Universal Background Subtraction based on Arithmetic Distribution Neural Network

Chenqiu Zhao, Student Member, IEEE Kangkang Hu and Anup Basu, Senior Member, IEEE

Abstract—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.

Index Terms—Background Subtraction, Deep Learning, Distribution Learning, Arithmetic Distribution Operations

I. 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. Moreover, several excellent networks for background subtraction perform poorly for unseen videos, because of their dependence on the scene information in training videos. Finally, 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 [1]) 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 paper, we focus on learning the distribution of comparisons to classify pixels into foreground or background, based on a new Arithmetic Distribution Neural Network (ADNN), as shown in Fig. 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

Fig. 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.
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 of this paper are:

- 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 V-A; 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 V-C.

II. RELATED WORK

In this section, we outline background subtraction algorithms briefly into two categories. In Section II-A, the algorithms without using a deep learning network are introduced. Then, several excellent deep learning networks for background subtraction are discussed in Section II-B.

A. Non-Deep-Learning Algorithms

Background subtraction is a fundamental research topic in computer vision [2]–[5], and a large number of methods have been proposed on this topic [6]–[43]. Throughout the development of background subtraction algorithms, the distribution of temporal pixels has played an important role since it is a good representation of background information. In particular, the Gaussian mixture model proposed by Zivkovic et al. [8] is one of the most popular techniques, where the background observations are described by several Gaussian functions [8], [23], with a large number of extensions being proposed [21], [22]. For example, Lee et al. [7] utilized an adaptive learning rate for each Gaussian function to improve the convergence rate during clustering. Haines et al. [41] [42] used the Dirichlet processes with Gaussian mixture models to analyze pixel distributions. Recently, Chen et al. [33] [34] used Gaussian mixture models to represent the vertices of spanning trees and Akilan et al. [35] 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 [36], kernel density estimation [37] and artificial neural networks [38].

In addition, several excellent publications [19], [25]–[27] 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. [26], [27] utilized robust principal component analysis [25] to separate the background scenes based on the spatial and temporal subspaces. Yong et al. [19] proposed online matrix factorization for background subtraction. Machine-learning techniques have also been utilized for background subtraction [44]–[54]. Lin et al. [44] classified the pixels using a probabilistic support vector machine. Similarly, Han et al. [46] used density-based features into a classifier utilizing a support vector machine. Li et al. [45] formulated background subtraction as minimizing a constrained risk function and Culibrk et al. [49] proposed an unsupervised Bayesian classifier using a neural network architecture for background subtraction. Unfortunately, given the complexity and diversity of natural scenes, artificial models are not adequate to generate promising classifications of pixels for all videos captured from natural scenes. Thus, recent research has focused on using deep learning networks to perform background subtraction automatically.

B. Algorithms based on Deep Learning

Since convolutional neural networks have demonstrated an excellent ability to learn scene information, several approaches [1], [17], [52], [55]–[85] have used deep learning networks to learn the background scenes for subtraction. For example, Wang et al. [57] proposed a fully connected network to learn the background scenes. Zeng et al. [60] utilized a multi-scale strategy to improve the results. Similarly, Lim et al. [1] used a triplet convolutional neural network to extract multi-scale features from background scenes and Yang et al. [63] 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, Babaei et al. [61] 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. [59] utilized the foreground mask generated by the SubSENSE algorithm [17] rather than manual labeling for training, and Zeng et al. [64] 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. [73] incorporated temporal information by using a foreground saliency reinforcement block, and proposed the 3DCD network for unseen videos. Similarly, Tezcan et al. [76, 77] trained a U-net architecture to do subtractions between background images and current frames, and Giraldo [75] 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 this work, we focus on learning the distributions of temporal pixels for background subtraction by the proposed 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.

Although the focus of the proposed approach is related to the D-DPDL method proposed by Zhao et al. [66], there are several differences between our ADNN and the D-DPDL method proposed in [66]. The main difference between the proposed ADNN and D-DPDL is the mathematical method used for learning distributions. D-DPDL devised a convolutional neural network and the entries of input patches were randomly permuted to force the network to learn distributions. In the proposed ADNN, arithmetic distribution layers are proposed for learning distributions, which demonstrate better ability to classify distributions compared to the convolutional layers in the experiments presented in Section III-A. In addition, the input of the proposed ADNN and D-DPDL are also completely different. In D-DPDL, the input of the network is patches of temporal pixels that are randomly permuted. By contrast, the input of the proposed ADNN is the histograms of subtractions between current pixels and their historical counterparts, since the histogram is a proper discrete representation of the probability density function which is used for computations in the proposed arithmetic distribution layers. In addition, the number of temporal pixels is limited by the size of patches in the D-DPDL method, but the information of all temporal pixels at a particular location can be compressed into one histogram, which has better global information. Finally, there is a big difference between the number of parameters used in D-DPDL and the proposed ADNN. According to the network architecture published in [66], there are around 7 million learning parameters in the D-DPDL model. In contrast, as shown in Table I the ADNN devised for comparisons with state-of-the-art methods in this paper contains only around 0.1 million parameters.

III. 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 [86] 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 formulæ 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 \( \bar{x}, \bar{w}, \bar{b} \) and \( \bar{z} \) 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 \( \bar{x}, \bar{w}, \bar{b} \) and \( \bar{z} \), respectively, where \( n, i, k, j \) are indices. \( \text{loss} \) is the final scalar output of the network using the arithmetic distribution layers. During training, we want to minimize the value of \( \text{loss} \) to approach 0 if possible.
\[ \delta S = \frac{\partial \text{loss}}{\partial z} \] 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 \( Z = X W \). In order to implement the product distribution layer, the expressions for \( f_Z(z) \) and the gradient of \( f_Z(z) \) 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:

\[
F_Z(z) \overset{\text{def}}{=} P(Z \leq z) = P(XW \leq z) \\
= P(XW \leq z, W \geq 0^+) + P(XW \leq z, W \leq 0^-) \\
= P(X \leq \frac{z}{W}, W \geq 0^+) + P(X \geq \frac{z}{W}, W \leq 0^-) \\
\therefore XW \leq z, W \in [-\infty 0^-] \cup [0^+ \infty] \\
\Rightarrow X \leq \frac{z}{W}, W \geq 0^+ \text{ or } X \geq \frac{z}{W}, W \leq 0^- 
\]

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:

\[
F_Z(z) = \int_{0^-}^{\infty} f_X(w) \int_{-\infty}^{\frac{z}{w}} f_W(x) dx dw + \int_{0^+}^{\infty} f_X(w) \int_{\frac{z}{w}}^{\infty} f_W(x) dx dw \\
\therefore P(X \leq \frac{z}{W}, W \geq 0^+) = \int_{0^-}^{\infty} f_X(w) \int_{-\infty}^{\frac{z}{w}} f_W(x) dx dw \\
P(X \geq \frac{z}{W}, W \leq 0^-) = \int_{0^+}^{\infty} f_X(w) \int_{\frac{z}{w}}^{\infty} f_W(x) dx dw, 
\]

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:

\[
\frac{d}{dz} \left( \int_{a(z)}^{b(z)} f(z, x) dx \right) = f(z, b(z)) \cdot \frac{d}{dz} b(z) \\
- f(z, a(z)) \cdot \frac{d}{dz} a(z) + \int_{a(z)}^{b(z)} \frac{\partial}{\partial z} f(z, x) dx.
\]

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:

\[
f_Z(z) = \frac{d(F_Z(z))}{dz} \\
= d \left( \int_{0^-}^{\infty} f_W(w) \int_{-\infty}^{\frac{z}{w}} f_X(x) dx dw + \int_{0^+}^{\infty} f_W(w) \int_{\frac{z}{w}}^{\infty} f_X(x) dx dw \right) / dz \\
= \int_{0^-}^{\infty} f_W(w) \frac{df}{dz} f_X(x) dx dw + \int_{0^+}^{\infty} f_W(w) \frac{df}{dz} f_X(x) dx dw \\
= \int_{0^-}^{\infty} f_W(w) f_X\left(\frac{z}{w}\right) \frac{1}{w} dw + \int_{0^+}^{\infty} f_W(w) f_X\left(\frac{z}{w}\right) \frac{1}{w} dw \\
= \int_{0^-}^{\infty} f_W(w) f_X\left(\frac{z}{w}\right) \frac{1}{w} dw \\
\]

\[
\therefore \frac{df}{dz} f_X(x) dx dw = f_X\left(\frac{z}{w}\right) \frac{1}{w} - f_X(z) \frac{1}{w} + 0 + 0 \\
\Rightarrow z_j = \sum_{i=-\infty}^{\infty} w_i f_X\left(\frac{z_j}{i}\right) \frac{1}{|i|} \cdot 1 \quad \because dw = 1, \ f_W(i) = w_i, 
\]

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:

\[
\frac{\partial \text{loss}}{\partial w_i} = \frac{\partial \text{loss}}{\partial Z} \cdot \frac{\partial Z}{\partial w_i} = \sum_{j=-\infty}^{\infty} \frac{\partial \text{loss}}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_i} \\
= \sum_{j=-\infty}^{\infty} \frac{\partial \text{loss}}{\partial z_j} \cdot \frac{\partial \left( \sum_{i=-\infty}^{\infty} w_i f_X\left(\frac{z_j}{i}\right) \frac{1}{|i|} \right)}{\partial w_i} \\
= \sum_{j=-\infty}^{\infty} \delta z_j f_X\left(\frac{z_j}{i}\right) \frac{1}{|i|} \\
\Rightarrow \delta w_i = \sum_{j=-\infty}^{\infty} \delta z_j f_X\left(\frac{z_j}{i}\right) \frac{1}{|i|}, 
\]

where \( \delta w_i \) and \( \delta z_j \) 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:

$$f_Z(z) = \sum_{i=-\infty}^{\infty} f_B(b) f_X(z - b) \cdot db$$

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:

$$\frac{\partial \text{loss}}{\partial b_k} = \frac{\partial \text{loss}}{\partial Z} \cdot \frac{\partial Z}{\partial b_k} = \sum_{j=-\infty}^{\infty} \frac{\partial \text{loss}}{\partial z_j} \cdot \frac{\partial z_j}{\partial b_k}$$

$$= \sum_{j=-\infty}^{\infty} \frac{\partial \text{loss}}{\partial z_j} \cdot \delta_{z_j} f_X(z_j - k)$$

$$\Rightarrow \frac{\partial \text{loss}}{\partial b_k} = \sum_{j=-\infty}^{\infty} \delta_{z_j} f_X(z_j - k),$$

where $\delta_{z_j}$ are the gradients of entries in histograms corresponding to $f_B(b)$ and $f_Z(z)$ respectively, and $k$ are indices. $\text{loss}$ is the final output of the network using the sum distribution layer.

With the help of Eqn. [4] the forward pass and the backpropagation of arithmetic distribution layers can be easily implemented in PyTorch [87]. In particular, the gradient of the learning kernels of the arithmetic distribution layers is computed and input into the “Autograd package” of PyTorch [87] for backpropagation. Because of the limit on the length of this paper, the validation experiment for the proposed arithmetic distribution layers are outlined in the appendices in the supplementary material. 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 I 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 paper.

### IV. 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. 2 with details of the network architecture presented in Table I in which the first two convolutions are 1D convolutions, since one of the values of the kernel size is 1. Starting with a given frame of a video, denoted as $I(t)$, the network architecture 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.

| Type          | Filters | Layer size | Data size |
|---------------|---------|------------|-----------|
| Input         | Product Distribution | 2 | $3 \times 201 \times 1$ |
| sum distribution | 2 | $3 \times 201 \times 2$ |
| Convolution   | 1 | $3 \times 1 \times 2$ |
| Convolution   | 512 | $1 \times 201 \times 1$ |
| Rectified linear unit | 512 | $1 \times 1 \times 1$ |
| Convolution   | 2 | $512 \times 1 \times 1$ |
| Softmax       | 2 | $2 \times 1 \times 1$ |

B: Batch size.
of these two layers is used as the input of the classification architecture. Mathematically:

$$M(x, y) = L(F_p(H_{x,y}) + F_u(H_{x,y})),$$  \hspace{1cm} (9)$$

where $H_{x,y}$ is the input histogram; $F_p$ and $F_u$ denote the product distribution and the sum distribution layer; $L$ is the convolution procedure; and $\mathcal{C}$ 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 $L, C, F_p$ and $F_u$ 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 $L$ to generate the label $M(x, y)$ 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 [66] 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 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:

$$F(I(x, y), M) = \arg \max_{a_i} P(I(x, y)|a_i) P(a_i)$$

$$= \arg \max_{a_i} P(a_i) \sum_{k=1}^{K} \pi_k \mathcal{N}_p(v_k|\mu_k, \Sigma_k)$$

$$= \arg \max_{a_i} \mathcal{N}(x|\mu_i, \Sigma_i)$$

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 through a mixture of Gaussian approximation functions $\mathcal{N}_p(v_k|\mu_k, \Sigma_k)$. 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. $\mu_k$ and $\Sigma_k$ denote the mean and variance of features of a pixel in a local rectangular range with center at $(x, y)$ and radius $R = 4$. $\pi_k$ is the weight to mix the Gaussian approximation functions $\mathcal{N}_p$ which is mathematically expressed as:

$$\mathcal{N}_p(x|\mu, \sigma) = \left\{ \begin{array}{ll} 1 + \frac{x-\mu}{n\sigma} & |x - u| \leq n\sigma \\ 0 & \text{otherwise} \end{array} \right.$$

where $\mu$ and $\sigma$ denote the mean and variance, and $n$ is a user parameter. During experiments, $n = 2$ gives us the best results. As shown in Fig. 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, since the original Bayesian refinement model involves Euclidean
distance and adaptive weights, 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. In particular, the output of the arithmetic distribution neural network is used as the initial binary mask for the convolutional layer. Thus, the proposed ADNN is better than convolutional layers in distribution classification. However, the proposed ADNN is only around 0.1 million. Thus, it is fair to claim that the proposed arithmetic distribution layers are better than convolutional layers to learn global information, the comparisons between ADNN and CNN are improved by increasing 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 V, CNN8 is the largest network with 2.9 million parameters and it is constructed purely by fully connected layers, and CNN11–14 are devised by inserting extra Relu and Gelu layers into CNN10. 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 V, we can conclude that the results of CNN3 are better than CNN2 which are better than CNN1, since when the training data is fixed, a network with more parameters is supposed to have better learning ability. In addition, the results of CNN3 are better than the ones of CNN4–7 which are devised by adding Relu and Gelu layers into CNN3. This is because Relu or Gelu drops several entires of input vectors which may result in losing useful information. In contrast, the proposed ADNN1 achieves better results on the average Fm value compared to CNN3–7, 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 ADNN2 and CNN8–14 are presented in Table V. In particular, since the learning ability of CNN and ADNN are improved by increasing 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 V, CNN8 is the largest network with 2.9 million parameters and it is constructed purely by fully connected layers, and CNN11–14 are devised by inserting Relu and Gelu in these fully connected layers. However, the proposed ADNN2 still achieves better results than CNN8–14 and the number of parameters in ADNN2 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.

A. Ablation Study

Although the comparisons between CNNs and the proposed ADNNs have been proposed in Section V-A, in which the benefit of using the proposed arithmetic distribution layers...
is demonstrated, it is still not clear how the kernel size, the filter number as well as the depth of arithmetic distribution layers contribute to the results. Thus, in this section, we devise another 7 ADNNs (ADNN_{3−9}), which are revised from ADNN_2, to compare with ADNN_2 for ablation study. The details of network architectures of ADNN_{3−9} are shown in Table VII and their quantitative comparisons are shown in Table VIII. During the evaluations of ADNN_{3−9}, the training and testing setting are the same as those of ADNN_2.

Compared to ADNN_2, ADNN_{6−9} have more parameters, but their overall Fm values are close to those of ADNN_2. This demonstrates that ADNN_2 has enough learning ability to handle the histograms of temporal pixels for background subtraction. In particular, ADNN_6 and ADNN_7 are devised by increasing the filter number and the width of arithmetic distribution layers, respectively. ADNN_9 is devised by increasing the filter number as well as the width of the arithmetic distribution layer. ADNN_9 is devised by increasing the depth of arithmetic distribution layers. Note that the overall Fm value of ADNN_9 is a little lower than ADNN_2, this decline may be from the accumulated error or the over-fitting problem, since ADNN_9 has a deeper architecture. In contrast, the number

| TABLE III |
| --- |
| 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. |

| Type | Filters Size | Type | Filters Size | Type | Filters Size | Type | Filters Size | Data Size |
|---|---|---|---|---|---|---|---|---|
| Input | | | | | | | | |
| Conv | 8 | 3 × 1 | 1 | Conv | 8 | 3 × 1 | 1 | Conv | 8 | 3 × 1 | 1 |
| Gelu | 16 | 8 × 1 | 1 | Gelu | 16 | 8 × 1 | 1 | Gelu | 16 | 8 × 1 | 1 |
| Relu | 32 | 16 × 201 | 1 | Relu | 32 | 16 × 201 | 1 | Relu | 32 | 16 × 201 | 1 |
| Conv | 128 | 32 × 1 | 1 | Conv | 128 | 32 × 1 | 1 | Conv | 128 | 32 × 1 | 1 |
| Gelu | 201 | 128 × 1 | 1 | Gelu | 201 | 128 × 1 | 1 | Gelu | 201 | 128 × 1 | 1 |
| Relu | 2 | 1 × 201 | 1 | Relu | 2 | 1 × 201 | 1 | Relu | 2 | 1 × 201 | 1 |
| NTP | 405 | NTP | 29,370 | NTP | 133,290 | NTP | 133,290 |

| Type | Filters Size | Type | Filters Size | Type | Filters Size | Type | Filters Size | Data Size |
|---|---|---|---|---|---|---|---|---|
| Input | | | | | | | | |
| Conv | 256 | 3 × 201 | 1 | Conv | 256 | 3 × 201 | 1 | Conv | 256 | 3 × 201 | 1 |
| Gelu | 512 | 256 × 1 | 1 | Gelu | 512 | 256 × 1 | 1 | Gelu | 512 | 256 × 1 | 1 |
| Relu | 1024 | 512 × 1 | 1 | Relu | 1024 | 512 × 1 | 1 | Relu | 1024 | 512 × 1 | 1 |
| Conv | 2048 | 1024 × 1 | 1 | Conv | 2048 | 1024 × 1 | 1 | Conv | 2048 | 1024 × 1 | 1 |
| Gelu | 2 | 2048 × 1 | 1 | Gelu | 2 | 2048 × 1 | 1 | Gelu | 2 | 2048 × 1 | 1 |
| Relu | 2 | 2048 × 1 | 1 | Relu | 2 | 2048 × 1 | 1 | Relu | 2 | 2048 × 1 | 1 |
| NTP | 2,910,976 | NTP | 2,910,976 | NTP | 2,910,976 | NTP | 2,910,976 |

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
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 [55], LASIESTA [78] and SBI2015 [92] datasets.

The deep learning networks compared include DeepBS [61], GuidedBS [59], CNN-SFC [63], D-DPD [66], 3DCD [73], FgSN [1], MSFS [91], DVTN [72] and BSUV [76, 77]. In particular 3DCD [73] and BSUV [76, 77] 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 [83]. 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 re-

| Videos | CNN1  | CNN2  | CNN3  | CNN4  |
|--------|-------|-------|-------|-------|
| 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, 0.9040, 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) | (0.9646, 0.9532, 0.9498) | (0.9439, 0.9663, 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, 0.7802) | (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, 0.9661, 0.8000) |
| sidewalk | (0.5490, 0.3675, 0.4403) | (0.8392, 0.3368, 0.4807) | (0.8314, 0.3578, 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, 0.9904) | (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)
TABLE V
QUANTITATIVE COMPARISON BETWEEN CNNS AND ADNN USING RE, PR, AND FM METRICS BASED ON SEEN AND UNSEEN VIDEOS.

| Training Sets | Testing Sets | CNN8 | CNN9 | CNN10 | CNN11 |
|---------------|--------------|------|------|-------|-------|
| Category Video | Category Video | Re  | Pr   | Fm   | Re  | Pr   | Fm   |
| Baseline highway | 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 pedestrians | Baseline pedestrians | 0.9579, 0.8828, 0.9186 | 0.9290, 0.8972, 0.9123 | 0.9362, 0.9070, 0.9213 | 0.9486, 0.8929, 0.9199 |
| Baseline office | 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 | Baseline PETS2006 | 0.9076, 0.7932, 0.8465 | 0.8872, 0.8183, 0.8514 | 0.8885, 0.7979, 0.8408 | 0.8946, 0.8065, 0.8475 |

2 ground-truth frames from every video of the CDnet2014 [55] dataset. ADNN-IB_{c8fm} is trained by a mixture of 8 down-sampling ground-truth frames from every video of the CDnet2014 [55] 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_{c8fm} and ADNN-IB_{C20fs} are actually the same, but the training set of ADNN-IB_{c8fm} includes more temporal information. Both ADNN-IB_{C20fs} and ADNN-IB_{c8fm} are trained on CDnet2014 [55], LASIESTA [78], and SBMI2015 [92] datasets. During the evaluations of ADNN-IB_{c8fm} and ADNN-IB_{C20fs}, only one network is trained and the parameters of the network are thus fixed for all testing videos.

ADNN-IB_{L3fs}, ADNN-IB_{B2fs} and ADNN-IB_{C20fs} are evaluated on LASIESTA [78], SBMI2015 [92] and CDnet2014 [55] 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_{L3fs}, ADNN-IB_{B2fs} and ADNN-IB_{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.
TABLE VII

| Training Sets | Baseline highway | Baseline pedestrians | Baseline fountain | Dyn. Bg. | Shadow | Overall |
|---------------|------------------|----------------------|------------------|---------|--------|---------|
|             | Category Video   | Category Video       | Category Video   | Category Video | Category Video | Category Video |
| Baseline highway | Baseline highway | (0.9303, 0.9378, 0.9340) | (0.9296, 0.8883, 0.9085) | (0.9307, 0.8764, 0.9027) | (0.9284, 0.8920, 0.9098) |
| Baseline pedestrians | Baseline pedestrians | (0.9710, 0.9214, 0.9455) | (0.9491, 0.9172, 0.9329) | (0.9441, 0.9096, 0.9265) | (0.9407, 0.9344, 0.9375) |
| Baseline office | Baseline office | (0.9250, 0.8720, 0.8977) | (0.6860, 0.7487, 0.8040) | (0.8678, 0.7769, 0.8199) | (0.8966, 0.7859, 0.8256) |
| Baseline PET'S | Baseline PET'S | (0.9081, 0.8137, 0.8583) | (0.8945, 0.7993, 0.8442) | (0.8776, 0.8158, 0.8455) | (0.8701, 0.8378, 0.8537) |
| Baseline fountain | Baseline fountain | (0.9216, 0.9841, 0.9318) | (0.8964, 0.9481, 0.9208) | (0.9221, 0.9352, 0.9379) | (0.9280, 0.9350, 0.9367) |
| Dyn. Bg. | Dyn. Bg. | (0.8840, 0.8033, 0.8417) | (0.8760, 0.5174, 0.6506) | (0.8610, 0.6040, 0.7099) | (0.8710, 0.6325, 0.7329) |
| Shadow bungalows | Shadow bungalows | (0.8197, 0.9787, 0.8922) | (0.8271, 0.9821, 0.8979) | (0.8217, 0.9854, 0.8961) | (0.8171, 0.9842, 0.8929) |
| Baseline pedestrians | Baseline pedestrians | (0.9834, 0.6666, 0.7190) | (0.9894, 0.5880, 0.7376) | (0.9867, 0.7843, 0.8739) | (0.9850, 0.6478, 0.7816) |
| Dyn. Bg. canoe | Dyn. Bg. canoe | (0.9007, 0.8919, 0.8963) | (0.9007, 0.4670, 0.6151) | (0.8861, 0.5586, 0.6852) | (0.9049, 0.6272, 0.7409) |
| Shadow peopleInShade | Shadow peopleInShade | (0.9062, 0.9742, 0.9390) | (0.7739, 0.8831, 0.8249) | (0.8626, 0.9127, 0.8889) | (0.8418, 0.9230, 0.8860) |

TABLE VIII

| Approach | Baseline | Dyn. Bg. | Cam. Jitt. | Int. Mot. | Shadow | Ther. | Bad Wea. | Low Fr. | Pr. | Vid. | PTZ | Turbul. | Overall |
|---------|----------|----------|-----------|-----------|--------|-------|----------|--------|-----|------|------|---------|---------|
| IUTIS-5 | 0.9567 | 0.8902 | 0.8332 | 0.7296 | 0.9084 | 0.8303 | 0.8428 | 0.7743 | 0.5290 | 0.4282 | 0.7836 | 0.7171 |
| PAWCS | 0.9397 | 0.8938 | 0.8137 | 0.7764 | 0.8913 | 0.8324 | 0.8152 | 0.6588 | 0.4152 | 0.4615 | 0.6450 | 0.7403 |
| BSUV2 | 0.9063 | 0.7967 | 0.7743 | 0.7499 | 0.9233 | 0.8581 | 0.8713 | 0.6797 | 0.6987 | 0.6187 | 0.7051 | 0.7868 |
| BSUV21 | 0.9020 | 0.9057 | 0.9004 | 0.8263 | 0.9562 | 0.8932 | 0.7037 | 0.9702 | 0.5587 | 0.7037 | 0.8174 | 0.8287 |
| FgSN | 0.9026 | 0.3634 | 0.4266 | 0.2002 | 0.5295 | 0.6038 | 0.3277 | 0.2482 | 0.8472 | 0.2931 | 0.7620 |
| ADNN-IB | 0.9522 | 0.9166 | 0.8245 | 0.6978 | 0.9054 | 0.8058 | 0.8243 | 0.7028 | 0.4872 | 0.2931 | 0.7620 |

TABLE IX

| Videos | ADNN-IB | BSUV2.0 | 3DCD | FgSN | CueV2 |
|--------|---------|---------|------|------|-------|
| L_Sf_02 | 0.91 | 0.89 | 0.86 | 0.53 | 0.64 |
| L_CA_02 | 0.79 | 0.60 | 0.49 | 0.58 |
| L_OC_02 | 0.95 | 0.93 | 0.25 | 0.86 |
| L_IL_02 | 0.53 | 0.89 | 0.85 | 0.65 |
| L_Mb_02 | 0.88 | 0.76 | 0.79 | 0.63 | 0.92 |
| L_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 |

1) 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-IBUsfs, ADNN-IBCfs and ADNN-IBcfs are trained and evaluated. The quantitative results of ADNN-IBUsfs are shown in Table VII. ADNN-IBCfs 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 mainly why ADNN-IBUsfs has better results than BSUV2.0 and IUTIS-5. Unfortunately, the overall results of ADNN-IBcfs 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-IBCfs and ADNN-IBcfs are trained to compare with BSUV2.0 and 3DCD across datasets. The quantitative results of ADNN-IBCfs and ADNN-IBcfs on several videos from LASIESTA dataset are shown in Table X. 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-IBCfs achieves the best results compared to BSUV2.0 and 3DCD. In particular, not only is the proposed ADNN-IBCfs, 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 in Table X. The proposed ADNN-IBCfs achieves better results than BSUV2.0 and 3DCD on the entire.
TABLE X
QUANTITATIVE EVALUATION OF THE PROPOSED APPROACH FOR SEEN AND UNSEEN VIDEOS ON THE LASIESTA [78] DATASET, USING THE Fm METRIC.

| Videos | ADNN-IB$_{C2fs}$ | D-DFDL | CueV2 | Lai | Cae | OC | ADNN-IB$_{C2fs}$ | ADNN-IB$_{c8fm}$ | BSUV2.0 | 3DCD-55 | MSFS-55 |
|--------|------------------|--------|-------|-----|-----|----|------------------|------------------|--------|--------|--------|
| LSI_01 | 0.9764           | 0.9596 | 0.9208 | 0.9622 | 0.8143 |
| LSI_02 | 0.9309           | 0.8687 | 0.8403 | 0.8130 | 0.7576 |
| LCA_01 | 0.9807           | 0.9309 | 0.9062 | 0.9220 | 0.8424 |
| LCA_02 | 0.9201           | 0.8850 | 0.7826 | 0.8656 | 0.6296 |
| LOC_01 | 0.9783           | 0.9710 | 0.7013 | 0.8920 | 0.8271 |
| LOC_02 | 0.9735           | 0.9677 | 0.8600 | 0.9526 | 0.8781 |
| LIL_01 | 0.7702           | 0.7161 | 0.6452 | 0.8861 | 0.7966 |
| LIL_02 | 0.7620           | 0.8972 | 0.6523 | 0.8122 | 0.7864 |
| LMB_01 | 0.9874           | 0.9699 | 0.9543 | 0.9816 | 0.7779 |
| LMB_02 | 0.9731           | 0.9195 | 0.9204 | 0.7064 | 0.6797 |
| LBS_01 | 0.9787           | 0.8371 | 0.7132 | 0.6285 | 0.5065 |
| LBS_02 | 0.9626           | 0.6178 | 0.6156 | 0.7333 | 0.6607 |
| OCL_01 | 0.9840           | 0.9792 | 0.9508 | 0.6946 | 0.9280 |
| OCL_02 | 0.9788           | 0.9800 | 0.9045 | 0.9588 | 0.8995 |
| ORA_01 | 0.9896           | 0.9072 | 0.8453 | 0.8225 | 0.7462 |
| ORA_02 | 0.9892           | 0.9083 | 0.8886 | 0.9590 | 0.8699 |
| OSN_01 | 0.9733           | 0.9690 | 0.9317 | 0.3054 | 0.8214 |
| OSN_02 | 0.9562           | 0.9341 | 0.6256 | 0.0426 | 0.0895 |
| OSU_01 | 0.9186           | 0.9065 | 0.6774 | 0.8115 | 0.6527 |
| OSU_02 | 0.9413           | 0.9388 | 0.7669 | 0.9021 | 0.8074 |
| Average| 0.9460           | 0.9068 | 0.8051 | 0.7826 | 0.7386 |

| Videos | 0.9293 | 0.9335 | 0.9200 | 0.8700 | 0.3900 |
|--------|--------|--------|--------|--------|--------|
| LSI_01 | 0.8225 | 0.7316 | 0.6800 | 0.8200 | 0.4000 |
| LSI_02 | 0.9600 | 0.9481 | 0.9600 | 0.9100 | 0.3700 |
| LCA_01 | 0.5216 | 0.4869 | 0.8800 | 0.9200 | 0.3500 |
| LCA_02 | 0.9272 | 0.9262 | 0.8100 | 0.8900 | 0.6400 |
| LOC_01 | 0.9126 | 0.8884 | 0.7700 | 0.7200 | 0.3600 |
| LOC_02 | 0.9594 | 0.9354 | 0.9300 | 0.8700 | 0.4100 |
| LIL_01 | 0.8686 | 0.8744 | 0.9400 | 0.9000 | 0.3500 |
| LIL_02 | 0.8300 | 0.8327 | 0.8400 | 0.6900 | 0.3100 |
| LMB_01 | 0.9868 | 0.8829 | 0.7900 | 0.8500 | 0.3700 |
| LMB_02 | 0.8715 | 0.8459 | 0.8500 | 0.8400 | 0.4000 |

LASIESTA dataset. Thus, the ability of the proposed approach to generalize is demonstrated.

2) 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. [1], 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. [73], 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 [X]. 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$_{C2fs}$, ADNN-IB$_{C2fs}$, and ADNN-IB$_{C2fs}$, show in Table [XII] Table [XII] and Table [XI] 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$_{C2fs}$, which is the results of ADNN-IB$_{C2fs}$ without the improved Bayesian refinement model, demonstrates the contributions of the improved Bayesian refinement model. As shown in Table [XII], the improved Bayesian refinement gives the proposed approach around 5% improvement in Fm value.

3) 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$_{c8fm}$ and ADNN-IB$_{C2fs}$ are trained and evaluated. In particular, during the testing of ADNN-IB$_{c8fm}$ and ADNN-IB$_{C2fs}$, the parameters of the networks are fixed for all testing videos. From the quantitative evaluations on LASIESTA and SBMI2015, shown in Table [X] and Table [XI] respectively, both ADNN-IB$_{c8fm}$ and ADNN-IB$_{C2fs}$ 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 and ADNN-IB 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 XII, ADNN-IB-achieves better results than IUTIS-5 which is a combination of several excellent state-of-the-art methods. Since the parameters of ADNN-IB-achieve fix 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-achieve and IUTIS-5 should be considered to be fair. Also, the good performance of ADNN-IB-achieve 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 X and XI.

The proposed approach is implemented in Pytorch [87], and the source code is available on GitHub[1]. 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 [98] with default parameters is used for training.

D. Limitation and Future Work

1) 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 XII the proposed approach does not work well in the PTZ category where videos are obtained by moving a 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 “_IL_01” and “_IL_02”, as shown in Table X.
2) Complexity Analysis: Currently, the proposed approach takes around 3s total time to process a frame with resolution $240 \times 320$ 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.

3) Future Work: The arithmetic distribution neural network proposed in this paper is just a prototype. The architecture of the network is very simple and the number of parameters is very small. The network used for comparisons with state-of-the-art methods in this paper 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 approach still achieves 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 paper are the best we can present. In addition, as the comparisons demonstrated in Section V-A show, 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 can be another direction in our future work.

VI. 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.

REFERENCES

[1] L. A. Lim and H. Y. Keles, “Foreground segmentation using convolutional neural networks for multiscale feature encoding,” Pattern Recognition Letters, pp. 256 – 262, 2018.
[2] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayananurthy, “Robust subspace learning: Robust pca, robust subspace tracking, and robust subspace recovery,” IEEE signal processing magazine, vol. 35, no. 4, pp. 32–55, 2018.
[3] P. Narayananurthy and N. Vaswani, “A fast and memory-efficient algorithm for robust pca (merop),” in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, 2018, pp. 4684–4688.
[4] J. H. Giraldo, S. Javed, M. Sultana, S. K. Jung, and T. Bouwmans, “The emerging field of graph signal processing for moving object segmentation,” in Proc. Frontiers of Computer Vision, 2021, p. 31.
[5] T. Minematsu, A. Shimada, H. Uchiyama, and R.-i. Taniguchi, “Analytic- ics of deep neural network-based background subtraction,” Journal of Imaging, vol. 4, no. 4, p. 78, 2018.
[6] O. Barnich and M. Van Droogenbroeck, “Vibe: A universal background subtraction algorithm for video sequences,” IEEE Transactions on Image Processing, no. 6, pp. 1709–1724, 2011.
[7] 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.
[8] Z. Zivkovic, “Improved adaptive gaussian mixture model for background subtraction,” in Proc. International Conference on Pattern Recognition, vol. 2, 2004, pp. 28–31 Vol.2.
[9] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian, “Statistical modeling of complex backgrounds for foreground object detection,” IEEE Transactions on Image Processing, vol. 13, no. 11, pp. 1459–1472, 2004.
[10] C. R. Jung, “Efficient background subtraction and shadow removal for monochromatic video sequences,” IEEE Transactions on Multimedia, no. 3, pp. 571–577, 2009.
[11] T. S. F. Haines and T. Xiang, “Background subtraction with dirichlet-process mixture models,” IEEE Transactions on Pattern Analysis and Machine Intelligence, no. 4, pp. 670–683, April 2014.
[12] R. Wang, F. Buyuky, G. Seetharaman, and K. Palaniappan, “Static and moving object detection using flux tensor with split gaussian models,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops, June 2014, pp. 420–424.
[13] G. Ramirez-Alonso and M. I. Chacon-Murguia, “Auto-adaptive parallel som architecture with a modular analysis for dynamic object segmentation in videos,” Neurocomputing, 99, 990 – 1000, 2016.
[14] 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.
[15] S. Bianco, G. Ciocca, and R. Schettini, “How far can you get by combining change detection algorithms?” in Proc. International Conference on Image Analysis and Processing, 2017, pp. 96–107.
[16] Y. Chen, J. Wang, and H. Lu, “Learning sharable models for robust background subtraction,” in Proc. International Conference on Multimedia and Expo, June 2015, pp. 1–6.
[17] P. L. S. St-Charles, G. A. Bilodeau, and R. Bergevin, “Subsense: A universal change detection method with local adaptive sensitivity,” IEEE Transactions on Image Processing, no. 1, pp. 359–373, Jan 2015.
[18] X. Liu, J. Yao, X. HONG, X. Huang, Z. Zhou, C. Qi, and G. Zhao, “Background subtraction using spatio-temporal group sparsity recovery,” IEEE Transactions on Circuits and Systems for Video Technology, no. 99, pp. 1–1, 2017.
[19] H. Yang, D. Meng, W. Zuo, and L. Zhang, “Robust online matrix factorization for dynamic background subtraction,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 7, pp. 1726–1740, 2018.
K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis, “Background subtraction methods using gaussian mixture model for video surveillance systems,” Artificial Intelligence Review, vol. 50, no. 2, pp. 241–259, 2018.

C. Stauffer and W. Grimson, “Adaptive background mixture models for foreground detection in complex scenes,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 1999, pp. –252 Vol. 2.

T. Kim, T. Chalidabhongse, D. Harwood, and L. Davis, “Background modeling and subtraction by codebook construction,” in Proc. International Conference on Image Processing, Oct 2004, pp. 3061–3064.

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.

S. Javed, A. Mahmood, T. Bouwmans, and S. K. Jung, “Background–foreground modeling based on spatiotemporal sparse subspace clustering,” IEEE Transactions on Image Processing, vol. 26, no. 12, pp. 5840–5854, 2017.

S. Javed, A. Mahmood, S. Al-Maadeed, T. Bouwmans, and S. K. Jung, “Moving object detection in complex scene using spatiotemporal structural-sparse rPCA,” IEEE Transactions on Image Processing, vol. 28, no. 2, pp. 1007–1022, 2019.

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.

T. Huynh-The and O. B. et al., “Nic: A robust background extraction algorithm for foreground detection in dynamic scenes,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, pp. 1478–1490, July 2017.

S. Jiang and X. Lu, “Wesame: A weight-sample-based method for background subtraction,” IEEE Transactions on Circuits and Systems for Video Technology, no. 99, pp. 1–17, 2017.

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. 100204, 2020.

T. S. F. Haines and T. Xiang, “Background subtraction with dirichlet processes,” in Proc. European Conference on Computer Vision, Springer Berlin Heidelberg, 2012, pp. 99–113.

M. Chen, X. Wei, Q. Yang, Q. Li, G. Wang, and M. H. Yang, “Spatiotemporal gmm for background subtraction with superpixel hierarchy,” IEEE Transactions on Pattern Analysis and Machine Intelligence, no. 99, pp. 1–17, 2017.

M. Chen, Q. Yang, Q. Li, G. Wang, and M.-H. Yang, “Spatiotemporal background subtraction using minimum spanning tree and optical flow,” in Proc. European Conference on Computer Vision, 2014, pp. 521–534.

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.

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.

S. Liao, G. Zhao, V. Kellokumpu, M. Pietikäinen, and S. Z. Li, “Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 1301–1306.

G. Geninian 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.

Z. Zeng, J. Jia, D. Yu, Y. Chen, and Z. Zhu, “Pixel modeling using histograms based on fuzzy partitions for dynamic background subtraction,” IEEE Transactions on Fuzzy Systems, pp. 584–593, 2017.

S. Varadarajan, P. Miller, and H. Zhou, “Region-based mixture of gaussians model for foreground detection in dynamic scenes,” Pattern Recognition, no. 11, pp. 3488 – 3503, 2015.

A. Elgammal, D. Harwood, and L. Davis, “Non-parametric model for background subtraction,” in Proc. European Conference on Computer Vision, 2000, pp. 751–767.

Z. Liu, K. Huang, and T. Tan, “Foreground object detection using top-down information based on em framework,” IEEE Transactions on Image Processing, vol. 9, pp. 4294–4217, Sept 2012.

D. Li, L. Xu, and E. Goodman, “A fast foreground object detection algorithm using kernel density estimation,” in Proc. IEEE International Conference on Signal Processing, Oct 2012, pp. 703–707.

H.-H. Lin, T.-L. Liu, and J.-H. Chuang, “A probabilistic svm approach for background scene initialization,” in Proc. International Conference on Image Processing, June 2002, pp. 893–896 vol.3.

L. Cheng and M. Gong, “Realtime background subtraction from dynamic scenes,” in Proc. International Conference on Computer Vision, Sept 2009, pp. 2066–2073.

B. Han and L. Davis, “Density-based multifeature background subtraction with support vector machine,” IEEE Transactions on Pattern Analysis and Machine Intelligence, no. 5, pp. 1017–1023, May 2012.

X. Zhang, Z. Liu, H. Li, X. Zhao, and P. Zhang, “Statistical background subtraction based on imbalanced learning,” in Proc. IEEE International Conference on Multimedia and Expo, July 2014, pp. 1–6.

X. Zhang, C. Zhu, H. Wu, Z. Liu, and Y. Xu, “An imbalance compensation framework for background subtraction,” IEEE Transactions on Multimedia, no. 11, pp. 2425–2438, Nov 2017.

D. Culibrk, O. Marques, D. Socek, H. Kalva, and B. Furtih, “Neural network approach to background modeling for video object segmentation,” IEEE Transactions on Neural Networks, no. 6, pp. 1614–1627, Nov 2007.

X. Zhou, C. Yang, and W. Yu, “Moving object detection by detecting contiguous outliers in the low-rank representation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, no. 3, pp. 597–610, 2013.

B.-H. Do and S.-C. Huang, “Dynamic background modeling based on radial basis function neural networks for moving object detection,” in Proc. IEEE International Conference on Multimedia and Expo, 2011, pp. –4.

M. D. Gregorio and M. Giordano, “Change detection with weightless neural networks,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops, June 2014, pp. 409–413.

L. Maddalena and A. Petrosino, “A self-organizing approach to background subtraction for visual surveillance applications,” IEEE Transactions on Image Processing, no. 7, pp. 1168–1177, July 2008.

L. Maddalena and A. Petrosino, “The sobbs algorithm: What are the limits?” in Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops, June 2012, pp. 21–26.

Y. W. et al., “Cdfnet 2014: An expanded change detection benchmark dataset,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops, June 2014, pp. 393–400.

M. Braham and M. V. Droogenbroeck, “Deep background subtraction with scene-specific convolutional neural networks,” in Proc. International Conference on Systems, Signals and Image Processing, 2016, pp. 1–4.

Y. Wang, Z. Luo, and P.-M. Jodoin, “Interactive deep learning method for segmenting moving objects,” Pattern Recognition Letters, vol. 96, pp. 66–75, 2016.

C. Zhao, T. Cham, X. Ren, J. Cai, and H. Zhu, “Background subtraction based on deep learning,” in Proc. IEEE International Conference on Multimedia and Expo, July 2018.

X. Liang, S. Liao, X. Wang, W. Liu, Y. Chen, and S. Z. Li, “Deep background subtraction with guided learning,” in Proc. IEEE International Conference on Multimedia and Expo, July 2018.

D. Zeng and M. Zhu, “Background subtraction using multiscale fully convolutional network,” IEEE Access, vol. 6, pp. 16010–16021, 2018.

M. Babaei, D. T. Dinh, and G. Rigoll, “A deep convolutional neural network for video sequence background subtraction,” Pattern Recognition, pp. 635 – 649, 2018.

L. A. Lim and H. Y. Keles, “Learning multi-scale features for foreground segmentation,” Pattern Analysis and Applications, vol. 23, no. 3, pp. 1369–1380, 2020.

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. 97949–97958, 2019.

A. K. Dongdong Zeng, Ming Zhu, “Combining background subtraction algorithms with convolutional neural network,” Journal of Electronic Imaging, no. 1, pp. 1 – 6, 2019.

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, pp. 8 – 66, 2019.

C. Zhao and A. Basu, “Dynamic deep pixel distribution learning for background subtraction,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 11, pp. 4192–4206, 2020.

V. M. Mondejar-Guerra, J. Rouco, J. Novo, and M. Ortega, “An end-to-end deep learning approach for simultaneous background modeling and object detection,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019.

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.
