Distribution Learning for Video Segmentation Applications

by

Chenqiu Zhao

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

With the increase in the number of deep learning networks, many excellent methods have been proposed for video segmentation tasks. However, most of the these methods are for learning pattern information. Not as much work has been done in the area of distribution information, which is also useful for video segmentation. Therefore, this work focuses on learning statistical distributions via neural networks for video segmentation tasks including background subtraction, vessel segmentation and crack detection. In this thesis, we discuss four proposed methods in order to identify an effective way to learn statistical distributions. First, we propose a dynamic deep pixel distribution learning (D-DPDL) method for background subtraction. In D-DPDL, a random permutation of temporal pixels feature is used to force the network to learn the statistical distributions. Compared with previous background subtraction methods based on deep learning networks, the D-DPDL model only requires limited ground-truth frames for training, and it is effective even when training videos and testing videos are captured from different scenes. Then, we improve the D-DPDL method and apply it to vessel segmentation and crack detection, and we found that a wide rather than deep network works better. Finally, we proposed an arithmetic distribution neural network (ADNNet), which is based on arithmetic distribution layers, for learning distributions. The arithmetic distribution layers is the first work to propose network layers based on arithmetic distribution operations, which perform even better than convolutional layers in distribution classification.

Preface

The majority of this thesis has been published in peer-reviewed journals or conferences. The contents of Chapter 3 include the article "Dynamic deep pixel distribution learning for background subtraction," which was published in the IEEE Transactions on Circuits and Systems for Video Technology (2019). Chapter 4 incorporates the papers "Pixel Distribution Learning for Vessel Segmentation under Multiple Scales" and "Multi-scale deep pixel distribution learning for concrete crack detection," which were published in IEEE Engineering in Medicine and Biology Conference (2021) and International Conference on Pattern Recognition (2021), respectively. Finally, Chapter 5 includes our work in "Universal Background Subtraction based on Arithmetic Distribution Neural Network" which has been published by the IEEE Transactions on Image Processing (2022).

Journal Papers:

1. **Zhao, Chenqiu**, Kangkang Hu, and Anup Basu. "Universal background subtraction based on arithmetic distribution neural network." IEEE Transactions on Image Processing, 2022.

2. Ge, Yongxin, Junyin Zhang, Xinyu Ren, **Chenqiu Zhao**, Juan Yang, and Anup Basu. "Deep Variation Transformation Network for Foreground Detection." IEEE Transactions on Circuits and Systems for Video Technology 31, no. 9 (2020): 3544-3558.

3. **Zhao, Chenqiu**, and Anup Basu. "Dynamic deep pixel distribution learning for background subtraction." IEEE Transactions on Circuits and Systems for Video Technology 30, no. 11 (2019): 4192-4206.

Conference Papers:

1. **Zhao, Chenqiu**, and Anup Basu. "Pixel Distribution Learning for Vessel Segmentation under Multiple Scales." In 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), pp. 2717-2721. IEEE, 2021.

2. Wu, Xuanyi, Jianfei Ma, Yu Sun, **Chenqiu Zhao**, and Anup Basu. "Multi-scale deep pixel distribution learning for concrete crack detection." In 2020 25th International Conference on Pattern Recognition (ICPR), pp. 6577-6583. IEEE, 2021.

3. Ma, Yingnan, Guanfang Dong, **Chenqiu Zhao**, Anup Basu, and Zhengjiang Wu. "Background subtraction based on principal motion for a freely moving camera." In International Conference on Smart Multimedia, pp. 67-78. Springer, Cham, 2019.

4. Wu, Xuanyi, Xin Gao, **Chenqiu Zhao**, Jiangzheng Wu, and Anup Basu. "Background Subtraction by Difference Clustering." In International Conference on Smart Multimedia, pp. 45-56. Springer, Cham, 2019.

To the Count

For teaching me everything I need to know about math.

You must never confuse faith that you will prevail in the end—which you can never afford to lose—with the discipline to confront the most brutal facts of your current reality, whatever they might be.

– Admiral James Stockdale

Acknowledgements

I would like to sincerely acknowledge my supervisor, Prof. Anup Basu, for all of his support, kindness and guidance throughout my PhD program, especially for the opportunity to join his lab at the University of Alberta for my degree. Also, I would like to thank my co-adviser, Prof. Irene Cheng, who gave me many suggestions about my candidacy and defense.

I would like to thank the rest of my thesis committee, Prof. Nilanjan Ray, Prof. Adam White, Prof. Davood Rafiei and Prof. Bruce Cockburn for all the great feedback they gave on my research and thesis.

In addition, I thank all the anonymous reviewers of the papers that I have published. Their comments helped me to improve the quality of my research.

Finally, I wish to express my deep gratitude to my parents for their never-ending love and support along the way. From them, I have learned that one person can really make a difference.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Statement

Video segmentation has been increasingly attracting attention from researchers in computer vision during this period of explosive growth in video streaming. Especially with the increasing application of deep learning networks, the pattern information implied in videos is being more effectively extracted and a lot of excellent algorithms have been proposed for video segmentation. However, in contrast to the methods for learning pattern information, the methods for learning statistical distributions do not seem to have been as well investigated in the literature. Statistical distribution is supposed to have great potential for applications in academia and industry. Based on this insight, a few interesting questions have been presented: Is it possible to propose new techniques to effectively and automatically learn the statistical distributions and are these new techniques better than traditional deep learning networks in applications related to distribution analysis? The proposed methods [138], [151], [153], [154] discussed in this thesis are designed to provide answers, which should prove useful in distribution analysis.

Distribution analysis is arguably one of the oldest techniques used in computer vision applications. Tremendous distribution information is implied in videos, which is very useful for a lot of vision applications. For example, the distribution of temporal pixels can be approximated by several Gaussian functions, which can be used to model the background of video frames. The Gaussian mixing model [159] is thus proposed to use distribution information to model the background, which underlies one of the most popular techniques for background subtraction. Unfortunately, despite a comprehensive literature review, we have not found an effective distribution learning technique for video segmentation tasks. Therefore, in this thesis, we focused on learning statistical distributions and applying them to video segmentation applications. In order to automatically and effectively learn the statistical information, it was reasonable to incorporate convolutional neural networks, which have demonstrated excellent learning ability. However, to the best of our knowledge, the conventional convolutional neural network is not a perfect candidate for learning statistical distributions since the computational operations are mainly based on matrix operations, in which all involved entries are considered as feature vectors rather than as statistical distributions. For that reason, the main focus of this thesis is how to propose a network for learning statistical distributions. We proposed several methods to fill the gap that use new types of networks to learn statistical distributions. In particular, the arithmetic distribution neural network discussed in Chapter 5 was specifically devised for learning distributions automatically. In the background subtraction task, the arithmetic distribution neural network performed better than the traditional convolutional neural network.

In total, four of our proposed methods are discussed in this thesis. All are related to learning statistical distributions in video segmentation tasks. Two are proposed for solving the moving objects segmentation problem. One each is applied to vessel segmentation and crack detection. All are proposed for one purpose ot learn statistical distributions for video segmentation. They play different roles in the development of the arithmetic distribution neural network, which is ultimately the method of our choice for distribution learning in this thesis. The dynamic deep pixel distribution learning (D-DPDL) model is first discussed in Chapter 3, in which we attempt to use a convolutional neural network for learning statistical distributions. In the D-DPDL model, the temporal pixels are randomly permutated and input into a convolutional neural network. This forces the network to focus on learning statistical distributions since only statistical information is retained in the input after random permutation. Then, our pixel distribution learning technique is further refined and applied into into vessel segmentation and crack detection, which is introduced in Chapter 4. We devised a wide rather than a deep network for learning statistical distributions, and used a multiple scales strategy to improve accuracy. Finally, in order to figure out why random permutation works for distribution learning, we proposed the arithmetic distribution neural network, which is discussed in Chapter 5. The arithmetic distribution operations are used for the forward pass and back propagation. To the best of our knowledge, this is the first method to propose network layers based on arithmetic distribution operations for learning distributions during background subtraction.

1.2 Scope and Significance

This thesis focused on learning statistical distributions for video segmentation applications. However, since distribution analysis has a wide range of applications, we had to select a particular application evaluating the methods that we proposed. We used moving objects segmentation, also known as background subtraction, as the application of starting point. We did so because the moving objects segmentation problem can be considered as a binary classification of temporal pixels in which many methods were proposed to analyze the statistical distribution. In Chapter 3, we propose a D-DPDL learning model for background subtraction, which is the fundamental problem of computer vision. We further applied this model to vessel segmentation and crack detection, which we introduce in Chapter 4. A new type of network, the arithmetic distribution neural network, is proposed in Chapter 5 for background subtraction again, in order to compare it with the D-DPDL model and the traditional convolutional neural network.

1.3 Challenges

There are two primary challenges in this thesis. The first has to do with the involved video segmentation applications containing background subtraction, vessel segmentation and crack detection. Background subtraction is a fundamental problem in computer vision caused by the diversity and complexity of natural scenes. Vessel segmentation applications require the proposed methods can be trained with limited groundtruth frames. Crack detection algorithms have to deal with noisy points since crack images often included stains.

The second challenge comes mainly from the mathematical theory involved in the arithmetic distribution neural network, which is based on arithmetic distribution operations rather than matrix operations. The arithmetic distribution operations are used to compute the distribution of the arithmetic of two random variables with their particular distributions. It involves probability calculus under an infinite domain, which is quite challenging to a student like me, whose background is not in math.

1.4 Contribution

In order to figure out how to most effectively learn statistical distributions from video segmentation tasks, four works are proposed in this thesis, which are discussed in three different sections.

1. Dynamic Deep Pixel Distribution Learning for Background Subtraction: We propose a novel moving objects segmentation algorithm, named Dynamic Deep Pixel Distribution Learning (D-DPDL). The enties of patches, which is used the input of networks, are randomly permutated to force the network focus on statistical distributions. This work strongly suggests that networks can be used to learn statistical distribution. Among its strengths are that it only requires a limited groundtruth frames for training, and the training videos and testing videos can be different, unlike state-of-the-art methods that are based on traditional convolutional neural networks.

2. Pixel Distribution Learning for vessel segmentation and crack detection: When we saw how useful the D-DPDL method was for learning distributions in moving-objects segmentation, we further applied it to vessel segmentation and crack detection. This showed that our distribution

learning technique can be used for more than moving objects segmentation. In particular, we found that a wider rather than deeper network is more suitable for learning statistical distributions. To improve the accuracy, we captured features from multiple scales and input them into the network for learning distributions.

3. Universal Background Subtraction based on Arithmetic Distribution Neural Network: Although the distribution learning technique used in D-DPDL has demonstrated an ability for learning distribution information, it is still not clear how exactly the random permutation works for learning distribution. Therefore, we further propose a universal background subtraction framework based on the Arithmetic Distribution Neural Network (ADNN) for learning the distributions of temporal pixels. In our ADNN model, we used the arithmetic distribution operations to introduce the arithmetic distribution layers, including the product and sum distribution layers. Compared with the convolution layer, the arithmetic distribution layers worked better in distribution classification tasks with far fewer parameters. In short, the proposed ADNN performed better than the traditional convolutional neural network, and a comprehensive experiment was proposed for validation.

1.5 Organization of the thesis

The rest of the thesis is organized as follows: In Chapter 2, we cover existing research and related works in background subtraction, vessel segmentation and crack detection related to distribution analysis. In Chapter 3, we detail the D-DPDL method for background subtraction. In Chapter 4, we further apply the pixel distribution learning technique to vessel segmentation and crack detection. In Chapter 5, we introduce our optimal solution for learning distribution in videos: the ADNN. Chapter 6 contains the conclusion and the discussion of future work.

Chapter 2

Background and Related Work

In this chapter, we first briefly discussed the correlation between background subtraction methods and distribution learning technique in Section 2.1. A detailed review of existing background subtraction methods is proposed in Section 2.2. Next the methods related to vessel segmentation and crack detection are covered in Section 2.3. Finally, we further discussed the methods of background subtraction related to distribution learning in Section 2.4.

2.1 Distribution Learning and Moving Objects Segmentation

Statistical information includes several features including histograms, probability, and distributions. Because they provide a good representation of the background scene, the statistical distributions play an important role [2], [20], [22], [24], [36], [44], [50]–[52], [79], [80], [91], [103], [121], [129], [145], [159] throughout the development of moving objects segmentation algorithms. Many research papers have utilized statistical distribution models to find the background using temporal pixels. For example, the Gaussian mixture model proposed by Zivkovic et al. [159] is one of the most popular techniques. Lee et al. [79] utilized an adaptive learning rate for each Gaussian function to improve the convergence rate during clustering, and Haines et al. [50] [51] used the Dirichlet processes with Gaussian mixture models to analyze pixel distributions. Recently, Chen et al. [20] [22] used Gaussian mixture models to represent the vertices of spanning trees and Akilan et al. [2] proposed a foreground validation process through probability estimation of multivariate Gaussian model distribution. In addition to the Gaussian distribution, there are several other techniques to describe temporal pixels, such as Laplacian distribution [24], kernel density estimation [103] and artificial neural networks [44].

Unfortunately, artificial models do not have the ability to handle the complex distributions generated from diverse natural scenes which is the reason that learning-based algorithms [41], such as convolutional neural networks, are growing in popularity. However, most existing distribution learning algorithms are related to label distribution learning [41], [45], [59], [60], [64], [69]. They handle insufficient training data based on label ambiguity. In contrast, we intend to learn the statistical distribution by using deep learning networks. Then, these learned distributions will be used for video segmentation.

2.2 Background Subtraction

Traditionally background subtraction is done by modeling the variation of pixel observations over time (e.g. [6], [8], [18], [23], [50], [68], [71], [79], [82], [83], [90], [100], [111], [128], [129], [133], [143], [159]). Among them, one of the most popular techniques for background subtraction [48] is the Gaussian mixture model (GMM) [159], where pixel distribution is approximated by several Gaussian functions. Numerous extensions of GMM have been proposed. For example, Varadarajan et al. [129] approximate the distribution from regions instead of pixels for background subtraction, where the neighboring relationships between pixels are also considered. Similarly, Sriram et al. [121] extend approximation through the use of expectation maximization. Moreover, several sophisticated algorithms utilize the kernel density estimation (e.g. [36], [52], [80], [91], [103]) instead of Gaussians for approximation. Recently, Haines et al. [50], [51] proposed the use of Dirichlet processes with Gaussian mixture models to analyze pixel distribution, while Chen et al. [21], [22] used Gaussian mixture models to represent the vertices of spanning trees.

Besides GMMs, there are also several other recent advances (e.g., [18], [19],

[65], [70], [113], [145]) in background subtraction. Charles et al. [18] utilized temporal binary features and color information for background subtraction. Moreover, inspired by the traditional codebook algorithm [75], they also utilized word dictionaries for background subtraction [19]. Zeng et al. [145] proposed the use of a histogram based on strong uniform fuzzy partitions, where the threshold for background segmentation is set adaptively according to the shape of the histogram. Sajid et al. [113] proposed a universal multi-modal system that merged pixels together to form what the authors called mega-pixels. Javed et al. [67] proposed a robust principal component analysis framework, which incorporates spatial and temporal sparse subspaces. Finally, Huynh et al. [65] subtracted the moving objects from a dynamic background by analyzing the motion of pixels captured from the comparison between current and background frames, while Jiang et al. [70] utilized a weighted-sample method to rapidly adapt to changing scenarios, and Yong et al. [143] proposed online matrix factorization for background subtraction.

However, these hand-crafted models do not work effectively when applied to a wide range of scene categories. For example, GMM-based algorithms suffer from performance degradation in scenes with high complexity [67], where the distributions of pixels are too complex to be described even by several Gaussian components. It inspired us that a new distribution learning technique maybe needed, and in this thesis we mainly focus on how to automatically and effectively learn the distribution information from videos.

2.2.1 Algorithms based on Supervised Machine Learning

Besides traditional algorithms, there are also several background subtraction methods [25], [28], [32], [49], [53], [88], [96], [97], [148], [149], [158] that utilize machine learning, commonly involving support vector machines (SVM) and Bayesian methods. Lin et al. [88] proposed using a probabilistic SVM for background initialization, Cheng et al. [25] accommodated spatial interactions by minimizing a risk function generalization of the one class SVM, and Han et al. [53] utilized density-based features as the input to an SVM classifier. Zhang et al. [148], [149] proposed an imbalance compensation mechanism based on bilayer modeling and Bayesian classification. Culibrk et al. [28] constructed an unsupervised Bayesian classifier for background modeling and subtraction using a neural network architecture. The novel subspace learning method, proposed by Zhou et al. [158], extracted a sequence of regular video bricks. Then, it posed the background modeling problem as pursuing a subspace representation for video bricks while adapting to scene variations.

In addition to the above, there are also related works employing neural networks for background subtraction. Gregorio et al. [49] proposed a weightless neural network for dynamically adapting to background change, while Maddalena et al. [96], [97] proposed having background models be automatically generated through a self-organizing network, which appears to work well in several scenes containing gradual illumination variation and camouflage.

2.2.2 Algorithms based on Deep Learning

As we all know, deep learning has become popular in diverse applications of computer vision, such as image classification [57], [58], [110], image segmentation [5], video captioning [146] and so on. Since background subtraction can be considered as a pixel-wise classification, or image segmentation based on spatial information, it is not surprising that recent background subtraction methods [4], [10], [11], [18], [33], [84], [86], [136], [144] have embraced deep learning. Wang et al. [136] considered background subtraction as scene segmentation, and proposed a fully connected network with cross-entropy loss function to learn the background scenes directly. Similarly, Zeng et al. [144] proposed the multi-scale strategy to generate better results and Lim et al. [86] extracted multi-scale features of background scenes in the middle layers of a network. In addition, Braham et al. [11] employed deep networks to evaluate the difference between current frames and a background image, the latter being computed as the temporal median across multiple frames.

In addition, a more robust background model algorithm was proposed by Babaee et al. [4] for background subtraction, with a network used for subtracting the background from the current image. Liang et al. [84] proposed a guided learning strategy to avoid manual labeling, in which the network is initiated with the foreground mask generated by the SubSENSE algorithm [18]. Zeng et al. [33] combined several background subtraction methods with the convolutional neural network. Although some of these excellent advances have achieved almost perfect results in standard datasets, many frames are needed in the training phase to achieve accurate segmentation of moving objects. However, since every pixel of a frame needs a label of foreground or background in background subtraction, the ground-truth masks of background subtraction are expensive, which limits the applications of these excellent methods.

Therefore, the proposed dynamic deep pixel distribution learning (D-DPDL) method, which is the first work discussed in this thesis, departs from the conventional approach of creating an explicit representation of the background scene. Instead, it focus on the more fundamental concept underlying background subtraction, which is the classification of pixels in a time sequence [6], based on network-learned discriminative features that act directly on the distribution of pixels over time. Compared to existing methods based on deep learning networks, there are two excellent properties of in the proposed D-DPDL model. First, since a large quantity of distribution information can be generated with a limited number of ground-truth frames, the ground-truth data assumed by the proposed approach is much less than previous approaches. Moreover, the proposed D-DPDL model is effective when training videos and testing videos are from different sources, since distributions of temporal pixels have a considerable independence from scenes. As such, the distribution depends on a pixel itself even under the influence of interactions between neighbouring pixels.

2.3 Vessel Segmentation and Crack Detection

Since the proposed dynamic deep pixel distribution learning (D-DPDL) model have demonstrated a few excellent properties in background subtraction, we thus further applied it into vessel segmentation and crack detection applications.

2.3.1 Vessel Segmentation

Medical image processing is an important topic in computer vision [62], [92]. In particular, the segmentation of blood vessels is a challenging problem because of the extreme variations in the morphology of the vessels against the noisy background [35], [74], [116], [118]. Plenty of approaches have been developed into this topic, which includes many works based on the deep learning network. However, for brevity, only a few typical methods related to convolutional neural networks and distribution analysis are discussed here.

For vessel segmentation, several approaches based on the analysis of distributions have been proposed. For example, Hassouna et al. [55] utilized the stochastic modeling to segment cerebrovascular structure from time of flight magnetic resonance angiography. In this approach the histogram of pixel intensities is classified as the vessel or the background. Specifically, the Background class is approximated by the combination of two Gaussian and one Rayleigh distributions, while the Vessel class is approximated by one Gaussian function. Besides this work, Evgin et al. [47] used k-means for rough liver vessel segmentation. They used further iterative refinement steps based on morphological operations to refine the segmentation result. In addition, Oliveira et al. [105] segmented liver vessels in CT images utilizing region-growing. A pixel was incorporated within a region if its intensity fell within a predefined range, which was defined by approximating the image histogram with a Gaussian Mixture Model (GMM) [37]. However, due to the complexity and diversity of different vessel structures, the Gaussian distribution may not be strong enough to handle all these cases. In contrast, we focus on how to make the network learn the distribution information automatically.

In addition, there are also several other methods related to deep learning networks. For example, Dasgupta et al. [29] formulated the segmentation task as a multi-label inference problem, which combined a convolutional neural network with structured prediction. The network architecture was devised as a fully convolutional network to segment the retinal blood vessels from fundus images. In addition, Fu et al. [40] formulated the vessel segmentation problem as a boundary detection task. In particular, a multi-scale and multi-level convolutional neural network was utilized to learn a rich hierarchical representation, and the conditional random field was used to model the long-range interactions between pixels. Similarly, Luo et al. [95] combined the prediction ability of CNNs and the segmentation ability of CRFs. In their approach an end-to-end deep learning model was trained for retinal images with the ability to segment one image during one network forward computation. Moreover, Vega et al. [132] utilized a Lattice neural network with Dendritic processing, which does not require parameters and can automatically construct its structure. Related to this, Wang et al. [134] combined two superior classifiers: Convolutional Neural Networks (CNN) and Random Forests (RF). Where the CNN performs as a trainable hierarchical feature extractor [140] and ensemble RFs work as a trainable classifier. By contrast, we formulate the vessel segmentation as a spatial distribution classification problem. Then the deep pixel distribution learning (DPDL) model is revised to learn the spatial distribution for vessel segmentation.

2.3.2 Crack Detection

Recently, since a large number of roads, buildings of last century have reached their intended lifetime, crack detection becomes an interesting research topic in computer vision. Essentially, crack detection is a classification problem. Therefore, several different features have been used for crack detection, such as Gabor filters [17], wavelet features [157], Histogram of Oriented Gradient (HOG) [73], and Local Binary Pattern (LBP) [61]. These features are exracted from crack images and used as the input of the following classifier consisting of handle-crack model or machine learning methods. Moreover, Wrong et al. [31] proposed to use edge detection techniques in the frequency domain with the utilization of Sobel or gradient based operators used for identifying cracks on walls of buildings. In addition, Zou et al. [160] utilized the local patterns together under a global view, which have considered the photometric and geometric characteristics of the crack images, in order to reduce the noising points in the output and make the detected crack more continuous.

Recently, several works have tried to use deep learning networks [93], [106], [117], [139] for crack detection. For example, Young-Jin et al. [16] proposed to utilize convolutional neural networks for crack detection, which requires a large labelled training images. Xie et al. [139] proposed a bottom-up architecture for hierarchical feature extraction, and then applied several side networks combining VGGNet [117] and a fully convolutional network (FCN) [93] to detect the edges. Moreover, Yang et al. [141] proposed the Feature Pyramid and Hierarchical Boosting Network (FPHBN), which integrates a feature pyramid module and a hierarchical boosting module into HED [139]. The feature pyramid is added after the feature extraction step. It is designed to introduce context information from higher-level to lower-level feature maps via a top-down architecture. Fan et al. [38] incorporate an adaptive threshold strategy into a deep convolutional neural network for crack segmentation.

However, these methods related to deep learning networks usually assumed a large number of images for training, which is quite expensive in real application. In order to handle it, we applied our pixels distribution learning technique for crack detection, which only requires a limited ground-truth frames for training.

2.4 Distribution Learning and Background Subtraction

Unfortunately, it is still not clear what kind of distribution information is learned by the proposed deep pixel distribution learning (DPDL) technique. Therefore, in order to explain it, we further proposed the arithmetic distribution neural network for background subtraction, which also theoretically explain what has been learned by the DPDL model. Before that, we first discussed existing background subtraction methods related to statistical distributions.

2.4.1 Non-Deep-Learning Algorithms

As we mentioned above, throughout the development of background subtraction Algorithms [2], [6], [8], [9], [14], [18], [20], [22]–[24], [36], [42], [44], [50], [51], [65], [67], [68], [70], [71], [75], [79], [80], [82], [83], [90], [91], [100], [103], [111], [121], [123], [133], [143], [145], [150], [159], the distribution of temporal pixels has played an important role since it is a good representation of background information. Lee et al. [79] utilized an adaptive learning rate for each Gaussian function to improve the convergence rate during clustering. Haines et al. [50] [51] used the Dirichlet processes with Gaussian mixture models to analyze pixel distributions. Recently, Chen et al. [20] [22] used Gaussian mixture models to represent the vertices of spanning trees and Akilan et al. [2] proposed a foreground validation process through probability estimation of multivariate Gaussian model distribution. Besides the Gaussian distribution, there are also several other techniques for the description of temporal pixels, such as Laplacian distribution [24], kernel density estimation [103] and artificial neural networks [44].

In addition, several excellent publications [14], [67], [68], [143] have considered the background as a low-rank component of video frames, given the correlation between background scenes of frames over time. For example, Javed et al. [67], [68] utilized robust principal component analysis [14] to separate the background scenes based on the spatial and temporal subspaces. Yong et al. [143] proposed online matrix factorization for background subtraction. Machine-learning techniques have also been utilized for background subtraction [25], [28], [32], [49], [53], [88], [96], [97], [148], [149], [158]. Lin et al. [88] classified the pixels using a probabilistic support vector machine. Similarly, Han et al. [53] used density-based features into a classifier utilizing a support vector machine. Li et al. [25] formulated background subtraction as minimizing a constrained risk function and Culibrk et al. [28] proposed an unsupervised Bayesian classifier using a neural network architecture for background subtraction.

2.4.2 Algorithms based on Deep Learning

Since convolutional neural networks have demonstrated an excellent ability to learn scene information, several approaches [4], [10], [11], [18], [27], [30], [33], [43], [46], [49], [54], [56], [63], [77], [84], [86], [98], [99], [101], [104], [108], [109], [124]–[127], [135], [136], [142], [144], [151], [152], [156] have used deep learning networks to learn the background scenes for subtraction. For example, Wang et al. [136] proposed a fully connected network to learn the background scenes. Zeng et al. [144] utilized a multi-scale strategy to improve the results. Similarly, Lim et al. [86] used a triplet convolutional neural network to extract multi-scale features from background scenes and Yang et al. [142] improved the robustness of their method by using an end-to-end multi-scale Spatio-temporal (MS-ST) method to extract deep features from scenes. Unfortunately, these papers usually assume a large number of ground truth frames for training, which is very expensive in background subtraction applications. In contrast, Babaee et al. [4] proposed a robust model in which a network is used to subtract the background from the current frame, using only 5% of the labeled masks for training. Liang et al. [84] utilized the foreground mask generated by the SubSENSE algorithm [18] rather than manual labeling for training, and Zeng et al. [33] used a convolutional neural network to combine several background subtraction algorithms together. However, since these approaches rely substantially on the scene information, their performance decline considerably when an unseen video is tested for background subtraction. Recently, Mandal et al. [99] incorporated temporal information by using a foreground saliency reinforcement block, and proposed the 3DCD network for unseen videos. Similarly, Tezcan et al. [126], [127] trained a U-net architecture to do subtractions between background images and current frames, and Giraldo [46] solved an optimization problem of graph signals for background subtraction, utilizing a deep learning network for feature extractions. Such algorithms are effective in unseen videos since they use temporal information; however, a large number of ground-truth frames are still needed from other videos for training. In order to handle it, we proposed the arithmetic distribution neural network (ADNN).

The proposed ADNN requires less than 1% of the frames as ground-truth for training, and it is also effective for unseen videos. In addition, given the generality of distribution information, one ADNN can be trained for all seen or unseen videos and the parameters of networks are thus fixed for all testing videos.

Chapter 3

Dynamic Deep Pixel Distribution Learning for Background Subtraction

Abstract

Previous approaches to background subtraction usually approximate the distribution of pixels with artificial models. In this chapter, we focus on automatically learning the distribution, using a novel background subtraction model named Dynamic Deep Pixel Distribution Learning (D-DPDL). In our D-DPDL model, a distribution descriptor named Random Permutation of Temporal Pixels (RPoTP) is dynamically generated as the input to a convolutional neural network for learning the statistical distribution, and a Bayesian refinement model is tailored to handle the random noise introduced by the random permutation. Because the temporal pixels are randomly permutated to guarantee that only statistical information is retained in RPoTP features, the network is forced to learn the pixel distribution. Moreover, since the noise is random, the Bayesian theorem is naturally selected to propose an empirical model as a compensation based on the similarity between pixels. Evaluations using standard benchmark demonstrates the superiority of the proposed approach compared with the state-of-the-art, including traditional methods as well as deep learning methods.

3.1 Introduction

Motion is a powerful cue for image and scene segmentation in the human visual system. The human ability to detect, segment and understand motion is nearly instantaneous and works for diverse complex scenes. While there has been considerable recent progress related to understanding motion, it still appears that we are far from reaching human capabilities. Therefore, as an essential and fundamental research topic related to motion in computer vision, background subtraction has been attracting increasing attention. Especially, with the utilization of deep learning networks, several researches have recently achieved excellent performance. However, such sophisticated algorithms related to deep learning usually require hundreds of labelled frames for training, which is too expensive to be acceptable in real applications, since every pixel has to be labelled. Therefore, a general solution to background subtraction remains a challenging problem given limited labelled frames.

Essentially, background subtraction is a classification of pixels in a sequence of image frames, typically assuming a static camera, wherein every pixel of a frame is classified as foreground or background by comparing its current measurement with historical observations. This is a challenging problem when dealing with diverse and complex scenes, since the pixel measurements take the form of complex distributions. For example, as shown in R1, R2 and R3 of Fig. 5.1, the distributions of the pixel observations in the moving objects, the static and the dynamic backgrounds are completely different, and manually-tailored models have limited ability to cope with these contrasting distributions. To handle this problem, we focus on learning the distribution automatically instead of devising a sophisticated model. A deep learning network is naturally selected to learn the distribution, given its excellent learning ability. Then, a novel background subtraction model named Dynamically Deep Pixel Distribution Learning (D-DPDL) that needs limited groundtruth frames is proposed.

In our D-DPDL model, the distribution of pixels' observations is described by the Random Permutation of Temporal Pixels (RPoTP) feature, in which the

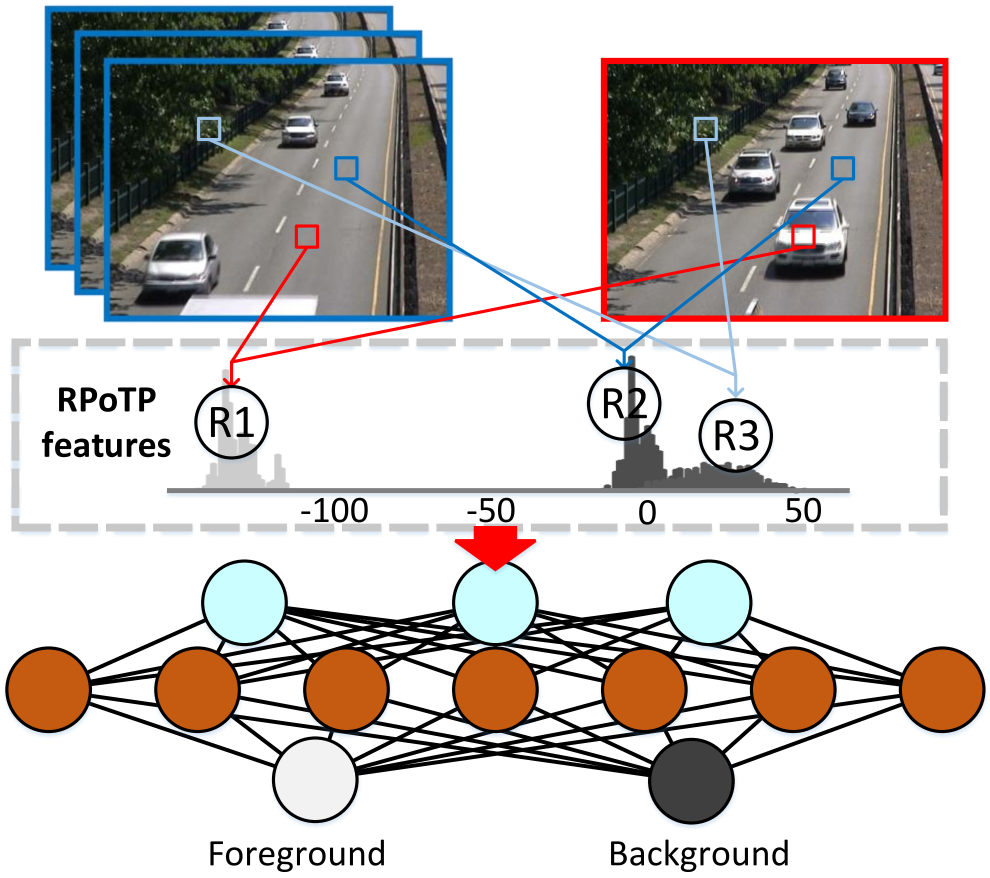


Figure 3.1: An illustration of the deep pixel distribution learning model. The RPoTP features encode the distributions of pixel observations that belong to moving objects R1, static background R2 and dynamic background R3 respectively. The RPoTP features from all pixels are then fed into a convolutional neural network to learn a classifier for background subtraction.

pixel observations are randomly permutated to guarantee that only the statistical information is retained. The RPoTP features are dynamically generated as the input of the convolutional neural network (CNN) for every training epoch. This indirectly forces the network to rely solely on the statistics of the pixel intensity distribution. However, the utilization of random permutations unavoidably introduces random noise, and a Bayesian refinement model based on similarity between pixels is proposed to compensate for the random noise and improve the accuracy of the proposed approach. Since each RPoTP feature only depends on a single pixel, a large number of features are obtained for training, even with limited groundtruth frames. Moreover, the proposed approach is valid even under the condition that training frames and testing frames are obtained from different scenes, because the distribution information is a general feature regardless of the scene. The main contributions of this work are:

1. We improve the procedure of distribution learning, in which the Random Permutation of Temporal Pixels (RPoTP) features are dynamically generated during the training to prevent the network from over fitting the pattern implied in random permutations.

2. We propose a Bayesian refinement model to compensate for the random noise generated from the utilization of random permutations in the extraction of the RPoTP features for training the network. The accuracy of the proposed approach is improved after compensating with the Bayesian refinement model, which is based on the assumption that similar neighboring pixels share similar labels of foreground or background.

3. Comprehensive experiments are proposed for evaluating the proposed approach, including *a)* the validation of the proposed approach under the condition that training videos and testing videos are different, as shown in Section 3.3.1, *b)* the effectiveness of the proposed Bayesian refinement model and strategy of dynamical RPoTP feature generation is shown in Section 3.3.2, *c)* a comprehensive comparison between the proposed approach and state-of-the-art methods including traditional and deep learning approaches is shown in Section 3.3.3.

3.2 Dynamic Deep Pixel Distribution Learning

The details of our Dynamic Deep Pixel Distribution Learning (D-DPDL) model are explained in this section. As shown in Fig. 3.4, the proposed approach is divided into two parts: "Pixel Distribution Learning," in which

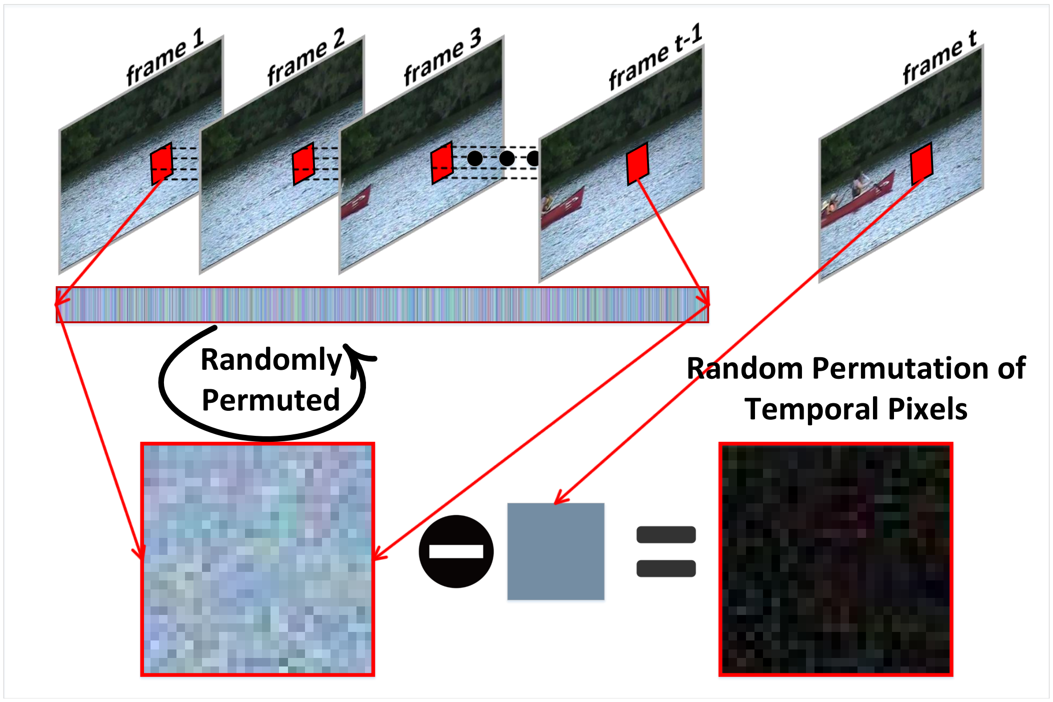


Figure 3.2: The extraction of the Random Permutation of Temporal (RPoTP) pixels feature in a particular pixel. The historical observations of pixels are randomly permutated and reshaped into a rectangle. The RPoTP feature is then captured from the difference between the pixels in the rectangle and the current pixel.

Table 3.1: Details of our network architecture, which consists of 4 convolutional layers, 3 batch normalization, 2 max pooling and a softmax operator.

Type	Filters	Layer size	Data size
The RPoTP feature			$25 \times 25 \times 3$
Convolution	64	$7 \times 7 \times 3$	$19 \times 19 \times 64$
Batch Normalization			$19 \times 19 \times 64$
Max pooling		4×4	$16 \times 16 \times 64$
Convolution	128	$7 \times 7 \times 64$	$10 \times 10 \times 128$
Batch Normalization			$10 \times 10 \times 128$
Max pooling		4×4	$7 \times 7 \times 128$
Convolution	1024	$7 \times 7 \times 128$	$1 \times 1 \times 1024$
Batch Normalization			$1 \times 1 \times 1024$
Rectified linear unit			$1 \times 1 \times 1024$
Convolution	255	$1 \times 1 \times 1024$	$1 \times 1 \times 255$
Softmax			

the RPoTP features are dynamically generated as the input of a convolutional neural network for learning the distribution of pixels (Section 3.2.1), and the

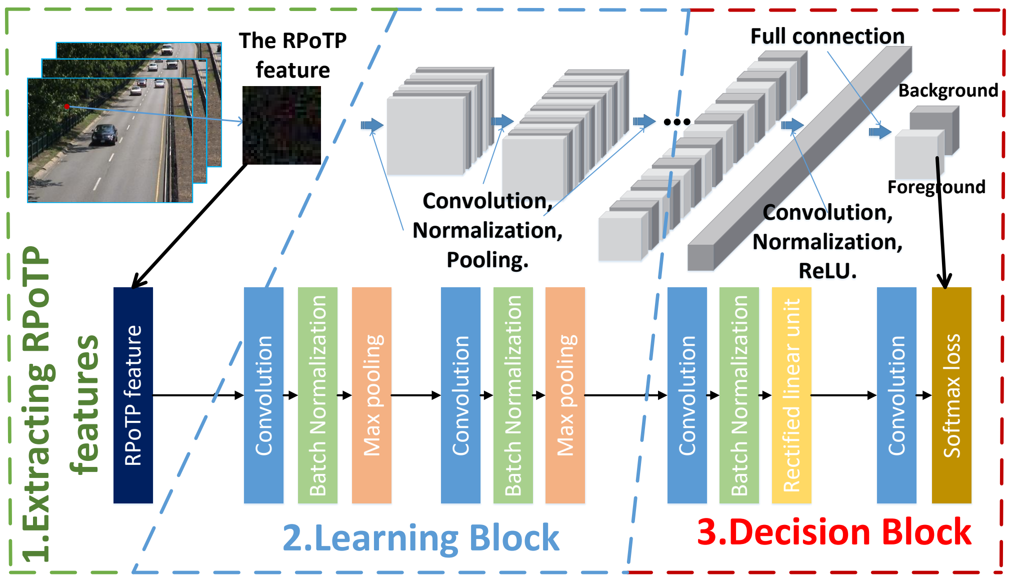


Figure 3.3: Illustration of the deep pixel distribution learning model. The RPoTP features from all pixels are fed into the convolutional neural network to learn a classifier for background subtraction.

"Bayesian Refinement Model," which is proposed to compensate for the random noise (Section 3.2.2).

3.2.1 Deep Pixel Distribution Learning

For the completeness, we first introduce our previous Deep Pixel Distribution Learning (DPDL) [152] model. Then, the dynamic training strategy, which is one of the main differences between D-DPDL model and our previous DPDL, is discussed. then the improvement of DPDL model proposed in this work is discussed.

In videos obtained from natural scenes, multiple factors contribute to the variation of observations in a particular pixel, such as illumination change, dynamic background and moving objects. More importantly, the distribution of observations is different depending on the causes of the variation. For example, variations resulting from illumination changes typically follow a smooth unimodal distribution, since the position of the sun changes slowly during a time sequence. Conversely, for pixels in a dynamic background, the variation often takes the form of a multi-peak distribution, due to the pattern of cyclical repetitions in background pixels, such as leaves fluttering. However, when the variation is caused by moving objects, the observations are typically outliers compared to the distribution formed by the observations of pixels in the background. Therefore, the main purpose of the proposed approach is learning the distribution of pixels in different scenarios to classify these pixels into foreground or background. In particular, the Random Permutation of Temporal Pixels (RPoTP) feature is proposed to discriminate among distributions. The process of extracting the RPoTP features is shown in Fig. 3.2. An image sequence is denoted as $\{I_t | t \in [1,\ T]\}$ where T is the number of frames, captured with a static camera. The temporal observations for a particular pixel are obtained as a vector and reshaped into a square matrix, in which the observations are randomly permutated by a random permutation $\mathcal{X} = \{a_i | i \in [1,\ N]\}$ where N is the length of the permutation. The square matrix includes a fair sample set of historical observations, and the random permutation guarantees that only statistical information remains in the matrix. Finally, the RPoTP feature is extracted by subtracting the current pixel value from the sample matrix, in order to explicitly describe the difference from historical observations, which is mathematically shown as:

$$RPoTP^{r}_{x,y}(m,n) = I_t(x,y) - I_{\mathcal{X}(m \cdot r+n)}(x,y) \tag{3.1}$$

where $RPoTP^{r}_{x,y}$ denotes the RPoTP feature extracted from the pixel located at (x,y), r is the user parameter to control the size of the RPoTP matrix, m,n are the indices of an entry in a matrix. During all the experiments, the radius r of the RPoTP feature is set to 25.

Essentially, background subtraction is a classification of pixels over time. Pixels are classified as foreground or background based on the comparison with their historical observations. The RPoTP features are actually the statistical representation of the comparison, which is used as the input to a convolutional neural network for classification. Therefore, the architecture of our network is devised for classification and the whole classification procedure is shown in Fig. 3.3, which can be divided into three parts, features extraction, learning block and decision block. First, the RPoTP feature is extracted for each pixel and used as the input of the learning block. The learning block consists of several stacks of convolutional, normalization and pooling layers. The decision block is a set of convolutional, normalization and ReLU layers, followed by a fully connected layer that implements the softmax operator. Mathematically, it is shown as follows:

$$\ell_{x,y} = D(\mathcal{L}^{\theta}(RPoTP^{r}_{x,y})), \tag{3.2}$$

where $\ell_{x,y}$ is a binary label of the pixel at location (x, y) identifying it as foreground or background, $\mathcal{L}$ is the learning block and $\mathcal{D}$ is the decision block. θ denotes the parameters of the learning block. The learning block $\mathcal{L}$ consists of several stacks of convolutional, normalization and pooling layers with θ determining the number of these layers. The decision block $\mathcal{D}$ is a set of convolutional, normalization and ReLU layers, followed by a fully connected layer that implements the softmax operator. During training, the logistic loss on the final output node is minimized.

Usually, the training instances of networks are pre-generated before training, it is possible that the network overfits to the pattern implied in random permutations rather than learns the statistical information included in RPoTP features. In order to address this issue, a dynamic training strategy is proposed as a compensation, in which the entries of RPoTP features are randomly re-permutated by new permutations for every training epoch. This strategy effectively prevents our network from overfitting, and improves the accuracy of the proposed approach in complex scenes. Section 3.3.1 demonstrates the effectiveness of the dynamical training strategy by comparing our previous DPDL and the proposed D-DPDL.

In addition, there are also several minor modifications in the network architecture in the proposed D-DPDL model compared with our previous DPDL model. First, we increased the width of network and decreased the depth inspired by [89] which suggests wider networks tend to learn pixel distribution features. Besides, the sizes of convolutional kernels are increased as well, to allow more temporal pixels to be input into the network for learning distributions. Moreover, the stride of the pooling layer is set to 1 to avoid missing useful information during the down-sampling procedure. The comprehensive details of the network architecture are shown in Table 5.1.

3.2.2 Bayesian Refinement Model

To the best of our knowledge, pixels are not independent but related to their neighborhoods, especially similar ones. It is common to accept that similar pixels should share similar labels of foreground or background. Such similarity is very useful against random noise. The Bayesian refinement model based on the similarity among pixels is thus proposed to compensate for random noise generated during distribution learning.

The Bayesian refinement model is devised as an iterative procedure to infer the labels of pixels according to their neighborhoods, with the Bayesian theorem as the kernel of inference. As shown in the right part of Fig. 3.4, the foreground mask produced by the pixel distribution learning, which is discussed in Section 3.2.1, is iteratively refined based on the Bayesian theorem. We explain the process of refinement in a pixel during one iteration, but the procedure is identical for every pixel during all iterations. Let us denote the features captured from the center pixel and their neighborhoods as v and $\mathcal{G}$ repectively. The procedure of inferring the label of the center pixel v is as follows:

$$P(a_i|v) = \frac{P(v|\mathcal{G})P(\mathcal{G}|a_i)P(a_i)}{P(v)}, \tag{3.3}$$

where $a_i \in \{0, 1\}$ denotes the labels of the pixels, which is either background represented by 0 or foreground represented by 1. $P(v)$ and $P(a_i)$ are the marginal probabilities corresponding to the feature and the label respectively. There is no need to consider $P(v)$ in the Bayesian inference procedure, since it is a constant. $P(v|\mathcal{G})$ and $P(\mathcal{G}|a_i)$ are the conditional probabilities, in which $P(\mathcal{G}|a_i)$ describes the probability of pixels with the label a_i belonging to $\mathcal{G}$, and $P(v|\mathcal{G})$ denotes the proximity between the center pixel and its neighborhoods. Next, the details of computing these probabilities are discussed.

The marginal probability $P(a_i)$ in Eqn. 3.3 represents the probability of label a_i in $\mathcal{G}$ which is the set of pixels in a rectangular region around the center

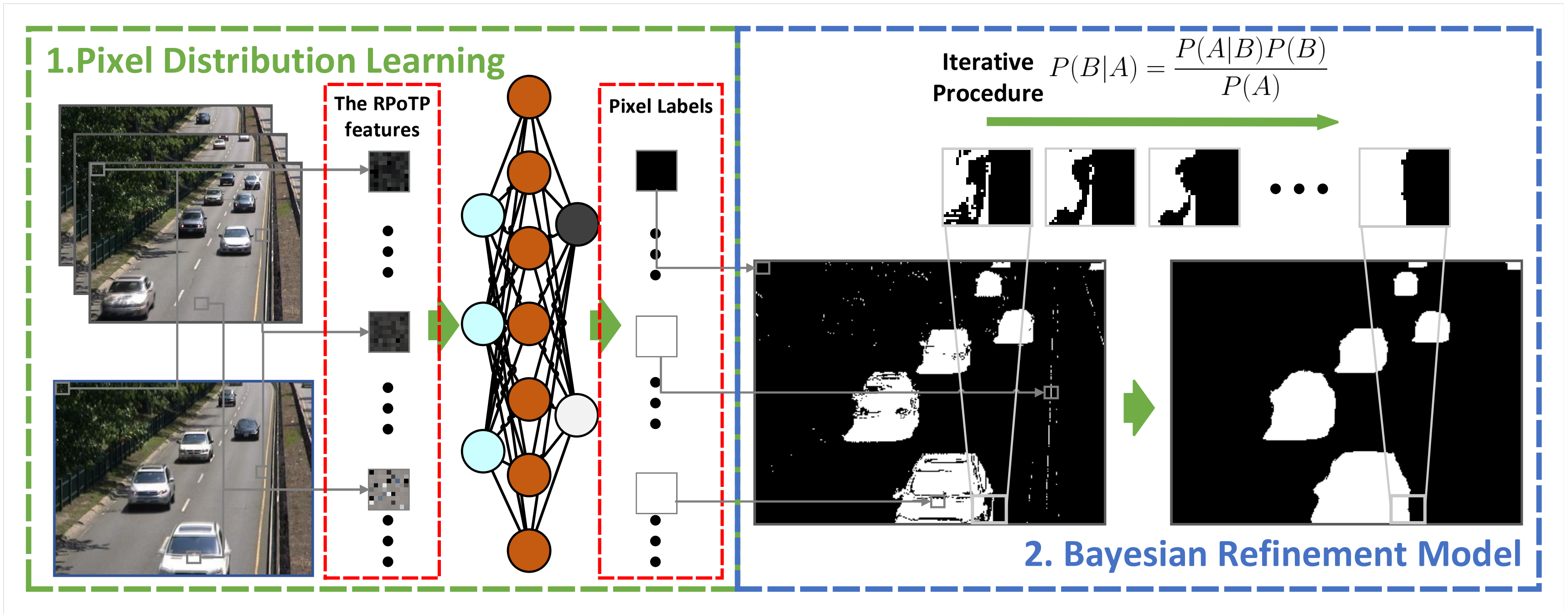

1.Pixel Distribution Learning
The RPoTP features
Pixel Labels
Iterative Procedure $P(B|A) = \frac{P(A|B)P(B)}{P(A)}$
2. Bayesian Refinement Model

Figure 3.4: The pipeline of the proposed approach, which is divided into two parts. The distribution included in the Random Permutation of Temporal Pixels (RPoTP) feature is learned by the convolutional neural network for background subtraction. The Bayesian refinement model is then proposed to compensate for the random noise introduced by random permutation.

pixel. During experiments, the size of the rectangular region is controlled by a user input radius R. Therefore, the statistics of pixels' labels in the region can be used to compute $P(a_i)$ as:

$$P(a_i) = \frac{1}{(2R+1)^2} \sum_{x'=x-R}^{x+R} \sum_{y'=y-R}^{y+R} |\mathcal{B}(x', y') \cap a_i|, \tag{3.4}$$

where $(2R+1)^2$ is the total number of pixels in the rectangular region, (x, y) represents the position of the center pixel and $\mathcal{B}(x', y')$ denotes the foreground mask as well as the label matrix corresponding to the pixels of the current frame, in which x', y' are the indices. A high value of $P(a_i)$ implies higher likelihood of the label a_i appearing in the region, and there is a high possibility that the label of the center pixel is a_i due to the correlation with their neighborhoods. In this condition, the label of the center pixel is correctly labelled as a_i by the marginal probability even though random noise is generated. The marginal probability $P(a_i)$ thus has the ability to isolate the influence of noise, and it is one of the main reasons motivating us to devise a statistical refinement model based on the Bayesian theorem.

In contrast to the marginal probability $P(a_i)$, the conditional probabilities $P(v|\mathcal{G})P(\mathcal{G}|a_i)$ tend to protect the border between the foreground and background, since it is based on the proximity among pixels. In particular, $\mathcal{G}$ represents the set of neighboring pixels, v is the feature of the center pixel and a_i denotes the label. Let us denote the pixels in $\mathcal{G}$ as $g_j \in \mathcal{G}$, and the procedure of capturing the conditional probability can be distributed by g_j as $\sum P(v|g_j)P(g_j|a_i)$. Due to the fact that the foreground and background are exclusive to each other, it is fair to assume that there is no correlation between pixels with different labels, and $P(g_j|a_i)$ becomes 0 corresponding to g_j whose label is not a_i. Based on this assumption, we divide $\mathcal{G}$ into two subsets G_i and $(G_i)^c$ according to the labels of pixels, where G_i is the set of pixels with the label a_i and $(G_i)^c$ is the complement. Then, the conditional probability

$P(v|\mathcal{G})P(\mathcal{G}|a_i)$ can be computed as:

$$\begin{aligned} P(v|\mathcal{G})P(\mathcal{G}|a_i) &= \sum_{g_j \in \mathcal{G}} P(v|g_j)P(g_j|a_i) \\ &= \sum_{g_j \in G_i} P(v|g_j)P(g_j|a_i) + \sum_{g_j \in (G_i)^c} P(v|g_j)P(g_j|a_i) \\ &= \sum_{g_j \in G_i} P(v|g_j)P(g_j|a_i) \end{aligned} \tag{3.5}$$

$$\because \quad P(g_j|a_i) = 0 \quad \forall g_j \in (G_i)^c.$$

In particular, $P(g_j|a_i)$ represents the probability of pixels in $\mathcal{G}$ under the condition that a_i is the label of pixels. It can be computed by a statistical procedure, which is:

$$P(g_j|a_i) = \frac{1}{||G_i||_0} = \left(\sum_{x'=x-R}^{x+R} \sum_{y'=y-R}^{y+R} |\mathcal{B}(x', y') \cap a_i| \right)^{-1}, \tag{3.6}$$

where $\mathcal{B}(x', y')$ is the label matrix introduced in Eqn. 3.4. $||G_i||_0$ is the l_0 norm of G_i, which is also the number of pixels in G_i. $P(v|g_j)$ represents the proximity between the center pixel v and its neighborhood g_j. In this work, the proximity is approximated by a distance function $\mathcal{L}()$, which is a measure of distance between v and g_j, and an activation function $\varphi()$, which convert the distance to proximity. Mathematically, it is shown as:

$$P(v|g_j) \sim \varphi(\mathcal{L}(v, g_j)). \tag{3.7}$$

Then, with the help of Eqn. 3.6 and Eqn. 3.7, the conditional probability $P(v|\mathcal{G})P(\mathcal{G}|a_i)$ in Eqn. 3.5 can be expressed as:

$$\begin{aligned} P(v|\mathcal{G})P(\mathcal{G}|a_i) &= \sum_{g_j \in G_i} P(v|g_j)P(g_j|a_i) \\ &\sim \frac{1}{||G_i||_0} \sum_{g_j \in G_i} \varphi(\mathcal{L}(v, g_j)) \end{aligned} \tag{3.8}$$

With several attempts with different activation functions, the following formula of φ achieves the best performance with acceptable complexity of computation:

$$\varphi(d) = \frac{1}{d}. \tag{3.9}$$

Moreover, inspired by the extraction of super-pixels [1], the distance function $\mathcal{L}(v, c_i)$ is devised as an Euclidean distance of the vectors consisting of five entries $\{l, a, b, x, y\}$, where (l, a, b) is the LAB color space of pixels and (x, y) is the spatial location. Mathematically, it is shown as follows:

$$\mathcal{L}(v, c) = \sqrt{\left(\frac{d_c}{\mu_c}\right)^2 + \left(\frac{d_s}{\mu_s}\right)^2} \tag{3.10}$$

where d_c and d_s denote the Euclidean distance of the LAB color and the spatial location between v and g_j. μ_c and μ_s are the means of the color and spatial distance, which are used to combine these two components. Mathematically, they are computed as:

$$\mu_c = \frac{1}{||\mathcal{G}||_0} \sum_{g_j \in \mathcal{G}} d_c(g_j, v), \ \mu_s = \frac{1}{||\mathcal{G}||_0} \sum_{g_j \in \mathcal{G}} d_s(g_j, v) \tag{3.11}$$

Now, we have the conditional probability, and the marginal probability. The label with the highest probability is used as the label of the center pixel located at (x, y). Therefore, our Bayesian refinement model is described as:

$$\begin{aligned} \mathcal{M}(x, y) = \mathcal{F}(\mathcal{U}, \mathcal{B}) &= \underset{a_i}{\operatorname{argmax}} \, P(a_i | \mathcal{U}(x, y)) \\ &= \underset{a_i}{\operatorname{argmax}} \frac{P(a_i)}{||G_i||_0} \sum_{g_j \in G_i} \varphi(\mathcal{L}(\mathcal{U}(x, y), g_j)) \\ &= \underset{a_i}{\operatorname{argmax}} \sum_{g_j \in G_i} \varphi(\mathcal{L}(\mathcal{U}(x, y), g_j)) \\ G_i &= \{\mathcal{U}(x', y') | \mathcal{B}(x', y') \cap a_i = 1\} \end{aligned} \tag{3.12}$$

$$\begin{aligned} \because \quad & \frac{P(a_i)}{||G_i||_0} = (2R + 1)^2 \\ \& \quad & \underset{x}{\operatorname{argmax}} \, Nx = \underset{x}{\operatorname{argmax}} \, x \quad \forall N > 0 \end{aligned}$$

where $\mathcal{M}(x, y)$ is the output of the refinement model in which $x' \in [x - R \; x + R]$ and $y' \in [y - R \; y + R]$ are the indices of pixels belonging to G_i and (x, y) is the location of the center pixel, $\mathcal{B}(x', y')$ is the foreground mask generated by distribution learning and $\mathcal{U}(x, y) = v$ denotes the feature captured from the pixel located as (x, y). $\mathcal{L}$ and φ are the distance function and the activation function discussed in Eqn. 3.10 and Eqn. 3.9.

The noise included in the input of the refinement model influences the output. Thus, our Bayesian refinement model cannot handle stubborn noise in one iteration. Fortunately, the output of the Bayesian refinement model is empirically better than the input, and can be used in the input again to generate better results iteratively, as shown in the right part of Fig. 3.4. Therefore, our Bayesian refinement model is devised as an iterative procedure as:

$$\mathcal{M}_n(x, y) = \mathcal{F}(\mathcal{U}(x, y), \mathcal{M}_{n-1}(x, y)), \tag{3.13}$$

where $\mathcal{M}_n(x, y)$ is the foreground mask in iteration n, $\mathcal{F}$ is the Bayesian refinement model and $\mathcal{U}$ is the set of features captured from pixels. Considering performance and computation cost, the number of iteration is set to 3 and the radius R is set to 4 during all experiments. In particular, With a larger iteration number, the performance of the proposed approach is supposed to be better, but the computational cost will increase. Moreover, the robustness of the refinement model is improved when R becomes larger, sicne more pixels are considered during refinement. However, a larger R increase the computational cost as well.

3.3 Experiments

In this section, comprehensive experiments for evaluating the proposed approach are presented. In Section 3.3.1, the transferability of our D-DPDL model is discussed, in which the training videos and testing videos are different. The efficiency of our dynamic training strategy and Bayesian refinement model, which are the two main differences compared with our previous work [152], is demonstrated in Section 3.3.2. Then, comprehensive comparison between the proposed approach and several state-of-the-art methods including deep learning methods is presented in Section 3.3.3. Finally, a discussion of the computational cost of the proposed approach is proposed in Section 3.3.4.

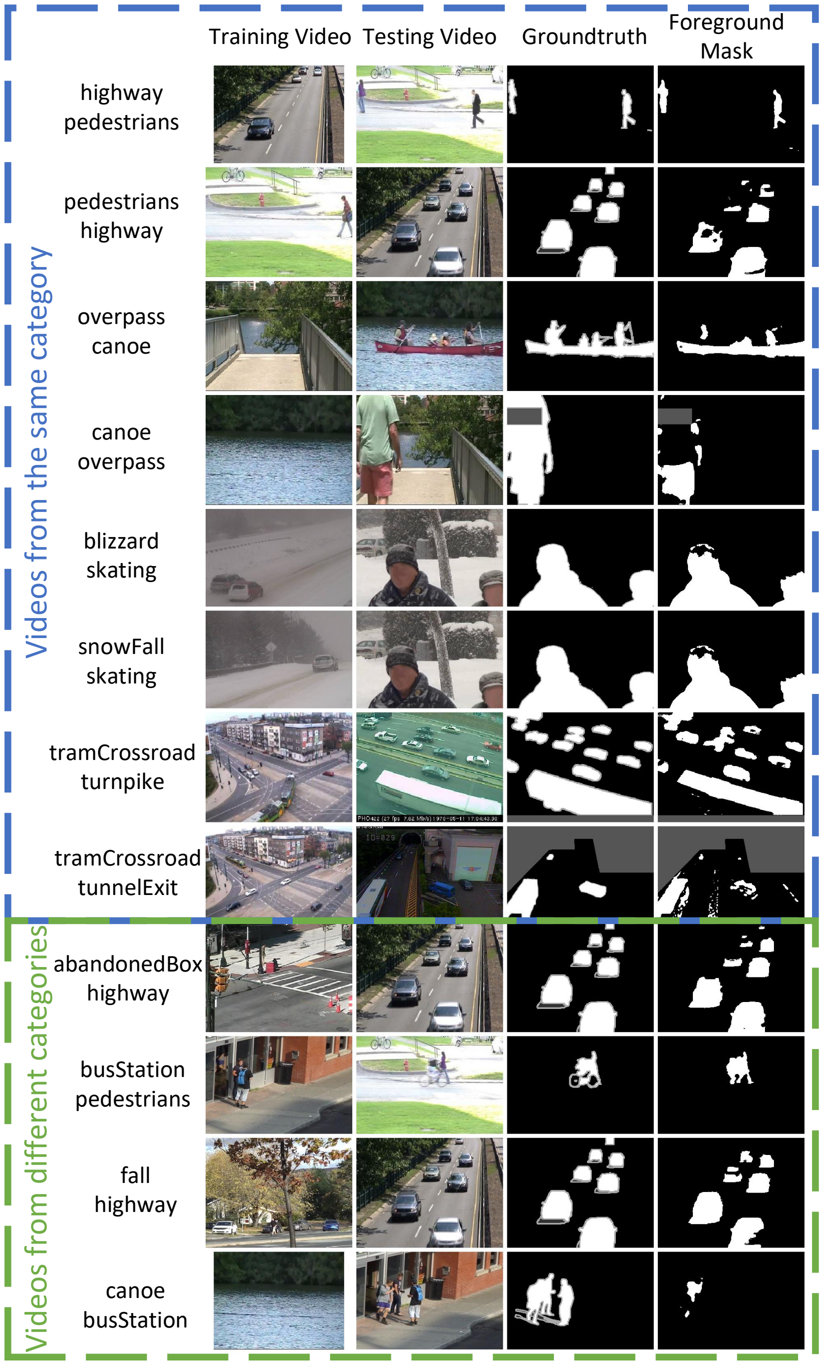


Figure 3.5: The qualitative evaluation of the transferability of the proposed approach in several videos of CDnet2014 [135] dataset. In particular, the training frames and testing frames are obtained from different videos.

Table 3.2: Quantitative transferability evaluation of the proposed approach using Re, Pr and Fm metrics, in which the training and testing videos are different.

Training Sets		Testing Sets		Performance
Video	Category	Video	Category	Re Pr Fm
highway	baseline	pedestrians	baseline	(0.9723, 0.8138, 0.8860)
pedestrians	baseline	highway	baseline	(0.6132, 0.9998, 0.7602)
overpass	Dyn. Bg.	canoe	Dyn. Bg.	(0.8231, 0.9948, 0.9008)
canoe	Dyn. Bg.	overpass	Dyn. Bg.	(0.3709, 0.9530, 0.5340)
blizzard	Bad Wea.	skating	Bad Wea.	(0.8471, 0.9870, 0.9117)
snowFall	Bad Wea.	skating	Bad Wea.	(0.7618, 0.9924, 0.8619)
tramCrossroad	Low Fr.	turnpike	Low Fr.	(0.9188, 0.9537, 0.9359)
tramCrossroad	Low Fr.	tunnelExit	Low Fr.	(0.7544, 0.3724, 0.4987)
abandonedBox	Int. Mot.	highway	baseline	(0.8103, 0.9996, 0.8951)
busStation	Shadow	pedestrians	baseline	(0.9747, 0.8668, 0.9176)
fall	Dyn. Bg.	highway	baseline	(0.8102, 0.9994, 0.8949)
canoe	Dyn. Bg,	busStation	Shadow	(0.0718, 0.9738, 0.1337)
Average				(0.7274, 0.9089, 0.7609)

3.3.1 Transferability of D-DPDL

In this section, the transferability of our D-DPDL algorithm is demonstrated. Previous approaches related to deep learning only guarantee the effectiveness under the assumption that the training frames and testing frames are obtained from the same scene. In contrast, our D-DPDL model is valid even when the testing frames are captured from a scene different from the one used for capturing training frames. This happens because the distribution of pixels, which are learned by the D-DPDL model, is a general feature regardless of the scene.

In order to demonstrate transferability, the proposed approach is evaluated for image frames which do not belong to the training video. During the experiments, several challenging videos are selected from Change Detection net 2014 (CDnet2014) dataset [135] for evaluation. In particular, 10 frames are extracted from the training video to learn the distribution information of pixels, and all frames of the testing video are utilized to evaluate the proposed approach. Quantitative and qualitative evaluations are shown in Table 3.2 and Fig. 3.5 respectively. The evaluations are divided into two parts. First,

Table 3.3: Quantitative demonstration of the effectiveness of the dynamic training strategy and the Bayesian refinement model using Re, Pr and Fm metrics. The performances of our D-DPDL with Bayesian refinement model are shown in the first row, the one without are presented in the second row and the third row demonstrates the performances of our previous DPDL method, in which the dynamic training strategy is not used, in CDnet2014 [135] dataset.

Videos		highway	office	sidewalk	boulevard	abandonedBox	parking	snowFall	wetSnow	bungalows	cubicle	Average
Bayesian Refinement	Re	**0.9666**	**0.9498**	**0.8007**	0.6404	0.7990	**0.8690**	0.7569	**0.6011**	0.8091	**0.9010**	**0.8094**
	Pr	**0.9978**	0.9104	**0.6304**	**0.6610**	**0.9759**	0.6524	**0.3754**	**0.8259**	**0.9923**	**0.6611**	**0.7683**
	Fm	**0.9819**	**0.9297**	**0.7054**	**0.6505**	**0.8787**	**0.7453**	**0.5019**	**0.6958**	**0.8914**	**0.7626**	**0.7743**
Dynamic Training	Re	0.9076	0.9275	0.6932	0.5906	0.7779	0.7641	0.7149	0.5435	0.7429	0.8224	0.7485
	Pr	0.9819	0.9061	0.3672	0.4222	0.8456	0.4211	0.2810	0.7082	0.9807	0.6230	0.6537
	Fm	0.9433	0.9166	0.4801	0.4924	0.8103	0.5430	0.4035	0.6150	0.8454	0.7090	0.6759
DPDL [152]	Re	0.9213	0.3351	0.6062	**0.7327**	**0.8063**	0.1180	**0.7970**	0.3772	**0.9523**	0.6880	0.6334
	Pr	0.9444	**0.9900**	0.2397	0.1490	0.6515	**0.9471**	0.1987	0.3175	0.2635	0.3984	0.5100
	Fm	0.9327	0.5007	0.3436	0.2477	0.7207	0.2098	0.3181	0.3448	0.4128	0.5046	0.4536

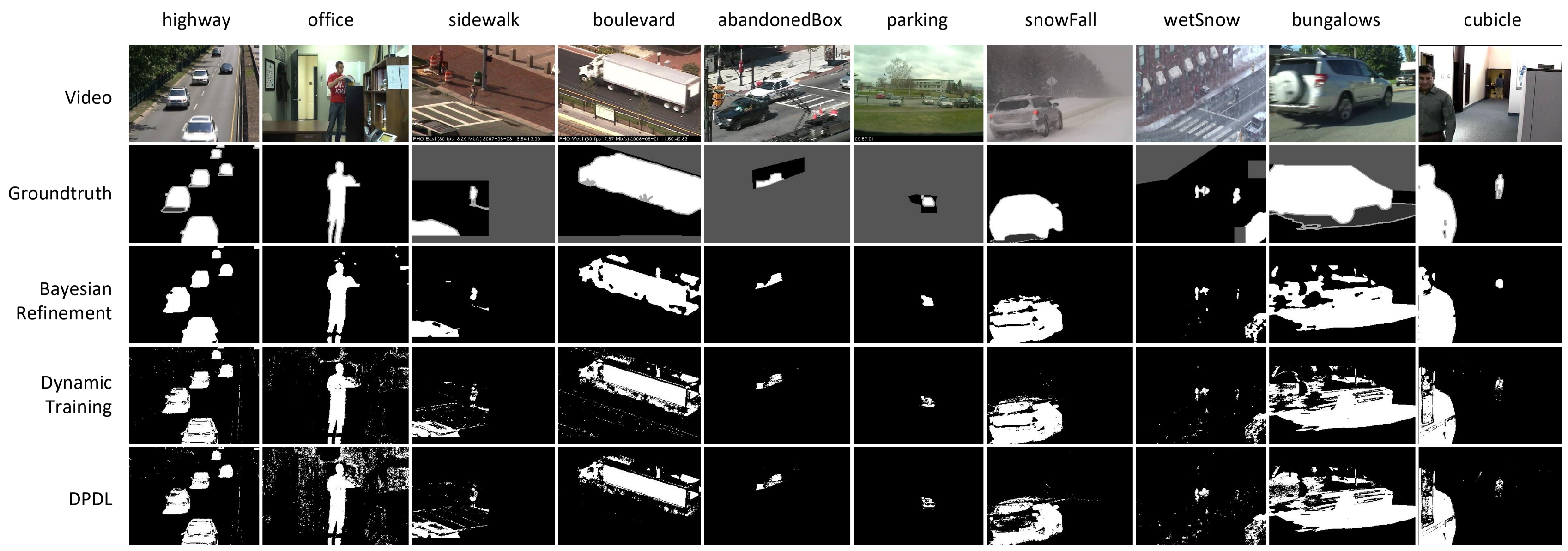


Figure 3.6: Qualitative comparison between D-DPDL with the Bayesian refinement model, the one without it and our previous DPDL [152] method in CDnet2014 [135] dataset. In the third row, both the Bayesian refinement model and dynamic training strategy are used to generate the foreground masks of “Bayesian Refinement”, only dynamic training strategy are utilized to produce the foreground masks of “Dynamic Training” which is shown in the fourth row, none of these techniques are used in our previous DPDL [152] model in the last row.

the training videos and testing videos are selected from the same category, which means the distribution of pixels from training videos and testing videos may share some common features. For example, most of the pixels on the background scene of videos in the category of "Dynamic Background" shared a repetitive pattern of variation in pixels' observations. In the second part, the training video and testing video are selected from different categories to emphasize the difference between training videos and testing videos for evaluating the transferability of the proposed approach.

As shown in Table 3.2 and Fig. 3.5, our D-DPDL model achieves 0.7609 on the average Fm metric of all testing videos, when the scenes in the training and testing videos are different. Therefore, it is fair to claim that our D-DPDL is effective when the training and testing videos are completely different, demonstrating the transferability of the proposed approach. Moreover, the performance of the transferability is reasonably different in different cases due to the diversity of scenes in these videos. For example, in the first case "highway-pedestrians," the proposed approach is trained on the video "highway" and tested on the video "pedestrians." Since the scenes for these two videos do share several similar parts like the plants and roads, the proposed approach generates promising foreground results in this case. However, when we simply switch these two videos in the second case "pedestrians-highway," the performance of the proposed approach has an obvious decline, due to the fact that the diversity of the scene with pedestrians is lower than the one with highways. The diversity of scenes thus plays an important role in the transferability of the proposed approach. This is more clear from the cases "overpass-canoe" and "canoe-overpass," in which the video "overpass" includes water, plants and roads, but the video "canoe" only contains water and plants. When the proposed approach is trained on the video "overpass" and tested by the video "canoe," it achieves 0.9008 for the Fm metric. On the other hand, the proposed approach only achieves 0.5340 when the training and testing videos are switched as shown in the case "canoe-overpass." Hence, it only achieves 0.1337 when the proposed approach is applied to a more diverse video "busStation" as shown in the case "canoe-busStation," in which the video "canoe" with low diversity is used for training and video "busStation" with high diversity is used for testing.

Similar phenomena are also shown in the cases "blizzard-skating" and "snowFall-skating," in which the testing videos are the same but the training videos are different. As we can see in Fig. 3.5, the scenes in the video "blizzard" and "snowFall" are similar to each other. Thus, the proposed approach learns similar distribution information when these videos are used for training. Consequently, the proposed approach achieves similar performances when it is tested on the same video "skating" in these two cases. Similar conditions are also shown in the cases "tramCrossroad-turnpike" and "tramCrossroad-tunnelExit" where the proposed approach is trained on the same video but tested on different videos. In summary, the transferability of the proposed approach is highly dependant on the diversity of scenes included in the training video, where more diverse distribution information can be learned by the proposed approach. When the diversity in the training video is higher than that of the testing video, the proposed approach generates good results. Otherwise, it fails like for the case "tramCrossroad-tunnelExit."

In the second part of Table 3.2 and Fig. 3.5, the transferability of the proposed approach is demonstrated under the condition that the training videos and testing videos are captured from different categories, including the cases "abandonedBox-highway," "busStation-pedestrians," "fall-highway" and "canoe-busStation." The proposed approach works well in the first three cases, as the diversity of scenes in the training videos is obviously higher than that of the testing videos. Therefore, the transferability of the proposed approach is affected by the diversity of scenes in the videos rather than their categories. Moreover, the proposed approach failed in the last case as the training videos only include plants and water, which do not appear in the testing videos. Since the objects in the training videos are not contained in the testing videos, the proposed approach cannot capture enough distribution information, which is the main reason for the poor performance in this case.

3.3.2 Bayesian Refinement Model and Dynamic Training

In this section, the validation of dynamic training and Bayesian refinement model is presented. The strategy of dynamic training is proposed to prevent the network from overfitting the pattern implied in random permutations. The Bayesian refinement model is devised to overcome the random noise generated by random permutations used in RPoTP features.

In order to present the validation of the Bayesian refinement model, we compare the performances of the proposed approach with or without the compensation of the Bayesian refinement model. As shown in Table 3.3 and Fig. 3.6, the performance of the proposed approach with refinement model is denoted by "Bayesian Refinement" and the performance without refinement model is denoted by "Dynamic Training." Moreover, for validating the dynamic training strategy, the performance of our previous work DPDL [152] is presented as well for comparison, since the dynamic training strategy is not applied in DPDL [152]. During comparison, only one groundtruth frame is used for training and several challenging videos are selected from CDnet2014 dataset [135] for comparison.

As shown in Table 3.3 and Fig. 3.6, the dynamic training strategy and Bayesian refinement model have significant contribution to the efficiency and accuracy of the proposed approach. Compared to DPDL [152], the dynamic training strategy contributes over 30% improvement on the average Fm value, since RPoTP features are dynamically generated to prevent the network from overfitting to the pattern implied in random permutations. Then, the efficiency of the dynamic training strategy is demonstrated. In particular, the improvement resulting from the dynamic training strategy is more clear in several complex videos such as the video "office," "bungalows" and "cubicle." In these videos, the motion of moving objects is complex, where the network generates more noise in the foreground and background when it overfits to the pattern in a random permutation. For example, in the visual foreground shown in Fig. 3.6, the background of the video "office" has many randomly incorrect foreground points compared to the video "highway."

The Bayesian refinement model further improves the performance of the proposed approach by handling the noise generated from the utilization of random permutations during capturing RPoTP features. This is shown in the visual foreground proposed in Fig. 3.6. The foreground mask produced by the dynamic training includes many noisy points in the background scenes as well as the foreground objects. By contrast, all these noisy points are removed in the foreground mask after refinement by the Bayesian refinement model. This is achieved by the marginal probability of the Bayesian refinement model discussed in Eqn. 3.4, which can handle random noise. Moreover, the conditional probability is utilized to protect the border between the foreground and background, and the Bayesian refinement model contributes an extra 10% improvement on the average Fm metrics. With the help of these two components, the proposed approach achieves much better performance with even less training frames. Specifically, our D-DPDL model with 10 groundtruth frames for training achieves higher performance than our previous DPDL [152] model with 40 groundtruth frames for training.

3.3.3 Comprehensive Evaluation of D-DPDL

In this section, comprehensive comparisons between our Dynamic Deep Pixel Distribution Learning (D-DPDL) model and several sophisticated state-of-the-art methods including traditional methods and deep learning methods on LASIESTA [27] and CDnet2014 [135] datasets are presented. The results of the compared algorithms are directly referred from the implementation provided by authors.

It should be noted that the training data is important to supervised methods and has direct contribution to their performances. Methods with more training data are desired to achieve better performance. Especially, with the utilization of deep learning network, a few excellent methods [86], [87], [136], [137] achieve almost perfect results in standardized datasets when many groundtruth foreground masks are used for training. For example, FgSegNet model proposed by Lim et al. [87] achieves over 98% F-scores when 200

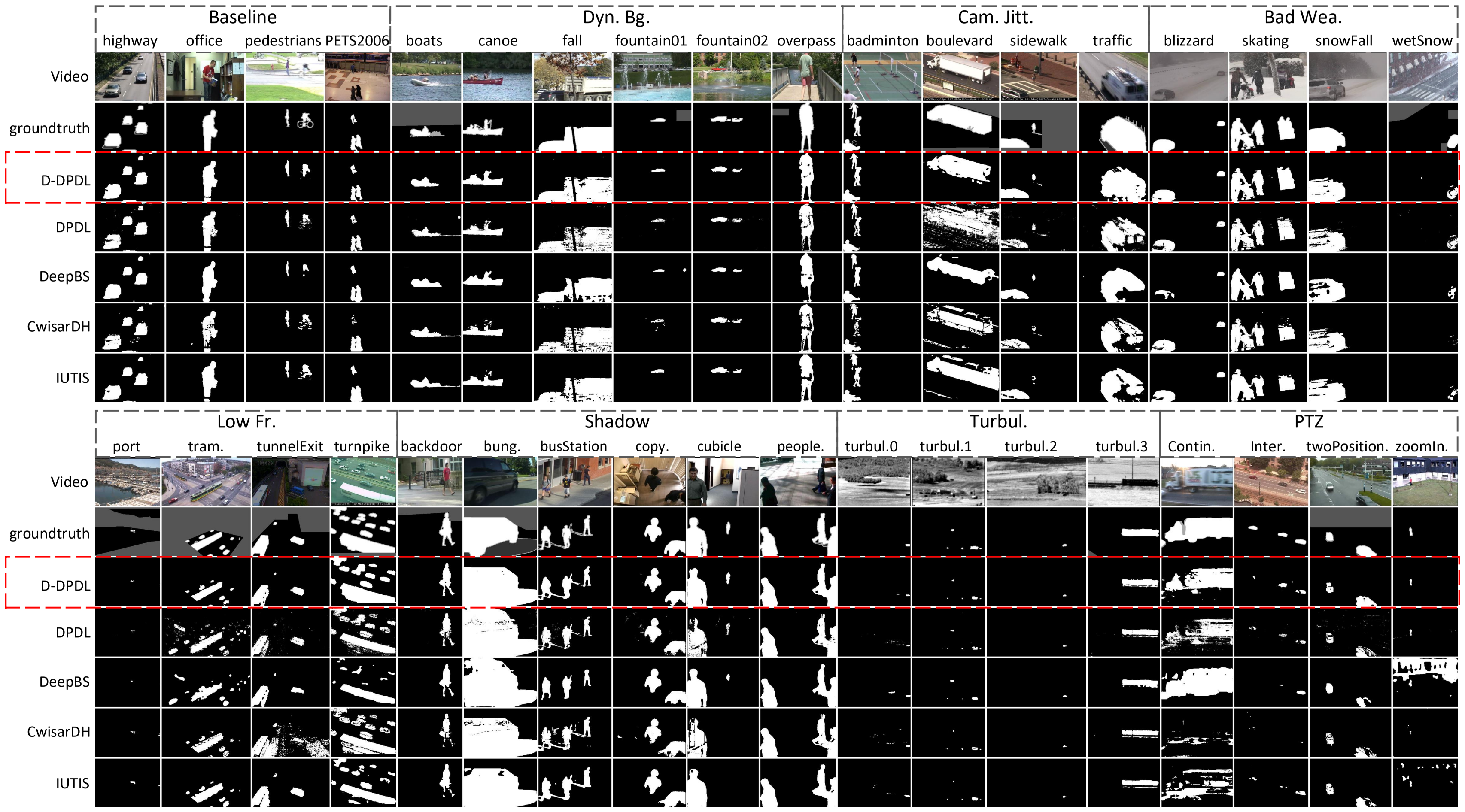


Figure 3.7: Qualitative comparison between the proposed approach and several state-of-the-art methods including DeepBS [4], CwisarDH[49] and IUTIS-5[8], on CDnet2014 [135] dataset, which contains 53 videos grouped into 11 categories.

Table 3.4: Quantitative comparison between the proposed approach and state-of-the-art using Fm metric on LASIESTA [27] dataset.

Videos	D-DPDL$_3$	D-DPDP$_1$	CueV2[7]	Haines[50]	Cue[26]	SOBS[97]
I_SI_01	0.9596	0.8335	0.9208	**0.9622**	0.8143	0.9559
I_SI_02	0.8687	0.7963	0.8403	0.8130	0.7576	**0.9409**
I_CA_01	**0.9309**	0.6881	0.9062	0.9220	0.8424	0.8416
I_CA_02	**0.8850**	0.8724	0.7826	0.8656	0.6296	0.8731
I_OC_01	**0.9710**	0.8094	0.7013	0.8920	0.8274	0.9573
I_OC_02	**0.9677**	0.9331	0.8600	0.9526	0.8781	0.9508
I_IL_01	0.7161	0.6398	0.6452	**0.8861**	0.7966	0.1898
I_IL_02	**0.8972**	0.8749	0.6523	0.8122	0.7864	0.2312
I_MB_01	0.9699	0.9627	0.9543	**0.9816**	0.7779	0.9728
I_MB_02	0.9195	0.5937	**0.9204**	0.7064	0.6797	0.8517
I_BS_01	**0.8371**	0.5609	0.7132	0.6285	0.5065	0.4015
I_BS_02	0.6178	**0.7393**	0.6156	0.7333	0.6607	0.4021
O_CL_01	**0.9792**	0.9584	0.9508	0.6946	0.9280	0.9657
O_CL_02	0.9800	**0.9819**	0.9045	0.9588	0.8995	0.9760
O_RA_01	**0.9072**	0.4047	0.8453	0.8225	0.7462	0.8353
O_RA_02	0.9803	**0.9812**	0.8886	0.9590	0.8699	0.9591
O_SN_01	**0.9690**	0.4590	0.9317	0.3054	0.8214	0.9093
O_SN_02	0.9341	**0.9421**	0.6256	0.0426	0.0895	0.7116
O_SU_01	**0.9065**	0.8783	0.6774	0.8115	0.6527	0.8742
O_SU_02	**0.9388**	0.9171	0.7669	0.9021	0.8074	0.8843
Average	**0.9068**	0.7913	0.8051	0.7826	0.7386	0.7842

groundtruth frames of each video are used for training. Though, these methods achieve better quantitative scores than the proposed approach, they also used much more groundtruth frames than our method requiring less than 10 groundtruth frames of each video. Therefore, in order to propose a fair comparison, besides unsupervised algorithms, several state-of-the-art methods claiming limited groundtruth frames are selected for comparison, which includes CwisarDH [49], DeepBS [4], GuidedBS [84], CNN-SFC [33] and our previous DPDL [152].

Among these supervised methods, CwisarDH [49] mentioned in their work that several pixel instances are taken in different frames of videos for training, but details are lacking. DeepBS [4] claims 5% of groundtruth frames of entire dataset are utilized for training, which means around $5\% \times 160000 = 8000$ groundtruth foreground masks are required to train the network, where

Table 3.5: Quantitative comparison between the proposed approach and state-of-the-art using Fm metric on CDnet2014 [135] dataset.

	Approach	Baseline	Dyn. Bg.	Cam. Jitt.	Int. Mot.	Shadow	Ther.	Bad Wea.	Low Fr.	Nig. Vid.	PTZ	Turbul.	Overall
unsupervised models	IUTIS-5[8]	0.9567	0.8902	0.8332	0.7296	0.9084	0.8303	0.8248	0.7743	0.5290	0.4282	0.7836	0.7717
	SemanticBGS[12]	0.9604	**0.9489**	0.8388	0.7878	0.9478	0.8219	0.8260	**0.7888**	0.5014	0.5673	0.6921	0.7892
	AAPSA[111]	0.9183	0.6706	0.7207	0.5098	0.7953	0.7030	0.7742	0.4942	0.4161	0.3302	0.4643	0.6179
	MBS[113]	0.9287	0.7915	0.8367	0.7568	0.7968	0.8194	0.7980	0.6350	0.5158	0.5520	0.5858	0.7288
	PAWCS[19]	0.9397	0.8938	0.8137	0.7764	0.8913	0.8324	0.8152	0.6588	0.4152	0.4615	0.6450	0.7403
	ShareM[23]	0.9522	0.8222	0.8141	0.6727	0.8898	0.8319	0.8480	0.7286	0.5419	0.3860	0.7339	0.7474
	SuBSENSE[18]	0.9503	0.8177	0.8152	0.6569	0.8986	0.8171	0.8619	0.6445	0.5599	0.3476	0.7792	0.7408
	WeSamBE[70]	0.9413	0.7440	0.7976	0.7392	0.8999	0.7962	0.8608	0.6602	0.5929	0.3844	0.7737	0.7446
	GMM[159]	0.8245	0.6330	0.5969	0.5207	0.7370	0.6621	0.7380	0.5373	0.4097	0.1522	0.4663	0.5707
	RMoG[129]	0.7848	0.7352	0.7010	0.5431	0.7212	0.4788	0.6826	0.5312	0.4265	0.2470	0.4578	0.5735
	MSCL-FL[67]	0.9400	0.9000	0.8600	0.8400	0.8600	0.8600	0.8800	N/A	N/A	N/A	N/A	N/A
	DSPSS[34]	0.9664	0.9057	0.8662	0.7870	0.9177	0.7328	N/A	N/A	N/A	N/A	N/A	N/A
	STSHBM[20]	0.9534	0.9120	0.8503	0.8349	0.8930	0.8579	N/A	N/A	N/A	N/A	N/A	N/A
	DPGMM[50]	0.9286	0.8137	0.7477	0.5418	0.8127	0.8134	N/A	N/A	N/A	N/A	N/A	N/A
supervised models	DeepBS[4]	0.9580	0.8761	0.8990	0.6098	0.9304	0.7583	0.8301	0.6002	0.5835	0.3133	**0.8455**	0.7548
	GuidedBS[84]	0.9467	0.8266	0.8818	0.6229	0.8910	0.7490	0.8711	0.6396	0.5048	0.6057	0.8114	0.7591
	CNN-SFC[33]	0.9497	0.9035	0.8035	0.7499	0.9127	0.8494	**0.9084**	0.7808	**0.6527**	**0.7280**	0.8288	0.8243
	CwisarDH[49]	0.9145	0.8274	0.7886	0.5753	0.8581	0.7866	0.6837	0.6406	0.3735	0.3218	0.7227	0.6812
	DPDL_1 [152]	0.7886	0.6566	0.5456	0.5115	0.6957	0.6697	0.6036	0.5966	0.3953	0.2942	0.6301	0.5807
	DPDL_{20} [152]	0.9620	0.8369	0.8627	0.8174	0.8763	0.8311	0.8107	0.6646	0.5866	0.4654	0.7173	0.7665
	DPDL_{40} [152]	0.9692	0.8692	0.8661	0.8759	0.9361	0.8379	0.8688	0.7078	0.6110	0.6087	0.7636	0.8106
	D-DPDL_1	0.8964	0.8348	0.7731	0.6956	0.8284	0.8000	0.7313	0.6476	0.4735	0.3414	0.7708	0.7084
	D-DPDL_{10}	**0.9743**	0.9008	**0.9098**	**0.8887**	**0.9526**	**0.8808**	0.9001	0.7527	0.6505	0.6362	0.8073	**0.8413**

160000 is the total number of groundtruth frames contained in CDnet2014 dataset [135]. Although GuidedBS [84] extracts no foreground masks from groundtruth frames for training, 200 foreground images of each video are assumed from the results generated by SubSENSE [18], which is a good background subtraction method. Therefore, we consider that $200 \times 53 = 10600$ foreground masks are required in GuidedBS [84]. This is around 6.625% of entire dataset. CNN-SFC [33] utilizes 4000 foreground masks from groundtruth data for training, which is around 2.5% of the entire dataset. For DPDL model [152], DPDL$_1$, DPDL$_{20}$ and DPDL$_{40}$ utilize 1, 20 and 40 groundtruth masks for each video respectively, which are 0.033%, 0.66% and 1.32% of entire dataset. In contrast to the results of D-DPDL model proposed in this chapter, 1 and 3 groundtruth frames are extracted for training in each video of LASIESTA [27] dataset to demonstrate the performance of D-DPDL$_1$ and D-DPDL$_3$ shown in Table 5.8, which takes around 0.24% and 0.72% of the groundtruth frames available in LASIESTA [27] dataset. In addition, in the CDnet [135] dataset, 1 and 10 frames are used for training in each video to demonstrate the performance of D-DPDL$_1$ and D-DPDL$_{10}$ shown in Table 5.9 and Fig. 3.7, which represents 0.033% and 0.33% of the groundtruth frames available in entire dataset.

Quantitative evaluation of the proposed approach on LASIESTA [27] dataset is shown in Table 5.8. The proposed approach achieves promising results even when only 1 groundtruth frame is used for training, and it generates the best performance when 3 groundtruth frames are used for training. There are 20 videos obtained from 10 different conditions in LASIESTA [27] dataset. Among them, videos "I_SI_01-02" are obtained from simple indoor scenes, in which the distribution of temporal pixels is easily classified, and the proposed approach achieves good performance. In addition, videos "I_CA_01-02," "I_OC_01-02," "I_IL_01-02," and "I_MB_01-02" include challenges of camouflage moving objects, occluded moving objects, moving objects with global illumination changes and modified background respectively. These challenges have significant influence on the distribution of pixels' observations. Fortunately, benefiting from the strong learning ability of a deep learning network, our D-DPDL model handles these challenges and achieves promising results. In particular, D-DPDL$_1$ does not achieve good performance in video "I_IL_01," since the illumination is changing during entire image sequences and 1 groundtruth frame may not contain enough information for the network to learn. When the number of groundtruth frames is increased to 3, the results of the proposed approach becomes promising.

However, in video "I_BS_02," the performance of D-DPDL$_3$ is not good and it is even worse than their counterpart with 1 groundtruth frame for training. The reason is the imbalance between training instances for foreground and background. During the training of the proposed approach, the groundtruth frames are randomly selected from image sequences. When only 1 groundtruth frame is used for training, only the frames with foreground objects are used as candidates to make sure there are foreground instances in the training data. While for 3 groundtruth frames, the extra 2 groundtruth frames are randomly selected from the entire video, and it is possible that the selected groundtruth frames do not include any moving object to cause foreground and background instances being extremely imbalanced. This condition becomes even worse when the moving objects are relatively small and only appear in a few frames, which happens in the video "I_BS_02." In contrast, videos "O_CL_01-02," "O_RA_01-02," "O_SN_01-02" and "O_SU_01-02" are obtained from outdoor scenes under different weather conditions, such as cloudy condition, rainy condition and so on. Our D-DPDL$_3$ achieves over 90% F-scores in all these videos, since the Bayesian refinement model has the ability to handle these isolated distractions from weather conditions.

Quantitative and qualitative evaluations of the proposed approach on CDnet2014 [135] dataset are shown in Table 5.9 and Fig. 3.7 respectively. To the best of our knowledge, CDnet2014 [135] dataset is the largest dataset for background subtraction, which contains 11 categories of videos. Therefore, the evaluation on CDnet2014 [135] dataset is fair and complete. As shown in Table 5.9, our D-DPDL$_{10}$ has the best overall performance using 10 groundtruth frames compared to unsupervised methods as well as the supervised methods. Moreover, even with only one groundtruth frame, the proposed

D-DPDL$_1$ still produces promising results which are close to the ones of MBS [113] or SuBSENSE [18]. Both of these demonstrate efficiency and accuracy of the proposed approach assuming limited groundtruth frames. In the “Baseline” category, the complexity of the distribution in pixel is not high, and the statistical information is easily learned by the proposed approach. The proposed D-DPDL$_{10}$ thus achieves excellent performance in this category with 10 groundtruth frames used for training. It is higher than our previous DPDL$_{40}$ training by 40 groundtruth frames. This improvement demonstrates the effectiveness of our strategy of dynamic RPoTP feature generation, as well as the Bayesian refinement model. In addition, the improvement is more clear if we refer to the comparison between D-DPDL$_1$ and DPDL$_1$, where only one groundtruth frame is utilized for training. D-DPDL$_1$ achieves much higher performance compared to DPDL$_1$, since the dynamic training strategy is not applied in DPDL$_1$, the network tends to overfit to the pattern implied in random permutations when the training data is limited.

In the “Dynamic Background” category (Dyn. Bg.), the improvement of D-DPDL becomes more obvious due to the increasing complexity of pixel distribution compared to our previous DPDL model [152]. In a dynamic background, the observations of pixels have a pattern of repeating variation, which generates a multiple-peak distribution. Compared with the DPDL model, the proposed D-DPDL model can focus better on the distribution with the help of the proposed dynamic training strategy. Therefore, our D-DPDL$_1$, which is trained with one groundtruth, even achieves performance close to DPDL$_{20}$ taking 20 groundtruth frames for training. However, due to the complexity of the distribution information in the pixels of a dynamic background, the network is trained by RPoTP features which unavoidably generates random noise during the utilization of random permutations. Thus, the Bayesian refinement model is proposed to handle the random noise, and the proposed D-DPDL model gives better results. This is more clear in the qualitative comparison shown in Fig. 3.7, in which the foreground masks generated by the D-DPDL model have almost no noise compared to our previous DPDL [152] model. Similarly, in the Camera Jitter (Cam. Jitt.) category, except for random permutations, the jitter of cameras also generates random noise. In this condition, the distribution of pixels' observation have multiple peaks as well. Benefiting from the learning ability of deep learning networks and our Bayesian refinement model's ability to handle random noise, the proposed approach generates promising results.

In the "Shadow" category, the main challenge comes from the illumination condition. The observations of pixels have significant variations when the illumination is occluded by moving objects. In this condition, the intensity of pixels' observations have a constant decline, which has a small influence on the waveform of their distribution. Previous methods, such as SemanticBGS [12], usually utilize texture features to overcome the illumination change. In contrast, the proposed D-DPDL model focuses on the distribution itself led by the illumination change, and has a strong ability to learn the distribution of pixels in the shadow or unshaded regions. Therefore, the proposed approach produces excellent foreground results in this category. A similar situation is also shown in the "Thermal" (Ther.) category. The "bad weather" (Bad Wea.) category includes several challenging videos. In particular, raining and snowing cause several hurdles in learning distributions. However, with the help of our Bayesian refinement model, the proposed approach still achieves promising results with 10 groundtruth frames for training compared to other deep learning methods such as CNN-SFC [33]. A similar condition also prevails in the "Low Frames" (Low Fr.) category, in which the different frame rates of cameras produce challenges to the methods related to distribution analysis such as GMM [159]. Fortunately, with the strong learning ability of our D-DPDL model, the proposed approach still generates promising results.

However, the proposed approach does not work well in the "Night Video" and "PTZ" category. It is understandable why the proposed approach cannot work in the PTZ category, since the videos in this category are obtained by moving cameras. For videos captured by a moving camera, the pixels no longer maintain their positions. The statistical information in the RPoTP feature is thus meaningless for learning the distribution. In the "Night Video" category, the proposed approach is still the best of two and close to CNN-SFC [33]. The distribution learning still works in this category, but the lighting in a video is too complex to be learned with limited groundtruth frames. Thus, the performance of the proposed approach is not that good.

3.3.4 Implementation details and Computational Cost

The D-DPDL model is implemented in Matlab with MatConvNet [131], and the source code is available at `https://github.com/zhaochenqiu/D_DPDL`. The experiments are run on a PC with i7 CPU processor and GeForce GTX 1080 GPU processor in the Matlab 2018b environment under Ubuntu 16.04. In the evaluation experiments, all images have not been normalized. During training, 40 epochs are set as the maximum and the objective-epochs plots on four videos from different categories are shown in Fig. 3.8. Correspondingly, the processing time of the D-DPDL model on these videos are shown in Table 3.6.

As shown in Table 3.6, our D-DPDL is divided into three parts:

- 1) RPoTP features extraction: Observations of pixels are converted to RPoTP features and used as the input to the network.
- 2) Foreground segmentation: RPoTP features captured from pixels are classified by a convolutional neural network to generate foreground mask.
- 3) Bayesian refinement: The foreground mask generated from last step is iteratively refined by a Bayesian refinement model.

In total, the proposed D-DPDL model needs around 5 seconds to process an image frame with a resolution of 320×240, excluding the I/O time (i.e., reading frames from disk), since the proposed method is implemented purely in Matlab without any code optimization. In particular, the Bayesian refinement procedure uses most of the runtime, because it is an iterative procedure implemented in Matlab. In future, the refinement procedure will be re-implemented in C++, which will save considerable time. In addition, in the foreground segmentation part, we believe a multiple GPU implementations will accelerate the proposed approach. In the procedure of RPoTP features extraction,

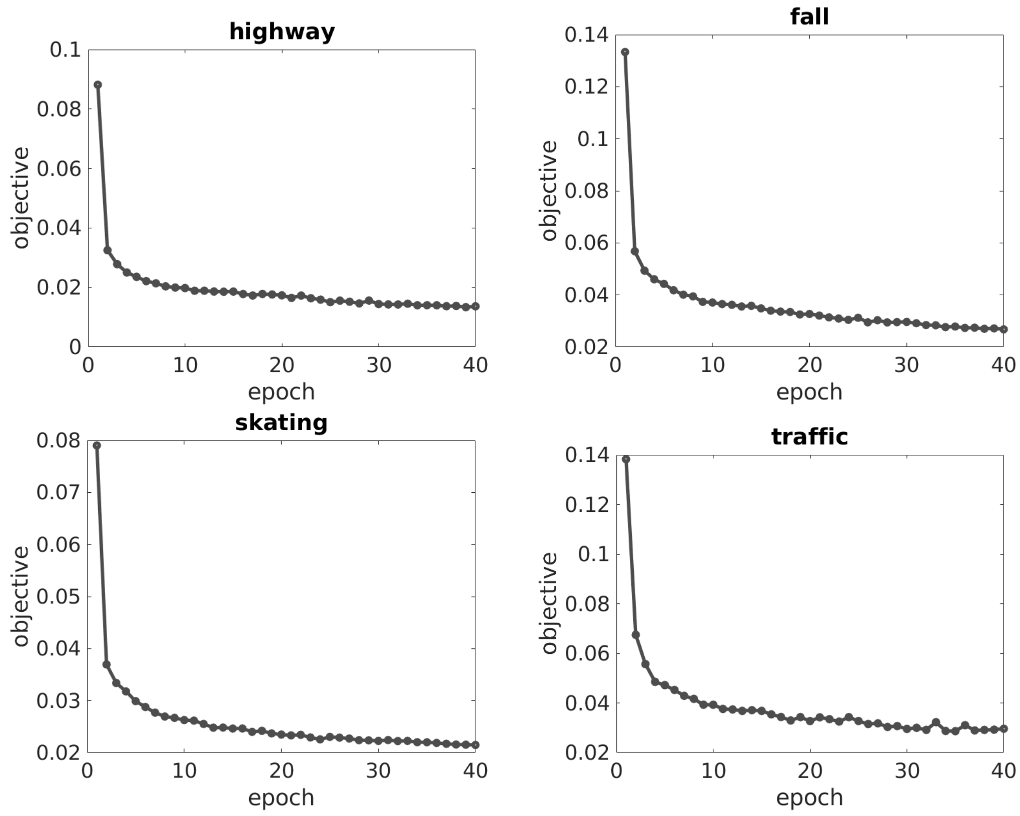


Figure 3.8: The objective-epoch plots on four videos from different categories, in which video “highway” comes from category “Baseline,” video ”fall” is extracted from category “Dynamic Background,” video “skating” is from category “Bad Weather” and video “traffic” belong to category “Camera Jitter.”

Table 3.6: The runtime of different parts of D-DPDL model in frames captured from several videos with varying resolutions (seconds/frame).

Videos	Resolution	RPoTP Features Extraction	Foreground Segmentation	Bayesian Refinement	Overall
highway	320×240	0.5577	1.5453	3.3475	5.4505
fall	720×480	2.4810	10.0575	14.5516	27.0901
skating	540×360	1.3670	6.6319	7.2662	15.2651
traffic	320×240	0.5109	1.7026	2.7228	4.9363

since loop computations are involved, the re-implementation by C++ will accelerate the proposed approach. Finally, the GPU implementation for random permutation should also accelerate the proposed approach.

3.4 Conclusion

In this chapter we proposed the Dynamic Deep Pixel Distribution Learning (D-DPDL) model for background subtraction. In particular, Random Permutation of Temporal Pixels (RPoTP) features is proposed to indirectly force the convolutional neural network to learn statistical distributions. Also, a Bayesian refinement model was proposed to handle the random noise generated during random permutations. Due to the pixel-wise representation of the RPoTP feature, a large number of RPoTP features can be captured with limited groundtruth frames for training the proposed approach. Moreover, since the statistical distribution is a general feature regardless of the scene, our D-DPDL model is effective even under the condition that the training videos and testing videos are completely different. Comprehensive evaluations comparing with other state-of-the-art methods demonstrate the superior efficiency of the proposed approach. In the future, we will work on optimizing the code and consider GPU implementation so that the proposed approach can get close to real-time performance.

Chapter 4

Pixel Distribution Learning for Vessel Segmentation and Crack Detection

Abstract

The dynamic deep pixel distribution learning (D-DPDL) model proposed in last chapter has demonstrated a few excellent properties such as less training data, training videos and testing videos can be different, which demonstrate a good potential of wide range of applications more than background subtraction. Therefore, in this chapter, we further applied the pixel distribution learning technique into vessel segmentation and crack detection which are related to medical images and architecture images respectively. Moreover, in order to improve the accuracy of the proposed approach, a multiple scale strategy is utilized for capturing features. In addition, Based on our preliminary experiments, we currently believe that a wide network, rather than a deep one, is better for distribution learning. Therfore, there is only one convolutional layer, one rectified linear layer and one fully connected layer followed by a softmax loss in our network. Evaluations using standard benchmark datasets demonstrate that the proposed approach achieves promising results compared with state-of-the-art methods.

4.1 Vessel Segmentation

4.1.1 Introduction

Vessel segmentation is a fundamental problem of medical image processing, with has a wide range of applications; such as oncology [15], ophthalmology [13] and neurosurgery [102]. Previous approaches usually devised an artificial model to analyze the distribution of pixels for vessel segmentation. However, because of the diversity of the medical images, which comes from the profiles of different patients, or machines, vessel segmentation is still a challenging problem in computer vision. Vessel segmentation is essentially a binary pixel-wise classification problem. A pixel is classified as a vessel or background based on the comparison with its neighborhood. In this work, the distributions of spatial pixels are used for vessel segmentation, and a novel method based distribution learning is proposed.

In our previous work [151], [152], we demonstrated that a convolutional neural network can be guided to learn a statistical distribution by randomly permutating the temporal pixels. In this work, our previous technique is extended for vessel segmentation, since vessels in an image can be segmented by classifying the distributions of spatial pixels. The pixels are subtracted from their neighborhoods, and the distributions of the subtraction results are input into the network for vessel segmentation, as is shown as Fig. 4.1.

In the proposed approach, a spatial distribution descriptor named the Random Permutation of Spatial Pixels (RPoSP) feature is proposed, in which the spatial pixels are randomly permutated to guarantee that only the statistical information is retained. The RPoSP features are dynamically generated as the input to the convolutional neural network (CNN) for every training epoch. It indirectly forces the network to rely solely on the statistics of the distribution of spatial pixels. Moreover, several RPoSP features captured under different scales [114] are combined and input to the network, in order to improve the accuracy of the proposed approach. The main differences between this work and our previous work [151], [152] are:

- Random Permutation of Spatial Pixels: In this work, the distribution

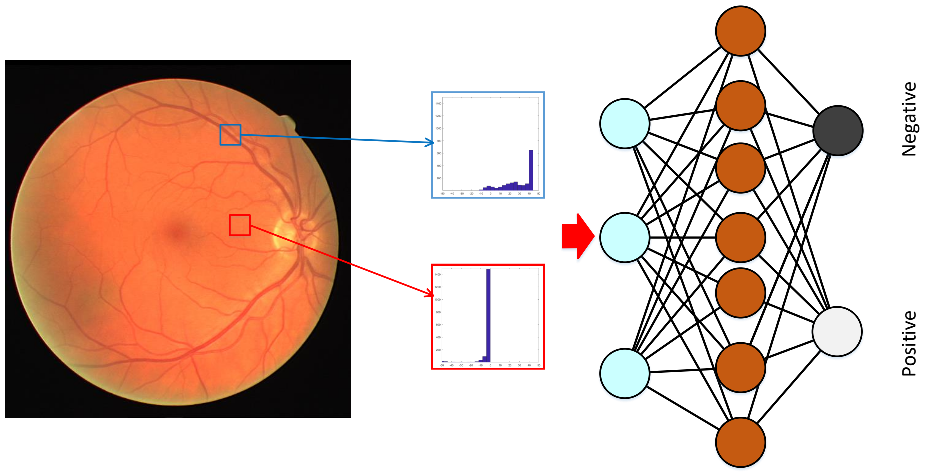


Figure 4.1: Pixels Distribution Learning for Vessel Segmentation.

of spatial pixels rather than temporal pixels is learned by the network. Compared to the temporal pixels, the variation of spatial pixels includes higher complexity and diversity, which is one of the motivations behind capturing the distribution information under multiple scales.

- Multiple Scales: We captured the distribution information at multiple scales rather than only one scale. This strategy provides the network with better information for learning the distribution and improves the accuracy of the proposed approach.

- Network architecture: We simplified the network architecture considering the computational cost. Our architecture is quite simple; thus, it should probably not be considered "deep." It only includes one convolutional layer, one rectified linear layer, and one fully connected layer.

4.1.2 Spatial Pixel Distribution Learning under Multiple Scales

In this section, the details of the proposed approach for vessel segmentation are discussed. The flowchart of the proposed approach is shown in the Fig. 4.2, in which the Random Permutation of Spatial Pixel (RPoSP) features are

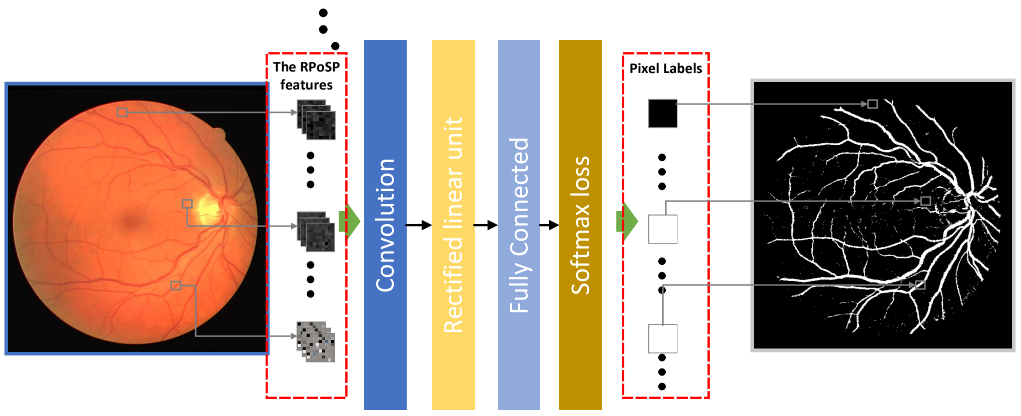


Figure 4.2: The flowchart of the proposed approach.

captured first and then input into the network to label if a pixel belongs to a vessel or is part of the background.

For vessel segmentation in medical image processing, it is reasonable to segment vessels based on comparisons between the center pixel and its neighborhood. This is because the intensity of pixels in a vessel is significantly different from ones in its neighboring pixels. In this work, we focus on learning the distribution of these comparisons for segmenting vessels, utilizing convolutional neural networks. In addition, motivated by our previous work [152] [151], it is possible to force a network to solely focus on the statistical distribution, by randomly permutating the temporal pixels which are used as the input to the network. However, in this work, the distribution information is derived from spatial pixels instead of temporal pixels. Therefore, a new distribution descriptor named Random Permutation of Spatial pixels (RPoSP features), which is an extension of our previous work, is proposed. Since the complexity of the distribution captured from spatial pixels is higher than the one from temporal pixels, a multiple scale strategy is proposed to extract the multiple RPoSP features to improve the robustness of the proposed approach. The combination of several RPoSP features captured from a particular pixel under different scales is input into the network for learning the distribution. The network architecture is devised as a classification network to classify pixels into the categories of a vessel or the background scene.

Table 4.1: Details of our network architecture, which consists of 4 convolutional layers, 3 batch normalization, 2 max pooling and a softmax operator.

Type	Filters	Layer size	Data size
Input Data			$15 \times 15 \times 15$
Convolution	10024	$15 \times 15 \times 15$	$1 \times 1 \times 10024$
Rectified linear unit			$1 \times 1 \times 10024$
Convolution	2	$1 \times 1 \times 10024$	$1 \times 1 \times 2$
Softmax			

The procedure of extracting RPoSP features under multiple scales is shown in Fig. 4.3. We introduce the extraction of RPoSP features for one pixel, but the procedure is identical for each pixel. Let us denote a given vessel image as $I(x, y)$, where x and y represent locations of pixels. The patches with the center location of (x, y) under multiple scales are extracted, and the intensity of the center pixel is subtracted from them. Following this, the RPoSP features are captured by randomly permutating entries of subtracted patches at a particular scale. Mathematically, this can be described as follows:

$$\begin{aligned} &RPoSP_{x,y}(m, n : R_i, R_o) = I(x, y) - I(x + r(m), y + r(n)), \\ &m, n \in [1\ R_i], \quad r(m), r(n) \in [1\ R_o] \end{aligned} \tag{4.1}$$

where $RPoSP_{x,y}(m, n : R_i, R_o)$ denotes the RPoSP feature extracted from the pixel located at (x, y). m, n are the indices of an entry in a patch and $r()$ is the random permutation to generate a random position according to the input indices. R_i and R_o are the parameters to control the size of RPoSP features under multiple scales. In particular, R_o is the radius of patches under different scales, and R_i is the radius of the RPoSP features. The reason we use two parameters is that the RPoSP features captured from different scales need to be linked together to input into the network. First, a patch with radius of R_o is captured and randomly permutated. Then, a download-sampling is used to captured a patch with radius of R_i and used as the RPoSP feature captured at a particular scale. This step guarantees that the RPoSP features at different scales have the same size. Thus, these features thus can be linked together and input into the network.

The network architecture in the proposed approach is devised for a classification network, which is shown in Table 5.1. The input of the network is the combination of RPoSP features captured at multiple scales; and the output is the label corresponding to the pixel where these RPoSP features are extracted. Mathematically, are steps can be shown as follows:

$$\begin{aligned} \ell_{x,y} &= D(\mathcal{L}^{\theta}(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_N)), \\ \mathcal{F}_n &= RPoSP_{x,y}(m, n : R^i_n, R^o_n) \end{aligned} \tag{4.2}$$

where $\ell_{x,y}$ is a binary label of the pixel at location (x, y) identifying it as a vessel or the background, $\mathcal{L}$ is the learning block and $\mathcal{D}$ is the decision block. θ denotes the parameters of the learning block. The learning block $\mathcal{L}$ consists of convolutional, and Rectified linear layers. The decision block $\mathcal{D}$ includes a fully connected layer linked with a Softmax loss.

There are several differences between the proposed approach and our previous work during network training. For our previous work, the data input into the network for training is only generated once. Thus, the input of the network is the same for every training epoch. Under this condition, it is possible that the network overfits the pattern implied in random permutations rather than learn the statistical information included in RPoSP features. In order to address this issue, a dynamic training strategy is proposed as a compensation. In this approach, the entries of RPoSP features are randomly re-permutated by new permutations for every training epoch. This strategy effectively prevents our network from overfitting, and improves the accuracy of the proposed approach in complex scenes.

4.1.3 Experiments

In this section, we evaluate the proposed approach. Our approach is compared with several state-of-the-art methods [29], [39], [40], [95], [132], [134] on the DRIVE [122] dataset. In particular, both of these methods are based on deep learning network. It should be noted that the training data is important for supervised methods and has a direct contribution in their performance. Methods with more training data are expected to achieve better results. This is especially true when deep learning networks are utilized. In the DRIVE [122]

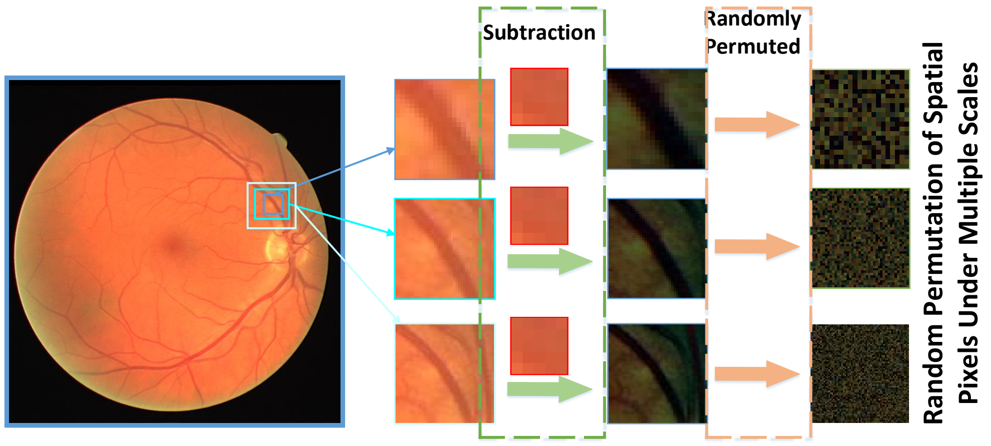


Figure 4.3: The extraction of Random Permutation of Spatial Pixel (RPoSP) features under multiple scales.

dataset, there are 20 images included in the training set, with another 20 images contained in the testing set. For the methods compared, all the 20 images in the training set are used for training the network. In contrast, since the proposed approach extracts training instances at a pixel-level, many training instances can be captured within one image. Considering this, our network is trained with 15 images, accounting for the limitations on our computational resources.

During the experiments, several metrics are used for evaluation, including Acc, Se, Sp, DSC and MCC. The definitions of these metrics are shown as follows:

$$
\begin{aligned}
Acc &= \frac{TP+TN}{n}, Se = \frac{TP}{TP+FN}, Sp = \frac{TN}{TN+FP},\\
DSC &= \frac{2TP}{FP+FN+2TP},\\
MCC &= \frac{(TP \times TN) - (FP \times TN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}
\end{aligned}
$$

where TP and FP are True Positive and False Positive. Here, positive refers to a vessel, while negative represents background. True denotes that the result of this detection is correct, while False means otherwise. Thus, TP means that the result of the detection is a vessel as well as being the ground-truth.

The quantitative comparisons between the proposed approach and state-

Table 4.2: Quantitative comparison between the proposed approach and other state-of-the-art methods on the DRIVE [122] dataset.

Methods	Acc	Se	Sp	DSC	MCC
Luo et al. [95]	0.95	0.75	-	-	-
Dasgupta et al. [29]	0.95	0.75	-	-	-
Fu et al. [40]	0.95	0.76	-	-	-
Vega et al. [132]	0.94	0.74	0.96	0.69	0.66
Wang et al. [134]	0.95	0.74	0.98	-	-
Fraz et al. [39]	0.95	0.74	0.98	-	-
Proposed approach	0.95	0.76	0.97	0.78	0.51

of-the-art methods are shown in Table 4.2. In addition, since the qualitative results of these compared methods are not available, only the qualitative results of the proposed approach are shown in Fig. 4.4. As shown in Table 4.2, the proposed approach achieves promising results compared to other state-of-the-art methods. In particular, the proposed approach achieves the highest scores in Se and Acc, which are considered as the completeness and the accurateness of the vessel mask generated.

There are some disadvantages to the proposed approach. Although the proposed approach achieves good scores in the Acc metric, which is considered as the accuracy, there are still several noisy points that can be seen in Fig. 4.4. These noisy points are a result of the random permutation utilized in the proposed approach. Since the entries of the RPoSP features are randomly permutated during every training epoch, it is possible that the RPoSP features fed into the network during the testing cases are never shown in the training procedure. In this condition, the network may falsely generate some random noise. In addition, the computational cost is another disadvantage of the proposed approach. Since the proposed approach is devised a pixel-wise classification network, every pixel of an image need to be classified by the network, which is time-consuming.

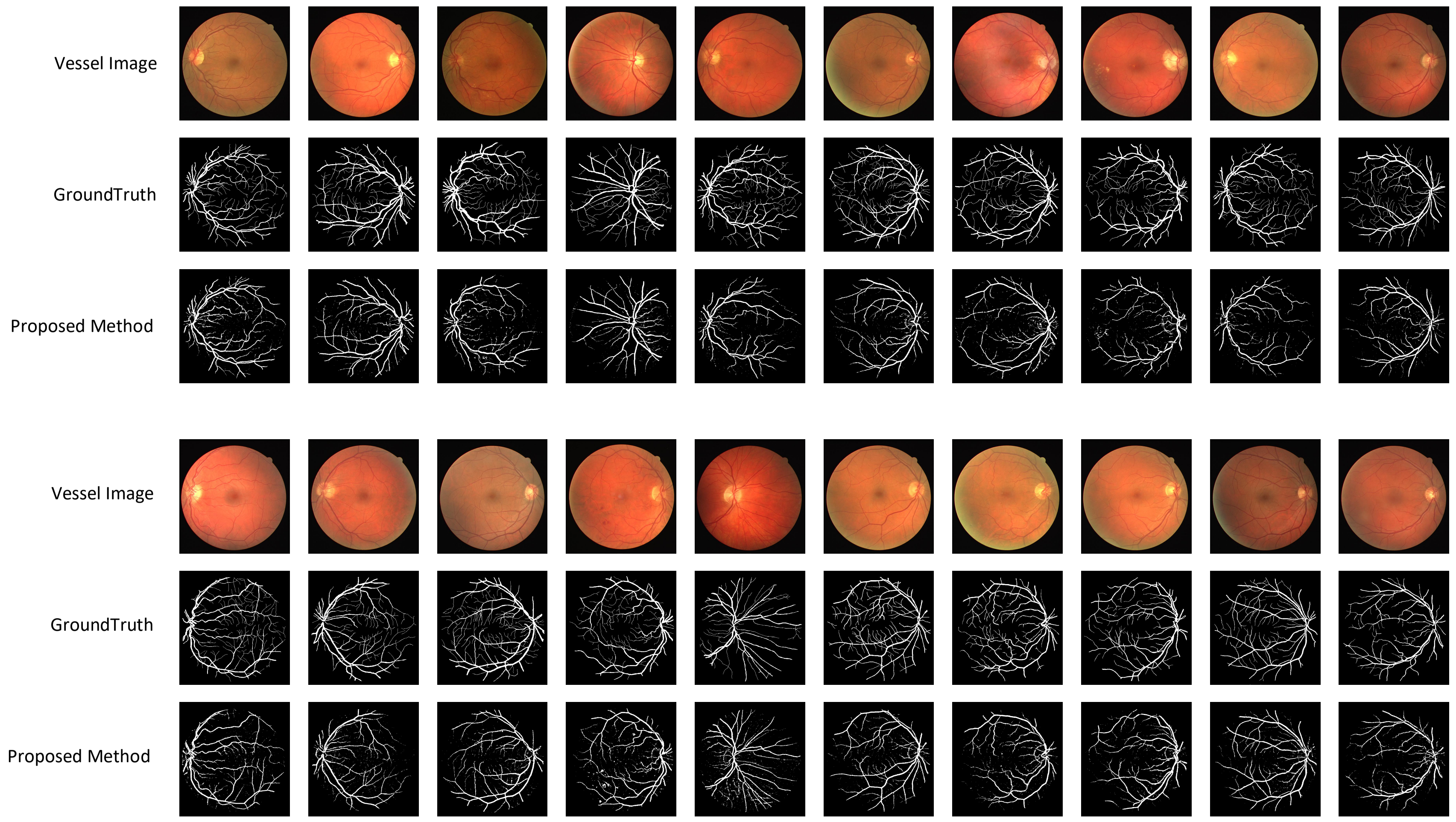


Figure 4.4: Qualitative results of the proposed approach on the DRIVE [122] dataset.

4.2 Crack Detection

4.2.1 Introduction

A number of roads, bridges, and buildings built in the last century have already reached their intended lifetime. Cracks are one of the common defects in cement surfaces and pose a potential threat to road and building safety. For maintenance purposes, people need to inspect these structures regularly and crack detection is usually the first step. However, due to the diversity and complexity of cracks, detecting cracks manually is extremely labor-intensive. How to effectively and automatically detect cracks is an interesting problem. Recently, the DPDL method has been proposed for background subtraction [152]. This method demonstrates a few good properties such as transferability, which means that the training images and testing images can be captured from different scenes. This is suitable for crack detection applications, since there is significant diversity in cracks, and training data is limited. Therefore, in this work, we applied deep pixel distribution learning for concrete crack detection.

Unlike common deep-learning applications such as face recognition [155] or object detection [83], the ground-truth data of crack detection is quite expensive. In addition, since cracks in different environments are very diverse, it is important to test the pre-trained model on a new dataset. In this work, we proposed a multiple scales deep pixel distribution learning (MS-DPDL) model for crack detection, and evaluated it using different training and testing datasets. To improve accuracy, we used a multiple scale strategy to learn the distribution of pixels from different scales. As shown in Fig. 4.5, the proposed method was first trained on the CRACK500 [141], [147] dataset, which is a dataset with 500 crack images of size 3264 by 2448 pixels. Then, this pre-trained MS-DPDL model was tested on the Concrete Crack Images dataset [107] for a classification task. Therefore, the training and testing datasets of the proposed method were completely different. During the testing, the proposed method achieved an accuracy of 0.97525 on the entire classification dataset (of 40,000 images), which demonstrates good precision and transferability.

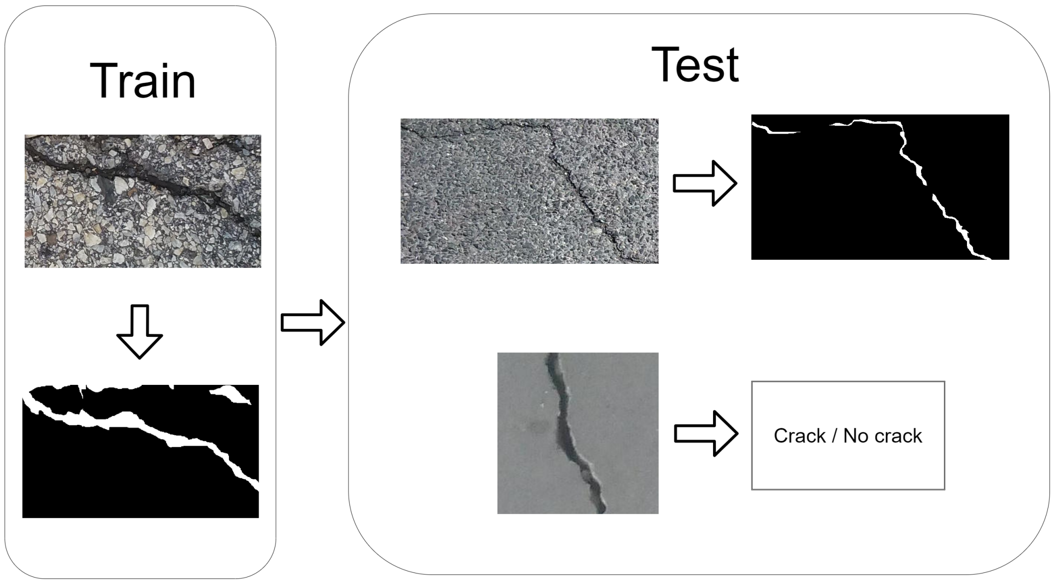


Figure 4.5: The demonstration of using pixels distribution learning for crack detection. The proposed method is first trained on CRACK500 dataset [141], [147], then the Concrete Crack Images dataset [107] is used for testing.

4.2.2 Pixel Distribution Learning for Crack Detection

The framework of the proposed approach is quite straightforward, as shown in Fig. 4.6. Crack images from multiple scales were extracted and converted into the random permutation of spatial pixel (RPoSP) features, which were discussed in the previous section. Then, the RPoSP features from different scales were input into convolutional layers whose output was combined for final classification. The final classification block consisted of a relu layer and fully connected layer followed by a logic loss such as the softmax function combined with a negative log likelihood loss.

4.2.3 Experiments

During the evaluation of the proposed method, the CRACK500 [141], [147] dataset was used. CRACK500 is a pavement crack dataset with 500 images (pixel size 3264 by 2448) taken using cell phones on the main campus of Temple University in Philadelphia, Pennsylvania, in the United States. It is currently the largest publicly accessible pavement crack dataset with pixel-wise annotation. Due to the large size, the authors cropped each image into

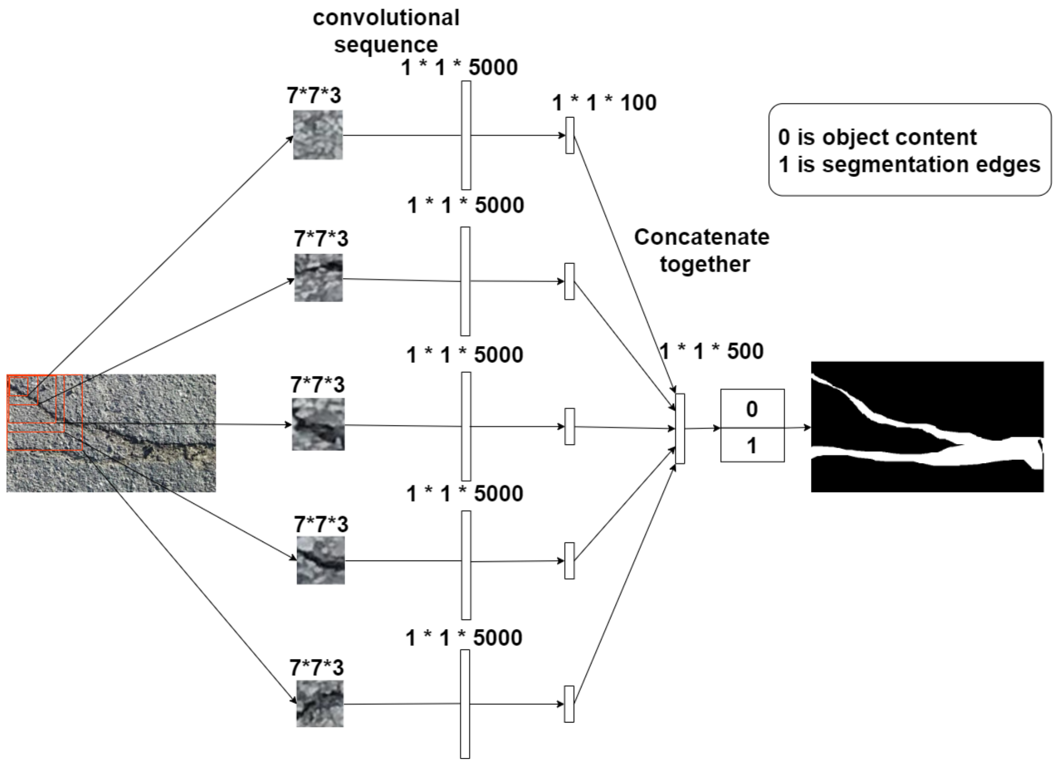


Figure 4.6: The network architecture of the pixel distribution learning network for crack detection.

16 non-overlapped image regions. There are, in total, 3368 cropped images. Fig. 4.7 shows some sample images in CRACK500. As shown, there are different kinds of pavement concrete. Some is in different colors and patterns, some cracks are obvious while some are tiny. Also, there may be deceptive factors like cigarette butts and yellow lines. This makes the dataset really complex. This dataset is used for training, and measures the performance for crack segmentation. In addition, in order to demonstrate the transferability of the proposed method, we evaluated our method in the Concrete Crack Images [107] dataset, which is proposed for crack classification. The dataset contains concrete images both with and without cracks. The data was collected from various METU campus buildings. For image classification, it is divided into two types of crack images: positive and negative. The sample images in the two classes are shown in Fig. 4.8 and Fig. 4.9 respectively. Each class has 20k images with a total of 40k images of size 227 by 227 pixels with RGB channels. The dataset was generated from 458 high-resolution images (4032

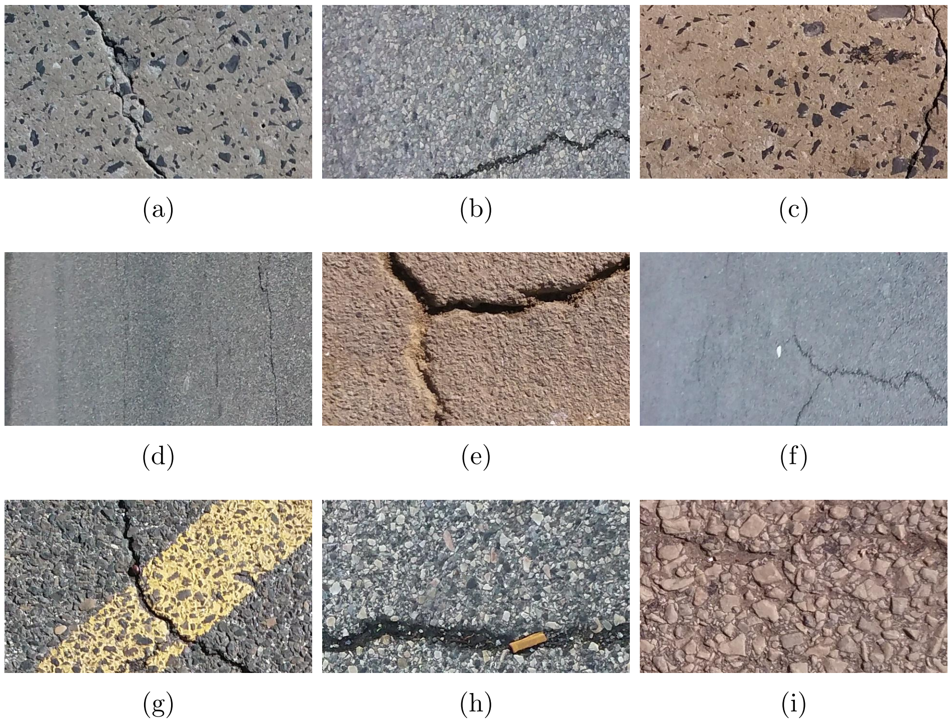


Figure 4.7: Examples of cropped images in CRACK500 dataset.

by 3024 pixels) with the method proposed by Zhang et al. [147].

Evaluation metrics

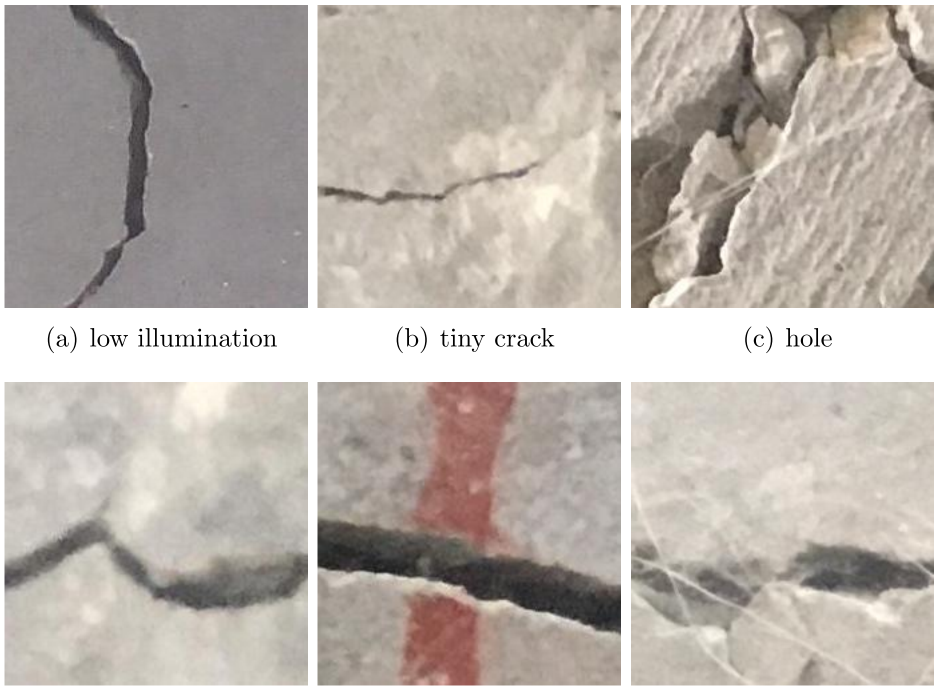
To validate the quality of our method result, we use four measures: recall[38], precision, f-measure and IoU for crack segmentation. These measurements are in the range from 0 to 1, a higher value means a better performance.

We use recall, precision, f-measure and accuracy to evaluate crack classification, which are mathematically shown as follows:

$$precision = \frac{n_{\mathrm{tp}}}{n_{\mathrm{tp}} + n_{\mathrm{fp}}} \qquad recall = \frac{n_{\mathrm{tp}}}{n_{\mathrm{tp}} + n_{\mathrm{fn}}}$$

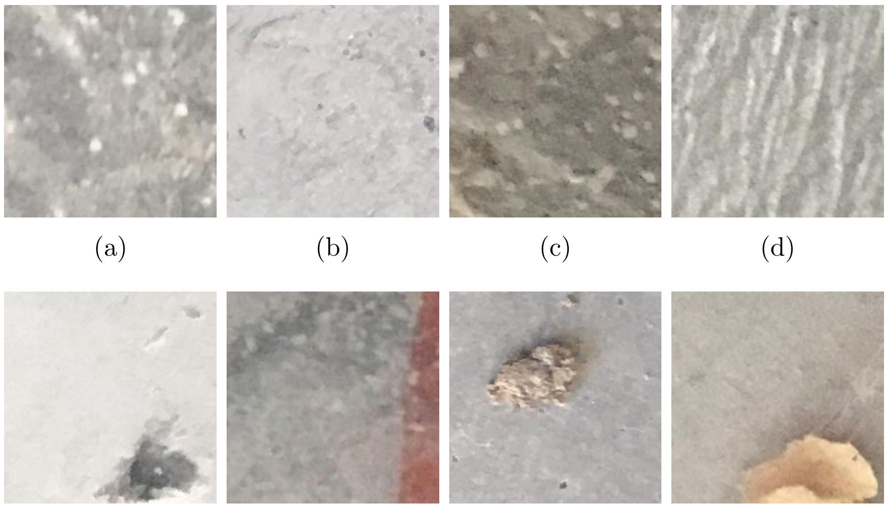
$$accuracy = \frac{n_{\mathrm{tp}} + n_{\mathrm{tn}}}{n_{\mathrm{tp}} + n_{\mathrm{tn}} + n_{\mathrm{fp}} + n_{\mathrm{fn}}}$$

$$\mathrm{F}_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} IoU = \frac{\text{target} \cap \text{prediction}}{\text{target} \cup \text{prediction}}$$


(a) low illumination (b) tiny crack (c) hole

(d) blurred (e) red line (f) dust

Figure 4.8: Examples of cracked images.


(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.9: Examples of non-cracked images.

where n_{tp}, n_{fp}, n_{tn} and n_{fn} represent the true positive, false positive, true negative and false negative respectively. In particular, positive means the

Table 4.3: A quantitative comparison of between the propose approach and previous methods

Measurements	Recall	Precision	Fm	Accuracy
Proposed method	**0.9916**	**0.9918**	**0.9916**	**0.9920**
CNN(Sitara) [119]	0.94	0.95	0.94	0.99
VGG16 [119]	0.92	0.93	0.92	0.96
VGG19 [119]	0.73	0.80	0.76	0.81
Inception ResNet [119]	0.93	0.93	0.93	0.98
SVM [66] [78]	0.7333	0.6875	0.7096	0.7187
CNN [66] [130]	0.7802	0.8875	0.8304	0.8187
FCN(Manjurul) [66]	0.941	0.913	0.927	0.928
CNN-AT(Rui) [38]	0.9992	0.9992	0.9992	0.9992

output of network is crack, and true means this output is correct.

Comparison with Previous Work

The proposed method was compared with FPHBN [141], CrackForest [115], and DPDL [152]. The results of the compared algorithms were captured by the implementation provided by the authors. In particular, the FPHBN [141] was trained with 1,896 images on the CRACK500 dataset for 12,000 iterations. CrackForest was trained on the same configuration. The proposed approach was trained with 200 images for 200 epochs. During training, 200 images were used and 200 was given as the maximum epoch number. Fig. 4.10 shows the visualization comparison of some sample cases. We performed better than CrackForest and DPDL. And even though the proposed approach was trained on far fewer images than FPHBN, we achieved similar results.

Crack classification Results

In order to demonstrate the transferability of the proposed approach, we used the pre-trained model of the proposed approach (trained on 20 images for 80 epochs) to do the crack classification on the Concrete Crack Images for Classification dataset [107]. A threshold $K = 40$ was used during all experiments. When the number of pixels labelled as 1(crack) in an image is greater than K, we say there is crack in the image. Otherwise the image is a non-crack.

Table 4.3 shows the quantitative comparisons between the proposed ap-

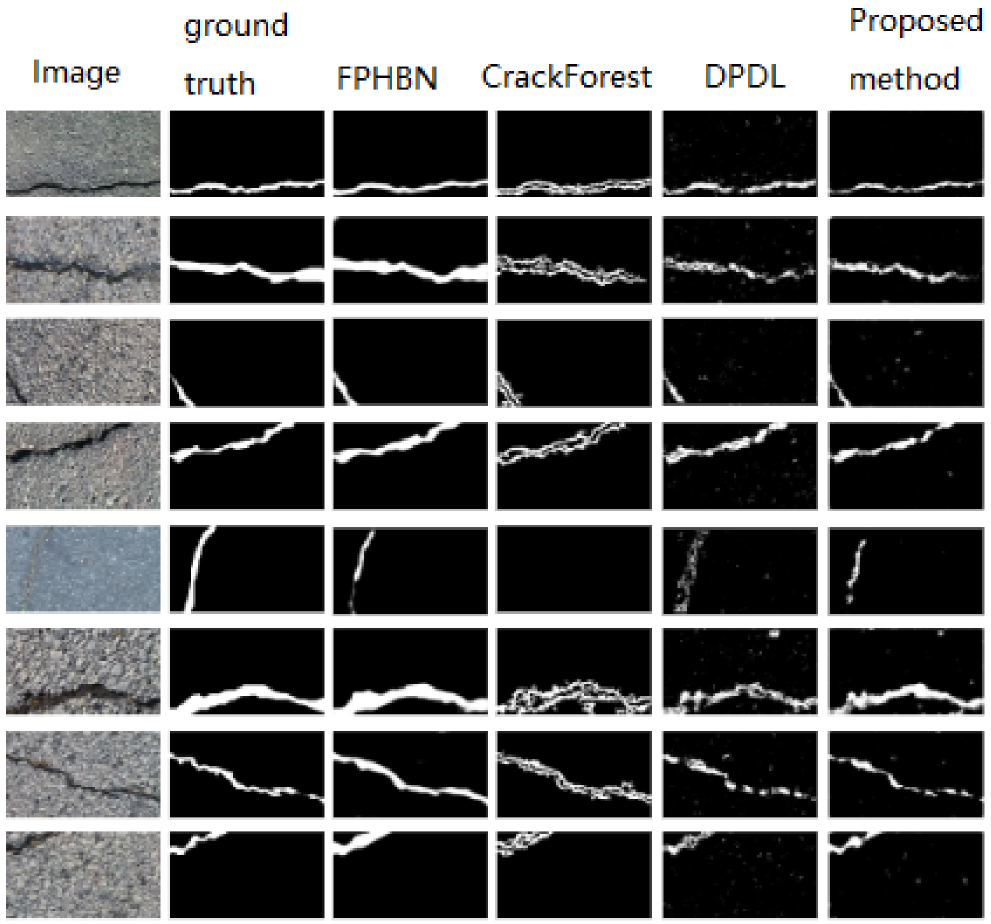


Figure 4.10: Results of compared methods on different kind of concrete surfaces

proach and state-of-the-art methods. There are 20k cracked images and 20k non-cracked images in that dataset. To compare with Rui's [38] model, which tests only 10k images from the dataset, we used stratified sampling to select 5k from the cracked images and 5k from the non-cracked images. In contrast, Rui [38] trained their net on 30k images (15k positive and 15k negative) and tested on the remaining 10k images. Sheerin's network [119] trained on 32k images(16k positive and 16k negative) and tested the remaining 8k images. Sheerin's work shows results for VGG16, VGG19 and Inception ResNet as well. Manjurul [66] used 24k images(12k positive and 12k negative) for training and validation, and tested on the remaining 16k images. As shown in Table 4.3, the proposed approach achieved 0.992 overall accuracy over the

10k sampled test images. It should be noted that the proposed approach was trained on CRACK500 dataset only. We used the trained model of the proposed approach, which was trained only on the CRACK500 dataset.

Chapter 5

Universal Background Subtraction based on Arithmetic Distribution Neural Network

Abstract

In this chapter, we propose a universal background subtraction framework based on the Arithmetic Distribution Neural Network (ADNN) for learning the distributions of temporal pixels. In our ADNN model, the arithmetic distribution operations are utilized to introduce the arithmetic distribution layers, including the product distribution layer and the sum distribution layer. Furthermore, in order to improve the accuracy of the proposed approach, an improved Bayesian refinement model based on neighboring information, with a GPU implementation, is incorporated. In the forward pass and backpropagation of the proposed arithmetic distribution layers, histograms are considered as probability density functions rather than matrices. Thus, the proposed approach is able to utilize the probability information of the histogram and achieve promising results with a very simple architecture compared to traditional convolutional neural networks. Evaluations using standard benchmarks demonstrate the superiority of the proposed approach compared to state-of-the-art traditional and deep learning methods. To the best of our knowledge, this is the first method to propose network layers based on arithmetic distribution operations for learning distributions during background subtraction.

5.1 Introduction

Background subtraction is a fundamental research topic in computer vision, which has attracted increasing attention during a period of explosive growth in video streaming. Recently, several sophisticated models based on deep learning networks have achieved excellent performance. Unfortunately, to the best of our knowledge, there are still a few challenges that limit the use of deep learning networks in real applications of background subtraction. First, such algorithms usually require a large number of ground-truth frames for training; however, creating ground-truth frames is quite expensive since every pixel of each frame has to be labelled. Second, several excellent networks for background subtraction perform poorly for unseen videos, because of their dependence on the scene information in training videos. Last but not least, various networks are trained with different videos and the parameters of networks for different testing videos are thus different. Therefore, there is no single well-trained network that can be applied for all testing videos. In order to address these challenges, a universal background subtraction method based on the Arithmetic Distribution Neural Network (ADNN) is proposed.

In background subtraction, pixels are classified as foreground or background based on comparisons with their historical counterparts. Thus, previous approaches captured a background image to represent the historical observations of pixels. Several recent methods used deep learning networks to learn the background representation for subtraction. However, such networks usually require a large number of ground-truth frames to learn the background representation and need to train a particular network for every video, to deal with the diversity in scene information for different videos. In addition, although a few excellent research (e.g., FgSN [86]) have achieved almost perfect results, their performance declines when an unseen video is introduced for segmentation. Background subtraction is essentially a classification of temporal pixels. The distributions of comparisons between temporal pixels are useful features that can be directly input into the network for classification. Therefore, in this work, we focus on learning the distribution of comparisons

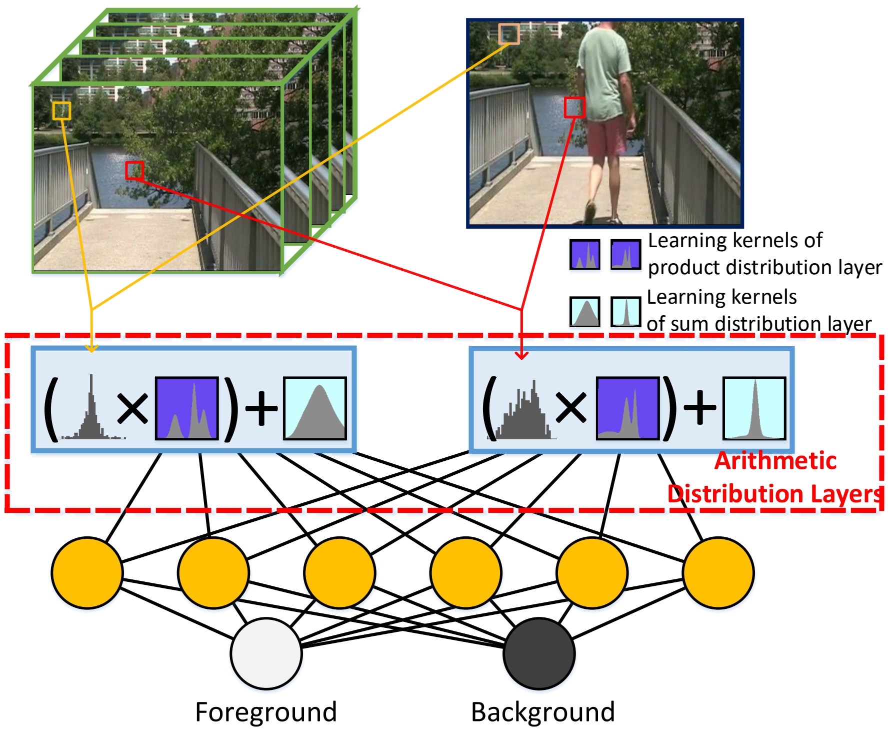

Learning kernels of product distribution layer
Learning kernels of sum distribution layer
Arithmetic Distribution Layers
Foreground
Background

Figure 5.1: Illustration of the arithmetic distribution neural network for background subtraction. Histograms of subtractions between the current observations and their historical counterparts in pixels are input into the arithmetic distribution layers, containing the product and sum distribution layers for distribution learning. In particular, the learning kernels of arithmetic distributions layers are also distributions described by histograms. A classification architecture is then attached to label the pixels according to the output of these layers.

to classify pixels into foreground or background, based on a new Arithmetic Distribution Neural Network (ADNN), as shown in Fig. 5.1.

The architecture of the proposed network is straightforward. Distributions described by histograms of subtractions between pixels' current observations and their historical counterparts are used as the input of the arithmetic distribution neural network. In particular, we propose the arithmetic distribution layers including the product and sum distribution layers as the first block of the network for learning distributions. A classification block is attached to label pixels according to the output of the arithmetic distribution layers. The architecture of the classification block is kept as simple as possible, with only one convolutional layer, one rectified linear unit layer, and one fully connected layer, to demonstrate that the good performance of the proposed approach comes from the proposed arithmetic distribution layers. Unfortunately, since pixels are classified independently, the proposed network is sensitive to noisy points that can be handled by neighboring information. An improved Bayesian refinement model, with a GPU implementation, is thus proposed for noise compensation. By utilizing the arithmetic distribution layers, histograms are considered as probability density functions, with the probability information being utilized. This helps the proposed approach achieve better distribution learning ability compared to even convolutional neural networks. Furthermore, since the histograms of temporal pixels are pixel-wise features, a large number of training instances can be captured. Thus, the proposed approach requires fewer than 1% of the ground truth frames during training. Finally, since the distribution information of temporal pixels is independent of scene information, the proposed network does not rely too much on the scenes where training frames are captured. Our ADNN can be trained with video frames obtained from different scenes, and it is valid even when no frame from the scenes of the testing videos is included in the training set. In addition, the independence of distribution information allows us to train only one network for all seen and unseen videos. The main contributions are shown as follows:

- We propose the Arithmetic Distribution Neural Network (ADNN) for background subtraction, utilizing the product distribution layer and the sum distribution layer.

- An improved Bayesian refinement model, with a GPU implementation, is proposed to improve the accuracy of our approach. In particular, an approximation of the Gaussian function is utilized to compute the correlation between neighboring pixels.

- Comprehensive experiments are conducted to evaluate the proposed approach, including: a) comparisons between the proposed ADNN and traditional convolutional neural networks on real data, as shown in Section 5.4.2; b) an ablation study of the proposed arithmetic distribution layers; c) a comprehensive comparison between the proposed approach and state-of-the-art methods including traditional and deep learning approaches on standard benchmarks, as shown in Section 5.4.3.

5.2 Arithmetic Distribution Layers

In this section, the mathematical details of the proposed arithmetic distribution layers are discussed. To the best of our knowledge, distributions have to be converted to histograms for convolutions in which the matrix arithmetic operations are used; and, all the objects involved in the operations are considered as vectors. Under this condition, the correlation between the entries of a histogram as well as their probability information are ignored. Essentially, histograms can be considered as the discrete approximation of the probability density functions that describe the distributions of the observed values of random variables. When a histogram of a distribution is input into a network for classification, it can be considered as a classification of random variables that have the input distributions. Based on this insight, we assume that the arithmetic distribution operations [120] are better than matrix arithmetic operations for distribution analysis, because histograms are considered as distributions rather than vectors during arithmetic distribution operations. Thus, a new type of network layers named arithmetic distribution layers, which contain the product and sum distribution layers, is proposed as a better substitute for the convolution layers for distribution classification. During the forward pass of the proposed arithmetic distribution layers, the input distributions are computed with the distributions in the learning kernels to generate the output distributions. In contrast to the backpropagation of the arithmetic distribution layers, the gradient of the distributions in the learning kernels with respect to the network output is computed to update the learning kernels. In particular, all these distributions are described by histograms and all computations are based on the arithmetic distribution operations in the proposed arithmetic distribution layers.

Notation: Before discussing the mathematical formulae of the forward pass and backpropagation of the proposed arithmetic distribution layers, the notation used throughout the rest of this section is first introduced. Let X and Z denote the random variables following the distributions of the input and output of the proposed arithmetic distribution layers, respectively. Let W and B denote the random variables following the distributions in the learning kernels of the product distribution layer and sum distribution layer, respectively. Let $f_X(x)$, $f_W(w)$, $f_B(b)$ and $f_Z(z)$ denote the probability density functions of random variables X, W, B and Z, respectively, where x, y, b and z denote the observed values of X, Y, B and Z, respectively. Let $\vec{x}$, $\vec{w}$, $\vec{b}$ and $\vec{z}$ denote the histograms used to describe $f_X(x)$, $f_W(w)$, $f_B(b)$ and $f_Z(z)$, respectively. In particular, x_n, w_i, b_k and z_j are the entries of histograms $\vec{x}$, $\vec{w}$, $\vec{b}$ and $\vec{z}$, respectively, where n, i, k, j are indices. $loss$ is the final scalar output of the network using the arithmetic distribution layers. During training, we want to minimize the value of $loss$ to approach 0 if possible. $\delta S \equiv \frac{\partial loss}{\partial S}$ is the gradient of variable S with the respect to $loss$. S can be the entries of histograms in the learning kernels, such as w_i or b_k.

The product distribution layer is used to compute the distribution of the product of random variables X and W having distributions $f_X(x)$ and $f_W(w)$, respectively. The input of the product distribution layer is a histogram describing $f_X(x)$. Then, the histogram of the learning kernel in the layer is used to describe $f_W(w)$. Finally, the output of the layer is a histogram of $f_Z(z)$, which is the probability density function of the random variable $Z = XW$. In order to implement the product distribution layer, the expressions for $f_Z(z)$ and the gradient of $f_W(w)$ must be obtained for the forward pass and the backpropagation, respectively. This is discussed in the remaining part of this section. The forward pass of the product distribution layer is the procedure to compute $f_Z(z)$ by the product of $f_X(x)$ and $f_W(w)$. In order to capture the expression for $f_Z(z)$, the definition of the cumulative distribution function of

Z is proposed first, as shown below:

$$\begin{aligned}
F_Z(z) &\overset{def}{=} \mathbb{P}(Z \le z) = \mathbb{P}(XW \le z) \\
&= \mathbb{P}(XW \le z, W \ge 0^+) + \mathbb{P}(XW \le z, W \le 0^-) \\
&= \mathbb{P}(X \le \frac{z}{W}, W \ge 0^+) + \mathbb{P}(X \ge \frac{z}{W}, W \le 0^-) \\
\because\ & XW \le z, W \in [-\infty\ 0^-] \cup [0^+\ \infty] \\
&\Rightarrow X \le \frac{z}{W}, W \ge 0^+ \ or\ X \ge \frac{z}{W}, W \le 0^-
\end{aligned} \tag{5.1}$$

where $F_Z(z)$ is the cumulative distribution function of the random variable Z. $\mathbb{P}$ is a cumulative distribution under a particular condition. Next, assuming X, W and Z are between negative infinity and positive infinity, the expression of $F_Z(z)$ is converted into an expression following the cumulative distribution function. Mathematically, this can be shown as:

$$\begin{aligned}
F_Z(z) &= \int_{0^+}^{\infty} f_W(w) \int_{-\infty}^{\frac{z}{w}} f_X(x) dx dw + \int_{-\infty}^{0^-} f_W(w) \int_{\frac{z}{w}}^{\infty} f_X(x) dx dw \\
\because \quad \mathbb{P}(X \le \frac{z}{w}, W \ge 0^+) &= \int_{0^+}^{\infty} f_W(w) \int_{-\infty}^{\frac{z}{w}} f_X(x) dx dw \\
\mathbb{P}(X \ge \frac{z}{w}, W \le 0^-) &= \int_{-\infty}^{0^-} f_W(w) \int_{\frac{z}{w}}^{\infty} f_X(x) dx dw,
\end{aligned} \tag{5.2}$$

where dx and dw are the delta of x and w respectively. Then, the formula of $f_Z(z)$ can be obtained by the derivative of the cumulative distribution function $F_Z(z)$ with respect to z. However, since z is under the integral sign, the Leibniz integral rule is applied. In calculus, the Leibniz integral rule is used for differentiating under the integral sign. For example, the derivative of $\int_{a(z)}^{b(z)} f(z,x)dx$ with respect to z, where $-\infty < a(z), b(z) < \infty$, can be expressed as:

$$\begin{aligned}
\frac{d}{dz}\left(\int_{a(z)}^{b(z)} f(z,x)dx\right) &= f(z, b(z)) \cdot \frac{d}{dz} b(z) \\
&\quad - f(z, a(z)) \cdot \frac{d}{dz} a(z) + \int_{a(z)}^{b(z)} \frac{\partial}{\partial z} f(z,x) dx.
\end{aligned} \tag{5.3}$$

Thus, with the help of the Leibniz integral rule, the expression for $f_Z(z)$ can be obtained as the derivative of the cumulative distribution function $F_Z(z)$

with respect to z. This is expressed as:

$$\begin{aligned}
f_Z(z) &= \frac{d(F_Z(z))}{dz} \\
&= d\left(\int_{0^+}^{\infty} f_W(w) \int_{-\infty}^{\frac{z}{w}} f_X(x)dxdw + \int_{-\infty}^{0^-} f_W(w) \int_{\frac{z}{w}}^{\infty} f_X(x)dxdw\right)/dz \\
&= \int_{0^+}^{\infty} f_W(w) \left[\frac{d\int_{-\infty}^{\frac{z}{w}} f_X(x)dx}{dz}\right] dw + \int_{-\infty}^{0^-} f_W(w) \left[\frac{d\int_{\frac{z}{w}}^{\infty} f_X(x)dx}{dz}\right] dw \\
&= \int_{0^+}^{\infty} f_W(w) f_X(\frac{z}{w}) \frac{1}{w} dw - \int_{-\infty}^{0^-} f_W(w) f_X(\frac{z}{w}) \frac{1}{w} dw \\
&= \int_{0^+}^{\infty} f_W(w) f_X(\frac{z}{w}) \frac{1}{w} dw + \int_{-\infty}^{0^-} f_W(w) f_X(\frac{z}{w}) \frac{1}{-w} dw \\
&= \int_{0^+}^{\infty} f_W(w) f_X(\frac{z}{w}) \frac{1}{|w|} dw + \int_{-\infty}^{0^-} f_W(w) f_X(\frac{z}{w}) \frac{1}{|w|} dw \\
&= \int_{-\infty}^{\infty} f_W(w) f_X(\frac{z}{w}) \frac{1}{|w|} dw \\
&\because \frac{d\int_{-\infty}^{\frac{z}{W}} f_X(x)dx}{dz} = f_X(\frac{z}{w}) \frac{d}{dz}(\frac{z}{w}) - f(-\infty)\frac{d}{dz}(-\infty) \\
&\qquad + \int_{-\infty}^{\frac{z}{W}} \frac{\partial}{\partial z}(f_X(x)dx) = f_X(\frac{z}{w})\frac{1}{w} - 0 + 0 \\
&\quad \frac{d\int_{\frac{z}{W}}^{\infty} f_X(x)dx}{dz} = f_X(\infty)\frac{d}{dz}(\infty) - f_X(\frac{z}{w})\frac{d}{dz}(\frac{z}{w}) \\
&\qquad + \int_{\frac{z}{W}}^{\infty} \frac{\partial}{\partial z}(f_X(x)dx) = 0 - f_X(\frac{z}{w})\frac{1}{w} + 0 \\
&\quad |w| = -w \ if \ w \in [-\infty \ 0^-], \ |w| = w, \ if \ w \in [0^+ \ \infty] \\
\Rightarrow z_j &= \sum_{i=-\infty}^{\infty} w_i f_X(\frac{z_j}{i}) \frac{1}{|i|} \cdot 1 \quad \because dw = 1, \ f_W(i) = w_i,
\end{aligned} \tag{5.4}$$

where w_i and z_j are the entries of histograms $\vec{w}$ and $\vec{z}$ that are used to describe $f_W(w)$ and $f_Z(z)$, respectively. The formula for the forward pass of the product distribution layer is thus derived. Then, the gradient of $f_W(w)$, which is used to update w_i during backpropagation, is obtained by partial derivatives and the chain rule. Mathematically:

$$\begin{aligned}
\frac{\partial loss}{\partial w_i} &= \frac{\partial loss}{\partial Z} \cdot \frac{\partial Z}{\partial w_i} = \sum_{j=-\infty}^{\infty} \frac{\partial loss}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_i} \\
&= \sum_{j=-\infty}^{\infty} \frac{\partial loss}{\partial z_j} \cdot \frac{\partial \left(\sum\limits_{i=-\infty}^{\infty} w_i f_X(\frac{z_j}{i}) \frac{1}{|i|} \right)}{\partial w_i} \\
&= \sum_{j=-\infty}^{\infty} \delta z_j f_X(\frac{z_j}{i}) \frac{1}{|i|} \\
\Rightarrow \delta w_i &= \sum_{j=-\infty}^{\infty} \delta z_j f_X(\frac{z_j}{i}) \frac{1}{|i|},
\end{aligned} \tag{5.5}$$

where δw_i and δz_j are the gradients of entries of histograms of $f_W(w)$ and $f_Z(z)$ respectively, i and j are indices. *loss* is the final scalar output of the network having product distribution layers, which is also the output of loss functions such as Mean Squared Error or Cross-entropy Loss during training. This way, the formula for backpropagation of the proposed product distribution layer is derived.

Similarly, the sum distribution layer is used to compute the distribution of the sum of random variables X and B, which are described by $f_X(x)$ and $f_B(b)$, respectively. Similar to the product distribution layer, $f_X(x)$ and $f_B(b)$ are represented by histograms as well. Utilizing the same mathematical procedure as the product distribution layer, the expression of the probability density function of the sum $Z = X + B$ of X and B is obtained. Mathematically:

$$\begin{aligned}
f_Z(z) &= \int_{-\infty}^{\infty} f_B(b) f_X(z-b) db \\
\Rightarrow z_j &= \sum_{i=-\infty}^{\infty} b_i f_X(z_j - i) \cdot 1 \quad \because db = 1,\ f_B(i) = b_i,
\end{aligned} \tag{5.6}$$

where b_i and z_j are the entries of histograms utilized to describe $f_B(b)$ and $f_Z(z)$ corresponding to random variables B and Z, respectively. i and j are indices. Also, the formula for backpropagation is:

$$
\begin{aligned}
\frac{\partial loss}{\partial b_k} &= \frac{\partial loss}{\partial Z} \cdot \frac{\partial Z}{\partial b_k} = \sum_{j=-\infty}^{\infty} \frac{\partial loss}{z_j} \cdot \frac{\partial z_j}{\partial b_k} \\
&= \sum_{j=-\infty}^{\infty} \frac{\partial loss}{z_j} \cdot \frac{\partial(\sum_{i=-\infty}^{\infty} b_i f_X(z_j - i))}{\partial b_k} \\
&= \sum_{j=-\infty}^{\infty} \delta z_j f_X(z_j - k) \\
\Rightarrow \delta b_k &= \sum_{j=-\infty}^{\infty} \delta z_j f_X(z_j - k),
\end{aligned}
\tag{5.7}
$$

where δb_k and δz_j are the gradients of entries in histograms corresponding to $f_B(b)$ and $f_Z(z)$ respectively, and k and j are indices. $loss$ is the final output of the network using the sum distribution layer.

With the help of Eqn. 5.4–5.7, the forward pass and the backpropagation of arithmetic distribution layers can be easily implemented in Pytorch [3]. In particular, the gradient of the learning kernels of the arithmetic distribution layers is computed and input into the "Autograd package" of PyTorch [3] for backpropagation. In our implementation of the arithmetic distribution layers, since the distributions of temporal pixels whose values have been normalized within $[0, 1]$, are used as the network input, the x coordinate of histograms is narrowed into $[-1, 1]$ with a bin interval of 0.01. This means that there are $(1 - (-1))/0.01 + 1 = 201$ bins in the histograms. This is the reason why the size of the arithmetic distribution layers shown in Table 5.1 is $3 \times 201 \times 1$, where 3 represents the RGB channels of images. Furthermore, the sum of probability values falling into bins are normalized to 1, before the histograms are used as network input. It should be noted that a larger range of the x coordinate of histograms can give us more accurate results. However, this implementation setting is enough to generate promising results, and is used for all experiments proposed in this work.

Table 5.1: Details of the proposed arithmetic distribution neural network architecture for background subtraction.

Type	Filters	Layer size	Data size
Input			B$\times$3 $\times$ 201 $\times$ 1
Product Distribution	2	3 $\times$ 201 $\times$ 1	B$\times$3 $\times$ 201 $\times$ 2
Sum Distribution	2	3 $\times$ 201 $\times$ 1	B$\times$3 $\times$ 201 $\times$ 2
Convolution	1	3 $\times$ 1 $\times$ 2	B$\times$10 $\times$ 201 $\times$ 1
Convolution	512	1 $\times$ 201 $\times$ 1	B$\times$512 $\times$ 1 $\times$ 1
Rectified linear unit			
Convolution	2	512 $\times$ 1 $\times$ 1	B$\times$2 $\times$ 1 $\times$ 1
Softmax			

B: Batch size.

5.3 Arithmetic Distribution Neural Network for Background Subtraction

Utilizing the proposed product and sum distribution layers, the arithmetic distribution neural network is devised for background subtraction. Background subtraction is a binary classification of temporal pixels; thus, the distributions of temporal pixels play an important role. In this work, the distributions of subtractions between pixels and their historical counterparts are used for classification. In particular, histograms are utilized to describe the distributions of subtractions and also directly used as the input of the proposed arithmetic distribution neural network. The network architecture is quite straightforward: histograms are first input into the product distribution layer and the sum distribution layer. Then, the outputs of these layers are combined by a convolution followed by a classification architecture which consists of a convolution, a rectified linear unit (Relu) layer, and a fully connected layer. The classification architecture is deliberately kept as simple as possible, with only 3 layers, in order to demonstrate that the good results come from the proposed arithmetic distribution layers.

The components of the proposed arithmetic distribution neural network for background subtraction are illustrated in Fig. 5.2, with details of the network architecture presented in Table 5.1, in which the first two convolutions are 1D convolution, since one of the values of the kernel size is 1. Starting with a given frame of a video, denoted as $\mathcal{I} = \{I_1, I_2, \cdots, I_T\} = \{I_t | t = [1,\ T] \cap \mathbb{N}\}$,

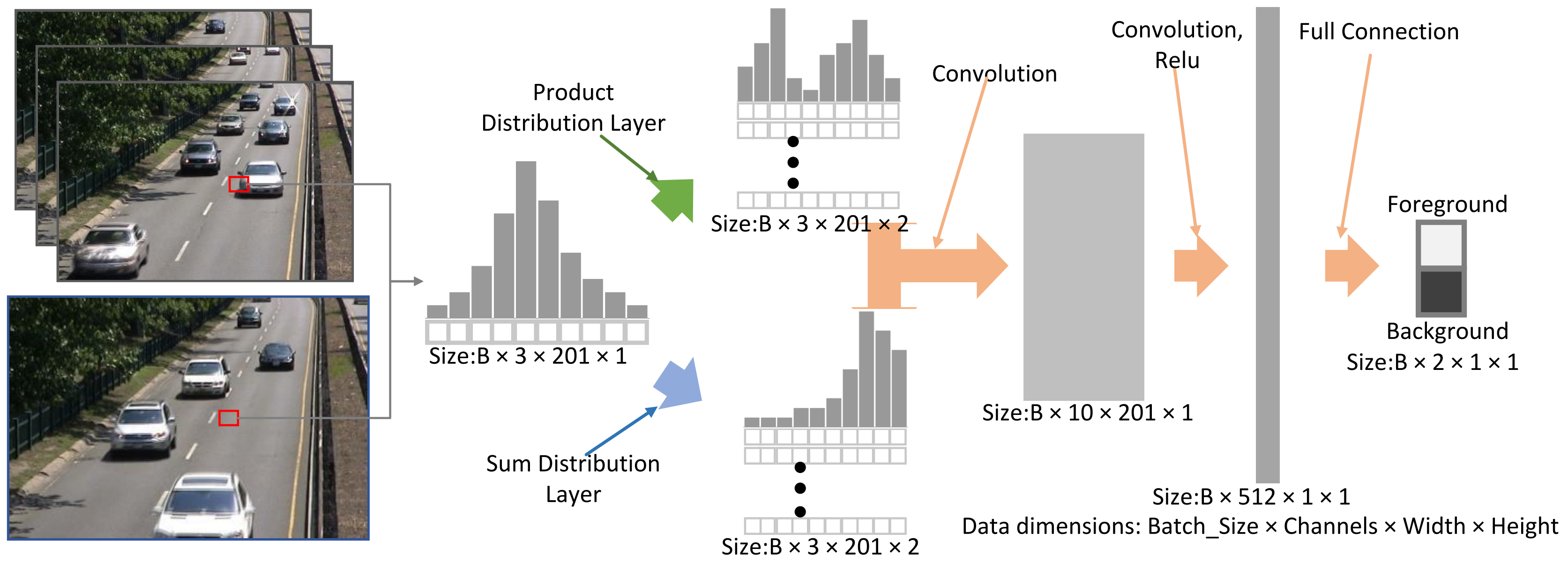


Figure 5.2: An illustration of the arithmetic distribution neural network for background subtraction. Histograms of subtractions between a pixel's current observation and its historical counterparts are used as the input to arithmetic distribution layers for learning distributions. The output histograms of arithmetic distribution layers are combined by a convolution and input into a classification architecture containing a convolution layer, a rectified linear unit (Relu) layer, and a fully connected layer for classification.

where t is the frame index, T is the number of frames, and $\mathbb{N}$ is the set of all natural numbers. To perform background subtraction for a particular pixel located at (x, y) on frame t, the histogram of subtractions between pixels' current observation and their historical counterparts is captured for classification. Mathematically:

$$H_{x,y}(n) = \sum_{i=1}^{T} (I_i(x, y) - I_t(x, y)) \cap n, \tag{5.8}$$

where $H_{x,y}$ is the histogram of subtractions, and n is the index of entries of the histogram. $I_i(x, y)$ denotes historical observations of the pixel located at (x, y), and $I_t(x, y)$ denotes its current observation. The distributions of subtractions are directly used as the input to the product and sum distribution layers for distribution learning. Then, the sum of the outputs of these two layers is used as the input of the classification architecture. Mathematically:

$$\mathcal{M}(x, y) = \mathcal{L}(\mathcal{C}(\mathcal{F}_p(H_{x,y}) + \mathcal{F}_a(H_{x,y}))), \tag{5.9}$$

where $H_{x,y}$ is the input histogram; $\mathcal{F}_p$ and $\mathcal{F}_a$ denote the product distribution and the sum distribution layer; $\mathcal{C}$ is the convolution procedure; and $\mathcal{L}$ is the classification architecture consisting of a convolution, a rectified linear unit, and a fully connected layer attached with a softmax function. In particular, the negative log likelihood loss (NLLLoss) is used to update the parameters in $\mathcal{L}$, $\mathcal{C}$, $\mathcal{F}_p$ and $\mathcal{F}_a$ during the training process. In contrast to the testing process, the arguments of the maxima (argmax) function is attached on the last layer of the classification architecture $\mathcal{L}$ to generate the label $\mathcal{M}(x, y)$ of the histogram captured from the pixel located at (x, y).

Unfortunately, the histograms utilized for classification are captured from independent pixels. Thus, the correlation between pixels is ignored. In order to improve the accuracy of the proposed approach, an improved Bayesian refinement model is introduced. For completeness, we briefly introduce the Bayesian refinement model; please check [151] for more details. In the Bayesian refinement model, the labels of pixels are re-inferred according to the correlations with their neighborhoods, and the Bayesian theory is utilized during inference. In particular, Euclidean distance is used to compute the correlation. In

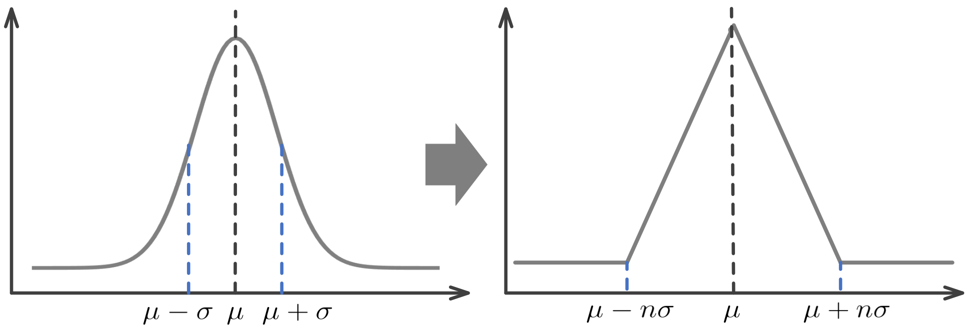


Figure 5.3: An illustration of the Gaussian approximation function which approximates using a piecewise function controlled by the parameters of the Gaussian function.

contrast, we utilize a mixture of Gaussian approximation functions to capture the correlation. This is the main difference compared to the original Bayesian refinement model. Mathematically:

$$\begin{aligned} \mathcal{F}(I(x,y),\mathcal{M}) &= \underset{a_i}{argmax}\,\frac{P(I(x,y)|a_i)P(a_i)}{P(I(x,y))} \\ &= \underset{a_i}{argmax}\,P(a_i)\sum_{k=1}^{K}\pi_k\mathcal{N}_p(v_k|\mu_{k,i},\Sigma_{k,i}) \\ &\because P(\mathcal{I}(x,y)) = N \\ &\quad \&\,\underset{x}{argmax}\,Nx = \underset{x}{argmax}\,x \\ &P(I(x,y)|a_i) = \sum_{k=1}^{K}\pi_k\mathcal{N}_p(v_k|\mu_{k,i},\Sigma_{k,i}), \end{aligned} \tag{5.10}$$

where $\mathcal{F}$ denotes the proposed improved Bayesian refinement model. $a_i \in \{0,1\}$ denotes the labels of foreground or background; $I(x,y)$ is a pixel located at (x,y); and, $P(I(x,y)|a_i)$ is the probability that the label of this pixel is a_i, which is captured though a mixture of Gaussian approximation functions $\mathcal{N}_p(v_k|\mu_{k,i}\Sigma_{k,i})$. In particular, v_k denotes the feature vector consisting of the Lab color and spatial position of the pixel $I(x,y)$, and k is the index of entries in a vector. u_k and Σ_k denote the mean and variance of features of a pixel in a local rectangular range with center at (x,y) and radius $R=4$. π_k is the weight to mix the Gaussian approximation functions $\mathcal{N}_p$ which is mathematically expressed as:

$$\mathcal{N}_p(x|\mu,\sigma) = \begin{cases} |1 + \frac{x-\mu}{n\sigma}| & |x-u| \leq n\sigma \\ 0 & otherwise \end{cases} \tag{5.11}$$

where μ and σ denote the mean and variance, and n is a user parameter. During experiments, $n = 2$ gives us the best results. As shown in Fig. 5.3, the Gaussian approximation function is actually a rough estimate of the Gaussian function. We use a piecewise function to approximate the waveform of the Gaussian function considering the computational cost. Also, it is more convenient for a GPU implementation, which significantly accelerates the refinement procedure.

Finally, the output binary mask is used in the input again to generate better results iteratively. The Bayesian refinement model is utilized to iteratively refine the foreground mask. Mathematically:

$$\mathcal{M}_n(x,y) = \mathcal{F}(I(x,y), \mathcal{M}_{n-1}), \tag{5.12}$$

where n is the iteration number and $\mathcal{M}_{n-1}$ is the binary mask from the last iteration. Using a GPU implementation, with the number of iterations set to 20, which is used in all the evaluation experiments proposed in this work, the entire refinement procedure only takes a few seconds.

The improved Bayesian refinement model (IBRM) runs much faster than the Bayesian refinement model (BRM) with almost no loss in accuracy. A comparison between them on a few frames for videos at different resolutions is shown in Table 5.2. In particular, the running time of BRM and IBRM with iteration numbers 1, 20 and 50 are presented, as well as the Fm value of their corresponding output masks after refinement. As shown in Table 5.2, when the number of iterations is 50, although the Fm value of the output mask shows obvious improvement, the run time also increases to 52s, which is too long for real applications. In contrast, IBRM needs only 3.5s of processing time, and the Fm value of the output mask is still close to the one for BRM. Actually, improved Bayesian refinement is devised for GPU implementation with the motivation of accelerating the refinement procedure. Thus, the superiority of the proposed IBRM is demonstrated. For comparisons, both BRM and IRBM

Table 5.2: Comparison between Bayesian refinement model and our improved Bayesian refinement model on frames of videos at different resolutions.

Video	Resolution	NI	Time/s		Fm value	
			BRM	IBRM	BRM	IBRM
highway	320×240	1	**1.2533**	1.4429	0.9554	**0.9612**
		20	21.1842	**2.2524**	0.9826	**0.9828**
		50	52.7545	**3.5712**	**0.9911**	0.9905
canoe	320×240	1	**1.1496**	1.4215	**0.9552**	0.9528
		20	19.9488	**2.2968**	**0.9535**	0.9482
		50	50.4685	**3.5780**	0.9534	**0.9453**
wetSnow	720×540	1	5.2601	**1.6945**	0.7252	**0.7279**
		20	93.1602	**5.8695**	**0.7731**	0.7646
		50	234.7885	**12.5385**	**0.7750**	0.7732

NI: Number of iterations BRM: Bayesian refinement model
I_BRM:Improved Bayesian refinement model.

are run on GPU devices, all data are moved into video memory (GPU memory) to guarantee the running environment of BRM and IBRM are the same.

5.4 Experiments

5.4.1 Verification of Arithmetic Distribution Layers

In this section, we verify the correctness of the proposed arithmetic distribution layers including the product distribution layer and sum distribution layer. In the experiments, synthetic data is used for verification. In particular, two continuous independent random variables X and W, which are described by two different probability density functions representing two histograms, are generated. In addition, the product $Z_p = XW$ and the sum $Z_s = X + W$ of the two variables X and W are computed for use as the target output. The values of X and W as well their product and sum are generated over one million times to capture the target histograms of Z_p and Z_s. The verification experiment is quite straightforward: the histogram of X is input into the product distribution layer and the sum distribution layer, to compute with the histogram of W' in layers to output the histograms of Z'_p and Z'_s. The output histograms are then compared with the target histograms of Z_p and Z_s to capture the gradients to train the arithmetic distribution layers. Finally, the correlation values between the output histograms and target histograms

Table 5.3: Details of the network architecture of arithmetic distribution neural networks (ADNNs) and convolutional neural networks (CNNs), used for comparison to demonstrate the superiority of the proposed approach.

CNN_1			CNN_2			CNN_3			CNN_4			Data Size
Type	Filters	Size	Type	Filters	Size	Type	Filters	Size	Type	Filters	Size	
Input												B$\times 3 \times 201 \times 1$
									Conv	8	$3 \times 1 \times 1$	
						Conv	8	$3 \times 1 \times 1$	Conv	16	$8 \times 1 \times 1$	
						Conv	16	$8 \times 1 \times 1$	Conv	32	$16 \times 201 \times 1$	
			Conv	8	$3 \times 1 \times 1$	Conv	32	$16 \times 201 \times 1$	Gelu			
			Conv	16	$8 \times 201 \times 1$	Conv	128	$32 \times 1 \times 1$	Conv	128	$32 \times 1 \times 1$	
Conv	1	$3 \times 1 \times 1$	Conv	201	$16 \times 1 \times 1$	Conv	201	$128 \times 1 \times 1$	Conv	201	$128 \times 1 \times 1$	B$\times 1 \times 201 \times 1$
Relu			Relu			Relu			Relu			
Conv	2	$1 \times 201 \times 1$	Conv	2	$1 \times 201 \times 1$	Conv	2	$1 \times 201 \times 1$	Conv	2	$1 \times 201 \times 1$	B$\times 2 \times 1 \times 1$
Softmax			Softmax			Softmax			Softmax			
NTP	405		NTP	29,370		NTP	133,290		NTP	133,290		

CNN_5			CNN_6			CNN_7			$ADNN_1$			Data Size
Type	Filters	Size	Type	Filters	Size	Type	Filters	Size	Type	Filters	Size	
Input												B$\times 3 \times 201 \times 1$
			Conv	8	$3 \times 1 \times 1$	Conv	8	$3 \times 1 \times 1$				
			Gelu			Relu						
			Conv	16	$8 \times 1 \times 1$	Conv	16	$8 \times 1 \times 1$				
Conv	8	$3 \times 1 \times 1$	Gelu			Relu						
Conv	16	$8 \times 1 \times 1$	Conv	32	$16 \times 201 \times 1$	Conv	32	$16 \times 201 \times 1$				
Conv	32	$16 \times 201 \times 1$	Gelu			Relu						
Relu			Conv	128	$32 \times 1 \times 1$	Conv	128	$32 \times 1 \times 1$	ProDis	1	$3 \times 201 \times 1$	
Conv	128	$32 \times 1 \times 1$	Gelu			Relu			SumDis	1	$3 \times 201 \times 1$	
Conv	201	$128 \times 1 \times 1$	Conv	201	$128 \times 1 \times 1$	Conv	201	$128 \times 1 \times 1$	Conv	1	$3 \times 1 \times 1$	B$\times 1 \times 201 \times 1$
Relu			Relu			Relu			Relu			
Conv	2	$1 \times 201 \times 1$	Conv	2	$1 \times 201 \times 1$	Conv	2	$1 \times 201 \times 1$	Conv	2	$1 \times 201 \times 1$	B$\times 2 \times 1 \times 1$
Softmax			Softmax			Softmax			Softmax			
NTP	133.290		NTP	133,290		NTP	133,290		NTP	1,611		

ProDis: product distribution layer SumDis: sum distribution layer Conv: Convolution Gelu: Gaussian Error Linear Unit Relu: Rectified Linear Unit NTP: Number of total parameters B: batch size

Table 5.4: Details of the network architecture of arithmetic distribution neural networks (ADNNs) and convolutional neural networks (CNNs), used for comparison to demonstrate the superiority of the proposed approach.

CNN_8			CNN_9			CNN_{10}			CNN_{11}			Data Size
Type	Filters	Size	Type	Filters	Size	Type	Filters	Size	Type	Filters	Size	
Input												B×3 × 201 × 1
									Conv	256	3 × 201 × 1	
Conv	1	3 × 1 × 1				Conv	256	3 × 201 × 1	Conv	512	256 × 1 × 1	
Conv	201	201 × 1 × 1	Conv	128	3 × 201 × 1	Conv	512	256 × 1 × 1	Relu			
Conv	201	201 × 1 × 1	Conv	256	128 × 1 × 1	Conv	1024	512 × 1 × 1	Conv	1024	512 × 1 × 1	
Conv	512	201 × 1 × 1	Conv	512	256 × 1 × 1	Conv	2048	1024 × 1 × 1	Conv	2048	1024 × 1 × 1	
Relu			Relu			Relu			Relu			
Conv	2	512 × 1 × 1	Conv	2	512 × 1 × 1	Conv	2	2048 × 1 × 1	Conv	2	2048 × 1 × 1	B×2 × 1 × 1
Softmax			Softmax			Softmax			Softmax			
NTP	184,741		NTP	242,048		NTP	2,910,976		NTP	2,910,976		
CNN_{12}			CNN_{13}			CNN_{14}			$ADNN_2$			Data Size
Type	Filters	Size	Type	Filters	Size	Type	Filters	Size	Type	Filters	Size	
Input												B×3 × 201 × 1
			Conv	256	3 × 201 × 1	Conv	256	3 × 201 × 1				
			Gelu			Relu						
Conv	256	3 × 201 × 1	Conv	512	256 × 1 × 1	Conv	512	256 × 1 × 1				
Conv	512	256 × 1 × 1	Gelu			Relu			ProDis	2	3 × 201 × 1	
Relu			Conv	1024	512 × 1 × 1	Conv	1024	512 × 1 × 1	SumDis	2	3 × 201 × 1	
Conv	1024	512 × 1 × 1	Gelu			Relu			Conv	1	3 × 1 × 2	
Conv	2048	1024 × 1 × 1	Conv	2048	1024 × 1 × 1	Conv	2048	1024 × 1 × 1	Conv	512	1 × 201 × 1	
Relu			Relu			Relu			Relu			
Conv	2	2048 × 1 × 1	Conv	2	2048 × 1 × 1	Conv	2	2048 × 1 × 1	Conv	2	512 × 1 × 1	B×2 × 1 × 1
Softmax			Softmax			Softmax			Softmax			
NTP	2,910,976		NTP	2,910,976		NTP	2,910,976		NTP	106,354		

ProDis: product distribution layer SumDis: sum distribution layer Conv: Convolution Gelu: Gaussian Error Linear Unit Relu: Rectified Linear Unit NTP: Number of total parameters B: batch size

Table 5.5: Quantitative comparison between CNNs and ADNN using Re, Pr, and Fm metrics based on real data.

Videos	CNN_1			CNN_2			CNN_3			CNN_4		
	Re	Pr	Fm	Re	Pr	Fm	Re	Pr	Fm	Re	Pr	Fm
highway	(0.9086 ,	0.8133 ,	0.8583)	(0.8732	, 0.7599	, 0.8126)	(0.9532	, 0.8017	, 0.8709)	(0.9085	, 0.8805	, 0.8943)
pedestrians	(0.9370 ,	0.9404 ,	0.9387)	(0.9536	, 0.9062	, 0.9293)	(0.9474	, 0.8942	, 0.9201)	(0.8963	, 0.8518	, 0.8735)
fountain01	(0.1404 ,	0.1988 ,	0.1646)	(0.6434	, 0.4435	, 0.5250)	(0.5944	, 0.5001	, 0.5432)	(0.6538	, 0.4665	, 0.5445)
canoe	(0.8134 ,	0.8398 ,	0.8264)	(0.9464	, 0.9532	, 0.9498)	(0.9439	, 0.9663i	, 0.9550)	(0.9422	, 0.9560	, 0.9490)
fountain02	(0.7289 ,	0.8812 ,	0.7978)	(0.7850	, 0.8584	, 0.8201)	(0.7670	, 0.9186	, 0.8360)	(0.7391	, 0.9505	, 0.8316)
peopleInShade	(0.9893 ,	0.2478 ,	0.3963)	(0.9054	, 0.4649	, 0.6144)	(0.8941	, 0.4758	, **0.6211**)	(0.8921	, 0.3713	, 0.5243)
backdoor	(0.7665 ,	0.5195 ,	0.6192)	(0.9275	, 0.2987	, 0.4519)	(0.9033	, 0.3420	, 0.4962)	(0.9302	, 0.2640	, 0.4113)
traffic	(0.6881 ,	0.9008 ,	0.7802)	(0.7009	, 0.9541	, 0.8081)	(0.7167	, 0.9499	, **0.8170**)	(0.6851	, 0.9611	, 0.8000)
sidewalk	(0.5490 ,	0.3675 ,	0.4403)	(0.8392	, 0.3368	, 0.4807)	(0.8314	, 0.3587	, **0.5012**)	(0.8475	, 0.3100	, 0.4539)
busStation	(0.8587 ,	0.7870 ,	0.8212)	(0.8846	, 0.8297	, 0.8563)	(0.8913	, 0.8544	, 0.8725)	(0.8769	, 0.7349	, 0.7996)
bungalows	(0.8636 ,	0.9604 ,	0.9094)	(0.8535	, 0.9607	, 0.9039)	(0.8493	, 0.9649	, 0.9034)	(0.8576	, 0.9560	, 0.9041)
library	(0.9101 ,	0.9702 ,	0.9392)	(0.9075	, 0.9701	, 0.9378)	(0.9193	, 0.9444	, 0.9444)	(0.9274	, 0.9680	, 0.9472)
Average	(0.7628 ,	0.7022 ,	0.7076)	(0.8517	, 0.7280	, 0.7575)	(0.8509	, 0.7476	, 0.7734)	(0.8464	, 0.7225	, 0.7444)
Videos	CNN_5			CNN_6			CNN_7			$ADNN_1$		
	Re	Pr	Fm	Re	Pr	Fm	Re	Pr	Fm	Re	Pr	Fm
highway	(0.9604	, 0.7476	, 0.8408)	(0.9586	, 0.8054	, 0.8753)	(0.9604	, 0.8562	, 0.9054)	(0.9603 ,	0.8697 ,	**0.9127**)
pedestrians	(0.9275	, 0.8974	, 0.9122)	(0.9432	, 0.8860	, 0.9137)	(0.9319	, 0.8585	, 0.8937)	(0.9399 ,	0.9690 ,	**0.9542**)
fountain01	(0.6178	, 0.4950	, 0.5497)	(0.6851	, 0.4128	, 0.5152)	(0.7136	, 0.5602	, **0.6277**)	(0.5953 ,	0.5311 ,	0.5613)
canoe	(0.9443	, 0.9659	, 0.9550)	(0.9442	, 0.9565	, 0.9503)	(0.9500	, 0.9707	, **0.9603**)	(0.9450 ,	0.9654 ,	0.9551)
fountain02	(0.7837	, 0.9284	, 0.8500)	(0.7937	, 0.8818	, 0.8354)	(0.7656	, 0.9148	, 0.8335)	(0.7738 ,	0.9534 ,	**0.8543**)
peopleInShade	(0.9043	, 0.3778	, 0.5329)	(0.8814	, 0.4172	, 0.5664)	(0.9135	, 0.3774	, 0.5342)	(0.9808 ,	0.4542 ,	0.6209)
backdoor	(0.9583	, 0.1473	, 0.2553)	(0.9464	, 0.1469	, 0.2544)	(0.9388	, 0.2543	, 0.4002)	(0.9134 ,	0.5555 ,	**0.6908**)
traffic	(0.6650	, 0.9648	, 0.7874)	(0.6716	, 0.9609	, 0.7906)	(0.6667	, 0.9690	, 0.7899)	(0.6927 ,	0.9670 ,	0.8072)
sidewalk	(0.8465	, 0.3441	, 0.4893)	(0.8309	, 0.3233	, 0.4655)	(0.8399	, 0.3443	, 0.4884)	(0.7851 ,	0.3437 ,	0.4781)
busStation	(0.8612	, 0.7972	, 0.8280)	(0.8896	, 0.7384	, 0.8070)	(0.8772	, 0.7636	, 0.8164)	(0.9097 ,	0.9057 ,	**0.9077**)
bungalows	(0.8543	, 0.9473	, 0.8984)	(0.8476	, 0.9610	, 0.9008)	(0.8164	, 0.9658	, 0.8848)	(0.9073 ,	0.9788 ,	**0.9417**)
library	(0.9275	, 0.9883	, **0.9570**)	(0.9080	, 0.9724	, 0.9391)	(0.8838	, 0.9733	, 0.9264)	(0.9358 ,	0.9746 ,	0.9548)
(Average	(0.8543	, 0.7168	, 0.7380)	(0.8584	, 0.7052	, 0.7345)	(0.8548	, 0.7340	, 0.7551)	(0.8616 ,	0.7890 ,	**0.8032**)

Table 5.6: Quantitative comparison between CNNs and ADNN using Re, Pr, and Fm metrics based on seen and unseen videos.

Training Sets	Testing Sets		CNN_8	CNN_9	CNN_{10}	CNN_{11}
Category Video	Category	Video	Re Pr Fm	Re Pr Fm	Re Pr Fm	Re Pr Fm
	Baseline	highway	(0.9343, 0.7418, 0.8270)	(0.9321, 0.9275, 0.9298)	(0.9341, 0.9134, 0.9237)	(0.9341, 0.9056, 0.9196)
Baseline highway	Baseline	pedestrians	(0.9574, 0.8828, 0.9186)	(0.9280, 0.8972, 0.9123)	(0.9362, 0.9070, 0.9213)	(0.9486, 0.8929, 0.9199)
Baseline pedestrians	Baseline	office	(0.9314, 0.7209, 0.8128)	(0.8751, 0.7307, 0.7964)	(0.8848, 0.7318, 0.8011)	(0.8784, 0.7377, 0.8019)
	Baseline	PETS2006	(0.9076, 0.7932, 0.8465)	(0.8872, 0.8183, 0.8514)	(0.8885, 0.7979, 0.8408)	(0.8946, 0.8051, 0.8475)
	Baseline	highway	(0.8434, 0.9077, 0.8744)	(0.9116, 0.9306, 0.9210)	(0.9212, 0.9660, 0.9430)	(0.9197, 0.9683, 0.9434)
Baseline fountain01	Dyn. Bg.	overpass	(0.8277, 0.4266, 0.5630)	(0.8616, 0.7802, 0.8189)	(0.8691, 0.8089, 0.8379)	(0.8664, 0.8169, 0.8410)
Dyn. Bg.overpass	Shadow	bungalows	(0.7638, 0.9926, 0.8633)	(0.8012, 0.9837, 0.8832)	(0.8056, 0.9831, 0.8855)	(0.8026, 0.9855, 0.8847)
Shadow bungalows	Baseline	pedestrians	(0.9341, 0.8752, **0.9037**)	(0.9034, 0.7150, 0.7982)	(0.9117, 0.7625, 0.8304)	(0.9447, 0.7912, 0.8612)
	Dyn. Bg.	canoe	(0.8811, 0.4395, 0.5865)	(0.8853, 0.6174, 0.7275)	(0.9037, 0.8155, 0.8573)	(0.9064, 0.7349, 0.8117)
	Shadow	peopleInShade	(0.9351, 0.8929, 0.9135)	(0.8956, 0.8428, 0.8684)	(0.8972, 0.8689, 0.8828)	(0.9052, 0.8650, 0.8847)
Overall			(0.8916, 0.7673, 0.8109)	(0.8881, 0.8243, 0.8507)	(0.8952, 0.8555, 0.8724)	(0.9001, 0.8503, 0.8715)
Training Sets	Testing Sets		CNN_{12}	CNN_{13}	CNN_{14}	$ADNN_2$
Category Video	Category	Video	Re Pr Fm	Re Pr Fm	Re Pr Fm	Re Pr Fm
	Baseline	highway	(0.9374, 0.8506, 0.8919)	(0.9318, 0.9235, 0.9276)	(0.9341, 0.8665, 0.8991)	(0.9303, 0.9378, **0.9340**)
Baseline highway	Baseline	pedestrians	(0.9631, 0.8880, 0.9240)	(0.9224, 0.8994, 0.9108)	(0.9276, 0.9178, 0.9227)	(0.9710, 0.9214, **0.9455**)
Baseline pedestrians	Baseline	office	(0.8888, 0.7625, 0.8208)	(0.8666, 0.7294, 0.7921)	(0.8125, 0.7320, 0.7701)	(0.9250, 0.8720, **0.8977**)
	Baseline	PETS2006	(0.9058, 0.7999, 0.8496)	(0.8811, 0.8110, 0.8446)	(0.8911, 0.7807, 0.8323)	(0.9081, 0.8137, **0.8583**)
	Baseline	highway	(0.9292, 0.8916, 0.9100)	(0.9162, 0.9472, 0.9315)	(0.9296, 0.9023, 0.9158)	(0.9216, 0.9841, **0.9518**)
	Dyn. Bg.	overpass	(0.8680, 0.8141, 0.8402)	(0.8600, 0.8067, 0.8325)	(0.8683, 0.8460, **0.8570**)	(0.8840, 0.8033, 0.8417)
Baseline fountain01	Shadow	bungalows	(0.8163, 0.9857, **0.8930**)	(0.7995, 0.9849, 0.8826)	(0.8180, 0.9805, 0.8919)	(0.8197, 0.9787, 0.8922)
Dyn. Bg.overpass	Baseline	pedestrians	(0.9707, 0.5645, 0.7138)	(0.9424, 0.7899, 0.8595)	(0.9310, 0.6210, 0.7451)	(0.9834, 0.6616, 0.7910)
Shadow bungalows	Dyn. Bg.	canoe	(0.8303, 0.9236, 0.8744)	(0.9015, 0.6935, 0.7840)	(0.8224, 0.9412, 0.8778)	(0.9007, 0.8919, **0.8963**)
	Shadow	peopleInShade	(0.8871, 0.8236, 0.8542)	(0.9017, 0.8389, 0.8692)	(0.8786, 0.8155, 0.8459)	(0.9062, 0.9742, **0.9390**)
Overall			(0.8997, 0.8304, 0.8572)	(0.8923, 0.8425, 0.8634)	(0.8813, 0.8404, 0.8558)	(0.9150, 0.8839, **0.8948**)

Figure 5.4: Quantitative evaluation of the proposed approach for unseen videos on the CDnet2014 [135] dataset, using the Fm metric.

Approach	Baseline	Dyn. Bg.	Cam. Jitt.	Int. Mot.	Shadow	Ther.	Bad Wea.	Low Fr.	Nig. Vid.	PTZ	Turbul.	Overall
IUTIS-5[8]	0.9567	0.8902	0.8332	0.7296	0.9084	0.8303	0.8248	0.7743	0.5290	0.4282	0.7836	0.7717
PAWCS[19]	0.9397	0.8938	0.8137	0.7764	0.8913	0.8324	0.8152	0.6588	0.4152	0.4615	0.6450	0.7403
BSUV[127]	**0.9693**	0.7967	0.7743	0.7499	0.9233	0.8581	0.8713	0.6797	**0.6987**	0.6282	0.7051	0.7868
BSUV2.0[126]	0.9620	0.9057	**0.9004**	**0.8263**	**0.9562**	**0.8932**	**0.8844**	**0.7902**	0.5857	**0.7037**	**0.8174**	**0.8387**
FgSN[86]	0.6926	0.3634	0.4266	0.2002	0.5295	0.6038	0.3277	0.2482	0.2800	0.3503	0.0643	0.3715
ADNN-IB$_{U4fs}$	0.9522	**0.9166**	0.8245	0.6978	0.9054	0.8058	0.8245	0.7028	0.4872	0.2931	0.7620	0.7451

Figure 5.5: Quantitative evaluation of the proposed approach for seen and unseen videos on the SBMI2015 dataset, using the Fm metric.

Approach	Board	Cand.	CAVIAR1	CAVIAR2	CaVig.	Foliage	HallA.	HighwayI	HighwayII	HumanB.	IBMtest2	PeopleA.	Snellen	Overall
Vibe[6]	0.7377	0.5020	0.8051	0.7347	0.3497	0.5539	0.6017	0.4150	0.5554	0.4268	0.7001	0.6111	0.3083	0.5617
RPCA[72]	0.5304	0.4730	0.4204	0.1933	0.4720	0.4617	0.4525	0.5733	0.7335	0.5765	0.6714	0.3924	0.4345	0.4911
SuBSENSE [18]	0.6588	0.6959	0.8783	0.8740	0.4080	0.1962	0.7559	0.5073	0.8779	0.8560	0.9281	0.4251	0.2467	0.6391
FgSN-M-55[86]	0.8900	0.2100	0.7000	0.0500	0.5700	**0.9100**	0.7100	0.7500	0.3100	0.8300	0.8300	**0.9000**	0.5200	0.6300
MSFS-55[85]	0.9100	0.2600	0.5700	0.0800	0.5700	0.8000	0.5200	0.8200	0.5800	0.6100	0.6000	0.8700	0.6800	0.6100
3DCD-55[99]	0.8300	0.3500	0.7900	0.5600	0.4800	0.6900	0.5800	0.7300	0.7700	0.6500	0.7000	0.7800	0.7600	0.6700
ADNN-IB$_{C2fs}$	0.4680	0.5981	0.7801	0.8646	0.6994	0.1420	0.8176	0.7376	0.9786	0.9329	0.8967	0.3893	0.0472	0.6425
ADNN-IB$_{C8fm}$	0.4527	0.5222	0.9169	0.8429	0.7259	0.0722	0.8200	0.7493	0.9827	0.9431	0.9348	0.3071	0.0445	0.6396
ADNN-IB$_{S2fs}$	**0.9421**	**0.9242**	**0.9550**	**0.8865**	**0.9589**	0.7528	**0.9151**	**0.8689**	**0.9854**	**0.9525**	**0.9548**	0.7108	**0.7893**	**0.8920**

Table 5.7: Quantitative evaluation of the proposed approach for unseen videos on the LASIESTA [27] dataset, using the Fm metric.

Videos	ADNN-IB$_{\mathrm{C2fs}}$	BSUV2.0[126]	3DCD[99]	FgSN[86]	CueV2[7]
I_SI_02	**0.91**	0.89	0.86	0.53	0.84
I_CA_02	**0.79**	0.60	0.49	0.58	0.78
I_OC_02	**0.95**	**0.95**	0.93	0.25	0.86
I_IL_02	0.53	**0.89**	0.85	0.41	0.65
I_MB_02	0.88	0.76	0.79	0.63	**0.92**
I_BS_02	**0.92**	0.69	0.87	0.25	0.62
O_CL_02	**0.97**	0.89	0.87	0.54	0.90
O_RA_02	**0.97**	0.93	0.87	0.54	0.89
O_SN_02	**0.80**	0.70	0.49	0.05	0.63
O_SU_02	0.87	**0.91**	0.83	0.29	0.77
Average	**0.86**	0.82	0.79	0.41	0.79

are computed to verify if the output is close to the target. Mathematically, the correlation value is computed as:

$$v = \frac{H_Z \cdot H_{Z'}}{|H_Z||H_{Z'}|} \tag{5.13}$$

where H_Z and $H_{Z'}$ are the histograms of the target distribution and output distribution, and v is the correlation value.

The experimental results are shown in Fig. 5.6, in which the histograms of X and W, the histograms of output layers Z'_p and Z'_s, the target histograms of Z_p and Z_s, the histogram of the arithmetic distribution layers W', and the correlation-epoch plots are shown. After training hundreds of epochs, the correlation between the output and target distributions is almost equal to 1. Furthermore, the distribution learned by the arithmetic distribution layers is almost the same as the one for W, which is used to generate the target histogram. Both of these results demonstrate that the proposed arithmetic distribution layers have the ability to output the desired distributions. The correctness of the proposed arithmetic distribution layer, which contains the forward pass and backpropagation through layers, is thus verified.

5.4.2 Comparisons with Convolutional Neural Networks

In this section, we demonstrate the superiority of the proposed arithmetic distribution neural network (ADNN) compared to the convolutional neural

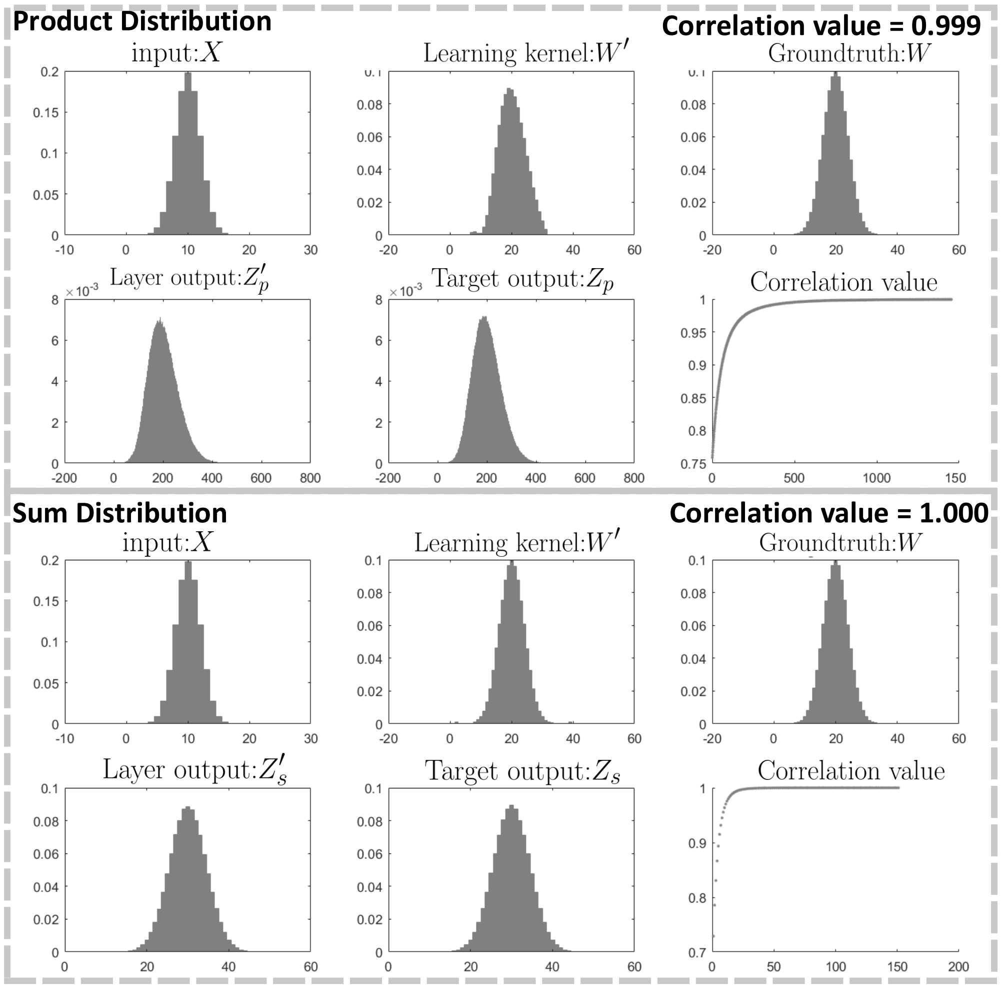


Figure 5.6: Validation of the arithmetic distribution layers including product and sum distribution layers. The histogram of input X, the histogram in the learning kernel W', the histogram of the ground truth W, the histograms of output layers Z'_p and Z'_s, the target histograms of Z_p and Z_s and the correlation-epoch plot are shown.

network (CNN). Arithmetic distribution layers are proposed to serve as a better substitute for the convolutional layer. Thus, the proposed ADNN is better than convolutional neural networks in distribution classification. In order to demonstrate this, we devise 2 arithmetic distribution neural networks (ADNN$_{1-2}$) to compare with 14 traditional convolutional neural networks (CNN$_{1-14}$). Details on the architecture of ADNN$_{1-2}$ and CNN$_{1-14}$ are shown in Table 5.4. CNN$_{1-3}$ are devised by replacing the arithmetic distribution layers in ADNN$_1$ by several convolutional layers. CNN$_{4-7}$ are devised by inserting extra nonlinear activation functions, such as Rectified Linear Unit (Relu) or Gaussian Error Linear Unit (Gelu), into CNN$_3$ to handle the nonlinear data. Similarly, CNN$_{8-10}$ are devised by replacing the arithmetic distribution layers in ADNN$_2$ with fully connected layers, and CNN$_{11-14}$ are devised by inserting extra Relu and Gelu layers into CNN$_{10}$. During the comparisons between ADNNs and CNNs, one ground-truth frame from training videos is extracted for training, and all training and testing settings of CNNs and ADNNs are the same, including the learning rate, training algorithms, maximum number of training epochs, random number seed as well as the input and output of networks. For evaluation, the Re (Recall), Pr (Precision), and Fm (F-measure) metrics are used.

From the quantitative comparisons shown in Table 5.5, we can conclude that the results of CNN$_3$ are better than CNN$_2$ which are better than CNN$_1$, since when the training data is fixed, a network with more parameters is supposed to have better learning ability. In addition, the results of CNN$_3$ are better than the ones of CNN$_{4-7}$ which are devised by adding Relu and Gelu layers into CNN$_3$. This is because Relu or Gelu drops several entires of input vectors which may result in losing useful information. In contrast, the proposed ADNN$_1$ achieves better results on the average Fm value compared to CNN$_{3-7}$, which uses around 100 times more parameters than the proposed ADNN. This clearly demonstrates that the block consisting of arithmetic distribution layers has better distribution learning ability than the one consisting of traditional convolutional layers attached or not attached with Relu or Gelu. Moreover, in order to further compare the proposed arithmetic distribution layers with fully connected layers, which has better ability than convolutional layers to learn global information, the comparisons between ADNN$_2$ and CNN$_{8-14}$ are presented in Table 5.6. In particular, since the learning ability of CNN and ADNN are improved by increasing of the number of parameters, both the seen and unseen videos are used for evaluation to obtain the quantitative results under challenging conditions. As shown in Table 5.6, CNN$_6$ is the largest network with 2.9 million parameters and it is constructed purely by fully con-

Table 5.8: Quantitative evaluation of the proposed approach for seen and unseen videos on the LASIESTA [27] dataset, using the Fm metric.

Videos	ADNN-IB$_{L3fs}$	D-DPDL[151]	CueV2[7]	Hai[50]	ADNN-IB$_{C2fs}$	ADNN-IB$_{c8fm}$	BSUV2.0[126]	3DCD-55[99]	MSFS-55[85]
I_SI_01	**0.9764**	0.9596	0.9208	0.9622	0.9293	0.9335	0.9200	0.8700	0.3900
I_SI_02	**0.9309**	0.8687	0.8403	0.8130					
I_CA_01	**0.9807**	0.9309	0.9062	0.9220	0.8225	0.7316	0.6800	0.8200	0.4000
I_CA_02	**0.9201**	0.8850	0.7826	0.8656					
I_OC_01	**0.9783**	0.9710	0.7013	0.8920	0.9600	0.9481	0.9600	0.9100	0.3700
I_OC_02	**0.9735**	0.9677	0.8600	0.9526					
I_IL_01	0.7702	0.7161	0.6452	**0.8861**	0.5216	0.4869	0.8800	0.9200	0.3500
I_IL_02	0.7620	**0.8972**	0.6523	0.8122					
I_MB_01	**0.9874**	0.9699	0.9543	0.9816	0.9272	0.9262	0.8100	0.8900	0.6400
I_MB_02	**0.9731**	0.9195	0.9204	0.7064					
I_BS_01	**0.9787**	0.8371	0.7132	0.6285	0.9216	0.8884	0.7700	0.7200	0.3600
I_BS_02	**0.9626**	0.6178	0.6156	0.7333					
O_CL_01	**0.9840**	0.9792	0.9508	0.6946	0.9594	0.9534	0.9300	0.8700	0.4100
O_CL_02	0.9788	**0.9800**	0.9045	0.9588					
O_RA_01	**0.9896**	0.9072	0.8453	0.8225	0.8668	0.8744	0.9400	0.9000	0.3500
O_RA_02	**0.9839**	0.9803	0.8886	0.9590					
O_SN_01	**0.9733**	0.9690	0.9317	0.3054	0.8300	0.8327	0.8400	0.6900	0.3100
O_SN_02	**0.9562**	0.9341	0.6256	0.0426					
O_SU_01	**0.9186**	0.9065	0.6774	0.8115	0.8715	0.8829	0.7900	0.8500	0.3700
O_SU_02	**0.9413**	0.9388	0.7669	0.9021					
Average	**0.9460**	0.9068	0.8051	0.7826	0.8610	0.8459	0.8500	0.8400	0.4000

nected layers, and CNN_{11-14} are devised by inserting Relu and Gelu in these fully connected layers. However, the proposed ADNN_2 still achieves better results than CNN_{8-14} and the number of parameters in ADNN_2 is only around 0.1 million. Thus, it is fair to claim that the proposed arithmetic distribution layers are better than convolutional layers in distribution classification tasks.

5.4.3 Comparisons with State-of-the-art Methods

The proposed approach has three good properties for use in real applications. 1) *Generality*: the proposed approach is effective for unseen videos; 2) *Efficiency*: only limited ground-truth frames are required to generate promising results; 3) *Universality*: one network can be trained for all videos. In this section, these three properties are demonstrated through comparisons with state-of-the-art methods including unsupervised methods, deep learning methods for seen videos and unseen videos on CDnet2014 [135] LASIESTA [27] and SBMI2015 [94] datasets.

The deep learning networks compared include DeepBS[4], GuidedBS[84], CNN-SFC[33], D-DPDL [151], 3DCD[99], FgSN[86], MSFS[85], DVTN [43] and BSUV[126], [127]. In particular 3DCD[99] and BSUV[126], [127] are effective for unseen videos. After the rise of deep learning networks in the background subtraction field, the fairness of comparisons between deep learning methods has been a concern. It is commonly accepted that the quantity of training data and the number of parameters in a network have significant and direct contributions to the performance of various methods [77]. However, the assumptions on the training data, numbers of parameters in the network, and the utilization of pre-trained networks in these methods are completely different. In order to propose fair comparisons, the proposed ADNN is trained and tested under 6 conditions to propose different evaluation results which are named $\text{ADNN-IB}_{\text{U4fs}}$, $\text{ADNN-IB}_{\text{C2fs}}$, $\text{ADNN-IB}_{\text{L3fs}}$, $\text{ADNN-IB}_{\text{S2fs}}$, $\text{ADNN-IB}_{\text{c8fm}}$ and $\text{ADNN-IB}_{\text{C20fs}}$ for comparisons with various state-of-the-art methods.

$\text{ADNN-IB}_{\text{U4fs}}$ is trained following the partition of training and testing videos proposed in BSUV [127]. During training, 4 ground-truth frames from every training video and 4 binary masks of testing videos provided by IUTIS-5 [8] are mixed and used for training. Note that BSUV also manually extracted hundreds of frames from testing videos to generate reference images which can be used as background instances to train the proposed approach. ADNN-IB$_{\mathrm{C2fs}}$ is trained by a mixture of 2 ground-truth frames from every video of the CDnet2014 [135] dataset. ADNN-IB$_{\mathrm{c8fm}}$ is trained by a mixture of 8 down-sampling ground-truth frames from every video of the CDnet2014 [135] dataset. During the down-sampling procedure, only 25% of labels from the ground-truth frames are kept for training. Thus, the total number of training instances of ADNN-IB$_{\mathrm{c8fm}}$ and ADNN-IB$_{\mathrm{C2fs}}$ are actually the same, but the training set of ADNN-IB$_{\mathrm{c8fm}}$ includes more temporal information. Both ADNN-IB$_{\mathrm{C2fs}}$ and ADNN-IB$_{\mathrm{c8fm}}$ are tested on CDnet2014 [135], LASIESTA [27] and SBMI2015 [94] datasets. During the evaluations of ADNN-IB$_{\mathrm{c8fm}}$ and ADNN-IB$_{\mathrm{C2fs}}$, only one network is trained and the parameters of the network are thus fixed for all testing videos.

ADNN-IB$_{\mathrm{L3fs}}$, ADNN-IB$_{\mathrm{S2fs}}$ and ADNN-IB$_{\mathrm{C20fs}}$ are evaluated on LASIESTA [27], SBMI2015 [94] and CDnet2014 [135] datasets respectively. During the evaluations, 3, 2, and 20 ground-truth frames from videos of corresponding datasets are used for training, and the remaining frames of the same videos are used for testing. This means that various networks are trained for different videos to generate the results of ADNN-IB$_{\mathrm{L3fs}}$, ADNN-IB$_{\mathrm{S2fs}}$ and ADNN-IB$_{\mathrm{C20fs}}$. The training frames only take less than 1% of ground-truth frames of the corresponding datasets and all ground-truth frames used for training are randomly selected.

Generality

The distributions of temporal pixels are relatively independent of the scene information, demonstrating that the proposed approach is generalizable. In order to demonstrate this, ADNN-IB$_{\mathrm{U4fs}}$, ADNN-IB$_{\mathrm{C2fs}}$ and ADNN-IB$_{\mathrm{c8fm}}$ are trained and evaluated. The quantitative results of ADNN-IB$_{\mathrm{U4fs}}$ are shown in Table 5.4. ADNN-IB$_{\mathrm{U4fs}}$ achieves the best results in the "Dyn. Bg." category. In this category, single background images are not enough to describe background information. In contrast, distributions for temporal pixels have better description ability. This is main reason why ADNN-IB$_{\mathrm{U4fs}}$ has better results than BSUV2.0 and IUTIS-5. Unfortunately, the overall results of ADNN-IB$_{\mathrm{U4fs}}$ can only be considered as promising during comparisons with BSUV and BSUV2.0. However, such comparisons may not be fair to the proposed approach, since BSUV extracted 200 ground-truth frames from every training video, and BSUV2.0 also utilized synthetic ground-truth frames to train their network. The synthetic ground-truth frames used in BSUV2.0 are very close to the ground-truth of testing videos, since they are generated by the fusion of manually selected frames from the testing video and moving objects segmented from other videos with the help of ground-truth frames, and videos from the same dataset are correlated to each other. However, when the training videos and testing videos are extracted from different datasets, the contribution of the quantity of the training set is reduced due to the lower correlation between training videos and testing videos. Thus, ADNN-IB$_{\mathrm{C2fs}}$ and ADNN-IB$_{\mathrm{c8fm}}$ are trained to compare with BSUV2.0 and 3DCD across datasets. The quantitative results of ADNN-IB$_{\mathrm{C2fs}}$ on several videos from LASIESTA are shown in Table 5.7. The LASIESTA dataset contains 20 videos from indoor and outdoor scenes. In addition, it also includes videos with illumination changes (I_IL) or camouflage (I_CA), which are challenging scenes for background subtraction. However, ADNN-IB$_{\mathrm{C2fs}}$ achieves the best results compared to BSUV2.0 and 3DCD. In particular, not only is the proposed ADNN-IB$_{\mathrm{C2fs}}$, but also BSUV2.0 and 3DCD are trained on the CDnet2014 dataset and tested on the LASIESTA dataset. This demonstrates the superiority of the generality of the proposed approach across different datasets. Such superiority is also demonstrated by the quantitative results shown inTable 5.8. The proposed ADNN-IB$_{\mathrm{C2fs}}$ achieves better results than BSUV2.0 and 3DCD on the entire LASIESTA dataset. Thus, the ability of the proposed approach to generalize is demonstrated.

Table 5.9: Quantitative evaluation of the proposed approach for seen videos on CDnet2014 [135] dataset, using Fm metric.

	Approach	Baseline	Dyn. Bg.	Cam. Jitt.	Int. Mot.	Shadow	Ther.	Bad Wea.	Low Fr.	Nig. Vid.	PTZ	Turbul.	Overall
	IUTIS-5[8]	0.9567	0.8902	0.8332	0.7296	0.9084	0.8303	0.8248	0.7743	0.5290	0.4282	0.7836	0.7717
	MBS[113]	0.9287	0.7915	0.8367	0.7568	0.7968	0.8194	0.7980	0.6350	0.5158	0.5520	0.5858	0.7288
	PAWCS[19]	0.9397	0.8938	0.8137	0.7764	0.8913	0.8324	0.8152	0.6588	0.4152	0.4615	0.6450	0.7403
	ShareM[23]	0.9522	0.8222	0.8141	0.6727	0.8898	0.8319	0.8480	0.7286	0.5419	0.3860	0.7339	0.7474
	SuBSENSE[18]	0.9503	0.8177	0.8152	0.6569	0.8986	0.8171	0.8619	0.6445	0.5599	0.3476	0.7792	0.7408
	WeSamBE[70]	0.9413	0.7440	0.7976	0.7392	0.8999	0.7962	0.8608	0.6602	0.5929	0.3844	0.7737	0.7446
	GMM[159]	0.8245	0.6330	0.5969	0.5207	0.7370	0.6621	0.7380	0.5373	0.4097	0.1522	0.4663	0.5707
	3PDM [112]	0.8820	0.8990	0.7270	0.6860	0.8650	0.8410	0.8280	0.5350	0.4210	0.5010	0.7930	0.7253
	HMAO [81]	0.8200	N/A	0.6300	0.7200	0.8600	0.8400	0.7900	0.6000	0.3600	N/A	0.4600	N/A
	B-SSSR[68]	0.9700	**0.9500**	0.9300	0.7400	0.9300	0.8600	0.9200	N/A	N/A	N/A	0.8700	N/A
	MSCL-FL[67]	0.9400	0.9000	0.8600	0.8400	0.8600	0.8600	0.8800	N/A	N/A	N/A	N/A	N/A
	DSPSS[34]	0.9664	0.9057	0.8662	0.7870	0.9177	0.7328	N/A	N/A	N/A	N/A	N/A	N/A
	STSHBM[20]	0.9534	0.9120	0.8503	0.8349	0.8930	0.8579	N/A	N/A	N/A	N/A	N/A	N/A
deep learning methods	BMN-BSN [104]	0.9521	0.6371	0.6962	0.6369	0.7893	0.7849	0.8124	0.6426	0.6125	N/A	N/A	N/A
	DeepBS[4]	0.9580	0.8761	0.8990	0.6098	0.9304	0.7583	0.8301	0.6002	0.5835	0.3133	0.8455	0.7548
	CNN-SFC[33]	0.9497	0.9035	0.8035	0.7499	0.9127	0.8494	0.9084	0.7808	0.6527	0.7280	0.8288	0.8243
	CwisarDH[49]	0.9145	0.8274	0.7886	0.5753	0.8581	0.7866	0.6837	0.6406	0.3735	0.3218	0.7227	0.6812
	$DPDL_{40}$ [152]	0.9692	0.8692	0.8661	0.8759	0.9361	0.8379	0.8688	0.7078	0.6110	0.6087	0.7636	0.8106
	3DCD-55[99]	0.9100	0.8500	0.8100	0.8300	0.8800	0.8500	0.9500	0.7400	0.8700	N/A	0.9200	N/A
	FgSN-55[86]	0.9200	0.6900	0.7600	0.6100	0.8200	0.7300	0.7200	0.3200	0.8100	N/A	0.5700	N/A
	MSFS-55[85]	0.8000	0.5000	0.8900	0.7000	0.9300	0.8600	0.8000	0.5500	0.8200	N/A	0.6700	N/A
	DVTN [43]	**0.9811**	0.9329	0.9014	**0.9595**	0.9467	**0.9479**	0.8780	0.7818	0.7737	0.5957	**0.9034**	0.8789
	$ADNN_{c20fs}$	0.9729	0.9110	0.8997	0.8857	0.9305	0.9091	0.8512	0.7370	0.6215	0.6310	0.8405	0.8355
	$ADNN\text{-}IB_{c2fs}$	0.9514	0.8923	0.8309	0.6989	0.8814	0.7565	0.8635	0.6507	0.5089	0.1252	0.7007	0.7146
	$ADNN\text{-}IB_{c8fm}$	0.9562	0.8748	0.8532	0.8742	0.9347	0.8568	0.8764	0.7983	0.6161	0.2409	0.7826	0.7877
	$ADNN\text{-}IB_{c20fs}$	0.9797	0.9454	**0.9411**	0.9114	0.9537	0.9411	0.9038	**0.8123**	0.6940	0.7424	0.8806	**0.8826**

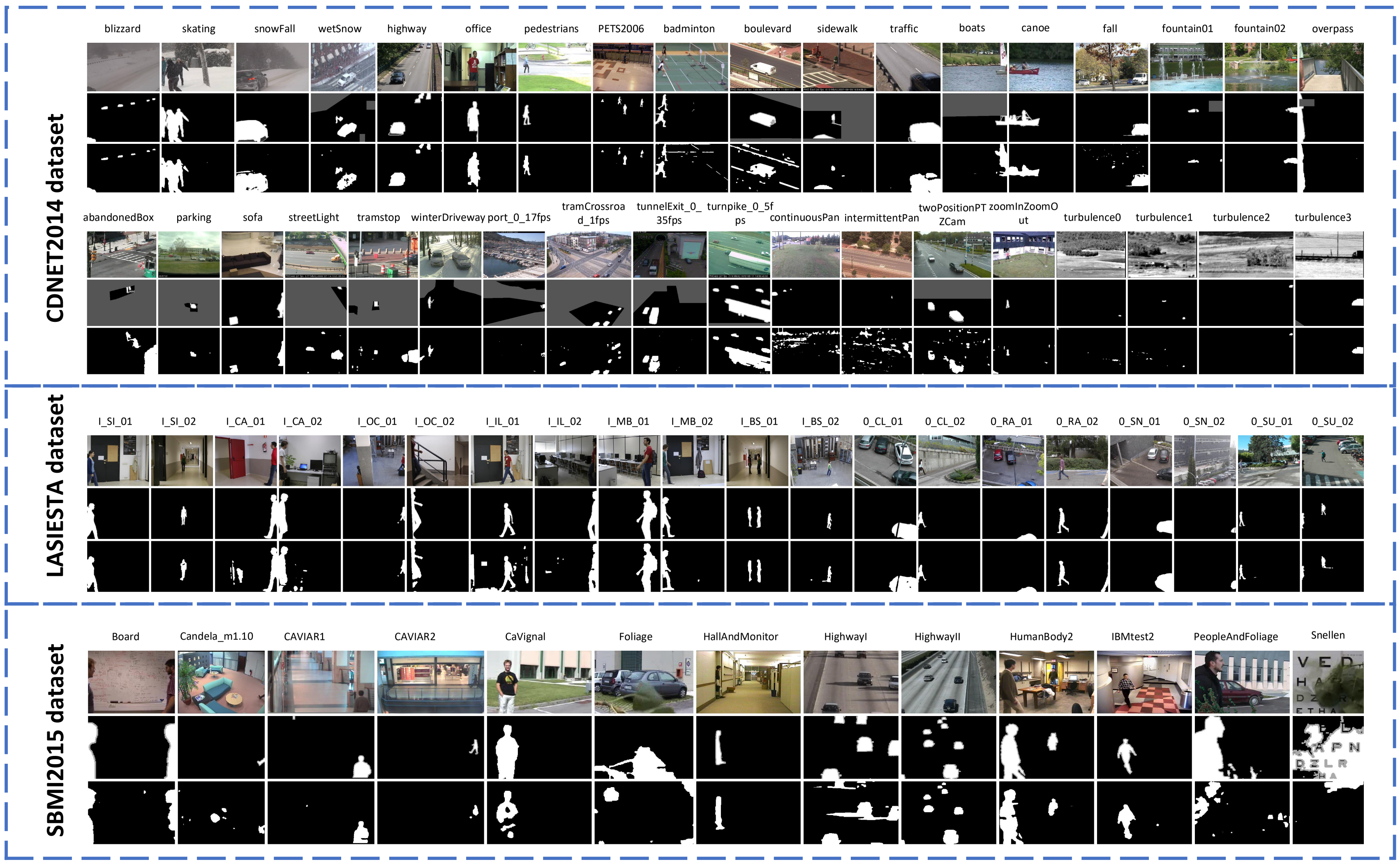


Figure 5.7: The visual results of the proposed ADNN-IB$_{\text{C8fm}}$ on CDNet2014 dataset, LASIESTA dataset and SBMI2015 dataset.

Efficiency

Recently, a few excellent deep learning networks, such as MSFS-55 and FgSN-55, have achieved almost perfect results when the training frames and computational resources are not limited. However, the utility of these methods is limited since creating ground-truth frames is very expensive in real applications. In addition, when the ground-truth frames used for training are limited or unseen videos are included for evaluation, such methods no longer work perfectly. For example, according to the results published by Lim et al. [86], the FgSN method attains over 98% in Fm value on CDnet2014. However, when FgSN is evaluated by the partition of training and testing frames proposed by Mandal et al. [99], the Fm value of FgSN decreases. Furthermore, once FgSN is applied to the unseen videos of LASIESTA, it only achieves 0.41 in Fm value, as shown in Table 5.7. In contrast, the proposed approach has good efficiency and only needs limited ground-truth frames to generate excellent results. As the quantitative results of ADNN-IB$_{\mathrm{L3fs}}$, ADNN-IB$_{\mathrm{S2fs}}$ and ADNN-IB$_{\mathrm{C20fs}}$, show in Table 5.8, Table 5.9 and Table 5.5, respectively, the proposed approach achieves the best overall Fm value for all three datasets, and the ground-truth frames used for training only take less than 1% of ground-truth frames of the corresponding datasets. By comparison, most of the compared methods based on deep learning networks use many more ground-truth frames than the proposed approach. For example, 3DCD-55 used 50% of the ground-truth frames of a particular video for training and the remaining frames from the same for testing. Compared to all these state-of-the-art methods based on deep learning networks, the proposed approach achieves the highest Fm value with the fewest number of ground-truth frames for training. This demonstrates the efficiency of the proposed approach. In addition, the results of ADNN$_{\mathrm{c20fs}}$, which is the results of ADNN-IB$_{\mathrm{C20fs}}$ without the improved Bayesian refinement model, demonstrates the contributions of the improved Bayesian refinement model. As shown in Table 5.9, the improved Bayesian refinement gives the proposed approach around 5% improvement in Fm value.

Universality

Traditionally, the parameters of background subtraction algorithms are fixed for all testing videos. This is reasonable for real applications since parameters adjustment is time-consuming. However, most of the methods based on deep learning actually trained various networks for different videos. Even for a few networks proposed for unseen videos, such as 3DCD or BSUV, various networks are still trained for videos from different categories or datasets. This is unfair for comparisons between unsupervised methods and methods based on deep learning networks. In fact, the results of unsupervised methods can be easily improved by manually adjusting their threshold values for different videos. Fortunately, due to the generality of distribution information as well as the efficiency of the proposed approach, one arithmetic distribution neural network can be trained for all videos. This demonstrates the universality of the proposed approach. In order to demonstrate this, ADNN-IB$_{\mathrm{c8fm}}$ and ADNN-IB$_{\mathrm{C2fs}}$ are trained and evaluated. In particular, during the testing of ADNN-IB$_{\mathrm{c8fm}}$ and ADNN-IB$_{\mathrm{C2fs}}$, the parameters of the networks are fixed for all testing videos. From the quantitative evaluations on LASIESTA and SBMI2015, shown in Table 5.8 and Table 5.5, respectively, both ADNN-IB$_{\mathrm{c8fm}}$ and ADNN-IB$_{\mathrm{C2fs}}$ achieve good results compared to unsupervised state-of-the-art methods. Since no frame from these two datasets are used for training and the parameters of the proposed approach are fixed for all testing videos, the comparisons between the proposed approach and unsupervised methods are completely fair. This way, the universality of the proposed approach is demonstrated.

From our point of view, when ground-truth frames are limited to an acceptable level and the network parameters are fixed, methods based on deep learning networks are appropriate for comparison with unsupervised methods. This is because researchers also adjust the parameters of their methods to achieve the best results for a particular dataset. During parameter adjustments, the ground-truth frames are also manually checked. Thus, both unsupervised methods and methods based on deep learning networks include prior knowledge from ground-truth frames. The difference is that deep learning networks directly extract the knowledge from ground-truth frames, while unsupervised methods capture the knowledge through parameter adjustments by researchers with the help of ground-truth frames. Based on this observation, although ADNN-IB$_{\text{c8fm}}$ and ADNN-IB$_{\text{C2fs}}$ are trained by frames from the CDnet2014 dataset, they are suitable for comparisons with unsupervised methods, such as SuBSENSE or IUTIS-5 on the CDnet2014 dataset. Note that the remaining frames used for testing takes over 99% of the ground-truth frames of the entire dataset. From the quantitative evaluation shown in Table 5.9, ADNN-IB$_{\text{c8fm}}$ achieves better results than IUTIS-5 which is a combination of several excellent state-of-the-art methods. Since the parameters of ADNN-IB$_{\text{c8fm}}$ are fixed for all testing videos and the amount of ground-truth labels used for training takes less than 1% of the entire dataset, the comparison between ADNN-IB$_{\text{c8fm}}$ and IUTIS-5 should be considered to be fair. Also, the good performance of ADNN-IB$_{\text{c8fm}}$ demonstrates the potential of the proposed approach in real applications, since a single well-trained network can be used for all testing videos even from different datasets, based on results shown in Tables 5.8 and 5.5. In addition, the visual results of ADNN-IB$_{\text{C8fm}}$ are shown in Fig. **??**.

The proposed approach is implemented in Pytorch[3], and the source code will be made available following acceptance of our paper. Experiments are run on a GeForce GTX 1080 GPU processor with 8 GB memory. During training 60 epochs are set as the maximum, the learning rate is set to 0.0001 and the Adam method [76] with default parameters is used for training.

5.4.4 Limitation and Complexity

Failure Case Analysis

Although the proposed approach achieves promising results, there are still a few failure cases which are discussed in this section. As shown in Table 5.9, the proposed approach does not work well in the PTZ category where videos are obtained by a moving camera. This happens because the histograms of temporal pixels are no longer useful features since pixels do not maintain their positions when a camera moves. Fortunately, this limitation can be addressed by using the histograms of optical flows rather than temporal pixels as the input of the proposed approach. In addition, since the proposed approach only takes less 1% of ground-truth frames for training, it is possible that these training frames are not enough to cover the illumination variation in videos, which results in the failure of the proposed approach on videos "I_IL_01" and "I_IL_02", as shown in Table 5.8. This problem can be handled by increasing the number of ground-truth frames for training.

Complexity Analysis

Currently, the proposed approach takes around 3s total time to process a frame with resolution 320×240 on our computer. In particular, the histogram generation takes 0.1727s, the computation of the arithmetic distribution layer takes 0.4292s, the computation of classification block followed by arithmetic distribution layers takes 0.078s, and the improved Bayesian refinement takes 2.2524s. Although 3s is too long for real-time applications, there are several ways to accelerate the proposed approach. First, a C++ implementation of the proposed approach should take much less time, since python is currently used for implementing the arithmetic distribution layers and improved Bayesian refinement model. In addition, the batch size of pixels used for processing is given as 1000, which means only 1000 pixels are processed simultaneously in one processing round, since our GPU card is GTX 1080 with 8GB memory and some of the memory has to be used for exception handling. Once a machine with a better GPU card and larger memory is used for evaluation, the proposed approach can be easily accelerated by increasing the batch size.

5.5 Conclusion

We proposed the Arithmetic Distribution Neural Network (ADNN) for background subtraction. Specifically, the arithmetic distribution layers, including the product and sum distribution layers, based on arithmetic distribution operations were proposed for learning the distributions of temporal pixels. Also, an improved Bayesian refinement model, based on neighborhood information with a GPU implementation, was proposed to improve the robustness and accuracy of the proposed approach. Utilizing the arithmetic distribution layers, histograms are considered as probability density functions. This probability information is used during the learning procedure of the proposed approach. Compared to previous approaches based on deep learning networks, the proposed approach has three advantages, including: a) being effective for unseen videos; b) promising results are obtained using limited ground-truth frames; c) one network can be trained for all testing videos even from different datasets. Comprehensive evaluations compared to state-of-the-art methods showed the superior performance of the proposed approach, and demonstrated its potential for use in practical applications.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we proposed several methods to demonstrate how to learn statistical distributions for video segmentation tasks. These included background subtraction, vessel segmentation and crack detection. We proposed the ADNN, which performed better than the traditional convolutional neural network in distributions classification.

The most important contribution of this thesis is the proposed ADNN, which is the method of our final choice for learning statistical distributions. In order to do propose the ADNN, we first proposed the D-DPDL method for background subtraction. The entries of input patches were randomly permutated to force the network to focus on statistical distributions. The proposed D-DPDL method demonstrates a few excellent properties including that it requires limited ground-truth frames for training, and that training videos and testing videos can be different. We further extended the D-DPDL model for vessel segmentation and crack detection, and the D-DPDL model achieved promising results on both applications. During the extension of the D-DPDL model, we had a better performance with a wider rather than deeper network. This interesting phenomenon motivated us to find an explanation, and arithmetic distribution operations theoretically explains it. Also, based on the arithmetic distribution operations, we proposed the ADNN, which demonstrates a better distributions classification ability than the conventional convolutional neural network.

6.2 Future Work

The proposed ADNN is just a prototype. The architecture of the network is very simple and the number of parameters correspondingly small. The network used for comparisons with state-of-the-art methods in this work has only 0.1 million parameters, the training set takes less than 1% of ground-truth frames from the entire dataset and no pre-trained networks are used. However, the proposed ADNN still achieved promising results compared to state-of-the-art methods based on deep learning networks with many more parameters than the proposed ADNN. Thus, there is still room for improvement in the performance of the proposed approach by increasing the size of the training dataset, the number of network parameters, and integrating pre-trained networks for features extraction. However, due to the limitation of our computational resources, the results proposed in this thesis are the best we can present. In addition, as shown by the comparisons in Section 5.4.2, the proposed arithmetic distribution layers are better at distributions analysis compared to the convolutional layer. This demonstrates excellent potential for the proposed approach since distribution analysis has a wide range of applications beyond background subtraction. This provides another direction for our future work.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012, ISSN: 0162-8828. DOI: `10.1109/TPAMI.2012.120`.

[2] T. Akilan, Q. J. Wu, and Y. Yang, "Fusion-based foreground enhancement for background subtraction using multivariate multi-model gaussian distribution," *Information Sciences*, pp. 414–431, 2018, ISSN: 0020-0255.

[3] P. et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019.

[4] M. Babaee, D. T. Dinh, and G. Rigoll, "A deep convolutional neural network for video sequence background subtraction," *Pattern Recognit.*, vol. 76, pp. 635–649, 2018, ISSN: 0031-3203.

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017, ISSN: 0162-8828. DOI: `10.1109/TPAMI.2016.2644615`.

[6] O. Barnich and M. Van Droogenbroeck, "Vibe: A universal background subtraction algorithm for video sequences," *IEEE Trans. Image Process.*, vol. 20, no. 6, pp. 1709–1724, Jun. 2011, ISSN: 1057-7149. DOI: `10.1109/TIP.2010.2101613`.

[7] D. Berjón, C. Cuevas, F. Morán, and N. García, "Real-time nonparametric background subtraction with tracking-based foreground update," *Pattern Recognit.*, vol. 74, pp. 156–170, 2018, ISSN: 0031-3203.

[8] S. Bianco, G. Ciocca, and R. Schettini, "How far can you get by combining change detection algorithms?" In *Int. Conf. Image Analysis and Process.*, 2017, pp. 96–107.

[9] T. Bouwmans, F. El Baf, and B. Vachon, "Statistical background modeling for foreground detection: A survey," in *Proc. Handbook of pattern recognition and computer vision*, World Scientific, 2010.

[10] 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, ISSN: 0893-6080.

[11] M. Braham and M. V. Droogenbroeck, "Deep background subtraction with scene-specific convolutional neural networks," in *Int. Conf. Systems, Signals and Image Process. (IWSSIP)*, May 2016, pp. 1–4. DOI: 10.1109/IWSSIP.2016.7502717.

[12] M. Braham, S. Piérard, and M. Van Droogenbroeck, "Semantic background subtraction," in *2017 IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 4552–4556.

[13] P. A. Campochiaro, "Molecular pathogenesis of retinal and choroidal vascular diseases," *Progress in retinal and eye research*, vol. 49, pp. 67–81, 2015.

[14] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM*, vol. 58, no. 3, pp. 1–37, 2011.

[15] P. Carmeliet and R. K. Jain, "Angiogenesis in cancer and other diseases," *nature*, vol. 407, no. 6801, p. 249, 2000.

[16] B. O. Cha Y. J. Choi W., "Deep learning-based crack damage detection using convolutional neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, 2017.

[17] S. Chanda, G. Bu, H. Guan, J. Jo, U. Pal, Y.-C. Loo, and M. Blumenstein, "Automatic bridge crack detection–a texture analysis-based approach," pp. 193–203, 2014.

[18] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin, "Subsense: A universal change detection method with local adaptive sensitivity," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 359–373, Jan. 2015, ISSN: 1057-7149. DOI: 10.1109/TIP.2014.2378053.

[19] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin, "Universal background subtraction using word consensus models," *IEEE Transactions on Image Processing*, vol. 25, no. 10, pp. 4768–4781, 2016. DOI: 10.1109/TIP.2016.2598691.

[20] M. Chen, X. Wei, Q. Yang, Q. Li, G. Wang, and M. Yang, "Spatiotemporal gmm for background subtraction with superpixel hierarchy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1518–1525, Jun. 2018, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2717828.

[21] M. Chen, X. Wei, Q. Yang, Q. Li, G. Wang, and M. H. Yang, "Spatiotemporal gmm for background subtraction with superpixel hierarchy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PP, no. 99, pp. 1–1, 2017, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2717828.

[22] M. Chen, Q. Yang, Q. Li, G. Wang, and M.-H. Yang, "Spatiotemporal background subtraction using minimum spanning tree and optical flow," in *Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 521–534, ISBN: 978-3-319-10584-0. DOI: `10.1007/978-3-319-10584-0_34`.

[23] Y. Chen, J. Wang, and H. Lu, "Learning sharable models for robust background subtraction," in *IEEE Int. Conf. Multimedia and Expo (ICME)*, Jun. 2015, pp. 1–6. DOI: `10.1109/ICME.2015.7177419`.

[24] F. Cheng, S. Huang, and S. Ruan, "Advanced background subtraction approach using laplacian distribution model," in *IEEE International Conference on Multimedia and Expo*, 2010, pp. 754–759.

[25] L. Cheng and M. Gong, "Realtime background subtraction from dynamic scenes," in *Int. Conf. Comput. Vis. (ICCV)*, Sep. 2009, pp. 2066–2073. DOI: `10.1109/ICCV.2009.5459454`.

[26] C. Cuevas and N. García, "Improved background modeling for real-time spatio-temporal non-parametric moving object detection strategies," *Image and Vision Computing*, vol. 31, no. 9, pp. 616–630, 2013, ISSN: 0262-8856. DOI: `https://doi.org/10.1016/j.imavis.2013.06.003`.

[27] C. Cuevas, E. M. Yáñez, and N. García, "Labeled dataset for integral evaluation of moving object detection algorithms: Lasiesta," *Computer Vision and Image Understanding*, vol. 152, pp. 103–117, 2016, ISSN: 1077-3142. DOI: `https://doi.org/10.1016/j.cviu.2016.08.005`.

[28] D. Culibrk, O. Marques, D. Socek, H. Kalva, and B. Furht, "Neural network approach to background modeling for video object segmentation," *IEEE Trans. Neural Networks*, vol. 18, no. 6, pp. 1614–1627, Nov. 2007, ISSN: 1045-9227. DOI: `10.1109/TNN.2007.896861`.

[29] A. Dasgupta and S. Singh, "A fully convolutional neural network based structured prediction approach towards the retinal vessel segmentation," in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, Apr. 2017, pp. 248–251.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE conference on computer vision and pattern recognition*, 2009, pp. 248–255.

[31] J. J. Dhule, N. B. Dhurpate, S. S. Gonge, and G. M. Kandalkar, "Edge detection technique used for identification of cracks on vertical walls of the building," pp. 263–268, 2015.

[32] B.-H. Do and S.-C. Huang, "Dynamic background modeling based on radial basis function neural networks for moving object detection," in *IEEE Int. Conf. Multimedia and Expo (ICME)*, Jul. 2011, pp. 1–4. DOI: `10.1109/ICME.2011.6012085`.

[33] A. K. Dongdong Zeng Ming Zhu, "Combining background subtraction algorithms with convolutional neural network," *Journal of Electronic Imaging*, vol. 28, no. 1, pp. 1 - 6 –6, 2019. DOI: 10.1117/1.JEI.28.1.013011.

[34] S. E. Ebadi and E. Izquierdo, "Foreground segmentation with tree-structured sparse rpca," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 9, pp. 2273–2280, Sep. 2018, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2745573.

[35] N. Eladawi, M. Elmogy, L. Fraiwan, F. Pichi, M. Ghazal, A. Aboelfetouh, A. Riad, R. Keynton, S. Schaal, and A. El-Baz, "Early diagnosis of diabetic retinopathy in octa images based on local analysis of retinal blood vessels and foveal avascular zone," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 3886–3891.

[36] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Eur. Conf. Comput. Vis. (ECCV)*, 2000, pp. 751–767.

[37] B. S. Everitt, "Finite mixture distributions," *Wiley StatsRef: Statistics Reference Online*, 2014.

[38] R. Fan, M. J. Bocus, Y. Zhu, J. Jiao, L. Wang, F. Ma, S. Cheng, and M. Liu, "Road crack detection using deep convolutional neural network and adaptive thresholding," pp. 474–479, Jun. 2019, ISSN: 1931-0587. DOI: 10.1109/IVS.2019.8814000.

[39] M. M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. R. Rudnicka, C. G. Owen, and S. A. Barman, "An ensemble classification-based approach applied to retinal blood vessel segmentation," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 9, pp. 2538–2548, 2012.

[40] H. Fu, Y. Xu, S. Lin, D. W. Kee Wong, and J. Liu, "Deepvessel: Retinal vessel segmentation via deep learning and conditional random field," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, Eds., Cham: Springer International Publishing, 2016, pp. 132–139, ISBN: 978-3-319-46723-8.

[41] B. Gao, C. Xing, C. Xie, J. Wu, and X. Geng, "Deep label distribution learning with label ambiguity," *IEEE Transactions on Image Processing*, pp. 2825–2838, 2017.

[42] B. Garcia-Garcia, T. Bouwmans, and A. J. Rosales Silva, "Background subtraction in real applications: Challenges, current models and future directions," *Computer Science Review*, vol. 35, p. 100 204, 2020, ISSN: 1574-0137.

[43] Y. Ge, J. Zhang, X. Ren, C. Zhao, J. Yang, and A. Basu, "Deep variation transformation network for foreground detection," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2020. DOI: 10.1109/TCSVT.2020.3042559.

[44] G. Gemignani and A. Rozza, "A robust approach for the background subtraction based on multi-layered self-organizing maps," *IEEE Transactions on Image Processing*, pp. 5239–5251, 2016.

[45] X. Geng, "Label distribution learning," *IEEE Trans. Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1734–1748, Jul. 2016, ISSN: 1041-4347. DOI: 10.1109/TKDE.2016.2545658.

[46] J. H. Giraldo, S. Javed, and T. Bouwmans, "Graph moving object segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020. DOI: 10.1109/TPAMI.2020.3042093.

[47] E. Goceri, Z. K. Shah, and M. N. Gurcan, "Vessel segmentation from abdominal magnetic resonance images: Adaptive and reconstructive approach," *International journal for numerical methods in biomedical engineering*, vol. 33, no. 4, e2811, 2017.

[48] K. Goyal and J. Singhai, "Review of background subtraction methods using gaussian mixture model for video surveillance systems," *Artificial Intelligence Review*, Jan. 2017, ISSN: 1573-7462. DOI: 10.1007/s10462-017-9542-x.

[49] M. D. Gregorio and M. Giordano, "Change detection with weightless neural networks," in *IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2014, pp. 409–413. DOI: 10.1109/CVPRW.2014.66.

[50] T. S. F. Haines and T. Xiang, "Background subtraction with dirichletprocess mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 670–683, Apr. 2014, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2013.239.

[51] T. S. F. Haines and T. Xiang, "Background subtraction with dirichlet processes," in *Eur. Conf. Comput. Vis. (ECCV)*, Springer Berlin Heidelberg, 2012, pp. 99–113, ISBN: 978-3-642-33765-9.

[52] B. Han, D. Comaniciu, and L. Davis, "Sequential kernel density approximation through mode propagation: Applications to background modeling," in *Asian Asian Conf. Comput. Vis. (ACCV)*, vol. 4, 2004, pp. 818–823.

[53] B. Han and L. Davis, "Density-based multifeature background subtraction with support vector machine," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 5, pp. 1017–1023, May 2012, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2011.243.

[54] M. ul Hassan, "Vgg16 convolutional network for classification and detection," *Neurohive. Dostopno 2019]*, 2018.

[55] M. S. Hassouna, A. Farag, S. Hushek, and T. Moriarty, "Cerebrovascular segmentation from tof using stochastic models," *Medical Image Analysis*, vol. 10, no. 1, pp. 2–18, 2006, ISSN: 1361-8415.

[56] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proc. IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[57] X. He and Y. Peng, "Weakly supervised learning of part selection model with spatial constraints for fine-grained image classification," in *AAAI Conf. on Artificial Intelligence*, 2017.

[58] X. He, Y. Peng, and J. Zhao, "Which and how many regions to gaze: Focus discriminative regions for fine-grained visual categorization," *Int. Jnl. Comput. Vis.*, vol. 127, no. 9, pp. 1235–1255, Sep. 2019, ISSN: 1573-1405. DOI: 10.1007/s11263-019-01176-2.

[59] Z. He, X. Li, Z. Zhang, F. Wu, X. Geng, Y. Zhang, M. Yang, and Y. Zhuang, "Data-dependent label distribution learning for age estimation," *IEEE Transactions on Image Processing*, 2017. DOI: 10.1109/TIP.2017.2655445.

[60] P. Hou, X. Geng, Z.-W. Huo, and J. Lv, "Semi-supervised adaptive label distribution learning for facial age estimation," in *AAAI*, 2017.

[61] Y. Hu and C.-x. Zhao, "A novel lbp based methods for pavement crack detection," vol. 5, no. 1, pp. 140–147, 2010.

[62] K. Huang, H. D. Cheng, Y. Zhang, B. Zhang, P. Xing, and C. Ning, "Medical knowledge constrained semantic breast ultrasound image segmentation," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 1193–1198.

[63] R. Huang, M. Zhou, Y. Xing, Y. Zou, and W. Fan, "Change detection with various combinations of fluid pyramid integration networks," *Neurocomputing*, vol. 437, pp. 84–94, 2021, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.01.030.

[64] Z. Huo, X. Yang, C. Xing, Y. Zhou, P. Hou, J. Lv, and X. Geng, "Deep age distribution learning for apparent age estimation," in *Proceedings of the Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jun. 2016.

[65] T. Huynh-The, O. Banos, S. Lee, B. H. Kang, E. S. Kim, and T. Le-Tien, "Nic: A robust background extraction algorithm for foreground detection in dynamic scenes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 7, pp. 1478–1490, Jul. 2017, ISSN: 1051-8215. DOI: 10.1109/TCSVT.2016.2543118.

[66] M. M. M. Islam and J. Kim, "Vision-based autonomous crack detection of concrete structures using a fully convolutional encoder–decoder network," *Sensors*, vol. 19, p. 4251, Sep. 2019. DOI: 10.3390/s19194251.

[67] S. Javed, A. Mahmood, T. Bouwmans, and S. K. Jung, "Background–foreground modeling based on spatiotemporal sparse subspace clustering," *IEEE Trans. Image Process.*, vol. 26, no. 12, pp. 5840–5854, Dec. 2017, ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2746268.

[68] 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 Trans. Image Process.*, vol. 28, no. 2, pp. 1007–1022, Feb. 2019, ISSN: 1057-7149. DOI: 10.1109/TIP.2018.2874289.

[69] X. Jia, W. Li, J. Liu, and Y. Zhang, *Label distribution learning by exploiting label correlations*, 2018.

[70] S. Jiang and X. Lu, "Wesambe: A weight-sample-based method for background subtraction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. PP, no. 99, pp. 1–1, 2017, ISSN: 1051-8215. DOI: 10.1109/TCSVT.2017.2711659.

[71] C. R. Jung, "Efficient background subtraction and shadow removal for monochromatic video sequences," *IEEE Trans. Multimedia*, vol. 11, no. 3, pp. 571–577, Apr. 2009, ISSN: 1520-9210. DOI: 10.1109/TMM.2009.2012924.

[72] Z. Kang, C. Peng, and Q. Cheng, "Robust pca via nonconvex rank approximation," in *Proc. IEEE International Conference on Data Mining*, 2015, pp. 211–220.

[73] R. Kapela, P. Sniatała, A. Turkot, A. Rybarczyk, A. Po˙zarycki, P. Rydzewski, M.Wyczałek, and A. Błoch, "Asphalt surfaced pavement cracks detection based on histograms of oriented gradients," pp. 579–584, 2015.

[74] V. Kaul, A. Yezzi, and Y. Tsai, "Detecting curves with unknown endpoints and arbitrary topology using minimal paths," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 1952–1965, Oct. 2012, ISSN: 1939-3539.

[75] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis, "Background modeling and subtraction by codebook construction," in *Int. Conf. Image Process. (ICIP)*, Oct. 2004, pp. 3061–3064. DOI: 10.1109/ICIP.2004.1421759.

[76] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations*, 2015.

[77] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[78] Y. LeCun, Y. Bengio, and H. G, "Deep learning," *Nature 521*, pp. 436–444, 2015. DOI: `https://doi.org/10.1038/nature14539`.

[79] D.-S. Lee, "Effective gaussian mixture learning for video background subtraction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 827–832, 2005. DOI: `10.1109/TPAMI.2005.102`.

[80] D. Li, L. Xu, and E. Goodman, "A fast foreground object detection algorithm using kernel density estimation," in *IEEE Int. Conf. Signal Process.*, Oct. 2012, pp. 703–707. DOI: `10.1109/ICoSP.2012.6491583`.

[81] L. Li, Q. Hu, and X. Li, "Moving object detection in video via hierarchical modeling and alternating optimization," *IEEE Transactions on Image Processing*, pp. 2021–2036, 2019.

[82] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian, "Statistical modeling of complex backgrounds for foreground object detection," *IEEE Trans. Image Process.*, vol. 13, no. 11, pp. 1459–1472, Nov. 2004, ISSN: 1057-7149. DOI: `10.1109/TIP.2004.836169`.

[83] D. Liang, S. Kaneko, M. Hashimoto, K. Iwata, and X. Zhao, "Co-occurrence probability-based pixel pairs background model for robust object detection in dynamic scenes," *Pattern Recognition*, vol. 48, no. 4, pp. 1374–1390, 2015, ISSN: 0031-3203. DOI: `http://dx.doi.org/10.1016/j.patcog.2014.10.020`.

[84] X. Liang, S. Liao, X. Wang, W. Liu, Y. Chen, and S. Z. Li, "Deep background subtraction with guided learning," in *IEEE Int. Conf. Multimedia and Expo (ICME)*, Jul. 2018, pp. 1–6. DOI: `10.1109/ICME.2018.8486556`.

[85] K. Lim Long Ang and H. Yalim, "Learning multi-scale features for foreground segmentation," *Pattern Analysis and Applications*, vol. 23, no. 3, pp. 1369–1380, 2020.

[86] L. A. Lim and H. Y. Keles, "Foreground segmentation using convolutional neural networks for multiscale feature encoding," *Pattern Recognit. Letters*, vol. 112, pp. 256–262, 2018, ISSN: 0167-8655. DOI: `https://doi.org/10.1016/j.patrec.2018.08.002`.

[87] ——, "Learning multi-scale features for foreground segmentation," *Pattern Analysis and Applications*, vol. 23, no. 3, pp. 1369–1380, 2020.

[88] H.-H. Lin, T.-L. Liu, and J.-H. Chuang, "A probabilistic svm approach for background scene initialization," in *Int. Conf. Image Process. (ICIP)*, vol. 3, Jun. 2002, 893–896 vol.3. DOI: `10.1109/ICIP.2002.1039116`.

[89] P. Liu and R. Fang, "Learning pixel-distribution prior with wider convolution for image denoising," *CoRR*, vol. abs/1707.09135, 2017. arXiv: 1707.09135.

[90] X. Liu, J. Yao, X. HONG, X. Huang, Z. Zhou, C. Qi, and G. Zhao, "Background subtraction using spatio-temporal group sparsity recovery," *IEEE Trans. Circuits Syst. Video Technol.*, vol. PP, no. 99, pp. 1–1, 2017, ISSN: 1051-8215. DOI: 10.1109/TCSVT.2017.2697972.

[91] Z. Liu, K. Huang, and T. Tan, "Foreground object detection using top-down information based on em framework," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 4204–4217, Sep. 2012, ISSN: 1057-7149. DOI: 10.1109/TIP.2012.2200492.

[92] Z. Liu, Y. Song, C. Maere, Q. Liu, Y. Zhu, H. Lu, and D. Yuan, "A method for pet-ct lung cancer segmentation based on improved random walk," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 1188–1192.

[93] J. Long, E. Shelhamer, and T. Darrell., "Fully convolutional networks for semantic segmentation," 2015.

[94] M. Lucia and P. Alfredo, "Towards benchmarking scene background initialization," in *New Trends in Image Analysis and Processing*, 2015, pp. 469–476.

[95] Y. Luo, L. Yang, L. Wang, and H. Cheng, "Efficient cnn-crf network for retinal image segmentation," in *Cognitive Systems and Signal Processing*, F. Sun, H. Liu, and D. Hu, Eds., Singapore: Springer Singapore, 2017, pp. 157–165, ISBN: 978-981-10-5230-9.

[96] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Trans. Image Process.*, vol. 17, no. 7, pp. 1168–1177, Jul. 2008, ISSN: 1057-7149. DOI: 10.1109/TIP.2008.924285.

[97] L. Maddalena and A. Petrosino, "The sobs algorithm: What are the limits?" In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012, pp. 21–26. DOI: 10.1109/CVPRW.2012.6238922.

[98] 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. DOI: 10.1109/TITS.2020.3030801.

[99] 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. DOI: 10.1109/TIP.2020.3037472.

[100] S. Minaee and Y. Wang, "An admm approach to masked signal decomposition using subspace representation," *IEEE Trans. Image Process.*, vol. 28, no. 7, pp. 3192–3204, Jul. 2019, ISSN: 1057-7149. DOI: 10.1109/TIP.2019.2894966.

[101] T. Minematsu, A. Shimada, and R.-i. Taniguchi, "Rethinking background and foreground in deep neural network-based background subtraction," in *International Conference on Image Processing*, 2020, pp. 3229–3233. DOI: 10.1109/ICIP40778.2020.9191151.

[102] S. Moccia, E. De Momi, S. El Hadji, and L. S. Mattos, "Blood vessel segmentation algorithms—review of methods, datasets and evaluation metrics," *Computer methods and programs in biomedicine*, vol. 158, pp. 71–91, 2018.

[103] "Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2010, pp. 1301–1306, ISBN: 9781424469840. DOI: 10.1109/CVPR.2010.5539817.

[104] V. M. Mondéjar-Guerra, J. Rouco, J. Novo, and M. Ortega, "An end-to-end deep learning approach for simultaneous background modeling and subtraction.," in *Proc. British Machine Vision Conference*, 2019.

[105] D. A. Oliveira, R. Q. Feitosa, and M. M. Correia, "Segmentation of liver, its vessels and lesions from ct images for surgical planning," *Biomedical engineering online*, vol. 10, no. 1, p. 30, 2011.

[106] Ç. Özgenel and A. Sorguc, "Performance comparison of pretrained convolutional neural networks on crack detection in buildings," Jul. 2018. DOI: 10.22260/ISARC2018/0094.

[107] Ç. F. Özgenel, "Concrete crack images for classification," 2019.

[108] P. W. Patil, A. Dudhane, A. Kulkarni, S. Murala, A. B. Gonde, and S. Gupta, "An unified recurrent video object segmentation framework for various surveillance environments," *IEEE Transactions on Image Processing*, vol. 30, pp. 7889–7902, 2021. DOI: 10.1109/TIP.2021.3108405.

[109] P. W. Patil, A. Dudhane, and S. Murala, "Multi-frame recurrent adversarial network for moving object segmentation," in *Proc. IEEE Winter Conference on Applications of Computer Vision*, 2021.

[110] Y. Peng, X. He, and J. Zhao, "Object-part attention model for fine-grained image classification," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1487–1500, Mar. 2018, ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2774041.

[111] G. Ramírez-Alonso and M. I. Chacón-Murguía, "Auto-adaptive parallel som architecture with a modular analysis for dynamic object segmentation in videos," *Neurocomputing*, vol. 175, pp. 990–1000, 2016, ISSN: 0925-2312. DOI: `http://dx.doi.org/10.1016/j.neucom.2015.04.118`.

[112] S. M. Roy and A. Ghosh, "Foreground segmentation using adaptive 3 phase background model," *IEEE Transactions on Intelligence Transportation Systems*, pp. 2287–2296, 2020.

[113] H. Sajid and S. C. S. Cheung, "Universal multimode background subtraction," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3249–3260, Jul. 2017, ISSN: 1057-7149. DOI: `10.1109/TIP.2017.2695882`.

[114] R. Shen, I. Cheng, and A. Basu, "Qoe-based multi-exposure fusion in hierarchical multivariate gaussian crf," *IEEE Transactions on Image Processing*, vol. 22, no. 6, pp. 2469–2478, 2012.

[115] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, "Automatic road crack detection using random structured forests," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3434–3445, 2016.

[116] Z. Shi, H. Xie, J. Zhang, J. Liu, and L. Gu, "Vessel enhancement based on length-constrained hessian information," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 2869–2874.

[117] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[118] A. Sironi, V. Lepetit, and P. Fua, "Multiscale centerline detection by learning a scale-space distance transform," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014.

[119] S. Sitara and K. Srinivasan, "Comparative analysis of deep neural networks for crack image classification," in. Jan. 2020, pp. 434–443, ISBN: 978-3-030-34079-7. DOI: `10.1007/978-3-030-34080-3_49`.

[120] M. D. Springer, "The algebra of random variables," Tech. Rep., 1979.

[121] P. M. Sriram Varadarajan and H. Zhou, "Region-based mixture of gaussians modelling for foreground detection in dynamic scenes," *Pattern Recognition*, vol. 48, no. 11, pp. 3488–3503, 2015, ISSN: 0031-3203.

[122] J. Staal, M. D. Abramoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken, "Ridge-based vessel segmentation in color images of the retina," *IEEE Transactions on Medical Imaging*, vol. 23, no. 4, pp. 501–509, Apr. 2004, ISSN: 1558-254X.

[123] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2, 1999, -252 Vol. 2. DOI: `10.1109/CVPR.1999.784637`.

[124] M. Sultana, A. Mahmood, T. Bouwmans, and S. K. Jung, "Dynamic background subtraction using least square adversarial learning," in *Proc. IEEE Conference on Image Processing*, 2020. DOI: 10.1109/ICIP40778.2020.9191235.

[125] M. Sultana, A. Mahmood, T. Bouwmans, and S. K. Jung, "Unsupervised adversarial learning for dynamic background modeling," in *Frontiers of Computer Vision*, 2020, pp. 248–261, ISBN: 978-981-15-4818-5.

[126] 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. DOI: 10.1109/ACCESS.2021.3071163.

[127] O. Tezcan, P. Ishwar, and J. Konrad, "Bsuv-net: A fully-convolutional neural network for background subtraction of unseen videos," in *Proc. IEEE Winter Conference on Applications of Computer Vision*, 2020.

[128] P. Tiefenbacher, M. Hofmann, D. Merget, and G. Rigoll, "Pid-based regulation of background dynamics for foreground segmentation," in *IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 3282–3286. DOI: 10.1109/ICIP.2014.7025664.

[129] S. Varadarajan, P. Miller, and H. Zhou, "Spatial mixture of Gaussians for dynamic background modelling," in *Proc. IEEE Int. Conf. Advanced Video and Signal Based Surveillance*, 2013, pp. 63–68, ISBN: 9781479907038. DOI: 10.1109/AVSS.2013.6636617.

[130] S. Varadharajan, S. Jose, K. Sharma, L. Wander, and C. Mertz, "Vision for road inspection," *In Proceedings of the IEEE Winter Conference on Applications of Computer Vision, Steamboat Springs, CO, USA, 24–26*, pp. 115–122, Mar. 2014.

[131] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *ACM Int. Conf. Multimedia*, 2015, pp. 689–692.

[132] R. Vega, G. Sanchez-Ante, L. E. Falcon-Morales, H. Sossa, and E. Guevara, "Retinal vessel extraction using lattice neural networks with dendritic processing," *Computers in Biology and Medicine*, vol. 58, pp. 20–30, 2015, ISSN: 0010-4825.

[133] R. Wang, F. Bunyak, G. Seetharaman, and K. Palaniappan, "Static and moving object detection using flux tensor with split gaussian models," in *IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2014, pp. 420–424. DOI: 10.1109/CVPRW.2014.68.

[134] S. Wang, Y. Yin, G. Cao, B. Wei, Y. Zheng, and G. Yang, "Hierarchical retinal blood vessel segmentation based on feature and ensemble learning," *Neurocomputing*, vol. 149, pp. 708–717, 2015, ISSN: 0925-2312.

[135] Y. Wang, P. M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, "Cdnet 2014: An expanded change detection benchmark dataset," in *IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2014, pp. 393–400. DOI: 10.1109/CVPRW.2014.126.

[136] Y. Wang, Z. Luo, and P.-M. Jodoin, "Interactive deep learning method for segmenting moving objects," *Pattern Recognit. Letters*, vol. 96, pp. 66–75, 2016, ISSN: 0167-8655.

[137] K. W. Wenbo Zheng and F.-Y. Wang, "A novel background subtraction algorithm based on parallel vision and bayesian gans," *Neurocomputing*, 2019, ISSN: 0925-2312.

[138] X. Wu, J. Ma, Y. Sun, C. Zhao, and A. Basu, "Multi-scale deep pixel distribution learning for concrete crack detection," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 6577–6583. DOI: 10.1109/ICPR48806.2021.9413312.

[139] S. Xie and Z. Tu, "Holistically-nested edge detection," pp. 39–1403. 2015.

[140] X. Xie and Z. Wang, "Multi-scale semantic segmentation enriched features for pedestrian detection," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 2196–2201.

[141] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling, "Feature pyramid and hierarchical boosting network for pavement crack detection," Jan. 2019.

[142] Y. Yang, T. Zhang, J. Hu, D. Xu, and G. Xie, "End-to-end background subtraction via a multi-scale spatio-temporal model," *IEEE Access*, vol. 7, pp. 97 949–97 958, 2019. DOI: 10.1109/ACCESS.2019.2930319.

[143] H. Yong, D. Meng, W. Zuo, and L. Zhang, "Robust online matrix factorization for dynamic background subtraction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 7, pp. 1726–1740, Jul. 2018, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2732350.

[144] D. Zeng and M. Zhu, "Background subtraction using multiscale fully convolutional network," *IEEE Access*, vol. 6, pp. 16 010–16 021, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2817129.

[145] Z. Zeng, J. Jia, D. Yu, Y. Chen, and Z. Zhu, "Pixel modeling using histograms based on fuzzy partitions for dynamic background subtraction," *IEEE Trans. Fuzzy Systems*, vol. 25, no. 3, pp. 584–593, Jun. 2017, ISSN: 1063-6706. DOI: 10.1109/TFUZZ.2016.2566811.

[146] J. Zhang and Y. Peng, "Object-aware aggregation with bidirectional temporal graph for video captioning," in *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8327–8336.

[147] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Image Processing (ICIP), 2016 IEEE International Conference on*, IEEE, 2016, pp. 3708–3712.

[148] X. Zhang, Z. Liu, H. Li, X. Zhao, and P. Zhang, "Statistical background subtraction based on imbalanced learning," in *IEEE Int. Conf. Multimedia and Expo (ICME)*, Jul. 2014, pp. 1–6. DOI: 10.1109/ICME.2014.6890245.

[149] X. Zhang, C. Zhu, H. Wu, Z. Liu, and Y. Xu, "An imbalance compensation framework for background subtraction," *IEEE Trans. Multimedia*, vol. 19, no. 11, pp. 2425–2438, Nov. 2017, ISSN: 1520-9210. DOI: 10.1109/TMM.2017.2701645.

[150] X. Zhang, H. Wu, M. Wu, and C. Wu, "Extended motion diffusion-based change detection for airport ground surveillance," *IEEE Transactions on Image Processing*, vol. 29, pp. 5677–5686, 2020. DOI: 10.1109/TIP.2020.2984854.

[151] C. Zhao and A. Basu, "Dynamic deep pixel distribution learning for background subtraction," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2019, ISSN: 1558-2205.

[152] C. Zhao, T. Cham, X. Ren, J. Cai, and H. Zhu, "Background subtraction based on deep pixel distribution learning," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, Jul. 2018, pp. 1–6.

[153] C. Zhao and A. Basu, "Pixel distribution learning for vessel segmentation under multiple scales," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, 2021, pp. 2717–2721. DOI: 10.1109/EMBC46164.2021.9629614.

[154] C. Zhao, K. Hu, and A. Basu, "Universal background subtraction based on arithmetic distribution neural network," *IEEE Transactions on Image Processing*, pp. 1–1, 2022. DOI: 10.1109/TIP.2022.3162961.

[155] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM computing surveys (CSUR)*, vol. 35, no. 4, pp. 399–458, 2003.

[156] W. Zheng, K. Wang, and F.-Y. Wang, "A novel background subtraction algorithm based on parallel vision and bayesian gans," *Neurocomputing*, pp. 178–200, 2020, ISSN: 0925-2312.

[157] J. Zhou, P. S. Huang, and F.-P. Chiang, "Wavelet-based pavement distress detection and evaluation," vol. 45, no. 2, pp. 007–027, 2006.

[158] X. Zhou, C. Yang, and W. Yu, "Moving object detection by detecting contiguous outliers in the low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 3, pp. 597–610, Mar. 2013, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.132.

[159] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Int. Conf. Pattern Recognit. (ICPR)*, vol. 2, 2004, ISBN: 0-7695-2128-2. DOI: 10.1109/ICPR.2004.1333992.

[160] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "Cracktree: Automatic crack detection from pavement images," vol. 33, no. 3, pp. 227–238, 2012.