Sensors*, vol. 18, p. 4269, 12 2018.

[54] Y. Gao, H. Cai, X. Zhang, L. Lan, and Z. Luo, “Background Subtraction via 3D Convolutional Neural Networks,” in *International Conference on Pattern Recognition (ICPR)*, 2018, pp. 1271–1276.

[55] D. Sakkos, H. Liu, J. Han, and L. Shao, “End-to-End Video Background Subtraction with 3D Convolutional Neural Networks,” *Multimedia Tools and Applications*, vol. 77, pp. 23 023–23 041, 2017.

[56] Z. Hu, T. Turki, N. Phan, and J. T. L. Wang, “A 3D Atrous Convolutional Long Short-Term Memory Network for Background Subtraction,” *IEEE Access*, vol. 6, pp. 43 450–43 459, 2018.

[57] T. Akilan, Q. J. Wu, A. Safaei, J. Huo, and Y. Yang, “A 3D CNN-LSTM-Based Image-to-Image Foreground Segmentation,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 959–971, 2020.

[58] M. Mandal, V. Dhar, A. Mishra, S. K. Vipparthi, and M. Abdel-Mottaleb, “3DCD: Scene Independent End-to-End Spatiotemporal Feature Learning Framework for Change Detection in Unseen Videos,” *IEEE Transactions on Image Processing*, vol. 30, pp. 546–558, 2021.

[59] B. Hou, Y. Liu, and N. Ling, “A Super-Fast Deep Network for Moving Object Detection,” in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Oct 2020, pp. 1–5.

[60] M. C. Bakkay, H. A. Rashwan, H. Salmane, L. Khoudour, D. Puig, and Y. Ruichek, “BScGAN: Deep Background Subtraction with Conditional Generative Adversarial

Networks," in *25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 4018–4022.

[61] W. Zheng and K. Wang, "Background Subtraction Algorithm with Bayesian Generative Adversarial Networks," *Zidonghua Xuebao/Acta Automatica Sinica*, vol. 44, 05 2018.

[62] W. Zheng, K. Wang, and F.-Y. Wang, "A Novel Background Subtraction Algorithm based on Parallel Vision and Bayesian GANs," *Neurocomputing*, vol. 394, pp. 178–200, 2020.

[63] P. Patil and S. Murala, "FgGAN: A Cascaded Unpaired Learning for Background Estimation and Foreground Segmentation," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 1770–1778.

[64] M. Sultana, A. Mahmood, T. Bouwmans, and S. K. Jung, "Dynamic Background Subtraction using Least Square Adversarial Learning," in *IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 3204–3208.

[65] P. W. Patil, A. Dudhane, and S. Murala, "End-to-End Recurrent Generative Adversarial Network for Traffic and Surveillance Applications," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 550–14 562, 2020.

[66] M. Sultana, A. Mahmood, T. Bouwmans and S. K. Jung, "Unsupervised Adversarial Learning for Dynamic Background Modeling," in *International Workshop on Frontiers of Computer Vision*, 04 2020, pp. 248–261.

[67] M. Sultana, A. Mahmood, S. Javed, and S. K. Jung, "Unsupervised Deep Context Prediction for Background Foreground Separation," *Machine Vision and Applications*, vol. 30, pp. 375–395, 2018.

[68] J. H. Giraldo and T. Bouwmans, “GraphBGS: Background Subtraction via Recovery of Graph Signals,” 2020. [Online]. Available: `arXiv: 2001.06404`

[69] J. H. Giraldo and T. Bouwmans, “Semi-Supervised Background Subtraction of Unseen Videos: Minimization of the Total Variation of Graph Signals,” in *IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 3224–3228.

[70] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A. Sorkine-Hornung, “Learning Video Object Segmentation from Static Images,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3491–3500.

[71] W. Jang and C. Kim, “Online Video Object Segmentation via Convolutional Trident Network,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7474–7483.

[72] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, “Fast Online Object Tracking and Segmentation: A Unifying Approach,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1328–1338.

[73] C.-H. Shih and W.-J. Tsai, “Hierarchical Embedding Guided Network for Video Object Segmentation,” in *IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 1124–1128.

[74] X. Lu, W. Wang, J. Shen, Y. Tai, D. J. Crandall, and S. C. H. Hoi, “Learning Video Object Segmentation from Unlabeled Videos,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8957–8967.

[75] J. S. Yoon, F. Rameau, J. Kim, S. Lee, S. Shin, and I. Kweon, “Pixel-Level Matching for Video Object Segmentation using Convolutional Neural Networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 10 2017.

[76] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," in *International Conference on Computer Vision, ICCV 2015*. Institute of Electrical and Electronics Engineers Inc., Feb 2015, pp. 4489–4497.

[77] S. Y. Kim, J. Lim, T. Na, and M. Kim, "3DSRnet: Video Super-Resolution using 3D Convolutional Neural Networks," 2018. [Online]. Available: `arXiv:1812.09079`

[78] A. Torfi, S. M. Iranmanesh, N. Nasrabadi, and J. Dawson, "3D Convolutional Neural Networks for Cross Audio-Visual Matching Recognition," *IEEE Access*, vol. 5, pp. 22 081–22 091, 2017.

[79] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations*, 2015.

[80] Y. Wang, P. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, "CDnet 2014: An Expanded Change Detection Benchmark Dataset," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 393–400.

[81] C. Lin, Y. Hung, R. Feris, and L. He, "Video Instance Segmentation Tracking with a Modified VAE Architecture," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 13 144–13 154.

[82] J. Zhang, Y. Li, F. Chen, Z. Pan, X. Zhou, Y. Li, and S. Jiao, "X-Net: A Binocular Summation Network for Foreground Segmentation," *IEEE Access*, vol. 7, pp. 71 412–71 422, 2019.

[83] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017. [Online]. Available: `arXiv:1704.04861`

[84] L. Sifre and S. Mallat, "Rigid-Motion Scattering for Texture Classification," *Ph.D. thesis*, 2014. [Online]. Available: `aXiv:1403.1687`

[85] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Computing Research Repository (CoRR)*, 2015.

[86] M. Wang, B. Liu, and H. Foroosh, "Factorized Convolutional Neural Networks," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 545–553.

[87] F. Chollet, "Xception: Deep Learning with Depthwise Separable Cnvolutions," 2017.

[88] A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, and K. Keutzer, "Squeezenet: Hardware-Aware Neural Network Design," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[89] A. Ignatov, R. Timofte, S. Ko, S. Kim, K. Uhm, S. Ji, S. Cho, J. Hong, K. Mei, J. Li, J. Zhang, H. Wu, J. Li, R. Huang, M. Haris, G. Shakhnarovich, N. Ukita, Y. Zhao, L. Po, T. Zhang, Z. Liao, X. Shi, Y. Zhang, W. Ou, P. Xian, J. Xiong, C. Zhou, W. Y. Yu, Y. Yubin, B. Hou, B. Park, S. Yu, S. Kim, and J. Jeong, "AIM 2019 Challenge on RAW to RGB Mapping: Methods and Results," in *IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 3584–3590.

[90] A. Ignatov, R. Timofte, Z. Zhang, M. Liu, H. Wang, W. Zuo, J. Zhang, R. Zhang, Z. Peng, S. Ren, L. Dai, X. Liu, C. Li, J. Chen, Y. Ito, B. Vasudeva, P. Deora, U. Pal, Z. Guo, Y. Zhu, T. Liang, C. Li, C. Leng, Z. Pan, B. Li, B.-H. Kim, J. Song, J. C. Ye, J. Baek, M. Zhussip, Y. Koishekenov, H. C. Ye, X. Liu, X. Hu, J. Jiang, J. Gu, K. Li, P. Tan, and B. Hou, "AIM 2020 Challenge on Learned Image Signal Processing Pipeline," 2020. [Online]. Available: `arXiv: 2011.04994`

[91] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4489–4497.

[92] D.-P. Fan, M.-M. Cheng, Y. Liu, T. Li, and A. Borji, "Structure-Measure: A New Way to Evaluate Foreground Maps," in *IEEE International Conference onComputer Vision (ICCV)*, 2017.

[93] D.-P. Fan, C. Gong, Y. Cao, B. Ren, M.-M. Cheng, and A. Borji, "Enhanced-Alignment Measure for Binary Foreground Map Evaluation," in *International Joint Conferences on Artificial Intelligence*, 2018, pp. 698–704.

[94] D.-P. Fan, Z. Lin, Z. Zhang, M. Zhu, and M.-M. Cheng, "Rethinking RGB-D Salient Object Detection: Models, Data Sets, and Large-Scale Benchmarks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, p. 2075–2089, May 2021.

[95] D. P. Kingma and M. Welling, 2019.

[96] B. Hou, Y. Liu, N. Ling, L. Liu, and Y. Ren, "A Fast Lightweight 3D Separable Convolutional Neural Network with Multi-Input Multi-Output for Moving Object Detection," *IEEE Access*, vol. 9, pp. 148 433–148 448, 2021.

[97] Z. Hu, Y. Hu, J. Liu, B. Wu, D. Han, and T. Kurfess, “3D Separable Convolutional Neural Network for Dynamic Hand Gesture Recognition,” *Neurocomputing*, vol. 318, 08 2018.

[98] R. Ye, F. Liu, and L. Zhang, “3D Depthwise Convolution: Reducing Model Parameters in 3D Vision Tasks,” 2018. [Online]. Available: `arXiv:1808.01556`

[99] D. Kim, H. Cho, H. Shin, S.-C. Lim, and W. Hwang, “An Efficient Three-Dimensional Convolutional Neural Network for Inferring Physical Interaction Force from Video,” *Sensors*, vol. 19, p. 3579, 08 2019.

[100] Z. Qiu, T. Yao, and T. Mei, “Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks,” in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5534–5542.

[101] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, “A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 724–732.

[102] B. Hou, Y. Liu, N. Ling, L. Liu, Y. Ren, and M. Hsu, “F3DsCNN: A Fast Two-Branch 3D Separable CNN for Moving Object Detection,” in *Proceedings of the International Conference on Visual Communications and Image Processing (VCIP)*, Dec 2021, pp. 1–5.

[103] R. P. K. Poudel, S. Liwicki, and R. Cipolla, “Fast-SCNN: Fast Semantic Segmentation Network,” in *30th British Machine Vision Conference (BMVC)*, 2019, p. 289.

[104] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, “ICNet for Real-Time Semantic Segmen-

tation on High-Resolution Images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[105] R. P. K. Poudel, U. Bonde, S. Liwicki, and C. Zach, "ContextNet: Exploring Context and Detail for Semantic Segmentation in Real-Time," in *British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, September 3-6, 2018*, 2018, p. 146.

[106] D. Mazzini, "Guided Upsampling Network for Real-Time Semantic Segmentation," in *British Machine Vision Conference (BMVC)*, 2018, p. 117.

[107] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 4510–4520.

[108] "Benchmark Video Object Segmentation on DAVIS2016." [Online]. Available: `https://davischallenge.org/davis2016/soa_compare.html`

[109] L. Duan, J. Liu, W. Yang, T. Huang, and W. Gao, "Video Coding for Machines: A Paradigm of Collaborative Compression and Intelligent Analytics," *IEEE Transactions on Image Processing*, vol. 29, pp. 8680–8695, 2020.

PUBLICATIONS

Journal

- B. Hou, Y. Liu, N. Ling, L. Liu and Y. Ren, ”A Fast Lightweight 3D Separable Convolutional Neural Network with Multi-Input Multi-Output for Moving Object Detection,” IEEE Access, vol. 9, pp. 148 433–148 448, 2021.
- R. Berns and B. Hou, ”RIT-DuPont supra-threshold color-tolerance individual color-difference pair dataset,” Color Research and Application,35:274–283, 2009.

Conference

- B. Hou, Y. Liu, and N. Ling, ”A Super-Fast Deep Network for Moving Object Detection,” in Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Oct, 2020.
- B. Hou, Y. Liu, N. Ling, L. Liu, Y. Ren, and M.K. Hsu, “F3DsCNN: A Fast Two-Branch 3D Separable CNN for Moving Object Detection,” in Proceedings of the International Conference on Visual Communications and Image Processing (VCIP), Dec, 2021.
- A. Ignatov, R. Timofte, S. Ko, S. Kim, K. Uhm, S. Ji, S. Cho, J. Hong,K. Mei, J. Li, J. Zhang, H. Wu, J. Li, R. Huang, M. Haris, G. Shakhnarovich,N. Ukita, Y. Zhao, L. Po, T. Zhang, Z. Liao, X. Shi, Y. Zhang, W. Ou, P. Xian,J. Xiong, C. Zhou, W. Y. Yu, Y. Yubin, B. Hou, B. Park, S. Yu, S. Kim, and J. Jeong, ”AIM 2019 Challenge on RAW to RGB Mapping: Methods and Results,” in IEEE/CVF International Conference on Computer Vision Workshop (ICCVW),pages 3584–3590, Oct, 2019.
- A. Ignatov, R. Timofte, Z. Zhang, M. Liu, H. Wang, W. Zuo, J. Zhang,R. Zhang, Z. Peng, S. Ren, L. Dai, X. Liu, C. Li, J. Chen, Y. Ito, B. Vasudeva,P. Deora, U. Pal,

Z. Guo, Y. Zhu, T. Liang, C. Li, C. Leng, Z. Pan, B. Li, B.-H.Kim, J. Song, J. C. Ye, J. Baek, M. Zhussip, Y. Koishekenov, H. C. Ye, X. Liu,X. Hu, J. Jiang, J. Gu, K. Li, P. Tan, and B. Hou, ”AIM 2020 Challenge on Learned Image Signal Processing Pipeline,” 2020. [Online]. Available: arXiv: 2011.04994.

Master Thesis

- B. Hou. ”Extending the RIT-DuPont Suprathreshold Data Set: Weighted Individual Discrimination Pair Data and New Chroma Dependency Visual Data,” Master’s thesis, Rochester Institute of Technology, 2010.

PATENTS

- “3D Separable Deep Convolutional Neural Network For Moving Object Detection”, B. Hou, Y. Liu, N. Ling, L. Liu, Y. Ren, M.K. Hsu, filed on November 20, 2021, U.S. non-provisional Patent, U.S. Patent Application No. 17/533,012, Ref. No.: 2020KI0066USUS.
- ”F3DsCNN: A Fast Two-Branch 3D Separable CNN for Moving Object Detection”, B. Hou, Y. Liu, N. Ling, L. Liu, Y. Ren, M.K. Hsu, in the application for U.S. provisional Patent.