UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Investigating the latency cost of statistical learning of a Gaussian mixture simulating on a convolutional density network with adaptive batch size technique for background modeling

Hung Ngoc Phan

INF-3981 Master's Thesis in Computer Science - May 2021

Abstract

Background modeling is a promising field of study in video analysis, with a wide range of applications in video surveillance. Deep neural networks have proliferated in recent years as a result of effective learning-based approaches to motion analysis. However, these strategies only provide a partial description of the observed scenes' insufficient properties since they use a single-valued mapping to estimate the target background's temporal conditional averages. On the other hand, statistical learning in the imagery domain has become one of the most widely used approaches due to its high adaptability to dynamic context transformation, especially Gaussian Mixture Models. Specifically, these probabilistic models aim to adjust latent parameters to gain high expectation of realistically observed data; however, this approach only concentrates on contextual dynamics in short-term analysis. In a prolonged investigation, it is challenging so that statistical methods cannot reserve the generalization of long-term variation of image data. Balancing the trade-off between traditional machine learning models and deep neural networks requires an integrated approach to ensure accuracy in conception while maintaining a high speed of execution.

In this research, we present a novel two-stage approach for detecting changes using two convolutional neural networks in this work. The first architecture is based on unsupervised Gaussian mixtures statistical learning, which is used to classify the salient features of scenes. The second one implements a light-weighted pipeline of foreground detection. Our two-stage system has a total of approximately 3.5K parameters but still converges quickly to complex motion patterns. Our experiments on publicly accessible datasets demonstrate that our proposed networks are not only capable of generalizing regions of moving objects with promising results in unseen scenarios, but also competitive in terms of performance quality and effectiveness foreground segmentation.

Apart from modeling the data's underlying generator as a non-convex optimization problem, we briefly examine the communication cost associated with the network training by using a distributed scheme of data-parallelism to simulate a stochastic gradient descent algorithm with communication avoidance for parallel machine learning

Acknowledgements

First and foremost, a special thank must be given to Professor Phuong Hoai Ha, my research supervisors for his professional guidance, valuable support and constructive recommendations on my thesis. His insightful comments, invaluable encouragement and profound belief in my work incented me to enlighten me the first glance of investigation and to widen this research from various perspectives during my Master's study.

Second, I would like to extend my sincere gratitude to the Department of Computer Science including Svein Tore Jensen, Jan Fuglesteg, and Jon Ivar Kristiansen for their administrative and technical supports at all levels. Your unrelenting support has been done to facilitate my research work.

Third, I cannot forget to express my thanks to Dr. Synh Viet-Uyen Ha, a computer vision researcher at International University – Vietnam National University, from whom I received enthusiastic encouragement on formulating the learning model to complete this thesis.

Last but not the least, words cannot describe how thankful I am to my parents and my brother who have encouraged and supported me through all my endeavors.

Contents

Abstract i

Acknowledgements iii

List of Figures vii

List of Tables ix

1 Introduction 1
- 1.1 Problem Definitions and Research Questions 3
- 1.2 Research Contributions 5
- 1.3 Thesis Roadmap 7

2 Problem Background 9
- 2.1 Background Modeling and Change Detection: A Review . . . 9
  - 2.1.1 Background Modelling with Statistical Learning on Image Intensity 10
  - 2.1.2 Change Detection via Exploiting Deep Visual Learning 11
- 2.2 Communication-Efficient Distributed Deep Learning: A Survey 13
  - 2.2.1 Energy-based Models and Stochastic Gradient Descent 13
  - 2.2.2 Parallelism Schemes of Distributed Training 14
  - 2.2.3 Approaches of Communication-Avoidance in Distributed Stochastic Gradient Descent 16

3 An Unsupervised, Two-stage Network for Background Density Analysis and Motion Difference Approximation 21
- 3.1 Background Modelling with a Convolution Density Network of Gaussian Mixture 23
- 3.2 Background Learning via Training a Density Network with a Unsupervised Loss Function 28
- 3.3 Foreground Segmentation with a Non-Linear Approximating Frame-Difference 32
  - 3.3.1 The Proposed Architectural Design of Foreground Detection Network 33

3.3.2 The Methodology of Network Training 36

4 Implementation and Evaluation 39
4.1 Method Implementation and Experimental Setup 39
4.1.1 The Description of Implementing Methodology 39
4.1.2 Experimental Setup and Evaluation Metrics 40
4.2 Overall summary of experimental results 42
4.3 Experimental Benchmark on CDnet 2014 dataset 46
4.4 Experimental Benchmark on Wallflower dataset without Parameter Fine-Tuning 50

5 Event-Triggered Distributed Training of Proposed Model with Communication Avoidance 53
5.1 An Introduction to Event-Triggered Communication 54
5.1.1 Existing Problems in Recently Proposed Frameworks . 54
5.1.2 A Framework of Event-Triggered Communication in Parallel Distributed Model Learning 55
5.2 An Extensive Analysis in Distributed Training of Motion Difference Approximation 56
5.2.1 A Framework of EventGraD with GPU-accelerated Computation 56
5.2.2 The Discussion of Implementing Perspective 58
5.2.3 The Experimental Analysis of Communication Efficiency 60

6 Discussion and Concluding Remarks 63
6.1 Discussion 63
6.2 Concluding Remarks 65

Bibliography 67

List of Figures

2.1 A scheme of model parallelism 15
2.2 An architectural design of data parallelism 16
2.3 The illustrative summary of for communication synchronization in distributed training of deep learning models by Tang *et al.* [54] 17

3.1 The overview of the proposed method for background modeling and foreground detection 23
3.2 The proposed architecture of Convolution Density Network of Gaussian Mixture Model 25
3.3 The proposed architecture of MEDAL-net grounded on convolutional autoencoder for foreground detection 34

4.1 The summary of computational speed and average F-measure comparison with state-of-the-art methods on CDnet-2014. The upper red region defines an evaluation space where methods satisfy the applicability criteria of accuracy. The leftmost green area indicates approaches that met the requirement of real-time processing speed. Our proposed scheme of CDN-MEDALnet balances the trade-off of accuracy and processing speed between two approaches. 45
4.2 Visual quality comparison for foreground detection on all video sequences in eleven categories in CDnet 2014. The columns include: (⋆) input frame, (⋄) corresponding groundtruth foreground, (a) GMM – S & G, (b) GMM – Zivkovic, (c) SuBSENSE, (d) PAWCS, (e) BMOG, (f) FTSG, (g) SWCD, (h) CDN-MEDAL-net, (i) FgSegNet_S, (j) FgSegNet_v2 (k) Cascade CNN, (l) DeepBS. Experimented scenarios include bad weather (*BDW*), baseline (*BSL*), camera jitter (*CJT*), dynamic background (*DBG*), intermittent object motion (*IOM*), low frame rate (*LFR*), night videos (*NVD*), shadow (*SHD*), thermal (*THM*), and turbulence (*TBL*) 47

4.3 Visual illustration of background modelling using convolutional density network on different contexts of CDnet-2014 dataset 48

5.1 The model of Partitioned Global Address Space (PGAS) in UPC++. 58
5.2 The illustration of the experimental data-parallelism scheme with UPC++ and PyTorch C++ API. 59
5.3 The illustration of the experimental results where we measure the number of transmitted messages with respect to various ranges of processes on different batch sizes through 50 epochs. We vary the number of batch size: 4, 8, 12, 16 with an appropriate number of processes that fits the experimented hardware configuration. 61

List of Tables

3.1 Architecture of Convolutional Density Network 26
3.2 Body Architecture of MEDAL-net 35

4.1 The overall evaluation of experimented methods regarding accuracy and execution speed 44
4.2 F - measure comparisons over all of eleven categories in the CDnet 2014 dataset 49
4.4 Result of quantitative evaluation on CDnet 2014 dataset . . 50
4.5 F - measure comparisons over the six sequences of Wallflower dataset with model parameters tuned on CDnet-2014 51

5.1 The number of exchanged messages with respect to a range of threshold where the MEDALnet was trained wich a batch size of 4 60
5.2 The number of exchanged messages with respect to a range of threshold where the MEDALnet was trained wich a batch size of 8 62

1

Introduction

Computer vision is an cross-disciplinary research area that concentrates on how to simulate high-level perception on computers and electronic devices from digitals images and videos [1]. The field involves techniques to collect, and gain the interpretation of cognitive data from real-world contexts with a goal to formulate visual understanding in form of numerical representation with the assistance of statistics, linear systems, and learning theories.

With the rapid advancement of computer vision, surveillance systems using static cameras are emerging as exciting technologies for performing advanced tasks such as behavior analysis [2], object segmentation [3], and motion analysis [4], [5]. Among their various functionalities, background reconstruction is critical for a proper understanding of scene dynamics and thus for the extraction of desired attributes. A background is a scene that is devoid of moving objects and elements that that are not of interest to the system (e.g., streets, houses, trees). Thus, by comparing visual inputs to the background, desired objects, called foregrounds (e.g., cars, pedestrians), can be located for further study. Due to the fact that real-world situations require varying degrees of dynamics, such as lighting shifts, scene dynamics, or bootstrapping, there are several approaches to model background scenes with context-specific adaptation [6].

A couple of closely related issue to background modelling are multi-contextual foreground detection and dynamic scene adaptation as they was explicated in Belmar's study [7]. The former problem, foreground detection or background

subtraction, is a particular cases of estimating the difference between two scenes: one is the observed image and the other one is the corresponding background scene. Accordingly, the changes, which are due to moving objects, are taken into consideration to extract regions of moving objects. The latter problem aims to examine the semantic dynamics regarding intensity spaces of color to incorporate a fast and simple pre-attentive adaptation of modelling learning. The importance of this strategy is emphasized when we need to avoid distortions from anomalous updates and facilitate consistent modelling amidst variations [8]. There is a fact that in the real-world scenarios, a system of background modelling is fundamentally susceptible to particular scenarios with a variety of confounding factors that make learning systems not discriminate the input, including but not limit to illumination changes, continuous motion, slow-moving objects, camouflage, camera shaking. Hence, a rendering of misperceive signal is unexpectedly performed, which ruins out the data distribution in the background model and make the system prone to prolonged noise effects.

Another significant attention of background modelling is related to data-driven learning. Typically, a background model is commonly proposed as an unified framework to speculate underlying scenes' properties via presenting optimization problems in form of generative models that basically are grounded on statistical physics to learning towards collected data. One of the most popular approaches to achieve the generalization of this energy-based model is to perform the gradient-based method for inference with a large-scale of training sample [9]. Besides concerns on accuracy due to the completeness of data, learning on a great deal of sampling peculiarities requires an appropriate scheme of resource usage which takes advantage of computational parallelism among leading-edge processing units to gain highly efficient model learning while balancing accuracy in evaluation [10].

In this research, we introduce a novel, unsupervised, background subtraction method that is capable of high adaptation to dynamic context transformation. The proposed framework deals with a situation where there is a lack of labelled data for both background and foreground estimation, while still maintaining rapid convergence to complex motion patterns. Besides modeling the underlying generator of the data as a non-convex optimization problem, we investigate the cost of communication when training the network with an adaptive batch size. The trade-off between energy efficiency and accuracy in the model will be investigated with an adaptive batch size throughout network training

1.1 Problem Definitions and Research Questions

One of the most prominent approaches in tackling the background modeling problem is to employ pixel-based statistical frameworks such as Gaussian Mixture Models (GMM) [11], [12], [13], [8]. These methods are based on the hypothesis that background intensities appear predominantly throughout a scene, thereby constructing usefully explicit mathematical structures for exploitation of the dominance. In addition, an important property of such approaches is their adaptability to changing conditions of real-world scenarios, even under illumination changes (e.g. moving clouds), view noises (e.g. rain, snow drops) and implicit motions (e.g. moving body of water). However, the generalization of the methods' correctness is hindered when the hypothesis fails under appearances of stopped objects or high degrees of view noises (e.g. camera shaking, abrupt view changes), thereby producing corruptive backgrounds that often leads to poor estimations of foregrounds. Furthermore, the statistical schemes still follow the sequential processing paradigm that under-utilizes modern parallel processing units in the presence of big data.

On the other hand, riding on the increasing advancement wave of specialized processing units for large-scale data, Deep Neural Networks (DNNs) have emerged as a prominent pattern matching and visual prediction mechanism. Deep learning approaches for the motion detection problem are rapidly demonstrating their effectiveness not only in utilizing tremendous sets of processing cores of modern parallel computing technologies, but also in producing highly accurate predictions from data-learning. However, the typical DNNs' architectures are very computationally expensive if they actually can produce highly accurate results, especially regarding those providing solutions to the problem of background modeling and foreground detection. Furthermore, the DNNs in the literature have experienced two primary shortcomings:

- *A requirement of a huge-scale dataset of labeled images*: DNNs-based models for motion detection exploit weak statistical regularities between input sequences of images and annotated background scenes. Thus, to generalize all practical scenarios in real life, a prohibitively large dataset consisting of all practical scenarios and effects is needed. With few training labels in video sequences for building generalized background models [14], there are currently no universal data-driven experiments to assure that the scenes' true properties are appropriately presented.

- *A prevailing fail on contextual variation*: Recently, foreground segmentation has been considered from the perspective of binary classification schemes. It has been proposed to minimize a sum-of-squares or a cross-entropy error function in DNNs-based approaches to reflect the motion analysis problem's true objective as closely as possible. In this approach,

models are usually trained to represent the semantics properties on the training sets when the actual aim is to generalize well to experimental datasets' specific target video sequence. This conditional average will be inadequate for various unseen contextual semantics and dynamics that might occur in real-world [15, 16]. In other words, DNNs-based methods usually perform well on experimental datasets of background modeling and change detection but can still fail on unseen situations in real-world scenarios.

Recent years have witnessed the proliferation of convolutional neural networks via effective learning-based approaches in change detection. However, these techniques only provide a limited description and the observed scenes' insufficient properties, where the single-valued mapping is learned to approximate the conditional averages of the target background conditioned on color and motion features of each image batch. On the other hand, statistical learning in imagery domains has become one of the most popular approaches with high adaptation to dynamic context transformation, especially the Gaussian Mixture Model. Simulating the statistical learning of a Gaussian mixture on a convolutional neural network exploits a statistical inference on scene analysis with only about a few hundreds of parameters while still maintaining rapid convergence to complex motion patterns. In this work, we exploits a statistical inference on scene analysis by simulating the statistical learning of a Gaussian mixture on a convolutional neural network with only about a few hundreds of parameters while still maintaining rapid convergence to complex motion patterns. Besides modeling the underlying generator of the data as a non-convex optimization problem, we investigate the cost of communication when training the network with an adaptive batch size technique. The trade-off between energy efficiency and accuracy in the model will be investigated with an adaptive batch size throughout network training. In summary, with an assumption that the background scene of a video sequence contains the most commonly seen intensity value at each image point, our contributions in this work are proposed towards three following research questions:

- The first part of this work investigates the modeling methodology to answer the research question: "*RQ1: How to represent a conditional probability density function of a Mixture of Gaussians, which models the time-related history at each pixel location on a feed-forward Convolutional Neural Network?*".

- The second study of this research answers the: "*RQ2: How to devise the underlying generator of the background scenes from observed sequences of images with a trainable, unsupervised Convolutional Neural Network of Gaussian Mixture?*".

- The third part of this thesis aims to address the research question: "*RQ3: How significant is the trade-off between energy efficiency and accuracy of the model with an adaptive batch size throughout the training of the neural network?*".

1.2 Research Contributions

The DNNs-based approach is particularly promising as the literature has rapidly demonstrated their ability to approximate any functions up to arbitrary accuracy within highly parallelizable architectures. In other words, we can exploit their parallelizable capability to approximate the mechanism behind the optimization of GMM, in a way that boosts the construction of statistical model estimations of our data using modern parallel computing technologies. Hence, it becomes possible to efficiently exploit GMM-based background models' characteristics, which are clear and consistent with their mathematical framework, for functional extension, i.e. tackling stopped objects and high-degreed view shifts via DNNs' common data-driven effectiveness. In this thesis, to address the issues of DNNs while also utilizing its benefits, we incorporate the mathematics of modeling statistical GMM into our processes, and introduce a novel, light-weighted, dual framework of two convolutional neural networks (CNN): (1) the **C**onvolutional **D**ensity **N**etwork of **G**aussian **M**ixtures (**CDN-GM**) for the task of generalistically modeling backgrounds; and (2) the **M**otion **E**stimation with **D**ifferencing **A**pproximation via **L**earning on a convolutional network (**MEDAL-net**), for context-driven foreground extraction. Specifically, our contributions are actually three-fold, and they are summarized as follows:

- Firstly, by leveraging existing technologies and being inspired by Bishop [17], we propose our CDN-GM, a feed-forward, highly parallelizable CNN representing a conditional probability density function that models the temporal history for each pixel location in the first pipeline of the proposed framework. In this architecture, conditioned on pixel-wise vectors of intensity values across a time period, the network approximates a Gaussian-Mixture statistical mapping function to efficiently produce models of their underlying multimodular distributions. Accordingly, at each pixel, the mixture is characterized by the weighted combination of its Gaussian components, where each capture and highlight a context-relevant range of pixel-wise values in the manner of a mean and variance. Thus, from statistical models of data in Gaussian Mixtures at pixel level, backgrounds are extracted from the most informative components, resulting in our compressed, light-weighted and efficient architecture (See Section 3.1).

- Secondly, with the goal of modeling the underlying generator of the data, we propose a loss function in the manner of unsupervised learning. This loss function serves to direct the proposed CDN-GM's architectural parameters into approximating the mathematical structure behind GMM-driven modeling of the data with expectation maximization. Thus, because of this, the resulting inferences will consist of mixtures of Gaussian components describing the data, and the most likely background description of actually observed data can be made, with the trained network being subsequently presented with new values of input. In conjunction with CDN-GM, the proposed background modeling architecture not only achieves higher degrees of interpretability compared to the idea of estimating an implicit hidden function in previous neural network methods, but it also gains better capability of adaptation under contextual dynamics with statistical learning, as it is able to utilize a virtually inexhaustible amount of data for incorporation of expectation maximization into the neural-network parameters (See Section 3.2).

- Thirdly, in the latter pipeline of the proposed framework, we design a compact convolutional auto-encoder for context-driven foreground extraction called MEDAL-net, which simulates a context-driven difference mapping between input frames and their corresponding background scenes. This is greatly encouraged because even though real-life scenarios involve various degrees of contextual variations that yet any existing mathematical framework can completely capture, we can construct consistently GMM-driven background models of those variations with CDN-GM to provide semantic understanding of the scene. Thus, we are able to make good use of information from features in images from the first module of background modeling, and even from features seemingly corruptive to motion extractions (e.g. stopped objects), for formulating foreground extraction from raw inputs, thereby resulting in a very light-weighted and efficient structure with high accuracy. The network is trained in a supervised manner in such a way that it maintains good generalization to various views, and to even unseen situations of similar scenery dynamics (See Section 3.3 and Chapter 4).

- Finally, we survey recent framework of distributed machine learning for data-driven model with a goal of communication reduction (See Section 2.2). Then, we investigate the training of the proposed model with an event-triggered scheme of communication with a strategy of data parallelism. The analysis concentrates on the communication cost among a distributed crew of processing elements regarding model training when we vary different batch sizes of feed-forwarding samples and the threshold of gradient slope that triggers the message passing among processes (See Section 5).

1.3 Thesis Roadmap

The content of this thesis is organized as follows:

- Chapter 2 encapsulates the problem background, including the synthesis of recent approaches in background initialization and foreground segmentation. In addition, a general survey of communication-efficient distributed deep learning is also provided in this chapter.

- Chapter 3 presents the details of our couple of neural networks. The principal research contributions which are explicated in this chapter includes a convolutional density network for background modelling, associating with an unsupervised approach of model training, and an auto-encoder pipeline to simulate frame-background differencing with a goal of motion extraction.

- Chapter 4 mentions the perspective of implementation and experimental evaluation of the proposed method. We aims to analyze the capability of generalization of the approach in both aspects: learning with shortage of labelled data and inferencing in unseen samples. The trade-off between the accuracy and the speed of execution, which is a significant concern in the most of neural-network-based studies, is also discuss at the end of this chapter.

- Chapter 5 extends the training scheme of the proposed model in terms of a distributed environment. Available techniques that support pipeline of distributed parallelism are introduced. Then, we describe a distributed scheme of model training with a data parallelism strategy to analyze the cost of message passing.

- Chapter 6 discusses the findings, outlines future work and concludes the thesis

2

Problem Background

In this section, we will outline the previous and current work that is related to the problem of background modelling and foreground detection. Also, a brief overview of communication-avoidance mechanism for distributed training of neural networks is provided.

2.1 Background Modeling and Change Detection: A Review

The new era of video analysis has witnessed a proliferation of methods that concentrate on background modeling and foreground detection. Prior studies in recent decades were encapsulated in various perspectives of feature concepts [7,15,18]. Among published methods that meet the requirements of robustness, adaptation to scene dynamics, memory efficiency, and real-time processing, two promising approaches of background subtraction are statistical methods and neural-network-based models. Statistical studies aim to characterize the history of pixels' intensities with a model of probabilistic analysis. On the other hand, neural-network-driven approaches implicitly estimate a mapping between an input sequence of observed scenes and hand-labeled background/foreground images on non-linear regularities.

2.1.1 Background Modelling with Statistical Learning on Image Intensity

In statistical approaches, the pixels' visual features are modeled with an explainable probabilistic foundation regarding either pixel-level or region-level in temporal and spatial resolution perspectives. In the last decades, there have been a variety of statistical models that were proposed to resolve the problem of background initialization. Stauffer and Grimson [11] proposed a pioneering work that handled gradual changes in outdoor scenes using pixel-level GMM with a sequential K-means distribution matching algorithm. Another modification on CIE L*a*b* color space is Boosted Gaussian Mixture Model (BMOG) [13] which was introduced to investigate the adaptation of GMM with different color schemes. To enhance the foreground/background discrimination ability regarding scene dynamics, Pulgarin-Giraldo *et al.* [19] improved GMM with a contextual sensitivity that used a Least Mean Squares formulation to update the parameter estimation framework. Validating the robustness of background modeling in a high amount of dynamic scene changes, Ha *et al.* [20] proposed a GMM with high variation removal module using entropy estimation. To enhance the performance, Lu *et al.* [21] applied a median filter on an input frame to reduce its spatial dimension before initializing its background. To address the sequential bottleneck among statistical methods in pixel-wise learning, an unsupervised, tensor-driven framework of GMM was proposed by Ha *et al.* [8] with balanced trade-off between satisfactory foreground mask and exceptional processing speed. However, the approach's number of parameters requires a lot of manual tuning. In addition to GMM, Cauchy Mixture Models (CMM) was exploited to detect foreground objects via eliminating noise and capturing periodical perturbations in varying lighting conditions and dynamic scenarios [22]. Overall, statistical models were developed with explicit probabilistic hypotheses to sequentially present the correlation of history observation at each image point or a pixel block, added with a global thresholding approach to extract foreground. This global thresholding technique for foreground detection usually leads to a compromise between the segregation of slow-moving objects and rapid adaptation to sudden scene changes within short-term measurement. This trade-off usually damages the image-background subtraction in multi-contextual scenarios, which is considered as a sensitive concern in motion estimation. Hence, regarding foreground segmentation from background modeling, it is critical to improve frame differencing from constructed background scene with a better approximation mechanism, and utilize parallel technologies. There are also other developments in background modeling with fuzzy concepts recently. A post-processing scheme that utilizes Gibb's Markov Random Field fuzzy clustering and gray level co-occurrence matrix features [23] was proposed to more effectively remove the moving shadows out of the moving objects. Zeng *et al.* [24] introduced an adaptive histogram learning method where histogram accumulation is monitored by a

fuzzy controller to address the susceptibility to outliers and the randomness of histogram partitioning. To improve the quality of foreground segmentation, Yu *et al.* [25] adopted a fuzzy adaptive background maintenance method with a dynamic fuzzy nearness degree threshold update. Subudhi *et al.* [26] were one of the groundbreakers who used higher space of kernelized fuzzy set-theoretic method via the projection of the highly non-linear 3-dimensional feature space to classify foreground-background image points.

2.1.2 Change Detection via Exploiting Deep Visual Learning

Recently, there have also been many attempts to apply DNNs into background subtraction and background modeling problems with supervised learning. Inspired from LeNet-5 [27] used for handwritten digit recognition, one of the earliest efforts to subtract the background from the input image frame was done by Braham *et al.* [28]. This work explores the potential of visual features learned by hidden layers for foreground-background pixel classification. Similarly, Wang *et al.* [29] proposed a deep CNN trained on only a small subset of frames as there is a large redundancy in a video taken by surveillance systems. The model requires a hand-labeled segmentation of moving regions as an indicator in observed scenes. This period has experienced an evolutionary development of background subtraction with novel architectures in data-driven learning theories. Zhenshen *et al.* [30] proposed one of pioneering work that exploits a context-encoder for the task of constructing background scenes via learning latent visual properties of motion context within a video. Lim *et al.* [31] improved the learning mechanism of auto-encoder by inheriting a set of feature encoding layers from VGG-16 [32]. The proposed encoder-decoder network takes a video frame, along its corresponding grayscale background and its previous frame as the network's inputs to compute their latent representations, and to deconvolve these latent features into a foreground binary map. An adaptive Restricted Boltzmann Machine [33] was proven to be efficient in structuring the correlation among a batch of consecutive images to capture background scenes. An improved variant of generative network, Unet, was augmented by Tao *et al.* [34] to give attention to the probabilistic heat map of intensity values in color spaces to extract a static scene via unsupervised approach. Bridging the gap point-wise, time-series analysis in traditional statistical learning and the spatialpyramid feature sampling, a learning model with attention-guided module, named STAM [35], was proposed by Liang *et al.* The solution takes advantage of the synthesis of features, which was derived from both traditional methods and a modern deep neural network, to bolster the feature of motion. The theoritical perspective of this approach is intriguing; however, the execution requires us a great computational resources to achieve real-world deployment. Another method is DeepBS [36] which was proposed by Babaee *et al.* to compute the background model using both SuBSENSE [37]

and Flux Tensor method [38]. The authors extract the foreground mask from a small patch from the current video frame and its corresponding background to feed into the CNN, and the mask is later post-processed to give the result. Nguyen *et al.* proposed a motion feature network [39] to exploit motion patterns via encoding motion features from small samples of images. The method's experimental results showed that the network obtained a promising results and well-performed on unseen data sequences. Another method that used a triplet convolutional autoencoder to learn multi-scale hidden representations for motion mask extraction of the observed scenes was proposed by Lim *et al.* as FgSegNet [40]. The authors utilized multi-scale feature space of an image with a network, whose decoder module was constructed with transposed convolutional layers. The main purpose of this method is to associate the latent feature space of moving components with the outcome image space. Experimental results showed that the work is sustainable to various scene dynamics in CDnet-2014 dataset with high accuracy, including camera jitter, camouflage, light switching Perceiving the barrier in model training of FgSegNet with the shortage of labelled foreground samples, the authors devised an improved version, FgSegNet_v2 [41]. This variant has a better grasp of motion characteristics via feature fusion inside the former edition with a section of feature pooling. Although this technique gains a state-of-the-art accuracy in experimental results, there is a compromise between the correctness and the execution speed with an exceptional hardware configurations. Recently, there is also a work from Chen *et al.* [42] which aims to exploit high-level spatial-temporal features with a deep pixel-wise attention mechanism and convolutional long short-term memory (ConvLSTM). Recently, Akilan *et al.* [43] demonstrate the effectiveness of ConvLSTM via the consideration of operations in 3D space. The model employed both 3D convolution and ConvLSTM operations to extract both short-term and long-term temporal features with double-encoding and slow-decoding. Another approach that adopts generative adversarial networks (GAN) was introduced by Sultana *et al.* [44]. The authors utilized two CNNs, which are a context prediction network to estimate the background in regions with moving objects removed after a pre-processing step and a texture optimizing network to enhance the predicted context. Particularly, the whole method is performed in an unsupervised way. An architecture of background modeling with a self-organizing neural network was investigated in RGBD-SOBS [45]. The work modeled the video frame's color and depth information separately to give the foreground mask in each model.

2.2 Communication-Efficient Distributed Deep Learning: A Survey

With the rapid development of data-driven models in perceptual learning, distributed machine learning has become a pivotal research field that focuses on minimizing the amount of training time via reducing the communication among computing units (e.g., CPUs, GPUs). As the scale of training datasets and model sizes increases dramatically, data exchange between processing units could potentially become a severe bottleneck in degrading the effectiveness of computing tasks. This section demystifies the recent research in this topic as a comprehensive review of communication-efficient distributed deep learning with two conceptions: system-level and algorithmic level in model optimization. Regarding the perspective of systems, we summarize distributed environments that are beneficial to the optimization of learning models. At the algorithmic level, we provide a taxonomy of data-parallel distributed pipelines in two approaches: message synchronization, and parallelism of computation and data communication.

2.2.1 Energy-based Models and Stochastic Gradient Descent

In general, deep learning problems can be formulated as an optimization model of data prediction. Given a data sample ξ_i, our goal is to predict an outcome value $y = F(\mathrm{x}; \xi_i)$ so that

$$\min_{\mathrm{x}\in\mathbb{R}^N} f_s(\mathrm{x}) := \mathbb{E}_{\xi_i \sim \mathcal{D}} F(\mathrm{x}; \xi_i) \tag{2.1}$$

where the training sample ξ_i has a probabilistic distribution $\mathcal{D}$ (i.e., sampling a data example from a learning dataset), the target model is constructed with learnable parameters x, and $f_s : \mathbb{R}^N \to \mathbb{R}$ present the objective function of our optimization problem.

One of the most popularly-used approach to solve the above problem is gradient-based optimization method. Bottou *et al.* [46] proved that with an assumption where $f_s(\cdot)$ is a non-convex function and is differentiable with Lipschitzian continuous gradients, Eq. (2.1) can be solved with an iterative process of optimization, which comprises of four elementary and repetitive steps:

1. A mini-batch of sample data, ξ_i, is sampled from the training data $\mathcal{D}$.

2. The loss value, $F_t(\mathrm{x}_t; \xi_t)$, is retrieved from the learning model via a

feed-forward pass of ξ_i.

3. The gradients with respect to model parameters, $\nabla F_t\left(\mathrm{x}_t; \xi_t\right)$, are computed via backward propagation.

4. The model parameters are updated along the estimated gradients.

From classical version, the stochastic gradient descent (SGD) technique was motivated with mini-batch learning, which has become wide-spread in distributed deep learning communities. The SGD with mini-batch is mathematically formulated as:

$$\begin{aligned} G_t\left(\mathrm{x}_t\right) &= \nabla F_t\left(\mathrm{x}_t; \xi_t\right) \\ \mathrm{x}_{t+1} &= \mathrm{x}_t - \gamma G_t\left(\mathrm{x}_t\right) \end{aligned} \tag{2.2}$$

where x_t and x_{t+1} respectively represent the model parameters at time step t and $t+1$. The model parameters are iteratively adjusted with a learning rate γ.

Obviously, with a large-scale dataset, the traditional pipeline of SGD becomes time-consuming. Hence, it is necessary to parallelize the mini-batch learning on multiple processing units to accelerate the speed of model training with a distributed perspective. As a result, data communication in training should be optimized in order to fully leverage the computational power of distributed clusters of devices. In other words, recent research in distributed learning of deep training model concentrates on maintaining a high ratio of the computation to communication ratio with different strategies.

2.2.2 Parallelism Schemes of Distributed Training

In model training, there are a couple of architectural design of implementing parallelism: model parallelism and data parallelism. Each of them is devised to deal with different routines. While model parallelism is proposed to cope with huge network design that does not fit on a single computational device, data parallel is originated to compartmentalize a large dataset with replicated entity of learning model on processing units.

Model Parallelism

Model parallelism aims to segregate parameters of a large model on multiple processing workers [47]. In this context, each processing unit hold a part

of the model as illustrated in Fig 2.1. There are two approaches regarding this technique: vertical splitting and horizon splitting. The key idea of these approaches is to make the training of a huge-sized neural network become feasible because we do not need to allocate all components of a model on a single device. However, there is a fact that because of the separated ownership of model parameters among workers, there is an inevitable dependency between different layers within model training. This means that a worker has to suffer a high latency when they must wait for the computing output results from other workers before initiating the local computation. Unbalance allocation of model parameters is considered as a critical problem in this issue. Model partitioning is a NP-complete [48] where we need to relax the computing dependency while the robustness of model parallelism must be conserved. A non-trivial solution for device placements has been recently suggested to assign the parameters to appropriate computing nodes [49].

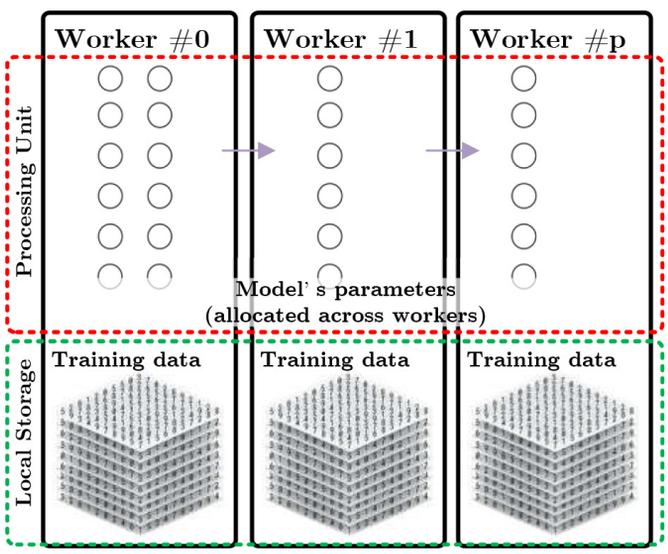

Figure 2.1: A scheme of model parallelism

Data Parallelism

Data parallelism is a type of distributed training where each of all computing nodes occupies a replication of a learning model. In this strategy, at each iteration, workers fetch different mini-batches of learning data, and the perform local SGD optimization as in conventional model. Fig 2.2 illustrates an architecture of data parallelism. Significantly, to ensure the consistence and the convergence of the model, they have to exchange the gradient updates to each other before updating new values for model parameters. However, the mechanism of gradient synchronization may vary in order to balance the trade-off

between communication cost and computational performance. Using a crew of n distributed workers for data parallelism, Eq. (2.2) is rewritten as:

$$f^* := \min_{x \in \mathbb{R}^d} [f(x) := \frac{1}{n} \sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi_i \sim \mathcal{D}_i} F_i(x; \xi_i)}_{=: f_i(x)}] \tag{2.3}$$

where $f_i : \mathbb{R}^d \to \mathbb{R}$ is a local objective function which a worker of an id i focuses on. At the end of each training step, the scheme aims to optimize the average of the target function at all processing nodes.

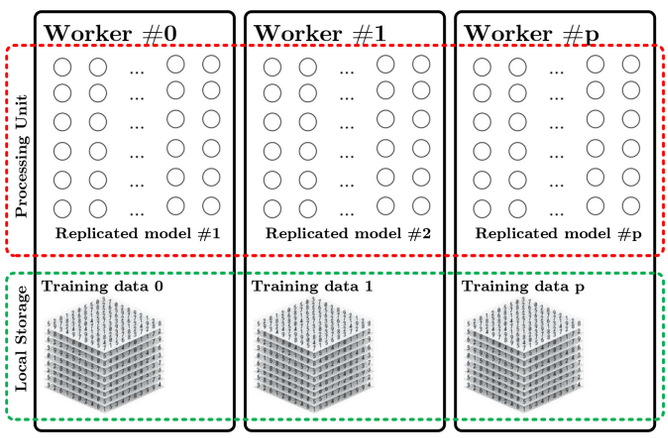

Figure 2.2: An architectural design of data parallelism

In following sections, we present communication-avoidance techniques in stochastic gradient descent regarding distributed training of multi-layered models with data parallelism.

2.2.3 Approaches of Communication-Avoidance in Distributed Stochastic Gradient Descent

The key idea of data parallelism is to parallelize SGD update in multiple processing nodes. A straightforward idea for this issue is to handle bulk synchronization of parallel SGD (BSP-SGD) [50] with an architecture of parameter server (PS) [51–53]. In this pipeline, at the same time, each processing element first reloads the global model parameters from PS, then the sample of training data is loaded and model gradients are calculated independently. At the end of each iteration, the gradient values are synchronized at all workers, and then they are aggregated on the PS. The global model is then updated after averaging all received local gradients at PS. This update scheme of distributed

SGD can be formulated as:

$$\begin{aligned} G_{i,t}\left(\mathrm{x}_t\right) &= \nabla F_{i,t}\left(\mathrm{x}_t; \xi_{i,t}\right) \\ \mathrm{x}_{t+1} &= \mathrm{x}_t - \gamma \frac{1}{n} \sum_{i=1}^{n} G_{i,t}\left(\mathrm{x}_t\right) \end{aligned} \tag{2.4}$$

where $G_{i,t}$ presents the local gradient of processing element i at time-step t, $F_{i,t}$.

From this approach, there are a triplet of critical factors that affect the communication cost among processing elements in distributed learning:

- The synchronization of communication and the frequency of data-exchange among participating elements
- The method of aggregation of model parameters in distributed memory
- The dimension of communication traffic among workers

To bolster the effectiveness of distributed training, recent research aims to address these issues with respect to the scalability of learning models.

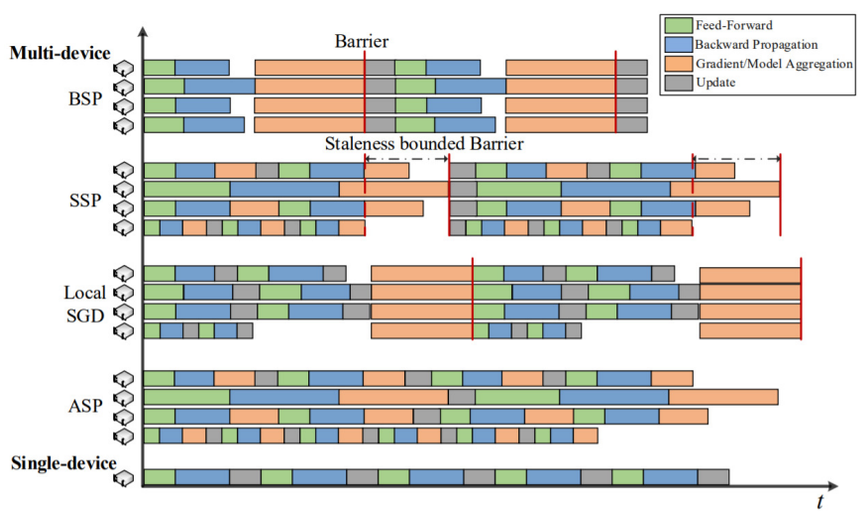

Figure 2.3: The illustrative summary of for communication synchronization in distributed training of deep learning models by Tang *et al.* [54]

Synchronization of Gradient Averaging

Regarding the synchonization of gradient averaging in communication-efficient distributed training, relaxing the synchronization among processing elements

is a research direction that defines how frequently workers exchange gradients with each other. The communication scheduling controls the frequency at which all of local models are synchronized with each others. This mechanism not only affects the communication traffic but also interferes in the convergence and the performance of learning model during training. Basically, there are four frameworks of gradient synchronization: synchronous SGD, stale-synchronous parallel (SSP), asynchronous parallel (ASP), and local GSD. Fig. 2.3 presents an illustration of approaches regarding communication-avoidance in distributed stochastic gradient descent.

Synchronous SGD Synchronous SGD is a classical data parallelism in distributed deep learning. They key idea of synchronous SGD is that, at each iteration, all workers have to wait until parameter transmission completes at every worker before continuing the next iteration of training. Bulk synchronization parallel is a conventional synchronous SGD [55]. This scheme ensures the model convergence as the synchronization of gradients are compulsory and consistent at all processing elements. However, persistence unavoidable mitigates the system output with the presence of stragglers. Hence, the framework introduces a great communication cost and limits the scalability of learning models. Synchronous SGD is employed in both kinds of architectures: centralize PS and decentralized systems. Regarding implementation of PS, followed with the completion of communication, all workers update to the same global model [56]. However, for decentralized arrangement, at the end of data exchange, all processing nodes do not necessarily keep the same learning model [57].

Stale-Synchronous Parallel The stale-synchrous parallel [51] is proposed to release the affect of the straggler problem while maintaining a consistence in synchronization. This framework lets the fast workers perform more updates than the slow ones with a goal of reducing the waiting time among workers. Particularly, a threshold of staleness is utilized to control the model consistency to ensure the convergence in distributed training. Given a staleness threshold s, the update at worker i at iteration $t+1$ is formulated as

$$\begin{aligned} \mathrm{x}_{i,t+1} = \mathrm{x}_0 &- \gamma \sum_{k=1}^{t} \sum_{j=1}^{n} G_{j,k}\left(\mathrm{x}_{j,k}\right) \\ &- \gamma \sum_{k=t-s}^{t} G_{i,k}\left(\mathrm{x}_{i,k}\right) - \gamma \sum_{(j,k)\in\mathcal{S}_{i,t+1}} G_{j,k}\left(\mathrm{x}_{j,k}\right) \end{aligned} \tag{2.5}$$

where $\mathcal{S}_{i,t+1}$ is some subset of the updates from other workers during period $t-s$.

Some improvement has been made to increase the effectiveness of this framework. To address the straggler problem, a novel model is proposed by Chen *et al.* [58], where the mini-batch SGD is only performed with a subset of extra workers. The primary idea is to let PS avoid waiting and let them update the model parameters from any arriving n workers. As a result, the slowest workers will be dropped as they arrive. Another research tackled the problem of stragglers with Round-Robin Synchronization Parallel [59] . The method reorder the arrangement of workers' updates by coordinating them with a fixed round-robin technique.

Asynchronous Parallel The asynchronous parallel (ASP) framework resolves the problem of degrading the system throughput by eliminating the synchronization. In this model, workers are independent to each other. Each of them continuously transfers the local gradients to the PS after the calculation completes. The PS updates the global model with received gradient values without waiting for any workers. Under mathematical formulation, the technique can be presented as

$$\mathrm{x}_{t+1} = \mathrm{x}_t - \gamma \sum_{i=1}^{n} G_{i,t-\tau_{i,k}} \left(\mathrm{x}_{i,t-\tau_{k,i}}\right) \tag{2.6}$$

where the $\tau_{i,k}$ is the delay period between the moment when worker i calculates the gradient at the current iteration

The most important characteristics of the asynchronous design is to make large-scale training system faster, more robust, and immune to the the failures of processing elements. To improve the efficience of the framework, Li *et al.* reduced the communication traffic by proposing Delayed Block Proximal Gradient Method where only a block of parameters is transmitted between master and workers, and they are asynchronously updated in each iteration. Grishchenko *et al.* exploited the sparsification of upward communications (i.e., from workers to master) in an asynchronous training system [60]. The authors simulated sparsification with a uniformly sampling of local update entries, which aims to make data communication more efficient. In general, without the synchronization among workers, asynchronous models usually end with a low degree of convergence in learning model [58].

Local SGD Local SGD [61] is a model of strict synchronization that allows all workers to perform local learning with several iterations before averaging all of them into a global model. The mathematical foundation of this framework

can be described as:

$$\mathrm{x}_{i,t+1} = \begin{cases} \mathrm{x}_{i,t} - \gamma G_{i,t}\left(\mathrm{x}_{i,t}\right), & \text{if } t+1 \notin \mathcal{I}_T \\ \mathrm{x}_{i,t} - \gamma \frac{1}{n} \sum_{i=1}^{n} G_{i,t}\left(\mathrm{x}_{i,t}\right), & \text{if } t+1 \in \mathcal{I}_T \end{cases} \tag{2.7}$$

Variants of this technique have been proposed to release the burden of huge communication cost. One of those methods is One-shot Averaging [10]. The authors aim to execute m independent workers in parallel for a number of iterations. After that, all model parameters are then averaged. Combining the distributed momentum SGD and the PR-SGD [62], Yu et al. [63] presented some improvement in the performance of local SGD with a linear speed-up of the training. Jiang et al. [64] reduced the communication complexity via exploiting the quantization method together with a local SGD framework. In general, with this framework, the less frequency of communication among workers leads to a degradation in training model's convergence. Hence this approach may require more training iterations than the vanilla SGD to achieve the same model accuracy. Hence, the trade-off of the communication must be scrutinized.

Parallelism of Computations and Communications

With a goal of making computational tasks and communication execute in parallel, studies focus on different algorithms of scheduling and pipelining. There is a fact that deep learning models are structured from sequential layers; so scheduling techniques take advantage of this characteristics to reduce the wasteful time of waiting between data-exchange and computational operations at each processing elements [65].

The common idea to diminish the communication cost is to find out the optimal order of computation and communication [66,67]. An representative technique of this approach is a wait-free backward propagation (WFBP) [68,69]. Another solution to this issue is to schedule a level of parallelism between compressing gradients' communications and letting processing elements compute on compression and backward propagation. [70].

In overall, there is not a complete solution to the communication-efficient learning. The recently proposed techniques generally reduce the amount of exchanged data among worker than conventional formulation; however, they have to penalize via degrading the convergence of the learning model.

3

An Unsupervised, Two-stage Network for Background Density Analysis and Motion Difference Approximation

Deep neural networks have shown their robustness of single-value mapping to generalize the contextual dynamics in background modelling [71, 72] and foreground detection [29, 36, 41]. Bouwmans *et al.* conducted a systematic survey with comparative evaluation of deep neural networks for background subtraction [15]. The authors pointed out five crucial requirements that any methods of background/foreground construction should address:

1. insensible to noise variation;

2. capable of high adaptation to contextual dynamics in background modelling;

3. able to cope with complicated changes in foreground segmentation;

4. coherent to the spatial and temporal arrangement of time-series data in sequences of images;

5. efficient to perform with real-time execution

First three conditions are ensured in deep neural networks. This is because convolutional architectures aims to learns contextual features of background images and foreground regions. Given a sufficient dataset of change detection, these pipelines of end-to-end networks are trained to learn all appearance of scene changes with dense layers. Deep features outperform traditional hand-crafted features of machine learning model with a high semantic conceptualization despite of high abstraction of comprehension [73]. Hence, with a sufficiently large amount of training, deep-learning-based methods is robust with a resistance to noise signals and a compliance with scene changes with complicated patterns without incremental learning. In addition to this, the processing with perspective of spatial and temporal space is one of the primary barriers of the techniques when moving objects present with diffent scales and aspect ratios. Several authors bridge this gap by adding spatial and/or temporal constraints by employing multiscale strategies in CNNs [29, 31, 36]. However, because deep neural networks formulate the learning process with single-valued mapping with a dense stack of computational layers, they are mostly time-consuming although acceleration of dedicated processing units (e.g., GPUs) is utilized. Criteria (4) and (5) regard challenges of deep learning methods in video processing. Nevertheless, traditional GMMs, which simulates the temporal learning on time-series data of images with statistical learning and difference thresholding [11, 12], are able to resolve these two lasts issues. In this situation, an integrated solution is an appropriate motivation to balance the trade-off between tremendous neural architectures and conventional machine learning models.

In this work, we propose a novel compact framework of CNN-based unsupervised background construction that exploits temporal information with lightweighted convolutional non-linear difference filtering to overcome the above drawbacks of previous studies in background subtraction, focusing on both background and foreground modeling. Comprehensively, the introduced convolutional background model captures attention towards static distributions while the CNN-based encoder-decoder filter focuses on representing the more general subtraction function, which is an idea commonly employed in post-processing steps of statistical methods to extract moving regions. The conditional density network are modeled in a completely general framework by combining the convention of CNN and a mixture of probabilistic functions. The key idea of this proposed technique is to exploit statistical inference to generalize the actual properties of observed sequences of scenes via modelling backgrounds with a learnable and explanable mixture of data distributions.

As shown with an overview in Fig. 3.1, the primary goal of our proposed framework is to address the previously listed problems of DNNs and statistical methods, via adaptively acquiring the underlying properties of a sequence of images to construct corresponding background scenes with CDN-GM (the left subfigure), and extract foregrounds of interest through data-driven learning with MEDAL-net (the right lower subfigure). Following pixel-wise temporal data reformation (the right upper subfigure), a batch of video frames is decompressed into a sequence of pixel histories to estimate each pixel's true background intensity with CDN-GM. After reconstructing the background image from the output intensity sequence of CDN-GM, the input frame is concatenated along the channel dimension with the background to estimate the final segmentation map. The concatenation before the foreground extraction step provides information to engender context-driven difference mapping within MEDAL-net, rather than memorizing the single-valued mapping between input frames and labeled foregrounds. This difference mapping idea effectively limits MEDAL-net's parameter search space, while enabling our proposed foreground extraction network to be more robust against various real-world motion dynamics.

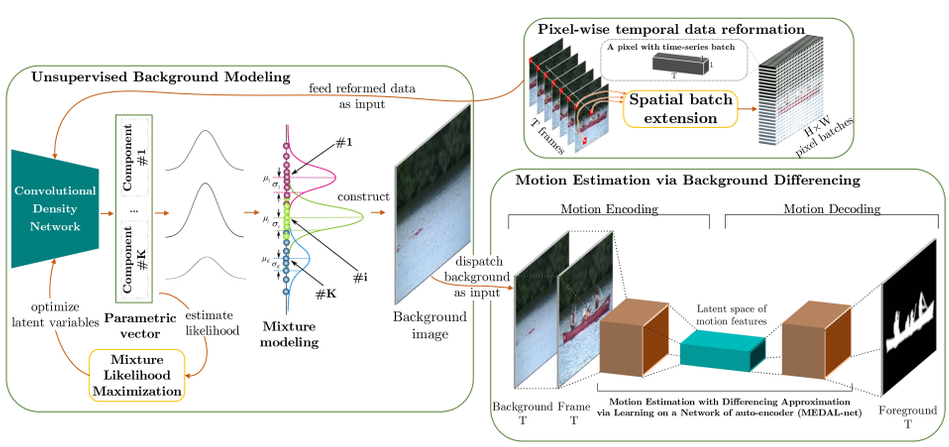

Figure 3.1: The overview of the proposed method for background modeling and foreground detection

3.1 Background Modelling with a Convolution Density Network of Gaussian Mixture

According to Zivkovic's study [12], let $\boldsymbol{\chi}_c^T = \{\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_T | \mathrm{x}_i \in [0, 255]^c\}$ be the time series of the T most recently observed color signals of a pixel where the dimension of the vector x_i in the color space is c, the distribution of pixel

intensity x_i can be modeled by a linear combination of K probabilistic components $\boldsymbol{\theta}_k$ and their corresponding conditional probability density functions $P(\mathrm{x}_i|\boldsymbol{\theta}_k)$. The marginal probability $P(\mathrm{x}_i)$ of the mixture is defined in:

$$P(\mathrm{x}) = \sum_{k=1}^{K} P(\boldsymbol{\theta}_k)P(\mathrm{x}|\boldsymbol{\theta}_k) = \sum_{k=1}^{K} \pi_k P(\mathrm{x}|\boldsymbol{\theta}_k) \tag{3.1}$$

where π_k is the non-negative mixing coefficient that sums to unity, representing the likelihood of occurrence of the probabilistic component $\boldsymbol{\theta}_k$.

Because of the multimodality of observed scenes, the intensity of target pixels is assumed to be distributed normally in a finite mixture. Regarding RGB space of analyzed videos, each examined color channel in $\mathbf{x}_i$ was assumed to be distributed independently and can be described with a common variance σ_k to avoid performing costly matrix inversion as indicated in [11]. Hence, the multivariate Gaussian distribution can be re-formulated as:

$$\begin{aligned} P(\mathrm{x}|\boldsymbol{\theta}_k) &= \mathcal{N}(\mathrm{x}|\boldsymbol{\mu}_k, \sigma_k) \\ &= \frac{1}{\sqrt{(2\pi)^c \sigma_k^c}} \exp\left(-\frac{\| \mathrm{x} - \boldsymbol{\mu}_k\|^2}{2\sigma_k}\right) \end{aligned} \tag{3.2}$$

where $\boldsymbol{\mu}_k$ is the estimated mean and σ_k is the estimated universal covariance of examined color channels in the k^{th} Gaussian component.

From this hypothesis, in this work, we propose an architecture of convolutional neural network, called Convolutional Density Network of Gaussian Mixtures (CDN-GM), which employs a set of non-linearity transformations $f_\theta(\cdot)$ to formulate a conditional formalism of GMM density function of x given a set of randomly selected, vectorized data points $\boldsymbol{\chi}_T$:

$$\mathrm{y}_T = f_\theta(\boldsymbol{\chi}_c^T) \sim P(\mathrm{x}|\boldsymbol{\chi}_\mathrm{c}^\mathrm{T}) \tag{3.3}$$

The ability of multilayer neural networks that was trained with an optimization algorithm to learn complex, high-dimensional, nonlinear mappings from large collections of examples increases their capability in pattern recognition via gathering relevant information from the input and eliminating irrelevant variabilities. With respect to problems of prediction, the conditional average represents only a very limited statistic. For applicable contexts, it is considerably beneficial to obtain a complete description of the probability distribution

of the target data. In this work, we incorporate the mixture density model with the convolutional neural network instead of a multi-layer perceptron as done by Bishop *et al.* in the vanilla research [17]. In the proposed scheme, the network itself learns to act as a feature extractor to formulate statistical inferences on temporal series of intensity values. First, regarding recently proposed CNN methods, the local connectivity characteristics in convolution layers motivate CNN to learn common visual patterns in a local region of images. Literally, a background image contains most frequently presented intensities in the sequence of observed scenes. Hence, in CDN-GM, we take advantage of this mechanism to exploit the most likely intensity value that will raise in the background image via consideration of temporal arrangement. Second, the memory requirement to store so many weights may rule out certain hardware implementations. In convolutional layers, shift invariance is automatically obtained by forcing the replication of weight configurations across space. Hence, the scheme of weight sharing in the proposed CNN reduces the number of parameters, making CDN-GM lighter and exploiting the parallel processing of a set of multiple pixel-wise analysis within a batch of video frames.

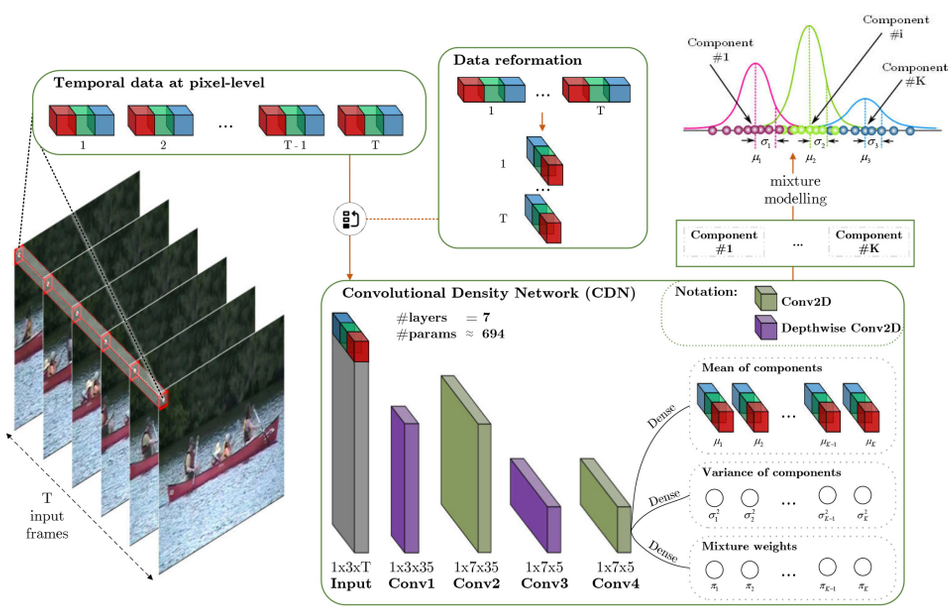

Figure 3.2: The proposed architecture of Convolution Density Network of Gaussian Mixture Model

The architecture of CDN-GM contains seven learned layers, not counting the input – two depthwise convolutional, two convolutional and three dense layers. Our network is summarized in Fig. 3.2. The input of our rudimentary architecture of the proposed network is a time series of color intensity at each pixel, which was analyzed with noncomplete connection schemes in four convolution layers regarding temporal perspective. Finally, the feature map of the last

convolution layer was connected with three different configurations of dense layers to form a three-fold output of the network which present the kernel parameter of the Gaussian Mixture Model.

The main goal of CDN-GM is to construct an architecture of CNN which presents multivariate mapping in forms of Gaussian Mixture Model with the mechanism of offline learning. With the simulated probabilistic function, we aim to model the description of the most likely background scenes from actual observed data. In other words, the regularities in the proposed CNN should cover a generalized presentation of the intensity series of a set of consecutive frames at pixel level. To achieve this proposition, instead of using separate GMM for each pixel-wise statistical learning, we consider to use a single GMM to formulate the temporal history of all pixels in the whole image. Accordingly, CDN-GM architecture is extended through a spatial extension of temporal data at image points with an extensive scheme defined in Table 3.1.

Table 3.1: Architecture of Convolutional Density Network

Type / Stride	Filter Shape	Output Size
Input	-	$(H * W) \times 1 \times T \times 3$
Conv dw / s7	$1 \times 7 \times 1$ dw	$(H * W) \times 1 \times 35 \times 3$
Conv / s1	$1 \times 1 \times 3 \times 7$	$(H * W) \times 1 \times 35 \times 7$
Conv dw / s7	$1 \times 7 \times 7$ dw	$(H * W) \times 1 \times 5 \times 7$
Conv / s1	$1 \times 1 \times 7 \times 7$	$(H * W) \times 1 \times 5 \times 7$
Dense / s1	$K \times C$	$(H * W) \times K \times d$
Dense / s1 / Softmax	K	$(H * W) \times K$
Dense / s1	K	$(H * W) \times K$

The network output y_T, whose dimension is $(c + 2) \times K$, is partitioned into three portions $y_\mu\left(\boldsymbol{\chi}_c^T\right)$, $y_\sigma\left(\boldsymbol{\chi}_c^T\right)$, and $y_\pi\left(\boldsymbol{\chi}_c^T\right)$ corresponding to the latent variables of GMM model:

$$\begin{aligned} y_T &= [y_\mu\left(\boldsymbol{\chi}_c^T\right), y_\sigma\left(\boldsymbol{\chi}_c^T\right), y_\pi\left(\boldsymbol{\chi}_c^T\right)] \\ &= [y_\mu^1, \ldots, y_\mu^K, y_\sigma^1, \ldots, y_\sigma^K, y_\pi^1, \ldots, y_\pi^K] \end{aligned} \tag{3.4}$$

With our goal of formulating the GMM, we impose a different restriction on threefold outputs from the network:

- First, as the mixing coefficients π_k indicate the proportion of data accounted for by mixture component k, they must be defined as independent and identically distributed probabilities. To achieve this regulation,

in principle, we activate the network output with a softmax activation function:

$$\pi_k(\boldsymbol{\chi}_c^T) = \frac{\exp(\mathrm{y}_\pi^k)}{\sum_{l=1}^{K} \exp(\mathrm{y}_\pi^l)} \tag{3.5}$$

- Second, in the realistic scenarios, the measured intensity of observed image signals may fluctuate due to a variety of factors, including illumination transformations, dynamic contexts and bootstrapping. In order to conserve the estimated background, we have to restrict the value of the variance of each component to the range $[\bar{\sigma}_{min}, \bar{\sigma}_{max}]$ so that each component does not span spread the entire color space, and does not focus on one single color cluster:

$$\sigma_k(\boldsymbol{\chi}_c^T) = \frac{\bar{\sigma}_{min} \times (1 - \hat{\sigma}_k) + \bar{\sigma}_{max} \times \hat{\sigma}_k}{255} \tag{3.6}$$

where $\sigma_k(\boldsymbol{\chi}_c^T)$ is normalized towards a range of $[0, 1]$ over the maximum color intensity value, 255; and $\hat{\sigma}_k$ is the normalized variance that was activated through a hard-sigmoid function from the output neurons y_σ that correspond to the variances:

$$\hat{\sigma}_k(\boldsymbol{\chi}_c^T) = \max\left[0, \min\left(1, \frac{\mathrm{y}_\sigma^k + 1}{2}\right)\right] \tag{3.7}$$

In this work, we adopt the hard sigmoid function because of the piecewise linear property and correspondence to the bounded form of linear rectifier function (ReLU) of the technique. Furthermore, this was proposed and proved to be more efficient in both in software and specialized hardware implementations by Courbariaux *et al.* [74].

- Third, the mean of the probabilistic mixture is considered on a normalized RGB color space where the intensity values retain in a range of $[0, 1]$ so that they can be approximated correspondingly with the normalized input. Similar to the normalized variance $\hat{\sigma}_k$, the mixture mean is standardized from the corresponding network outputs with a hard-sigmoid

function:

$$\mu_k(\boldsymbol{\chi}_c^T) = \max\left[0, \min\left(1, \frac{\mathrm{y}_\mu^k + 1}{2}\right)\right] \tag{3.8}$$

From the proposed CNN, we extract the periodical background image for each block of pixel-wise time series of data in a period of T. This can be done by selecting the means whose corresponding distributions have the highest degree of high-weighted, low-spread. To have a good grasp of the importance of a component in the mixture, we use a different treatment of weight updates with a ratio of $\pi_{k'}(\boldsymbol{\chi}_c^T) / \sigma_{k'}(\boldsymbol{\chi}_c^T)$. This is the manner of weighting components within a mixture at each pixel by valuing high-weighted, low-spread distributions in the mixture, thereby spotlighting the most significant distribution contributing to the construction of backgrounds.

$$BG(\boldsymbol{\chi}_c^T) = \max(\mu_k \cdot \hat{BG}_{k,T}), \quad \text{for } k \in [1, K] \tag{3.9}$$

where background mapping is defined at each pixel $\mathbf{x}$ as:

$$\hat{BG}_{k,T}(\boldsymbol{\chi}_c^T) = \begin{cases} 1, & \text{if } \underset{k'}{argmax}[\pi_{k'}(\boldsymbol{\chi}_c^T) / \sigma_{k'}(\boldsymbol{\chi}_c^T)] = k \\ & \text{for } k \in [1, K] \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

3.2 Background Learning via Training a Density Network with a Unsupervised Loss Function

In practice, particularly in each real-life scenario, the background model must capture multiple degrees of dynamics, which is more challenging by the fact that scene dynamics may also change gradually under external effects (e.g. lighting deviations). These effects convey the latest information regarding contextual deviations that may constitute new background predictions. Therefore, the modeling of backgrounds must not only take into account the various degrees of dynamics across multiple imaging pixels of the data source, but it must also be able to adaptively update its predictions with respect to semantic changes. Equivalently, in order to approximate a statistical mapping function for background modeling, the proposed neural network function has to be capable of approximating a conditional probability density function, thereby estimating a multi-modular distribution conditioned on its time-wise latest raw imaging inputs. The criteria for the neural statistical function to be instituted can be summarized as follows:

- As a metric for estimating distributions, input data sequences cannot be weighted in terms of order.

- Taking adaptiveness into account, the neural probabilistic density function can continuously interpolate predictions in evolving scenes upon reception of new data.

- The neural network function has to be generalizable such that its model parameters are not dependent on specific learning datasets.

Hence, satisfying the prescribed criteria, we propose a powerful loss function capable of directing the model's parameters towards adaptively capturing the conditional distribution of data inputs, thereby approximating a statistical mapping function in a technologically parallelizable form. At every single pixel, the proposed CNN estimates the probabilistic density function on the provided data using its GMM parameters. Specifically, given the set $\boldsymbol{\chi}_c^T$ randomly selected, vectorized data points, it is possible to retrieve the continuous conditional distribution of the data target x with the following functions:

$$P(\mathrm{x}) = \sum_{k=1}^{K} \pi_k(\boldsymbol{\chi}_c^T) \cdot \mathcal{N}(\mathrm{x}|\boldsymbol{\mu}_k, \sigma_k) \tag{3.11}$$

where the general disposition of this distribution is approximated by a finite mixture of Gaussians, whose values are dependent on our learnable neural variables:

$$\mathcal{N}(\mathrm{x}|\boldsymbol{\mu}_k, \sigma_k) = \frac{1}{\sqrt{2\pi \cdot \sigma_k(\boldsymbol{\chi}_c^T)^2}} \cdot \exp\left\{-\frac{\left\|\mathrm{x} - \mu_k(\boldsymbol{\chi}_c^T)\right\|^2}{2\sigma_k(\boldsymbol{\chi}_c^T)^2}\right\} \tag{3.12}$$

In our proposed loss function, the data distribution to be approximated is the set of data points relevant to background construction. This is rationalized by the proposed loss function's purpose, which is to direct the neural network's variables towards generalizing universal statistical mapping functions. Furthermore, even with constantly evolving scenes where the batches of data values also vary, this loss measure can constitute fair weighting on the sequence of inputs. Our proposed loss measure is designed to capture various pixel-wise dynamics over a video scene and to encompass even unseen perspectives via exploiting the huge coverage of multiple scenarios across more than one case with data. In other words, the order of the network's input does not matter upon loading, which is proper for any statistical function on estimating distribution.

For modeling tasks, we seek to establish a universal multi-modular statistical mapping function on the RGB color space, which would require optimizing the loss not just on any single pixel, but for b block of time-series image intensity data fairly into a summation value.

$$\mathcal{L} = \sum_{i}^{b} \sum_{j}^{T} \mathcal{L}_j^{(i)} \tag{3.13}$$

$$\text{where} \qquad \mathcal{L}_j^{(i)} = -\ln\left(\sum_{k=1}^{K} \pi_k^{(i)} \mathcal{N}(\mathrm{x}_j | \mu_k^{(i)}, \sigma_k^{(i)})\right) \tag{3.14}$$

where x_j is the j^{th} element of the i^{th} time-series data $\boldsymbol{\chi}_c^{T,(i)}$ of pixel values; $\pi^{(i)}$, $\mu^{(i)}$, and $\sigma^{(i)}$ are respectively the desired mixing coefficients, means, and variances that commonly model the distribution of $\boldsymbol{\chi}_c^{T,(i)}$ in GMM.

We define $\mathcal{L}_j^{(i)}$ as the error function for our learned estimation on an observed data point x_j, given the locally relevant dataset $\boldsymbol{\chi}_c^{T,(i)}$ for the neural function. $\mathcal{L}_j^{(i)}$ is based on the statistical log-likelihood function and is equal to the negative of its magnitude. Hence, by minimizing this loss measure, we will essentially be maximizing the expectation value of the GMM-based neural probabilistic density function $P(\mathrm{x})$, from the history of pixel intensities at a pixel position. Employing stochastic gradient descent on the negative logarithmic function $\mathcal{L}_j^{(i)}$ involves not only monotonic decreases, which are steep when close to zero, but also upon convergence it also leads to the proposed neural function approaching an optimized mixture of Gaussians probability density function.

In addition, since our loss function depends entirely on the input and the output of the network (i.e., without external data labels), the proposed work can be considered an unsupervised approach. This is because the objective of our network is to maximize the likelihood of the output on the data itself, not to any external labels. With this loss function, the optimization of the network to generalize on new data is available on the fly without needing any data labeled manually by humans. The key thing here is that whether the neural network can learn to optimize the loss function with the standard stochastic gradient descent algorithm with *back-propagation*. This can only be achieved if we can obtain suitable equations of the partial derivatives of the error $\mathcal{L}$ with respect to the outputs of the network. As we describe in the previous section, y_μ, y_σ, and y_π present the proposed CDN-GM's outputs that formulate to the latent variables of GMM model. The partial derivative $\partial \mathcal{L}_j^{(i)} \Big/ \partial \mathrm{y}^{(k)}$ can be evaluated for a particular pattern and then summed up to produce the derivative of the error function $\mathcal{L}$. To simplify the further analysis of the derivatives, it

is convenient to introduce the following notation that presents the posterior probabilities of the component k in the mixture, using Bayes theorem:

$$\Pi_k^{(i)} = \frac{\pi_k^{(i)} \mathcal{N}(\mathrm{x}_j | \boldsymbol{\mu}_k^{(i)}, \sigma_k^{(i)})}{\sum_{l=1}^{K} \pi_l^{(i)} \mathcal{N}(\mathrm{x}_j | \boldsymbol{\mu}_l^{(i)}, \sigma_l^{(i)})} \tag{3.15}$$

First, we need to consider the derivatives of the loss function with respect to network outputs y_π that correspond to the mixing coefficients π_k. Using Eq. (3.14) and (3.15), we obtain:

$$\frac{\partial \mathcal{L}_j^{(i)}}{\partial \pi_k^{(i)}} = \frac{\Pi_k^{(i)}}{\pi_k^{(i)}} \tag{3.16}$$

From this expression, we perceive that the value of $\pi_k^{(i)}$ explicitly depends on $\mathrm{y}_\pi^{(l)}$ for $l = 1, 2, ..., K$ as $\pi_k^{(i)}$ is the result of the softmax mapping from $\mathrm{y}_\pi^{(l)}$ as indicated in Eq. (3.5). We continue to examine the partial derivative of $\pi_k^{(i)}$ with respect to a particular network output $\mathrm{y}_\pi^{(l)}$, which is

$$\frac{\partial \pi_k^{(i)}}{\partial \mathrm{y}_\pi^{(l)}} = \begin{cases} \pi_k^{(i)}(1 - \pi_l^{(i)}), & \text{if } k = l \\ -\pi_l^{(i)} \pi_k^{(i)}, & \text{otherwise.} \end{cases} \tag{3.17}$$

By chain rule, we have

$$\frac{\partial \mathcal{L}_j^{(i)}}{\partial \mathrm{y}_\pi^{(l)}} = \sum_k \frac{\partial \mathcal{L}_j^{(i)}}{\partial \pi_k^{(i)}} \frac{\partial \pi_k^{(i)}}{\partial \mathrm{y}_\pi^{(l)}} \tag{3.18}$$

From Eq. (3.15), (3.16), (3.17), and (3.18), we then obtain

$$\frac{\partial \mathcal{L}_j^{(i)}}{\partial \mathrm{y}_\pi^{(l)}} = \pi_l^{(i)} - \Pi_l^{(i)} \tag{3.19}$$

For $\mathrm{y}_\sigma^{(k)}$, we make use of Eq. (3.2), (3.6), (3.7), (3.14), and (3.15), by differentiation, to obtain

$$\frac{\partial \mathcal{L}_j^{(i)}}{\partial \mathrm{y}_\sigma^{(k)}} = \frac{3.2}{255} \Pi_k^{(i)} \left(\frac{c}{2} \sqrt{(2\pi)^c (\sigma_k^{(i)})^{c+2}} - \frac{\| \mathrm{x}_j - \boldsymbol{\mu}_k \|^2}{2(2\pi)^c (\sigma_k^{(i)})^{c+2}} \right) \tag{3.20}$$

for $-2.5 < \mathrm{y}_{\sigma}^{(k)} < 2.5$. This is because the piece-wise property in the definition of the hard-sigmoid activation function.

Finally, for $\mathrm{y}_{\mu}^{(k)}$, let $\mu_{k,l}^{(i)}$ be the l^{th} element of the mean vector where l is an integer lies in $[0, c)$ and suppose that $\mu_{k,l}^{(i)}$ corresponds to an output o_k^{μ} of the network. We can get derivative of $\mu_{k,l}^{(i)}$ by taking Eq. (3.2), (3.8), (3.14), (3.15) into the differentiation process:

$$\frac{\partial \mathcal{L}_j^{(i)}}{\partial \mathrm{y}_{\mu}^{(k)}} = 0.2 \times \Pi_k^{(i)} \frac{x_{j,l} - \mu_{k,l}^{(i)}}{\sigma_k^{(i)}} \tag{3.21}$$

for $-2.5 < \mathrm{y}_{\mu}^{(k)} < 2.5$.

From Eq. (3.19), (3.20), and (3.21), when CDN-GM is performed data-driven learning individually on each video sequence using Adam optimizer with a learning rate of α, the process tries to regulate the values of laten parameters in the mixture model via minimizing the negative of log likelihood function. Hence, once the proposed model has been trained on video sequences, it is obviously seen that the network can predict the conditional density function of the target background, which is a statistical description of time-series data of each image point, so far, the foreground mask is then segmented correspondingly. The primary conceptualization in the model is to address the problems of DNNs as we mentioned above via online adaptively acquiring the underlying properties of a sequence of images to construct corresponding background scenes at concrete moments rather than memorizing the single-valued mapping between input frames and labelled backgrounds.

3.3 Foreground Segmentation with a Non-Linear Approximating Frame-Difference

In this section, we present the description of our proposed convolutional auto-encoder, called MEDAL-net, which simulates non-linear frame-background differencing for foreground detection. Traditionally, thresholding schemes are employed to find the highlighted difference between an imaging input and its corresponding static view in order to segment motion. For example, Stauffer and Grimson [11] employed variance thresholding on background - input pairs by modeling the static view with the Gaussian Mixture Model. While the experimental results suggest certain degrees of applicability due to its simplicity,

the approach lacks in flexibility as the background model is usually not static and may contain various motion effects such as occlusions, stopped objects, shadow effects, etc.

In practice, a good design of a difference function between the current frame and its background must be capable of facilitating motion segmentation across a plethora of scenarios and effects. However, for the countless scenarios in real life, where there are unique image features and motion behaviors to each, there is yet any explicit mathematical model that is general enough to cover them all. Because effective subtraction requires high-degreed non-linearity in order to compose a model for the underlying mathematical framework of many scenarios, following the Universal approximation theorem [75], we design the technologically parallelizable neural function for an approximation of such framework. Specifically, we make use of a CNN to construct a foreground segmentation network. The motive is further complemented by two folds:

- Convolutional Neural Networks have long been known for their effectiveness in approximating nonlinear functions with arbitrary accuracy.

- Convolutional Neural Networks are capable of balancing between both speed and generalization accuracy, especially when given an effective design and enough representative training data.

However, recent works exploiting CNN in motion estimation are still generating heavy-weighted models which are computationally expensive and not suitable for real-world deployment. In our proposed work, we exploit the use of a pair of the current video frame and its corresponding background as the input to the neural function and extract motion estimation. By combining this with a suitable learning objective, we explicitly provide the neural function with enough information to mold itself into a context-driven non-linear difference function, thereby restricting model behavior and its search directions. This also allows us to scale down the network's parameter size, width, and depth to focus on learning representations while maintaining generalization for unseen cases. As empirically shown in the experiments, the proposed architecture is light-weighted in terms of the number of parameters, and is also extremely resource-efficient, e.g. compared to FgSegNet [40].

3.3.1 The Proposed Architectural Design of Foreground Detection Network

The overall flow of the foreground detection network is shown in Fig. 3.3. We employ the encoder-decoder design approach for our segmentation function.

With this approach, data inputs are compressed into a low-dimensional latent space of learned informative variables in the encoder, and the encoded feature map is then passed into the decoder, thereby generating foreground masks. There are two components in our network: feature decoding and foreground upsampling. While the former element attends to compress the motion feature of observed scenes, the latter counterpart aims to reconstruct the foreground mask from extracted properties.

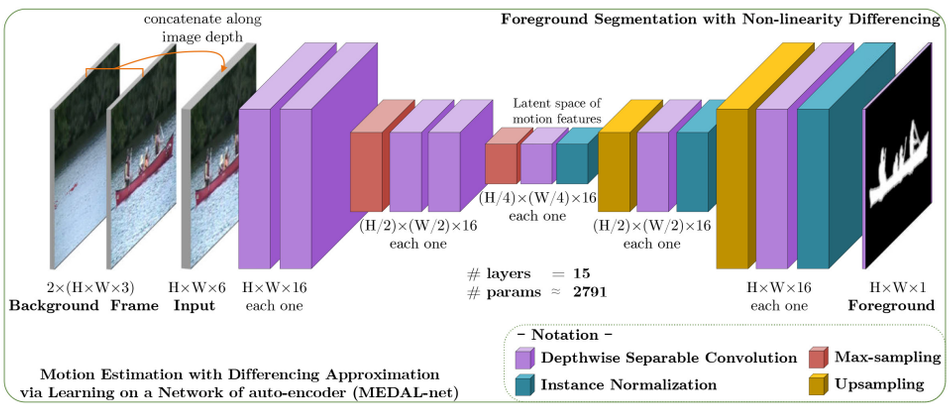

Figure 3.3: The proposed architecture of MEDAL-net grounded on convolutional autoencoder for foreground detection

In our design, we fully utilize the use of depthwise separable convolution introduced in MobileNets [76] so that our method can be suitable for mobile vision applications. Because this type of layer significantly scales down the number of convolutional parameters, we reduced the number of parameters of our network by approximately 81.7% compared to using only standard 2D convolution, rendering a light-weighted network of around 2,800 parameters. Interestingly, even with such a small set of parameters, the network still does not lose its ability to generalize predictions at high accuracy. Our architecture also employs normalization layers, but only for the decoder. This design choice is to avoid the loss of information in projecting the contextual differences of background-input pairs into the latent space via the encoder, while formulating normalization to boost the decoder's learning. The architecture of the proposed model is described in Table 3.2.

Encoder The encoder can be thought of as a folding function that projects the loaded data into an information-rich low-dimensional feature space. In our architecture, the encoder takes in pairs of video frames and their corresponding backgrounds concatenated along the depth dimension as its inputs. Specifically, the background image estimated by CDN-GM is concatenated with imaging signals such that raw information can be preserved for the neural network to freely learn to manipulate. Moreover, with the background image also in its raw form, context-specific scene dynamics (e.g. moving waves, camera jittering,

Table 3.2: Body Architecture of MEDAL-net

Type / Stride	Filter shape	Ouput size
Input	-	N x H x W x 6
DW conv / s1	3 x 3 x 1	N x H x W x 6
Conv / s1 / ReLU	1 x 1 x 6 x 16	N x H x W x 16
DW conv / s1	3 x 3 x 1	N x H x W x 16
Conv / s1 / ReLU	1 x 1 x 16 x 16	N x H x W x 16
Max pool / s2	2 x 2 x 1	N x (H / 2) x (W / 2) x 16
DW conv / s1	3 x 3 x 1	N x (H / 2) x (W / 2) x 16
Conv / s1 / ReLU	1 x 1 x 6 x 16	N x (H / 2) x (W / 2) x 16
DW conv / s1	3 x 3 x 1	N x (H / 2) x (W / 2) x 16
Conv / s1 / ReLU	1 x 1 x 16 x 16	N x (H / 2) x (W / 2) x 16
Max pool / s2	2 x 2 x 1	N x (H / 4) x (W / 4) x 16
DW conv / s1	3 x 3 x 1	N x (H / 4) x (W / 4) x 16
Conv / s1	1 x 1 x 16 x 16	N x (H / 4) x (W / 4) x 16
InstanceNorm / ReLU	-	N x (H / 4) x (W / 4) x 16
Upsampling	-	N x (H / 2) x (W / 2) x 16
DW conv / s1	3 x 3 x 1	N x (H / 2) x (W / 2) x 16
Conv / s1	1 x 1 x 16 x 16	N x (H / 2) x (W / 2) x 16
InstanceNorm / ReLU	-	N x (H / 2) x (W / 2) x 16
Upsampling	-	N x H x W x 16
DW conv / s1	3 x 3 x 1	N x H x W x 16
Conv / s1	1 x 1 x 16 x 16	N x H x W x 16
InstanceNorm / ReLU	-	N x H x W x 16
DW conv / s1	3 x 3 x 1	N x H x W x 16
Conv / s1 / Hard Sigmoid	1 x 1 x 16 x 1	N x H x W x 1

intermittent objects) are also captured. Thus, as backgrounds are combined with input images to formulate predictions, MEDAL-net may further learn to recognize motions that are innate to a scene, thereby selectively segmenting motions of interest based on the context.

In addition, by explicitly providing a pair of the current input frame and its background image to segment foregrounds, our designed network essentially constructs a simple difference function that is capable of extending its behaviors to accommodate contextual effects. Thus, we theorize that approximating this neural difference function would not require an enormous number of parameters. In other words, it is possible to reduce the number of layers and the weights' size of the foreground extraction network to accomplish the task. Hence, the encoder only consists of a few convolutional layers, with 2 max-pooling layers for downsampling contextual attributes into a feature-rich latent space.

Decoder The decoder of our network serves to unfold the encoded feature map into the foreground space using convolutional layers with two upsampling layers to restore the original resolution of its input data.

In order to facilitate faster training and better estimation of the final output, we engineered the decoder to include instance normalization, which is apparently more efficient than batch normalization [77]. Using upsampling to essentially expand the latent tensors, the decoder also employs convolutional layers to induce non-linearity like the encoder.

The final output of the decoder is a grayscale probability map where each pixel's value represents the chance that it is a component of a foreground object. This map is the learned motion segmentation results with pixel-wise confidence scores determined on account of its neighborhood and scene-specific variations. In our design, we use the hard sigmoid activation function because of its property that allows faster gradient propagation, which results in less training time.

At inference time, the final segmentation result is a binary image obtained by placing a constant threshold on the generated probability map. Specifically, suppose X is a probability map of size $N \times H \times W \times 1$, and let the set F be defined as:

$$F = \left\{(x, y, z) | \mathrm{X}_{x,y,z,0} \geq \epsilon\right\} \tag{3.22}$$

where $x \in [0, N]$, $y \in [0, H]$, $z \in [0, W]$, and ϵ is an experimentally determined parameter. In other words, F is a set of indices of X that satisfy the threshold ϵ. The segmentation map $\hat{\mathrm{Y}}$ of size $N \times H \times W$ is obtained by:

$$\hat{\mathrm{Y}}_{i,j,k} = \begin{cases} 1, & (i, j, k) \in F \\ 0, & otherwise \end{cases} \tag{3.23}$$

where 1 represents indices classified as foreground, and 0 represents background indices.

3.3.2 The Methodology of Network Training

We present the approach of training our proposed network in data preparation and the training procedure.

Data preparation CDnet[1] [78] is a large-scale database of change detection. The dataset contains 53 realistic, self-captured video sequences which are categorized in 11 different scenarios to examine the dynamic adaptation of algorithms regarding background subtraction. Each sequence in the dataset contains around 5000 consecutive frames of a scene on average. In each scenes, authors hand-labelled the groundtruth of foreground mask for all of frames in each sequence. However, at the beginning and at the end of each image set, around one hundred of frames are set aside for model initialization; so there is no labelled data for frames in these periods.

In this research, we chose CDnet-2014 as a base dataset for model training because of the diversity of scene dynamics. The training dataset for MEDAL-net is carefully chosen by hand so that the data maintains the balance between background labels and foreground labels since imbalance data will increase the model's likelihood of being overfitted. We choose just 200 labeled ground truths to train the model. This is only up to 20% of the number of labeled frames for some sequences in CDnet, and 8.7% of CDnet's labeled data in overall. During training, the associated background of each chosen frame is directly generated using CDN-GM as MEDAL-net is trained separately from CDN-GM because of the manually chosen input-label pairs.

Training procedure We penalize the output of the network using the cross-entropy loss function commonly used for segmentation tasks $[x, y, z]$, as the goal of the model is to learn a Dirac delta function for each pixel. The description of the loss function is as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{H}\sum_{k=1}^{W}[\mathrm{Y}_{i,j,k}\log(\hat{\mathrm{Y}}_{i,j,k}) + (1-\mathrm{Y}_{i,j,k})\log(1-\hat{\mathrm{Y}}_{i,j,k})] \tag{3.24}$$

where Y is the corresponding target set of foreground binary masks for $\hat{\mathrm{Y}}$, the batch of predicted foreground probability maps. The network is trained for about 1000 epochs for each sequence in CDnet using Adam optimizer with the learning rate = 0.005.

With this straightforward learning objective applied on our CNN, the designed architecture is enabled to learn not only pixel-wise motion estimates of the training set, but it also is taught to recognize inherent dynamics in its data, and perform as a context-driven neural difference function to accurately interpolate region-wise foreground predictions of unseen perspectives.

1. http://changedetection.net/

4

Implementation and Evaluation

4.1 Method Implementation and Experimental Setup

In this section, we presents the perspective of implementation for both of training and inferencing steps. The proposed scheme is then evaluated to examine the efficiency and the effective of the approach with respect to the state-of-the-art studies. A subjective evaluation is carried out on the CDnet-2014 dataset, on which our model is well-trained with few hundreds of labelled examples in each of image sequence. Moreover, an objective assessment is performed on another unseen dataset, Wallflower, to scrutinize the contextual generation of the framework with similar motion dynamics.

4.1.1 The Description of Implementing Methodology

From theoretical findings in the previous chapter, we simulate a scheme of learning-based model with TensorFlow [79], a open-source interface for machine learning development. TensorFlow provides a wide range of numerical operations that cover a degree of symbolic mathematics in tensor-driven algebras. The library was initiated by Google to facilitate the development of

user-defined neural networks with a particular focus on model training and practical inference. With a pre-defined dataflow and differentiable arithmetic, TensorFlow encourages a wide spread of customized machine learning methods in both research communities and practical industry via easing the development of learning modules across a diversity of cutting-edge computing platforms (e.g., CPUs, GPUs and TPUs).

The proposed model of background modelling and foreground subtraction was implemented into two separated components. As mentioned in Section 3.1, the key idea of the convolutional density network is to characterize the statistical learning with an auto-differentiable pipeline at pixel level. In other words, we presents a distinguished Gaussian Mixture model at each image point. Hence, there are up to $H \times W$ Mixtures in a processing prototype of the proposed framework. This means that we can disassemble a time-series data of T images into flexible batches of a certain number of pixels' temporal data. On the other hand, as described in Section 3.3, our network of foreground detection receives pairs of input frame and background as model's inputs. This means that the latter network must be pending until the former module completes the background generation if we combine two components in an end-to-end pipeline. Because of the difference in the dimension of input data between two models, the bottleneck will occurs at the middle of the framework as the network of foreground segmentation must be procrastinating until the pixel batch at density network fulfill a desired image frame during training procedure. Hence, we found that it is necessary to separate the proposed scheme into a couple of counterparts to optimize the training time.

Regarding model inferencing, we found that there is a considerable latency in model initialization when the framework starts execution. This is because there are some task of model allocation on the memory behind the top-level of implementation. To resolve this issue, after training the model on data, we convert all of the parameters from TensorFlow API to CUDA-accelerated, multidimensional array on Python by using Chainer [80–82]. Chainer is a open-souce, Python-based library that supports deep learning with a focus on versatility. In the interface, the implementation of neural networks are presented in form of a dynamic numerical graphs with automatic differentiation, which was facilitated regarding computational parallelism on CUDA/cuDNN. In the following part, we demonstrate the experimental evaluation regarding both accuracy in foreground segmentation and real-time processing speed.

4.1.2 Experimental Setup and Evaluation Metrics

In this section, we proceed to verify experimentally the capabilities of the proposed method via comparative evaluations in capturing motion attributes.

This is in order to evaluate the effectiveness of CDN-MEDAL-net in foreground detection. Our proposed scheme is designed to explicitly incorporate the probabilistic density properties into the architecture to achieve accurate adaptiveness, while taking advantage of parallel computing technologies often used with DNNs to compete with state-of-the-art works in speed given its light structure. Therefore, we compare the accuracy of the proposed framework not only with unsupervised approaches that are light-weighted and generalizable without pretraining: GMM – Stauffer & Grimson [11], GMM – Zivkovic [12], SuBSENSE [37], PAWCS [83], TensorMoG [8], BMOG [13], FTSG [38], SWCD [84], but also with the data-driven, supervised models which trade computational expenses for high accuracy performance: FgSegNet_S [40], FgSegNet [85], FgSegNet_v2 [41], Cascade CNN [29], DeepBS [36], STAM [35].

In terms of chosen metrics for measuring motion features, we employ quantitative analysis on values that can be appraised from confusion matrices [86], i.e. Precision, Recall, F-Measure, False-Negative Rate (FNR), False-Positive Rate (FPR) and Percentage of Wrong Classification (PWC). The evaluating measurement involves following pre-defined quantities:

- **True Positive (TP):** the number of accurately detected foreground pixels;
- **True Negative (TN):** the number of accurately detected background pixels;
- **False Positive (FP):** the number of background pixels that are mistaken as foreground;
- **False Negative (FN):** the number of foreground pixels which are incorrectly detected as background.

With the overall results being drawn from the combination of all confusion matrices across given scenarios, the benchmarks on CDnet-2014 [78] were performed by comparing foreground predictions against provided ground-truths. Then, we evaluate the proposed framework trained with CDnet-2014 on Wallflower [87] without any tuning or retraining laten parameters to examine the capability of our proposed approach in unseen scenarios having similar dynamics. Finally, we will also analyze all methods in terms of processing speed with the image resolution of 320×240 and draw final conclusions.

In our experiment, the number of Gaussians K is empirically and heuristically to balance the CDN-GM's capability of modeling constantly evolving contexts (e.g. moving body of water) under many effects of potentially corruptive noises. With K too big, many GMM components many be unused or they simply capture the various noises within contextual dynamics. As the Gaussian component

corresponding to the background intensity revolves around the most frequently occurring color subspaces to draw predictions, the extra components serve only as either placeholders for abrupt changes in backgrounds, be empty or capture intermittent noises of various degrees. In practice, noise Gaussian components in GMM are pulse-like as they would appear for short durations, and low-weighted because they are not as often matched as background components. Nevertheless, they still present corruptive effects to our model. Our proposed CDN-GM model was set up with the number of Gaussian components $K = 3$ for all experimented sequences, and was trained on CDnet-2014 dataset with Adam optimizer using a learning rate of $\alpha = 1e^{-4}$.

In addition, the constants $\bar{\sigma}_{min}$ and $\bar{\sigma}_{max}$ were chosen such that no Gaussian components span the whole color space while not contracting to a single point that represents noises. If the $[\bar{\sigma}_{min}, \bar{\sigma}_{\max}]$ interval is too small, all of the Gaussian components will be likely to focus on one single color cluster. Otherwise, if the interval is too large, some of the components might still cover all intensity values, making it hard to find the true background intensity. Based on this assumption and experimental observations, we find that the difference between color clusters usually does not exceed approximately 16 at minimum and 32 at maximum.

Regarding MEDAL-net, the value of ϵ was emperically chosen to be 0.3 in order to extract the foreground effectively even under high color similarity between objects and background.

4.2 Overall summary of experimental results

The proposed framework was implemented on a CUDA-capable machine with an NVIDIA GTX 1070 Ti GPU or similar, along with the methods that require CUDA runtime, i.e., TensorMoG, DeepBS, STAM, FgSegNet, and Cascade CNN. For unsupervised approaches, we conducted our speed tests on the configuration of an Intel Core i7 with 16 GB RAM. Our results are recorded quantitatively with execution performance in frame-per-seconds (FPS), and time (miliseconds) versus accuracy in Fig. 4.1. For the effectiveness in accuracy of methods, we examine F-measure score. The benchmark of correctness among approaches will be presented in Section 4.3.

At the speed of 129.4510 fps, it is apparent that CDN-MEDAL-net is much faster than other supervised deep learning approaches, of which the fastest - FgSegNet_S - runs at 23.1275 fps. By concatenating estimations of background scenes with raw signals for foreground extraction, our approach makes such efficient use of hardware resources due of its completely lightweight archi-

tecture and the latent-space-limitation approach. In contrast, other DNNs architectures are burdened with a large number of trainable parameters to achieve accurate input-target mapping. Furthermore, the proposed scheme dominates the mathematically rigorous unsupervised methods frameworks in terms of speed and accuracy such as SuBSENSE, SWCD, and PAWCS, as their paradigms of sequential processing is penalized by significant penalties in execution. Significantly, the average speeds of the top three methods dramatically disparate. With the objective of parallelizing the traditional imperative outline of rough statistical learning on GMM, TensorMoG reformulates a tensor-based framework that surpasses our duo architectures at 302.5261 fps. On the other hand, GMM - Zivkovic's design focuses on optimizing its mixture components, thereby significantly trading off its accuracy to attain the highest performance. Notwithstanding, our proposed framework gives the most balanced trade-off (top-left-most) in addressing the speed-and-accuracy dilemma. Our model outperforms other approaches of top accuracy ranking when processing at exceptionally high speed, while obtaining good accuracy scores, at over 90% on more than half of CDnet's categories and at least 84%.

In overall, from the evaluating results on average F-measure benchmark and execution speed of examining methods, we have a clear attention that supervised-learning or neural-network-based methods focus on the optimization of performance accuracy. On the contrary, traditional machine learning techniques aim to gain scene adaptation with a high processing speed despite of low correctness. Nevertheless, our proposed method, CDN-MEDAL-net, harmonizes the trade-off between both of the approaches. Our work perform foreground segmentation with an average F-measure of around 89.72% while maintaining a high processing speed at 129 frame-per-second. Although the results gained from our framework are not the best in both of the evaluating metrics, there is a major advantage that the method is capable of being put into practice with an impressive execution speed.

In order to measure the overall performance of examined methods, we estimate a metric of efficiency score, which is define as:

$$Efficiency\_score = \frac{\text{Avg.F} - \text{measure}}{\text{Inferencetime}} \tag{4.1}$$

where we penalize slow-processing method and promote approaches with high average F-measure score. In other words, we aim to evaluate the percentage of correct foreground segmentation on per time-step of processing time (e.g., a milisecond). Table 4.1 presents the detailed measurement of inference time (in mili-second), the accuracy with average F-measure score when we benchmark on the whole CDnet-2014 dataset with 53 video sequences, 11 difference

scenarios.

From the overall evaluation, there is a clear separation between unsupervised learning methods and supervised counterparts. Although deep learning networks produce results with high F-measure scores, they are penalized by low-speed execution. In this context, traditional machine learning techniques of unsupervised group dominate the evaluation because most of them address the requirement of high speed execution with a relatively considerable accuracy which varies from 55.57% to 77.38%. Significantly, our duo of neural networks slightly drops the F-score to 89.72% while we maintain a high processing rate of 129.451 frame-per-second. Hence, in overall evaluation of efficiency, our proposed methods balance the trade-off among two approaches. Using a small set of training data, and a light-weighted, explainable architectures, our work additional enhances the comprehension in a neural network which are being referred as a blackbox. Integrating statistical learning in the model present a better contextual adaptation in background modelling, facilitating the perception to scene dynamics for foreground segmentation. This section is followed by the experimental benchmark of accuracy in two aspects: evaluation with pre-training and examination of generalization on unseen scenes.

Table 4.1: The overall evaluation of experimented methods regarding accuracy and execution speed

	Method	Time‡	Avg. FM∓	Efficiency
Unsupervised	GMM – S & G	119.697$_{(3)}$	0.5688	0.0681
	GMM – Zivkovic	419.950$_{(1)}$	0.5557	0.2334$_{(2)}$
	SuBSENSE	15.717	0.7377	0.0116
	PAWCS	12.159	0.7529$_{(3)}$	0.0092
	TensorMoG	302.526$_{(2)}$	0.7738$_{(1)}$	0.2341$_{(1)}$
	BMOG	102.025	0.6542	0.0667
	FTSG	10.191	0.7256	0.0074
	SWCD	20.061	0.7540$_{(2)}$	0.0151
*	**CDN-MEDAL-net**	**129.451**	**0.8972**	0.1161$_{(3)}$
Supervised	FgSegNet_S	23.128$_{(1)}$	0.9803$_{(2)}$	0.0227
	FgSegNet	21.543$_{(2)}$	0.9771$_{(3)}$	0.0210
	FgSegNet_v2	18.015$_{(3)}$	0.9847$_{(1)}$	0.0177
	Cascade CNN	12.521	0.9193	0.0115
	DeepBS	10.015	0.7439	0.0075
	STAM	10.812	0.8959	0.0097

In each column, $Red_{(1)}$ is for the best, $Green_{(2)}$ is for the 2^{nd} best, and $Blue_{(3)}$ is for the 3^{rd} best. *Semi-Unsupervised; ‡The inference time is measured in frame-per-second unit (higher is better). ∓The average F-measure scores are evaluated on CDnet-2014 dataset (higher is better). The efficiency score measures the overall evaluation between accuracy and execution speed among methods (higher is better).

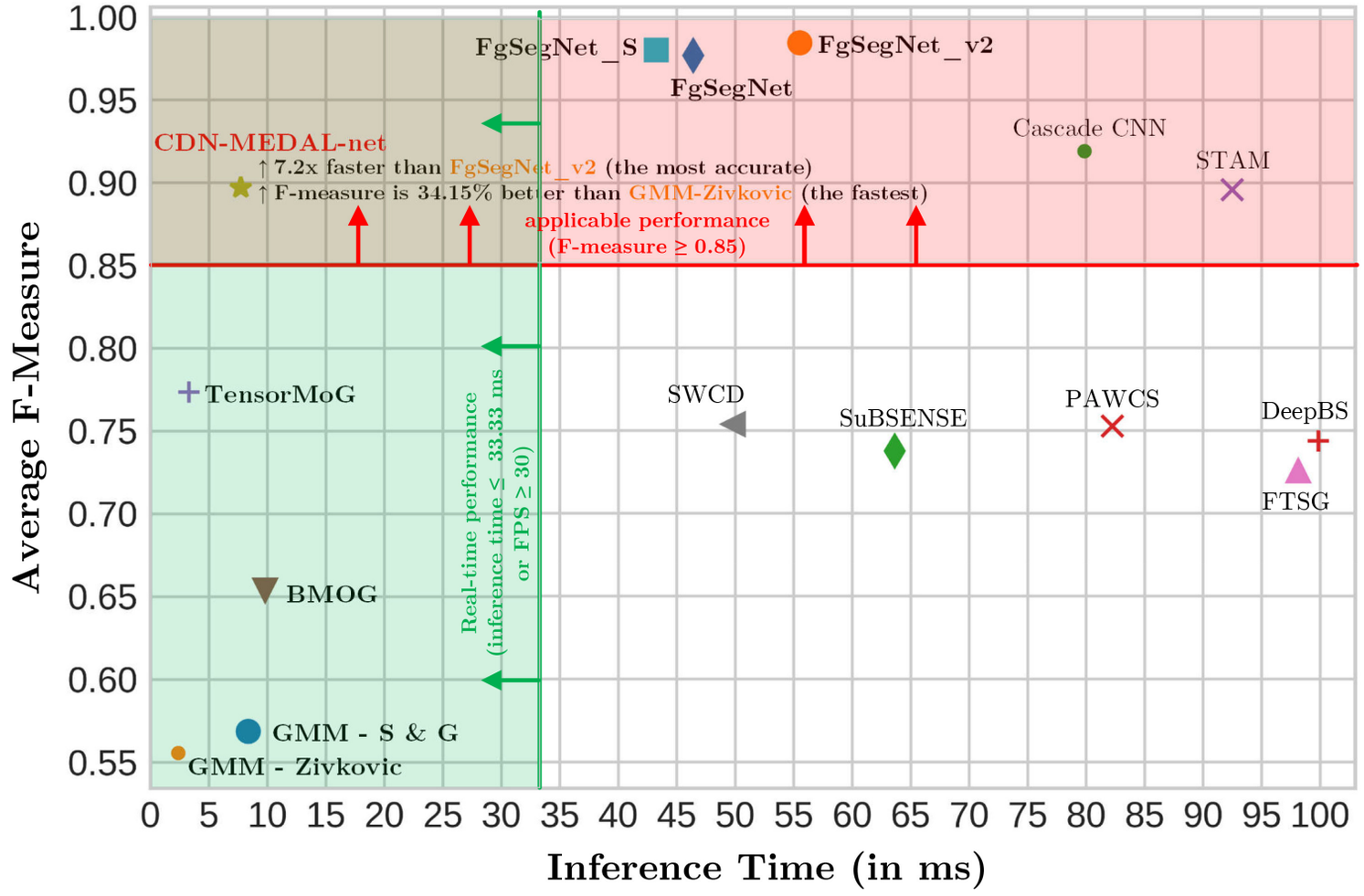

Figure 4.1: The summary of computational speed and average F-measure comparison with state-of-the-art methods on CDnet-2014. The upper red region defines an evaluation space where methods satisfy the applicability criteria of accuracy. The leftmost green area indicates approaches that met the requirement of real-time processing speed. Our proposed scheme of CDN-MEDALnet balances the trade-off of accuracy and processing speed between two approaches.

4.3 Experimental Benchmark on CDnet 2014 dataset

Using the large-scale CDnet-2014 dataset, we demonstrate empirically the effectiveness of our proposed approach across a plethora of scenarios and effects. For each thousands-frame sequence of a scenario, we sample only 200 foreground images for training our foreground estimator. This strategy of sampling for supervised learning is the same as that of FgSegNet's and Cascade CNN. The experimental results are summarized in Table 4.2, which highlights the F-measure quantitative results of our approach compared against several existing state-of-the-art approaches, along with Fig. 4.2 that provides qualitative illustrations. Despite its compact architecture, the proposed approach is shown to be capable of significantly outperforming unsupervised methods, and competing with complex deep-learning-based, supervised approaches in terms of accuracy on all but only the *PTZ* scenario. In this experimental dataset, we pass over the *PTZ* subdivision where our approach of CDN-GM is unsustainable to model the underlying description of the most likely background because of the fluctuation of actually observed data sequences when the recording camera rotates continuously. Accordingly, our MEDAL-net scheme of foreground segmentation encounters difficulty in estimating difference between input frames and corresponding background scenes.

In comparison with unsupervised models built on the GMM background modeling framework like GMM – Stauffer & Grimson, GMM – Zivkovic, BMOG and TensorMoG, the proposed approach is better augmented by the context-driven motion estimation plugin, without being constrained by simple thresholding schemes. Thus, it is able to provide remarkably superior F-measure results across the scenarios, especially on those where there are high degrees of noises or background dynamics like *LFR*, *NVD*, *IOM*, *CJT*, *DBG* and *TBL*. However, it is apparently a little worse than TensorMoG on *BDW*, *SHD*, *IOM* and *CJT*, which may be attributed to TensorMoG carefully tuned hyperparameters on segmenting foreground, thereby suggesting that the proposed method is still limited possibly by its architectural size and training data. Comparison with other unsupervised methods is also conducted, using mathematically rigorous approaches such as SuBSENSE, PAWCS, FTSG, SWCD that are designed to tackle scenarios commonly seen in real life (i.e. *BSL*, *DBG*, *SHD*, and *BDW*). Nevertheless, F-measure results of the proposed approach around 0.90 suggests that it is still able to outperform these complex unsupervised approaches, possibly ascribing to its use of hand-labeled data for explicitly enabling context capturing.

In comparison with supervised approaches, the proposed approach is apparently very competitive against the more computationally expensive state-of-

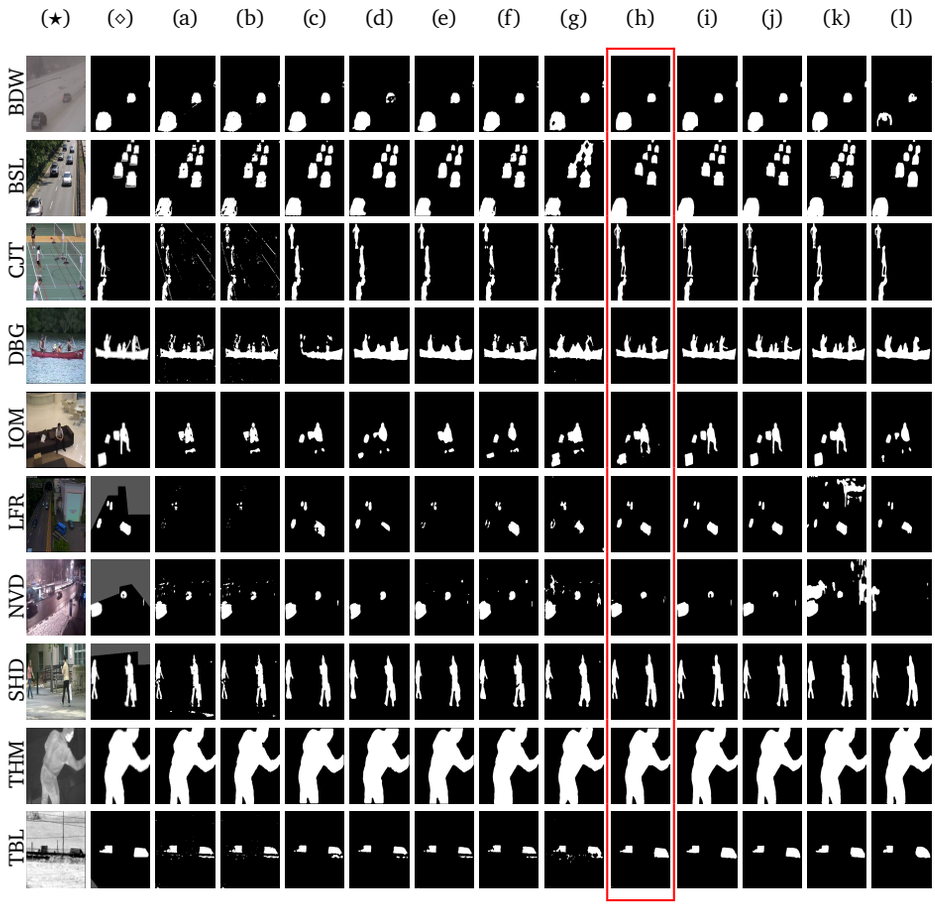

Figure 4.2: Visual quality comparison for foreground detection on all video sequences in eleven categories in CDnet 2014. The columns include: (★) input frame, (⋄) corresponding groundtruth foreground, (a) GMM – S & G, (b) GMM – Zivkovic, (c) SuBSENSE, (d) PAWCS, (e) BMOG, (f) FTSG, (g) SWCD, (h) CDN-MEDAL-net, (i) FgSegNet_S, (j) FgSegNet_v2 (k) Cascade CNN, (l) DeepBS. Experimented scenarios include bad weather (*BDW*), baseline (*BSL*), camera jitter (*CJT*), dynamic background (*DBG*), intermittent object motion (*IOM*), low frame rate (*LFR*), night videos (*NVD*), shadow (*SHD*), thermal (*THM*), and turbulence (*TBL*)

the-arts. For instance, our approach considerably surpasses the generalistic methods of STAM and DeepBS on *LFR* and *NVD*, but it loses against both of these methods on *SHD* and *CMJ*, and especially is outperformed by STAM on many scenarios. While STAM and DeepBS are constructed using only 5% of CDnet-2014, they demonstrate good generalization capability across multiple scenarios by capturing the holistic features of their training dataset. However, despite being trained on all scenarios, their behaviors showcase higher degrees of instability (e.g. with *LFR*, *NVD*) than our proposed approach on scenarios that deviate from common features of the dataset. Finally, as our proposed

method is compared against similarly scene-specific approaches like FgSegNet's, Cascade CNN, the results were within expectations for almost all scenarios that ours would not be significantly outperformed, as the compared models could accommodate various features of each sequence in their big architectures. However, surprisingly, our method surpasses even these computationally expensive to be at the top of the *LFR* scenarios. This suggests that, with a background for facilitating motion segmentation from an input, our trained model can better tackle scenarios where objects are constantly changing and moving than even existing state-of-the-arts.

Our proposed framework adopts semi-unsupervised approach, where the background generation is purely simulated with statistical learning and foreground segmentation is derived with few-shot learning on an architecture of an auto-encoder. With static scenes, we incorporate probabilistic inference with a neural network to emulate high adaptation of traditional machine learning techniques. There are two advantages that we can benefit from this component. First, the procedure of learning on observed scenes only enhances independence of labelled data for background generation, which is required at the second module. Second, simulating multi-valued mapping on a neural network overcomes the limitation of traditional least-square approach in conventional CNNs as describe by Bishop [17]. Fig. 4.3 illustrates the generated backgrounds of experimented scenes in CDnet-2014. Regarding foreground detection network, we aim to design a light-weighted architecture by superseding traditional convolutional layers with depthwise-separable alternatives to release the burden of memory consumption. Furthermore, because our outputs, foreground masks, are in binary domain, instead of using transpose convolution, we utilized the up-sampling techniques that perform up-scaling of intermediate feature maps with bi-linear interpolation. Although this manner will degrade the accuracy as we compare our results with those of FgSegNet's variants, our model ensures a sustainability in motion detection with a much smaller pipeline.

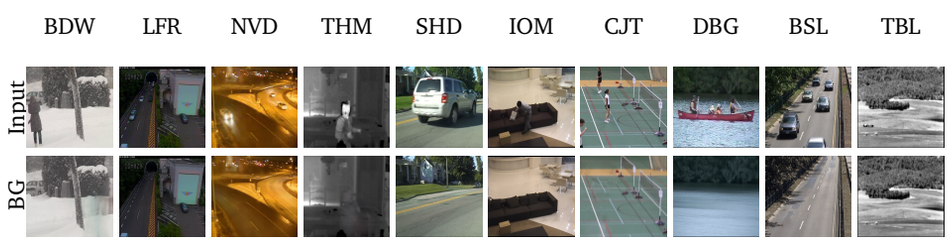

Figure 4.3: Visual illustration of background modelling using convolutional density network on different contexts of CDnet-2014 dataset

Table 4.2: F - measure comparisons over all of eleven categories in the CDnet 2014 dataset

	Method	*BDW*	*LFR*	*NVD*	*PTZ*	*THM*	*SHD*	*IOM*	*CJT*	*DBG*	*BSL*	*TBL*
Unsupervised	GMM – S & G	0.7380	0.5373	0.4097	0.1522	0.6621	0.7156	0.5207	0.5969	0.6330	0.8245	0.4663
	GMM – Zivkovic	0.7406	0.5065	0.3960	0.1046	0.6548	0.7232	0.5325	0.5670	0.6328	0.8382	0.4169
	SuBSENSE	$0.8619_{(2)}$	0.6445	$0.5599_{(3)}$	$0.3476_{(3)}$	$0.8171_{(3)}$	$0.8646_{(3)}$	0.6569	$0.8152_{(2)}$	0.8177	$0.9503_{(1)}$	$0.7792_{(2)}$
	PAWCS	0.8152	$0.6588_{(3)}$	0.4152	$0.4615_{(1)}$	$0.9921_{(1)}$	$0.8710_{(2)}$	$0.7764_{(3)}$	$0.8137_{(3)}$	$0.8938_{(1)}$	$0.9397_{(3)}$	0.6450
	TensorMoG	$0.9298_{(1)}$	$0.6852_{(2)}$	$0.5604_{(2)}$	0.2626	0.7993	$0.9738_{(1)}$	$0.9325_{(1)}$	$0.9325_{(1)}$	0.6493	$0.9488_{(2)}$	$0.8380_{(1)}$
	BMOG	0.7836	0.6102	0.4982	0.2350	0.6348	0.8396	0.5291	0.7493	0.7928	0.8301	0.6932
	FTSG	0.8228	0.6259	0.5130	0.3241	0.7768	0.8535	$0.7891_{(2)}$	0.7513	$0.8792_{(2)}$	0.9330	0.7127
	SWCD	$0.8233_{(3)}$	$0.7374_{(1)}$	$0.5807_{(1)}$	$0.4545_{(2)}$	$0.8581_{(2)}$	0.8302	0.7092	0.7411	$0.8645_{(3)}$	0.9214	$0.7735_{(3)}$
*	**CDN-MEDAL-net**	**0.9045**	**0.9561**	**0.8450**	-	**0.9129**	**0.8683**	**0.8249**	**0.8427**	**0.9372**	**0.9615**	**0.9187**
Supervised	FgSegNet_S	$0.9897_{(2)}$	$0.8972_{(2)}$	$0.9713_{(2)}$	$0.9879_{(1)}$	$0.9921_{(1)}$	$0.9937_{(3)}$	$0.9940_{(3)}$	$0.9957_{(2)}$	$0.9958_{(2)}$	$0.9977_{(1)}$	0.9681
	FgSegNet	$0.9845_{(3)}$	$0.8786_{(3)}$	$0.9655_{(3)}$	$0.9843_{(3)}$	$0.9648_{(3)}$	$0.9973_{(2)}$	$0.9958_{(1)}$	$0.9954_{(3)}$	$0.9951_{(3)}$	$0.9944_{(3)}$	$0.9921_{(2)}$
	FgSegNet_v2	$0.9904_{(1)}$	$0.9336_{(1)}$	$0.9739_{(1)}$	$0.9862_{(2)}$	$0.9727_{(2)}$	$0.9978_{(1)}$	$0.9951_{(2)}$	$0.9971_{(1)}$	$0.9961_{(1)}$	$0.9952_{(2)}$	$0.9938_{(1)}$
	Cascade CNN	0.9431	0.8370	0.8965	0.9168	0.8958	0.9414	0.8505	$0.9758_{(3)}$	0.9658	0.9786	0.9108
	DeepBS	0.8301	0.6002	0.5835	0.3133	0.7583	0.9092	0.6098	0.8990	0.8761	0.9580	0.8455
	STAM	0.9703	0.6683	0.7102	0.8648	0.9328	0.9885	0.9483	0.8989	0.9155	0.9663	$0.9907_{(3)}$

*Semi-Unsupervised; Experimented scenarios include bad weather (*BDW*), low frame rate (*LFR*), night videos (*NVD*), pan-tilt-zoom (*PTZ*), turbulence (*TBL*), baseline (*BSL*), dynamic background (*DBG*), camera jitter (*CJT*), intermittent object motion (*IOM*), shadow (*SHD*), and thermal (*THM*). In each column, higher is better; $Red_{(1)}$ is for the best, $Green_{(2)}$ is for the second best, and $Blue_{(3)}$ is for the third best.

From evaluation on F-measure scores, there is a significant difference in top-rating methods in both of approaches: supervised learning and unsupervised estimation. Compared with methods in the supervised approach, the experiment has demonstrated that unsupervised studies have no chance to dominate the data-driven, non-linear mapping of methods in the other counterpart. Neural-network based solutions, including FgSegNet, FgSegNet_S, FgSegNet v2, and Cascade CNN, exploit the high-level patterns of scene dynamics across various contexts to gain the generalization of regions that are in motion. However, they unexpectedly sink into a dataset-specific overfitting of a given sequence, where an intensive model fine-tuning is required to incorporate new contextual reformation. On the contrary, unsupervised-learning-based research concentrates on optimizing model's estimation in short-term analysis. Hence, when scenes change dramatically, like in *CJT*, *DBG*, and *LFR*, statistical models are distorted severely with low adaptation to the actual contexts. Our method is proposed to harmonize the trade-off between two approaches to gain both adaptation to circumstantial changes and generalization of motion patterns.

Overall, these comparisons serve to illustrate the superiority of the proposed approach in terms of accuracy over unsupervised approaches using only small training datasets, while cementing its practical use in its ability to compete with supervised ones despite its light-weighted structure. Table 4.4 presents evaluation metrics of a confusion matrix.

Table 4.4: Result of quantitative evaluation on CDnet 2014 dataset

	Method	*Average Recall*	*Average FPR*	*Average FNR*	*Average PWC*	*Average Precision*
Unsupervised	GMM – S & G	0.6846	0.0250	0.3154	3.7667	0.6025
	GMM – Zivkovic	0.6604	0.0275	0.3396	3.9953	0.5973
	SuBSENSE	0.8124	0.0096	$0.1876_{(1)}$	1.6780	0.7509
	PAWCS	$0.7718_{(3)}$	$0.0051_{(1)}$	0.2282	$1.1992_{(1)}$	$0.7857_{(2)}$
	TensorMoG	$0.7772_{(2)}$	0.0107	$0.2228_{(3)}$	2.3315	$0.8215_{(1)}$
	BMOG	0.7265	0.0187	0.2735	2.9757	0.6981
	FTSG	0.7657	$0.0078_{(3)}$	0.2343	$1.3763_{(3)}$	$0.7696_{(3)}$
	SWCD	$0.7839_{(1)}$	$0.0070_{(2)}$	$0.2161_{(2)}$	$1.3414_{(2)}$	0.7527
*	**CDN-MEDAL-net**	**0.9232**	**0.0039**	**0.0768**	**0.5965**	**0.8724**
Supervised	FgSegNet_S	$0.9896_{(1)}$	$0.0003_{(2)}$	$0.0104_{(1)}$	$0.0461_{(2)}$	0.9751
	FgSegNet	$0.9836_{(3)}$	$0.0002_{(1)}$	$0.0164_{(3)}$	$0.0559_{(3)}$	0.9758
	FgSegNet_v2	$0.9891_{(2)}$	$0.0002_{(1)}$	$0.0109_{(2)}$	$0.0402_{(1)}$	$0.9823_{(2)}$
	Cascade CNN	0.9506	0.0032	0.0494	0.4052	0.8997
	DeepBS	0.7545	0.0095	0.2455	1.9920	0.8332
	STAM	0.9458	$0.0005_{(3)}$	0.0542	0.2293	$0.9851_{(1)}$

*Semi-Unsupervised; For Recall and Precision, higher is better; For FPR, FNR, and PWC, lower is better; $Red_{(1)}$ is for the best, $Green_{(2)}$ is for the second best, and $Blue_{(3)}$ is for the third best.

4.4 Experimental Benchmark on Wallflower dataset without Parameter Fine-Tuning

In this section, using the Wallflower dataset, we aim to empirically determine our proposed approach's effectiveness on unseen sequences, using only trained weights from scenarios of similar dynamics in CDnet-2014. Wallflower[1] [87] is a realistic, self-captured dataset with outdoor and indoor sequences of images. Recalling from the previous section, we trained our model on CDnet-2014 with 200 labelled foreground samples for each input sequence. This accounts for around 20% of the number of labelled grouth-truth data in each sequence. The results apparently tend towards suggesting good degrees of our generalization from trained scenarios over to those unseen. Experimental evaluations are presented in Table 4.5, highlighting the F-measure quantitative results of our approach compared against some state-of-the-art methods in supervised, and unsupervised learning.

Specifically, on the *Camouflage* scenario, our approach presents a very high

1. https://www.microsoft.com/en-us/research/publication/wallflower-principles-and-practice-of-background-maintenance/

score of 0.97 in terms of F-measure using the *copyMachine* sequence of the *SHD* scenario in CDnet-2014. As the model learns to distinguish between object motions and the shadow effects of *copyMachine*, it even extends to recognizing object motions of similar colors. Under *Bootstrap* where motions are present throughout the sequence, we employ the straight-forward background subtraction function learned via the clear features of static-view-versus-motion of *highway* in *BSL*, giving an F-score of 0.768. Likewise, the model's capture of scene dynamics with *office* of *BSL*, *backdoor* of *SHD* and *fountain02* of *DBG* are extended towards respective views of similar features: *ForegroundAperture* of clear motions against background, *TimeOfDay* where there are gradual illumination changes and *WavingTrees* of dynamic background motions, providing decently accurate results. On the other hand, the *LightSwitch* scenario presents a big challenge where lightings are abruptly changed. As there is no scenario with this effect on the CDnet-2014 dataset, we chose the *SHD* simply for its ability to distinguish objects but the F-measure result is quite poor.

In comparison with existing methods whose aim are towards generalization like some unsupervised approaches GMM – Stauffer & Grimson, SuBSENSE, and CDnet-pretrained supervised approaches STAM, DeepBS, our proposed method yields very good results on *Camouflage* and *WavingTrees*, with even relatively better results on *Bootstrap*, *ForegroundAperture* and *TimeOfDay*. While obviously this does not evidence that our approach is capable of completely better generalization from training than others, it does suggest that the proposed framework is able to generalize to scenarios with dynamics similar to those learned, as supported by its relatively poor accuracy on *LightSwitch*.

Table 4.5: F - measure comparisons over the six sequences of Wallflower dataset with model parameters tuned on CDnet-2014

	Method	*Bootstrap*	*LightSwitch*	*WavingTrees*	*Camouflage*	*ForegroundAperture*	*TimeOfDay*
∓UnS.	GMM – Stauffer & Grimson	0.5306	0.2296	**0.9767**	0.8307	0.5778	0.7203
	SuBSENSE	0.4192	0.3201	0.9597	0.9535	0.6635	0.7107
*	**CDN-MEDAL-net**	**0.7680**	0.5400	0.8156	0.9700	**0.8401**	**0.7429**
∓Sup.	DeepBS [33]	0.7479	0.6114	0.9546	**0.9857**	0.6583	0.5494
	STAM	0.7414	**0.9090**	0.5325	0.7369	0.8292	0.3429

*Semi-Unsupervised; ∓UnS. = Unsupervised and Sup. = Supervised; In each column, higher is better; **Bold** is for the best within each scenario.

/5

Event-Triggered Distributed Training of Proposed Model with Communication Avoidance

Communication in distributed framework of deep model training imposes a major overhead which refrains the pipeline of parallel learning with a severe bottleneck among processing elements. In this section, we examine the effect of communication reduction in our proposed model of background subtraction when training with a distributed framework. Recent popularly-used application programming interfaces provide a set of synchronous message-passing operations in point-to-point level and collective communication, with which processing elements can exchange data with an esurance of message transmission. However, this technique ceases the computation flow at both ends of sending and receiving workers. Exploiting asynchronous communicating channels release the burden of process suspending because of data communication in processing groups. Hence, we present an implementation of data parallelism on CDnet-2014 dataset, where the communication overhead is reduced with presence of asynchronous behaviors and shared memory segments. This analysis aims to investigate the communication costs in visual learning models with a large-scale dataset using user-defined mechanisms.

5.1 An Introduction to Event-Triggered Communication

5.1.1 Existing Problems in Recently Proposed Frameworks

The presence of artificial neural networks initiated a revolution in learning-based approach thorough many aspects of data analysis and reasoning in a wide range of scientific communities. With gorgeous success in data-driven approximation, deep learning models superseded conventional techniques in machine learning thorough automating the parameter adjustment on a large scale of dataset. An attractive point of this methodology is that these multi-layered architectures are capable of approximating any functions with non-linear transformation with a space of interest [75]. Nevertheless, the research attention shifts towards these approach only when the models can perform computational tasks with an affordable amount of time. Accordingly, as the numerical perspective spans out of aspects of our life, we have to cope with scalability of greater model designs as well as larger training datasets. Optimizing the training steps becomes an existing challenge to research communication in both theoretical analysis and applicable implementation.

To accommodate the rapid scalability of learning models, we have to tailor the implementing designs of frameworks to leverage the computational capability and the memory space of large systems for machine learning models. This is referred as parallel computing or distributed implementation. The key idea of this manner is to scatter models or training datasets across processing elements. Using this pipeline, the biggest challenge of distributed training on large systems is the communication overhead between computing nodes [88]. Specifically, to ensure the accuracy and the convergence of the model, the weights and the bias of models' layers are exchanged and synchronized over the network at the end of each iteration. This compulsory requirement raises a bottleneck of the computational model as the size of the parallel processes increases [56, 89].

Recent years have witnessed a lot of research that optimize the communication tasks in distributed learning [59–62]. In this section, we adopt a model of triggered communication using event-based rule, called EventGraD [90]. The principal idea of this approach is to regulate the message exchange among processing workers with a fixed threshold of normalized values of parameters' values. With this framework, we re-implement our method of background subtraction with a distributed training scheme using asynchronous message-passing operators and global shared memory across processes. We aim to analyze the effect of communication avoidance as the number of exchanged messages among workers increase with respect to the control threshold during

the procedure of training. The perspective of implementation and experimental results are presented and discussed in the next section.

5.1.2 A Framework of Event-Triggered Communication in Parallel Distributed Model Learning

In conventional model of decentralized, distributed training, in each iterations, the local parameters at each workers have to be transmitted to and synchronized with nearest neighbors. This pipeline may cause a prodigality in allocated resources when the message is not necessarily communicated among processing units. Hence, we need to release the strictness of this communicating regulations. Accordingly, Ghosh *et al.* [90] proposed a method to trigger data exchanges among workers in the needed scenarios. In this section, we briefly recap the main skeleton of the techniques.

Similar to common data-parallelism approaches, in the EventGraD's model, each worker holds a replicated version of the learning model to perform parameters' updates with mini-batches of data. However, the key different of this method is that each worker have to perceive the changes in their local models. Specifically, when the normalized values of a parameter amplifies up to a threshold as compared with a old values in previous iterations, the node needs to communicate the new value to neighbors. Otherwise, the local model update is still performed as usual. At the end of each iteration, in stead of updating the model's parameter with local variables, we have to average the local values with those from corresponding neighboring workers. Supposing that at time-step k, the set of previously communicated values with respect to n layers of the learning model is:

$$\hat{X}_k := \begin{bmatrix} \hat{x}_{k,1} & \cdots & \hat{x}_{k,n} \end{bmatrix} \in \mathbb{R}^{N \times n} \tag{5.1}$$

where each element, $\hat{x}_{k,i}$, is a vector of the norm of N parameters:

$$\hat{x}_{k,i} = \begin{bmatrix} \hat{x}_{k,i,1} & \dots & \hat{x}_{k,i,N} \end{bmatrix}^\top \in \mathbb{R}^N. \tag{5.2}$$

The event-triggered mechanism of the EventGraD method is expressed as:

$$\hat{x}_{k+1,i,I} = \begin{cases} x_{k+1,i,I} & \text{if } \left\| \hat{x}_{k,i,I} - x_{k+1,i,I} \right\| \geq \delta_{k,i,I} \\ \hat{x}_{k,i,I} & \text{if } \left\| \hat{x}_{k,i,I} - x_{k+1,i,I} \right\| < \delta_{k,i,I} \end{cases} \tag{5.3}$$

where $x_{k+1,i}$ indicates the vector of norm of the same parameters in model layer I^{th} at time-step $k+1$, which is achieved from the local back-propagation; $\hat{x}_{k,i}$ denotes a vector of the previously communicated norm of the same parameters at time-step k; $\delta_{k,i,I}$ is a controlling threshold for worker i^{th} at iteration k

Accordingly, the update of model parameters is indicated as:

$$X_{k+1} = \hat{X}_k W - \gamma \partial F\left(\hat{X}_k; \xi_k\right) \tag{5.4}$$

Obviously, the selection of threshold estimation is important in this situation because they will control the number of message exchanges among processing elements. The authors chose the slope of local normalized values of parameters as a threshold for communication control. To be more detailed, the estimation of the slope is derived from the current value and the last communicated value in the same worker:

$$\delta_{k,i,I} = \underbrace{\frac{\left\|\hat{x}_{k,i,I} - x_{k+1,i,I}\right\|}{k - \hat{k}}}_{\text{Slope}} \times h, \tag{5.5}$$

where $\hat{k}$ is the time-step of the last data exchange.

In this framework, the authors analyzed the convergence rate of the learning model as well as the adaptation of the proposed threshold [90]. In our work, we utilize this model with a fixed value of threshold but we keep the semantic interpretation of the orginal work. Our goal is to examine the effect of communication-avoidance in a background subtraction model with different batch sizes, the number of processes and the various threshold values.

5.2 An Extensive Analysis in Distributed Training of Motion Difference Approximation

5.2.1 A Framework of EventGraD with GPU-accelerated Computation

The vanilla model of EventGraD [90] was proposed to formulate a control of data communication in distributed training. The work concentrated on the utilization of CPU-based computational capability to deal with small scale

of visual datasets. Two datasets that the authors used to experimented the method are MNIST [91], a data of $70,000$ images of hand-written digits in a resolution of 28×28, and CIFAR-10 [92], a dataset of $60,000$ 32×32 colour images in 10 classes for image classification. In our work, because the scope of video sequences is much greater than those of MNIST and CIFAR-10, we need to accelerate the training procedure with the computational power of GPUs. The illustration of this scheme is given in Fig. 5.2.

The difference between our architecture and the original model is that we allocate the learning model on GPUs. In each training step, the data is buffered from the local storage to the universal memory. Afterwards, the sampling data is copied to memory spaces on GPU. The estimation of gradient values is then performed directly on GPU. When communication events is triggered, the data is replicated on the central memory to exchange with other workers in the processing cluster. Finally, the last communicated parameters from neighbors are loaded from memory to graphical units to average all values before update new values for model variables. In this context, supposing that the model is initialized on memory spaces of graphical processing units, the communication cost covers some additional aspects:

- Fetch data from local storage to the primary memory spaces
- Form tensor-driven learning samples from system memory to processing units (e.g., GPUs, TPUs)
- Make a copy of model parameters from dedicated memory to main memory to facilitate the training synchronization among workers
- Communicate the model parameters between each of workers and their neighbors based on event-triggered regulation
- Allocate neighbors' model parameters on dedicated processing units to average the update of weights and bias of model's layers

In this analysis, we present a new perspective to implement this approach using asynchronous communicating operators with multi-GPU acceleration. After structuring the framework, we examine the affect of threshold-based control of communication within a group of processes with a varying range of training batch sizes.

5.2.2 The Discussion of Implementing Perspective

Regarding the implementation of experimental framework, we utilize UPC++ [93] and PyTorch [94, 95] with C++ API to simulate the training of our proposed background subtraction model with CUDA-compatible acceleration.

UPC++ is application programming interface which provides a support of Global Share Memory or Partitioned Global Address Space (PGAS) on C++ programming language. The library is structured with a goal to create a platform to develop more efficient, scalable, parallel programs on distributed-memory systems. One of the most component in UPC++ is the PGAS model, which is single program, multiple-data (SPMD). With this memory model, a shared memory, which is a global address spaces on system memory, is accessible to all of processing workers. In addition to this, each process has a private memory that is isolated with respect to other processes. Fig. 5.1 provides an illustration of PGAS memory model in UPC++.

To handle these kind of memory, UPC++ equips a variety of convenient methods for accessing and utilizing this global memory, as well as transmitting data from private space to global segments. A set of communication operations are also implemented, including remote memory access (RMA), remote procedure call (RPC), and collective communication (CC). All of these functions follow asynchronous mechanisms. Specifically, RMA engages one-side communication to facilitate memory access to remote, shared memory spaces, in which network interfaces are exploited to operate a data delivery with low latency. RMA operations use network hardware offload support to deliver low latency and high bandwidth. RPC enables user-defined functions to be executed at remote process. Significantly, UPC++ provides distributed objects as a mean to distribute objects with a same type across any subset of processes. Accordingly, RPC-based features translating distributed object arguments between global identifiers and the local representation in each process automatically and efficiently. In this context, explicit communication is required to obtain a global pointer from a distant instance of a distributed object.

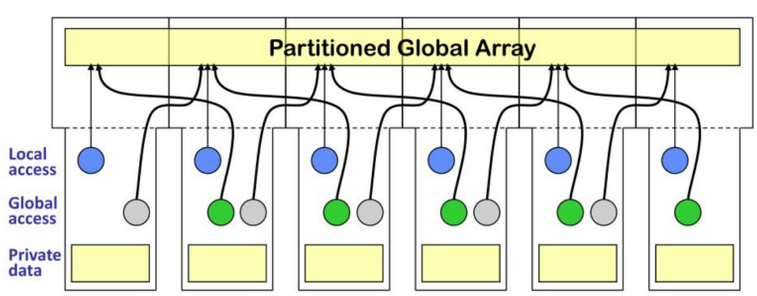

Figure 5.1: The model of Partitioned Global Address Space (PGAS) in UPC++.

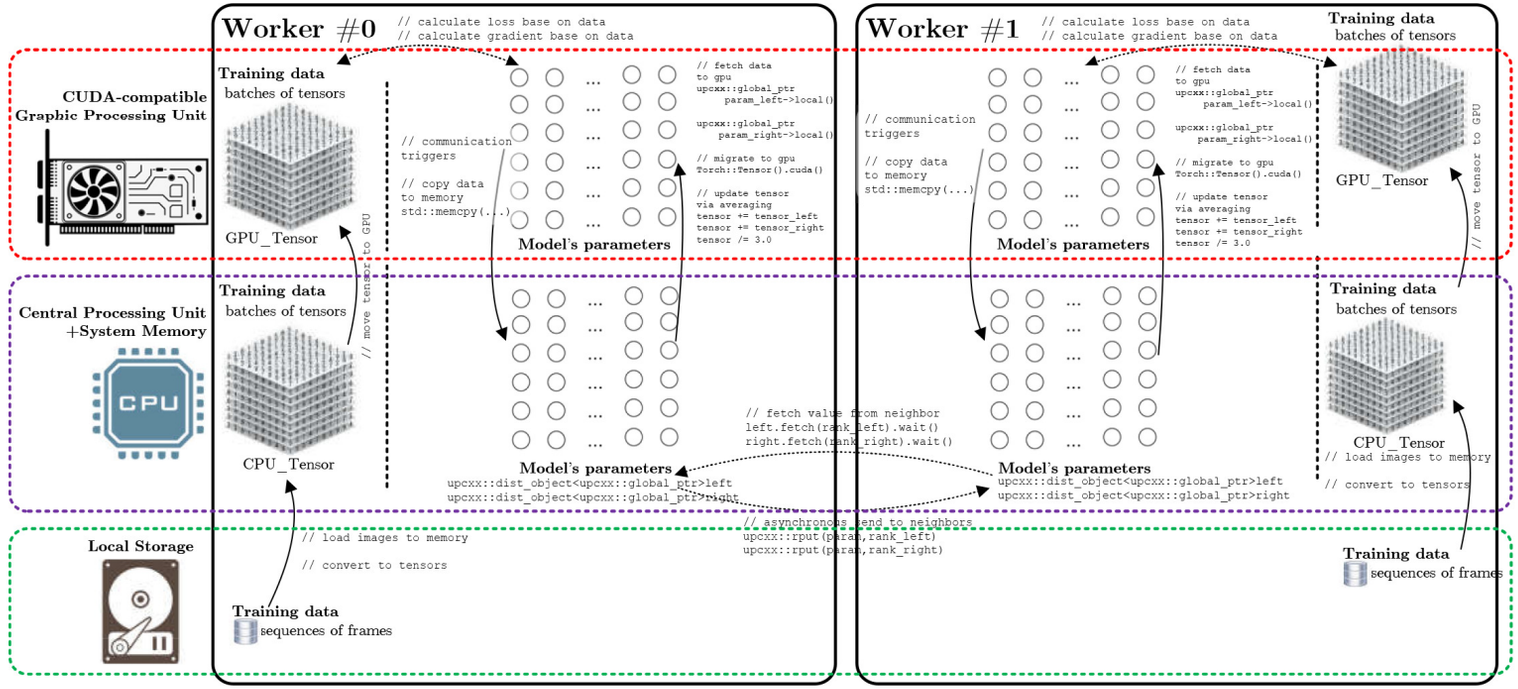

Figure 5.2: The illustration of the experimental data-parallelism scheme with UPC++ and PyTorch C++ API.

Using UPC++, we present the model's parameters in two kind of instances: tensors in PyTorch on GPUs and distributed arrays in UPC++ in main memory. When the communication is triggered by a threshold with respect to normalized values of our network's layers, the network's weight and bias is migrated from dedicated devices to local memory. Then, synchronous communications are performed with remote put operators (i.e., `upcxx::rput()`) to provide neighboring processes with a copies of local models. Similarly, a process retrieves the remote model from other neighbors using global pointers. In this design, we obviously separate communication and computation in two different kinds of processing units. In our implementation, we also extend the framework with multi-GPU processing for computation.

5.2.3 The Experimental Analysis of Communication Efficiency

In this section, we present our experimental results of the proposed analysis with the proposed framework of foreground detection. We investigate the communication avoidance via measuring the number of exchanged messages when training the model with different batch sizes and distributing the task on various number of processing elements. Specifically, we perform the investigation on a system with two NVIDIA RTX 6000. We execute the model with a range of batch sizes of 4, 8, 12, 16 through 50 training epochs. For each batch size of learning, we utilize an appropriate number of processes so that we can maximize the distributed scalability but still keep the analysis fitting the proposed systems. Fig. 5.3 illustrates a summary of our experimental results

Table 5.1 presents the number of communicated events that was counted on a group of distributed training workers with a batch size of 4. With the available resources, we experiment the framework with different number of UPC++ processes: 2, 4, 8, 12, 16. In overall, as the threshold rises gradually, the amount of exchanged messages reduces correspondingly regardless of the number of processes. As the training processes goes on further, the learning model tends to converge to fit the learning samples. At the moment, the difference of layers's weights and bias is diminished. Accordingly, workers do not perform unnecessary communicate to synchronize the learned parameters. Hence, at the end of the training, the number of messages approach to a convergence.

Table 5.1: The number of exchanged messages with respect to a range of threshold where the MEDALnet was trained wich a batch size of 4

Threshold	$\#proc = 2$	$\#proc = 4$	$\#proc = 8$	$\#proc = 12$	$\#proc = 16$
0.0001	57,478	114,518	226,258	22,842	40,280
0.0002	52,148	98,238	201,222	24,230	34,680
0.0004	41,688	85,076	153,052	15,756	30,078
0.0006	30,574	70,702	7,828	14,424	27,548
0.0008	28,726	53,976	22,496	12,842	23,880
0.001	21,210	48,162	12,960	16,156	23,508
0.002	9,096	33,444	9,308	11,368	18,750
0.004	5,688	9,564	9,992	11,188	16,248
0.006	3,142	5,752	9,288	11,150	15,536
0.008	2,782	3,906	8,626	13,182	15,012
0.01	2,748	3,782	11,014	11,136	14,940
0.02	2,110	3,724	8,388	11,528	14,848
0.04	1,996	3,712	7,424	11,136	15,042
0.06	1,860	3,712	7,504	11,136	14,848
0.08	1,856	3,712	7,580	11,136	14,848

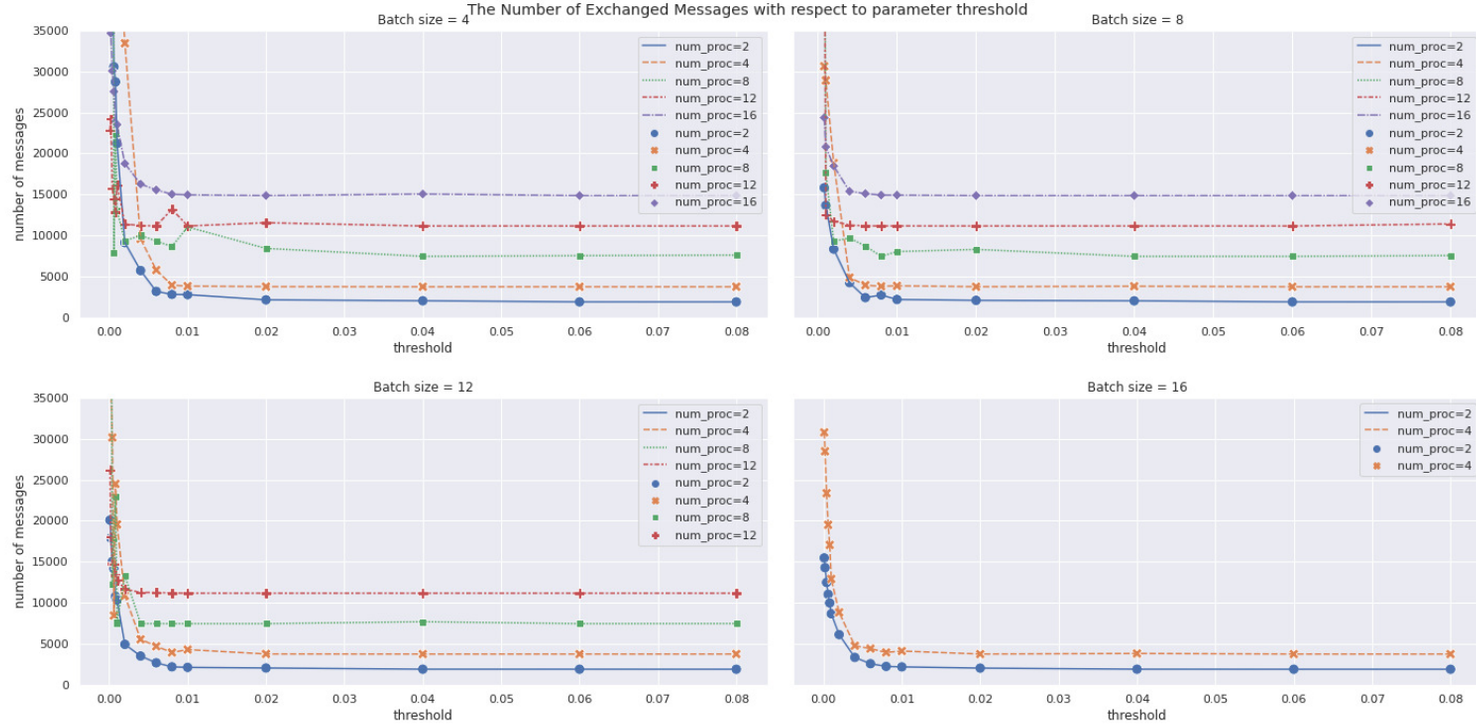

Figure 5.3: The illustration of the experimental results where we measure the number of transmitted messages with respect to various ranges of processes on different batch sizes through 50 epochs. We vary the number of batch size: 4, 8, 12, 16 with an appropriate number of processes that fits the experimented hardware configuration.

With a fixed threshold, when we increase the number of workers in the learning cluster, the communication cost experimented a significant growth. With a larger group of processing elements, the number of communicated messages linearly surges at the beginning of the training with respect to the size of the crew. This communication cost accumulates as the training goes by. Hence, we observe that the volume of data exchanges escalates when we enlarge the scale of distributed learning. The sample phenomena can be seen with batch size 8 in Table 5.2.

Table 5.2: The number of exchanged messages with respect to a range of threshold where the MEDALnet was trained wich a batch size of 8

Threshold	$\#proc = 2$	$\#proc = 4$	$\#proc = 8$	$\#proc = 12$	$\#proc = 16$
0.0001	29,224	57,148	113,096	20,582	39,696
0.0002	26,886	53,482	79,842	91,382	32,478
0.0004	22,008	44,726	7,774	15,400	28,630
0.0006	18,320	34,330	66,444	13,594	24,328
0.0008	15,796	30,650	65,284	57,006	24,356
0.001	13,648	28,910	17,626	12,514	20,784
0.002	8,304	18,822	9,278	11,696	18,458
0.004	4,208	4,836	9,662	11,244	15,368
0.006	2,380	3,902	8,682	11,152	15,088
0.008	2,702	3,762	7,424	11,140	14,906
0.01	2,152	3,820	8,000	11,142	14,908
0.02	2,046	3,716	8,266	11,136	14,848
0.04	1,990	3,766	7,424	11,136	14,848
0.06	1,860	3,712	7,424	11,136	14,848
0.08	1,856	3,712	7,520	11,382	14,848

From the experiment, we observe that the task of controlling communication threshold is a dilemma. With a large batch size, we can reduce the amount of communication cost significantly. However, in this situation, workers lacks of the awareness of learning at their neighbors. Accordingly, irregular synchronization among processing elements may lead to a degrading in model's inference. Our examination shows that with a batch size of 8, using 8 distributed learning processes, we witnessed a drop in evaluating accuracy from 95.1122% to 82.3151% when we increase the communication threshold from 0.0001 to 0.08. On the contrary, with a small threshold, we can make the synchronization of parameters more frequent, hence, ensuring the convergence and the generalization of the learning model. However, in this case, we have suffer a great amount of communication cost caused by unnecessary data exchanges among workers.

6

Discussion and Concluding Remarks

6.1 Discussion

In this work, since the literature has showed DNN's capacity to estimate any functions within highly parallelizable structures, the DNN-based method is objectively evaluated as a promising technique. We made use of this parallelizable computation on GMM to approximate the behind optimization of statistical learning in a way that enhances the creation of probabilistic model of our data with recent parallel computing technologies. As a result, it is possible to effectively exploit GMM-based background models' characteristics, which are clear and consistent with their mathematical framework via DNNs' similar data-driven performance. With experimental analysis, we demonstrated that the proposed method is robust to solve the issues of DNNs while utilizing their benefits by incorporating the mathematics of modeling statistical GMM into our processes. Specifically, we introduced a novel, lightweight, dual framework of two convolutional neural networks for the task of generically modeling backgrounds and segmenting motion regions. In this section, we aims to run over our results in this research, which include

First, we exploited the robustness of parallel computing to present a concept that leverages conventional GMM model with a feed-forward, highly parallelizable CNN to formulate a conditional probability density function. In this

design, the proposed network formulated statistical models of multi-modular distributions by approximating a Gaussian-Mixture statistical mapping function using pixel-wise vectors of intensity values that vary over time. Accordingly, a combination of weighted Gaussian components is presented, and each component captures and highlights a distinct contextual dynamics of pixel-wise values. Using Gaussian Mixtures at the pixel level, we aim to estimate the most frequently presented intensity of each pixel to model the background scenes. We adopt this network as a pre-processing module for our architecture of foreground detection.

Second, to make our network learnable, we mimic the underlying generator of the data of statistical learning. We design a loss function which follows unsupervised learning approach. This function is used to regulate the proposed density network in approximating the mathematical concepts behind GMM-driven modeling of the data using expectation maximization. In our network, all of the outputs aim to present of mixes of Gaussian components which represent the observed data, and the most probable background description of real data. The proposed background modeling architecture not only achieves higher degrees of interpretability when compared to the prior concept of estimating an implicit hidden function, but it also gains greater capability of adaptation when placed in a complex dynamic environment with statistical learning.

Thirdly, we develop a convolutional autoencoder for context-driven foreground extraction in the latter pipeline of the proposed framework, which replicates the context-driven mapping of differences between input frames and associated background scenes. This approach is strongly encouraged because real-life scenarios involve different degrees of contextual variations. We inherit the consistence that was formed by GMM-controlled background models to ensure a semantine understanding of the scene. Utilizing the information from features that are found in images from the first module of background modeling, we are able to formulate foreground extraction from raw inputs with a extremely lightweighted structure while ensuring a high accuracy. Experimental analysis showed that although our proposed approach is dominated by supervised learning methods, we present the sufficient degree of generalization of our model with unseen scenarios without performing model fine-tuning. In addition to a high accuracy, our method demonstrated with a high speed execution via exploiting the parallelism of advanced computing approaches on GPU-accelerated platform.

Finally, we reviewed the latest distributed machine learning techniques for data-driven models with the aim of reducing communication. Then, we perform a straightforward analysis of an event-triggered communication scheme to parallelize our proposed model of background subtraction. In this investigation,

we focus on the affect of communication avoidance when we vary the threshold of parameters' difference. Regarding the implementation, we extend the vanilla model with GPU-accelerated computation via segregating the computation and the communication task on different processing units to deal with CDnet, a large scale video dataset. The framework was implemented with asynchronous communication operators in UPC++. We experimented to train our model on various number of processes with different batch sizes. Observing the significant reduction in the number of exchanged message, we perceive that the scheme enhances the parallelism in our model training, so the procedure of training becomes faster with multiple scene-specific learning. Further, adjusting the threshold of parameters' difference, we are able to control the number of exchanged messages between processing elements.

6.2 Concluding Remarks

Our research has proposed a novel, two-stage framework with a GMM-based CNN for background modeling, and a convolutional auto-encoder MEDAL-net to simulate input-background subtraction for foreground detection, thus being considered as a search space limitation approach to compress a model of DNNs, while keeping its high accuracy. Our first and second contributions in this research include a pixel-wise, light-weighted, feed-forward CNN representing a multi-modular conditional probability density function of the temporal history of data, and a corresponding loss function for the CNN to learn from virtually inexhaustible datasets for approximating the mixture of Gaussian density function. In such a way, the proposed CDN-GM not only gains better capability of adaptation in contextual dynamics with humanly interpretable statistical learning for extension, but it is also designed in the tensor form to exploit technologically parallelizing modern hardware. Secondly, we showed that incorporating such statistical features into MEDAL-net's motion-region extraction phase promises more efficient use of powerful hardware, with prominent speed performance and high accuracy, along a decent generalization ability using a small-scale set of training labels, in a deep non-linear scheme of only a few thousands of latent parameters.

An investigation of communication-avoidance is carried out on our proposed model to examine the effectiveness of an event-triggered framework in controlling the number of exchanged message within a group of distributed processes. The experimental analysis showed that using an appropriate threshold of parameters' difference, we can control the cost of communication without degrading the accuracy of learning models. In this context, estimating an adaptive threshold to automate the control of communication cost within distributed training of deep learning models is a potential research in a near future.

Bibliography

[1] R. Szeliski, "Computer vision - algorithms and applications," in *Texts in Computer Science*, 2011.

[2] S. Zhang, H. Zhou, D. Xu, M. E. Celebi, and T. Bouwmans, "Introduction to the Special Issue on Multimodal Machine Learning for Human Behavior Analysis," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 16, no. 1s, pp. 1–2, apr 2020.

[3] S. Ammar, T. Bouwmans, N. Zaghden, and M. Neji, "Deep detector classifier (DeepDC) for moving objects segmentation and classification in video surveillance," *IET Image Processing*, vol. 14, no. 8, pp. 1490–1501, jun 2020.

[4] S. Park, M. Ji, and J. Chun, "2D human pose estimation based on object detection using RGB-D information," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 2, pp. 800–816, feb 2018.

[5] Y. C. Bilge, F. Kaya, N. I. Cinbis, U. Celikcan, and H. Sever, "Anomaly detection using improved background subtraction," in *2017 25th Signal Processing and Communications Applications Conference (SIU)*. IEEE, may 2017, pp. 1–4.

[6] T. Bouwmans, "Traditional and recent approaches in background modeling for foreground detection: An overview," *Computer Science Review*, vol. 11-12, pp. 31–66, may 2014.

[7] 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, feb 2020.

[8] S. V.-U. Ha, N. M. Chung, H. N. Phan, and C. T. Nguyen, "TensorMoG: A Tensor-Driven Gaussian Mixture Model with Dynamic Scene Adaptation for Background Modelling," *Sensors*, vol. 20, no. 23, p. 6973, dec 2020.

[9] Y. LeCun, S. Chopra, R. Hadsell, A. Ranzato, and F. Huang, "A tutorial on

energy-based learning," 2006.

[10] M. A. Zinkevich, M. Weimer, A. Smola, and L. Li, "Parallelized stochastic gradient descent," in *NIPS*, 2010.

[11] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. IEEE Comput. Soc, pp. 246–252.

[12] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* IEEE, 2004, pp. 28–31 Vol.2.

[13] I. Martins, P. Carvalho, L. Corte-Real, and J. L. Alba-Castro, "BMOG: boosted Gaussian Mixture Model with controlled complexity for background subtraction," *Pattern Analysis and Applications*, vol. 21, no. 3, pp. 641–654, aug 2018.

[14] R. Kalsotra and S. Arora, "A Comprehensive Survey of Video Datasets for Background Subtraction," *IEEE Access*, vol. 7, pp. 59 143–59 171, 2019.

[15] 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, sep 2019.

[16] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[17] C. Bishop, "Mixture density networks," Tech. Rep. NCRG/94/004, January 1994.

[18] T. Bouwmans, "Traditional Approaches in Background Modeling for Static Cameras," in *Background Modeling and Foreground Detection for Video Surveillance*. Chapman and Hall/CRC, aug 2014, pp. 1–1–1–54.

[19] J. D. Pulgarin-Giraldo, A. Alvarez-Meza, D. Insuasti-Ceballos, T. Bouwmans, and G. Castellanos-Dominguez, "GMM Background Modeling Using Divergence-Based Weight Updating," 2017, pp. 282–290.

[20] S. Viet-Uyen Ha, D. Nguyen-Ngoc Tran, T. P. Nguyen, and S. Vu-Truong Dao, "High variation removal for background subtraction in traffic surveillance systems," *IET Computer Vision*, vol. 12, no. 8, pp. 1163–1170, dec

2018.

[21] X. Lu, C. Xu, L. Wang, and L. Teng, "Improved background subtraction method for detecting moving objects based on GMM," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 13, no. 11, pp. 1540–1550, nov 2018.

[22] D. Sowmiya and P. Anandhakumar, "Cauchy Mixture Model-based Foreground Object Detection with New Dynamic Learning Rate Using Spatial and Statistical information for Video Surveillance Applications," *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, vol. 90, no. 5, pp. 911–924, dec 2020.

[23] B. N. Subudhi, S. Ghosh, S.-B. Cho, and A. Ghosh, "Integration of fuzzy Markov random field and local information for separation of moving objects and shadows," *Information Sciences*, vol. 331, pp. 15–31, feb 2016.

[24] 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*, vol. 25, no. 3, pp. 584–593, jun 2017.

[25] T. Yu, J. Yang, and W. Lu, "Dynamic Background Subtraction Using Histograms Based on Fuzzy C-Means Clustering and Fuzzy Nearness Degree," *IEEE Access*, vol. 7, pp. 14 671–14 679, 2019.

[26] B. N. Subudhi, T. Veerakumar, S. Esakkirajan, and A. Ghosh, "Kernelized Fuzzy Modal Variation for Local Change Detection From Video Scenes," *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 912–920, apr 2020.

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[28] M. Braham and M. Van Droogenbroeck, "Deep background subtraction with scene-specific convolutional neural networks," in *2016 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, may 2016, pp. 1–4.

[29] Y. Wang, Z. Luo, and P.-M. Jodoin, "Interactive deep learning method for segmenting moving objects," *Pattern Recognition Letters*, vol. 96, pp. 66–75, sep 2017.

[30] Z. Qu, S. Yu, and M. Fu, "Motion background modeling based on context-encoder," in *2016 Third International Conference on Artificial Intelligence*

and Pattern Recognition (AIPR), 2016, pp. 1–5.

[31] K. Lim, W.-D. Jang, and C.-S. Kim, "Background subtraction using encoder-decoder structured convolutional neural network," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, aug 2017, pp. 1–6.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–14, 2015.

[33] L. Xu, Y. Li, Y. Wang, and E. Chen, "Temporally adaptive restricted boltzmann machine for background modeling," in *AAAI*, 2015.

[34] Y. Tao, P. Palasek, Z. Ling, and I. Patras, "Background modelling based on generative unet," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6.

[35] D. Liang, J. Pan, H. Sun, and H. Zhou, "Spatio-Temporal Attention Model for Foreground Detection in Cross-Scene Surveillance Videos," *Sensors*, vol. 19, no. 23, p. 5142, nov 2019.

[36] M. Babaee, D. T. Dinh, and G. Rigoll, "A deep convolutional neural network for video sequence background subtraction," *Pattern Recognition*, vol. 76, pp. 635–649, apr 2018.

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

[38] R. Wang, F. Bunyak, G. Seetharaman, and K. Palaniappan, "Static and Moving Object Detection Using Flux Tensor with Split Gaussian Models," in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, jun 2014, pp. 420–424.

[39] T. P. Nguyen, C. C. Pham, S. V.-U. Ha, and J. W. Jeon, "Change Detection by Training a Triplet Network for Motion Feature Extraction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 433–446, feb 2019.

[40] L. A. Lim and H. Yalim Keles, "Foreground segmentation using convolutional neural networks for multiscale feature encoding," *Pattern Recognition Letters*, vol. 112, pp. 256–262, sep 2018.

[41] 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, aug 2020.

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

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

[44] M. Sultana, A. Mahmood, S. Javed, and S. K. Jung, "Unsupervised deep context prediction for background estimation and foreground segmentation," *Machine Vision and Applications*, vol. 30, no. 3, pp. 375–395, apr 2019.

[45] L. Maddalena and A. Petrosino, "Self-organizing background subtraction using color and depth data," *Multimedia Tools and Applications*, vol. 78, no. 9, pp. 11 927–11 948, may 2019.

[46] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization Methods for Large-Scale Machine Learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, jan 2018.

[47] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," *Advances in Neural Information Processing Systems*, vol. 2, no. October 2014, pp. 1223–1231, 2012.

[48] R. Mayer, C. Mayer, and L. Laich, "The tensorflow partitioning and scheduling problem," in *Proceedings of the 1st Workshop on Distributed Infrastructures for Deep Learning*. New York, NY, USA: ACM, dec 2017, pp. 1–6.

[49] A. Mirhoseini, H. Pham, Q. Le, M. Norouzi, S. Bengio, B. Steiner, Y. Zhou, N. Kumar, R. Larsen, and J. Dean, "Device placement optimization with reinforcement learning," 2017.

[50] L. G. Valiant, "A bridging model for parallel computation," *Commun. ACM*, vol. 33, no. 8, p. 103–111, Aug. 1990.

[51] Q. Ho, J. Cipar, H. Cui, J. K. Kim, S. Lee, P. B. Gibbons, G. A. Gibson, G. R. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," in *Proceedings of the 26th Interna-*

tional Conference on Neural Information Processing Systems - Volume 1, ser. NIPS'13. Red Hook, NY, USA: Curran Associates Inc., 2013, p. 1223–1231.

[52] M. Li, D. G. Andersen, A. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 19–27.

[53] B. C. Ooi, K.-L. Tan, S. Wang, W. Wang, Q. Cai, G. Chen, J. Gao, Z. Luo, A. K. Tung, Y. Wang, Z. Xie, M. Zhang, and K. Zheng, "Singa: A distributed deep learning platform," in *Proceedings of the 23rd ACM International Conference on Multimedia*, ser. MM '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 685–688.

[54] Z. Tang, S. Shi, X. Chu, W. Wang, and B. Li, "Communication-efficient distributed deep learning: A comprehensive survey," *CoRR*, vol. abs/2003.06307, 2020.

[55] R. Krizanc and A. Saarimaki, "Bulk synchronous parallel: practical experience with a model for parallel computing," in *Proceedings of the 1996 Conference on Parallel Architectures and Compilation Technique*, 1996, pp. 208–217.

[56] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep Gradient Compression: Reducing the communication bandwidth for distributed training," in *The International Conference on Learning Representations*, 2018.

[57] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms a case study for decentralized parallel stochastic gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 5336–5346.

[58] J. Chen, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous sgd," in *International Conference on Learning Representations Workshop Track*, 2016.

[59] C. Chen, W. Wang, and B. Li, "Round-robin synchronization: Mitigating communication bottlenecks in parameter servers," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 532–540.

[60] D. Grishchenko, F. Iutzeler, J. Malick, and M.-R. Amini, "Distributed learning with sparse communications by identification," 2020.

[61] S. U. Stich, "Local SGD converges fast and communicates little," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[62] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pp. 5693–5700, 2019.

[63] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 7184–7193. [Online]. Available: http://proceedings.mlr.press/v97/yu19d.html

[64] P. Jiang and G. Agrawal, "A linear speedup analysis of distributed deep learning with sparse and quantized communication," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 2530–2541.

[65] S. Shi, Q. Wang, X. Chu, and B. Li, "A dag model of synchronous stochastic gradient descent in distributed deep learning," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, 2018, pp. 425–432.

[66] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia, "Pipedream: Generalized pipeline parallelism for dnn training," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, ser. SOSP '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–15.

[67] S. H. Hashemi, S. A. Jyothi, and R. H. Campbell, "Tictac: Accelerating distributed deep learning with communication scheduling," in *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*, A. Talwalkar, V. Smith, and M. Zaharia, Eds. mlsys.org, 2019.

[68] A. A. Awan, K. Hamidouche, J. M. Hashmi, and D. K. Panda, "S-caffe: Co-designing mpi runtimes and caffe for scalable deep learning on modern gpu clusters," in *Proceedings of the 22nd ACM SIGPLAN Symposium on*

Principles and Practice of Parallel Programming, ser. PPoPP '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 193–205.

[69] H. Zhang, Z. Zheng, S. Xu, W. Dai, Q. Ho, X. Liang, Z. Hu, J. Wei, P. Xie, and E. P. Xing, "Poseidon: An efficient communication architecture for distributed deep learning on gpu clusters," in *Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference*, ser. USENIX ATC '17. USA: USENIX Association, 2017, p. 181–193.

[70] A. Dutta, E. H. Bergou, A. M. Abdelmoniem, C. Ho, A. N. Sahu, M. Canini, and P. Kalnis, "On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 3817–3824.

[71] A. Farnoosh, B. Rezaei, and S. Ostadabbas, "Deeppbm: deep probabilistic background model estimation from video sequences," *The Third International Workshop on Deep Learning for Pattern Recognition (DLPR20), in conjunction with the 25th International Conference on Pattern Recognition (ICPR 2020)*, 2020.

[72] J. Gracewell and M. John, "Dynamic background modeling using deep learning autoencoder network," *Multimedia Tools and Applications*, vol. 79, pp. 4639–4659, 2019.

[73] T. Georgiou, Y. Liu, W. Chen, and M. Lew, "A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision," *International Journal of Multimedia Information Retrieval*, vol. 9, no. 3, pp. 135–170, sep 2020.

[74] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 3123–3131.

[75] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, jan 1989.

[76] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand,

M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," apr 2017.

[77] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Improved Texture Networks: Maximizing Quality and Diversity in Feed-Forward Stylization and Texture Synthesis," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017, pp. 4105–4113.

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

[79] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[80] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a next-generation open source framework for deep learning," in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

[81] T. Akiba, K. Fukuda, and S. Suzuki, "ChainerMN: Scalable Distributed Deep Learning Framework," in *Proceedings of Workshop on ML Systems in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017.

[82] S. Tokui, R. Okuta, T. Akiba, Y. Niitani, T. Ogawa, S. Saito, S. Suzuki, K. Uenishi, B. Vogel, and H. Yamazaki Vincent, "Chainer: A deep learning framework for accelerating the research cycle," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 2002–2011.

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

[84] S. Isik, K. Özkan, S. Günal, and Ömer Nezih Gerek, "SWCD: a sliding window and self-regulated learning-based background updating method for change detection in videos," *Journal of Electronic Imaging*, vol. 27,

no. 2, pp. 1 – 11, 2018.

[85] L. A. Lim and H. Yalim Keles, "Foreground segmentation using convolutional neural networks for multiscale feature encoding," *Pattern Recognition Letters*, vol. 112, pp. 256–262, sep 2018.

[86] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427 – 437, 2009.

[87] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*. IEEE, 1999, pp. 255–261 vol.1.

[88] S. Jana, O. R. Hernandez, S. Poole, and B. M. Chapman, "Power consumption due to data movement in distributed programming models," in *Euro-Par 2014 Parallel Processing - 20th International Conference, Porto, Portugal, August 25-29, 2014. Proceedings*, ser. Lecture Notes in Computer Science, F. M. A. Silva, I. de Castro Dutra, and V. S. Costa, Eds., vol. 8632. Springer, 2014, pp. 366–378.

[89] Y. Zhang, J. C. Duchi, and M. J. Wainwright, "Communication-efficient algorithms for statistical optimization," *J. Mach. Learn. Res.*, vol. 14, no. 1, p. 3321–3363, Jan. 2013.

[90] S. Ghosh and V. Gupta, "EventGraD: Event-Triggered Communication in Parallel Stochastic Gradient Descent," in *2020 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC) and Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (AI4S)*. IEEE, nov 2020, pp. 1–8.

[91] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[92] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[93] J. Bachan, S. B. Baden, S. Hofmeyr, M. Jacquelin, A. Kamil, D. Bonachea, P. H. Hargrove, and H. Ahmed, "Upc++: A high-performance communication framework for asynchronous computation," in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2019, pp. 963–973.

[94] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 8024–8035.

[95] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, and S. Chintala, "Pytorch distributed: Experiences on accelerating data parallel training," *Proc. VLDB Endow.*, vol. 13, no. 12, p. 3005–3018, Aug. 2020.