Cooperative inference and learning for internet-of-things with limited resources

Wang, Yuan

2019

Wang, Y. (2019). Cooperative inference and learning for internet-of-things with limited resources. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/103663>

<https://doi.org/10.32657/10356/103663>

COOPERATIVE INFERENCE AND LEARNING FOR INTERNET-OF-THINGS WITH LIMITED RESOURCES

WANG YUAN

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2018

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

23-May-2019

.....

Date

.....

Wang Yuan

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

23 May 2019

.....
Date

.....
Tay Wee Peng

Authorship Attribution

Statement

This thesis contains material from 3 paper(s) published in the following peer-reviewed journal(s)/papers accepted at conferences where I was the first author.

Chapter 3 is based on the manuscript to be submitted to a journal which is an extension of a conference paper published as Y. Wang, W. P. Tay, and W. Hu, “An event-based diffusion LMS strategy,” *Proc. IEEE Sensor Array and Multichannel Signal Process. Workshop*, Sheffield, United Kingdom, Jul. 2018.

The contributions of the co-authors are as follows:

- Dr. Hu Wuhua suggested the research direction.
- I prepared the manuscript drafts. The manuscript was revised by Prof. Tay Wee Peng and Dr. Hu Wuhua.
- Prof. Tay Wee Peng and Dr. Hu Wuhua discussed the problem formulation with me, and they also assisted in proofs for theorems.

Chapter 4 is published as Y. Wang, W. P. Tay, and W. Hu, “A multitask diffusion strategy with optimized inter-cluster cooperation,” *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 3, pp. 504–517, April 2017, which is an extension of the conference paper Y. Wang, W. P. Tay, and W. Hu, “Multitask diffusion LMS with optimized inter-cluster cooperation,” *Proc. IEEE Statistical Signal Process. Workshop*, Palma de Mallorca, Spain, Jun. 2016.

The contributions of the co-authors are as follows:

- Dr. Hu Wuhua suggested the research direction.
- Prof. Tay Wee Peng discussed the problem formulation with me.

- Prof. Tay Wee Peng also assisted in the proofs for theorems in this work.
- I prepared the manuscript drafts. The manuscript was revised by Prof. Tay Wee Peng.
- Dr. Hu Wuhua also assisted in the revision of the manuscript.

23-May-2019

.....

Date

.....

Wang Yuan

Acknowledgments

After a six-year expedition of learning and research, eventually I put an end to my PhD study. This journey had never been easy and smooth, it is full of thorns, unexpected difficulties, and also a lot of priceless experiences as well as unforgettable memories. At the very beginning, I was too naive to know what exactly the word “PhD” means and what real research is about. Having learned many lessons and gone through trial and errors, I finally become a person who is in full awe of the scientific research. However, I definitely feel this procedure worthwhile without any regret. Hereby, I want to thank those people who had ever helped, guided, and also encouraged me to stick my PhD study out.

First of all, I would like to express my sincere thanks to my supervisor Prof. Tay Wee Peng. He guided me to navigate away from mist until I gradually reached the destination in the world of academic research. His knowledge, experience, instructions and patient helps have been the most important forces that molded what I am now. Our past interactions and communications cultivated my rigorous thinking and humble attitude for not only being a researcher, but also being a good person.

Besides, I want to thank Prof Hu Guoqiang who used to be my co-supervisor, and thank my thesis advisory committee Prof Tony Quek who is with the Singapore University of Technology and Design. Their nice instructions helped me to fulfill my PhD study. Especially, Prof Hu was the very first man appeared in my PhD career, thanks so much for introducing me to my supervisor Prof Tay.

I would also like to express my appreciation to Dr. Hu Wuhua who had offered me a lot irreplaceable help during the last three years of my PhD study. Thanks also go to my lab mates, colleagues, seniors and juniors of our research group. Especially, thanks to Dr. Zhao Lifan, Dr. Yang Lei, Dr. Tang Xin, and Dr. Huang

Tzuyi. Our discussions are not only nutritious for my research, but also greatly extended my horizon of knowledge. I really learned a lot from their expertise as well as unique perspectives.

Great friendships built during my time spent in Nanyang Technological University also brought me a lot happiness, lovely experience, and strong encouragement. Thanks to Hu Xiaolei, Tang Jianhua, Jin Chi, Du Pengfei, Wang Chengcheng, Chen Chen, Weng Renliang, Wang Benfei, Shen Shanlan, Hu Ting, Chen Yi. Special thanks to Zhong Nan, Liu Yuan, Leng Feng, and Chen Mengyue. I am also very grateful to my best friends Zhou Gefei and Liu Zhe, their own overseas study experience as well as our brotherhood friendship inspired and encouraged me a lot of times, without which I could not have made it through the hard times.

I dedicate this dissertation to my parents who are always my supporter no matter how difficult my life is. It is years of education received from them that cultivated my interests for exploring the unknown world and humble attitude towards the PhD study. Even when I had lost my hope and was about to give up, the love and power from them made me regain the confidence and re-engage into the fighting for my original dream. I will be forever grateful for their endless love.

Contents

Statement of Originality	i
Supervisor Declaration Statement	iii
Authorship Attribution Statement	v
Acknowledgment	vii
Table of Contents	ix
Abstract	xiii
List of Figures	xv
List of Tables	xvii
1 Introduction	1
1.1 The Emerging Internet-of-Things	1
1.2 Inference and Learning for IoT	2
1.2.1 Adaptive Filtering with LMS	2
1.2.2 Face Recognition with CNN	4
1.3 The Need of Local Intelligence and Cooperation	4
1.4 Motivations	7
1.5 Contributions and Outline of the Thesis	8
1.6 Notations	10
2 Background and Related Works	13
2.1 Diffusion LMS Strategies for Single-task Networks	13
2.1.1 Fundamentals of Single-task Networks and Distributed LMS	13

2.1.2	Prior Works in Energy Efficient Strategies for Single-task Networks	17
2.2	Diffusion Strategies for Multitask Networks	20
2.2.1	Fundamentals of Clustered Multitask Networks	20
2.2.2	Review of Prior Works	22
2.3	CNN-based Face Recognition Methods	24
3	Distributed LMS Estimation with Event-based Diffusion Cooperation	27
3.1	Introduction	27
3.2	Data Models and Preliminaries	28
3.2.1	Network and Data Model	28
3.2.2	ATC Diffusion Strategy for Distributed LMS Estimation . .	29
3.2.3	Proposed EB-ATC Diffusion Strategy	31
3.3	Performance Analysis	35
3.3.1	Network Error Recursions	35
3.3.2	Mean Error Analysis	37
3.3.3	Mean-square Error Analysis	38
3.4	Simulation Results	42
3.4.1	Simulation Setup	42
3.4.2	Stationary Environment	44
3.4.3	Non-stationary Environment	46
3.5	Summary	49
3.6	Derivations and Proofs	51
3.6.1	Derivations of (3.58)-(3.61)	51
3.6.2	Proof of the Theorem 1	54
3.6.3	Proof of the Theorem 2	56
4	Diffusion Strategy for Multitask Networks with Optimized Inter-Cluster Cooperation	59
4.1	Introduction	59
4.2	Models, Formulations, and Multitask Diffusion	61
4.2.1	Models and Assumptions	61

4.2.2	Multitask Diffusion Strategy with Normalized Inter-cluster Cooperation	64
4.2.3	The Multitask Adapt, Inter-cluster cooperate, and then Combine diffusion strategy	66
4.3	Convergence Analysis of the MAIC Strategy	68
4.3.1	Recursive Models of Network Errors	68
4.3.2	Mean Error Analysis	71
4.3.3	Mean-Square Error Analysis	72
4.4	Optimizing The Inter-cluster Cooperation	75
4.4.1	Formulations of Inter-cluster Weights Optimization	76
4.4.2	Adaptive Implementation of the Distributed Method	78
4.5	Simulation Results	80
4.5.1	Illustrative Examples	83
4.5.2	Benefits of Optimized Inter-cluster Cooperation	85
4.5.3	Performance Comparison under Different Means	86
4.6	Summary	90
4.7	Proofs	91
4.7.1	Proof of Theorem 3	91
4.7.2	Proof of Theorem 4	92
4.7.3	Proof of Lemma 1	93
4.7.4	Upper bounds of Steady-state Network MSD	94
5	Cooperative CNN-based Face Recognition using Low-cost Embedded IoT Devices	97
5.1	Introduction	97
5.2	Problem Statement and Overall Design	98
5.3	Design of the Pipeline Modules	101
5.3.1	Evaluation of Face Detection Methods	101
5.3.2	Selection of CNN Model	103
5.4	The Proposed CNN-based Cooperative Face Recognition	106
5.4.1	Training the CNN Model	106
5.4.2	Recognition Method	107
5.4.3	Cooperative Face Recognition by Multi-view Feature Fusion	109

5.5	Evaluation and Results	110
5.5.1	Experiment Setup	110
5.5.2	Recognition Accuracy	114
5.5.3	Pipeline Latency	116
5.6	Summary	117
6	Conclusions and Future Work	119
6.1	Conclusion	119
6.2	Future Work	121
A	Definitions, Lemmas and Proofs in References	123
A.1	Definition of BIBO Stability	123
A.2	Definition of the Block Maximum Norm and Related Lemmas . . .	124
A.3	Proof of the Mean Convergence of ATC Strategy	125
B	List of Acronyms	127
C	List of Publications	129
	Bibliography	131

Abstract

In the era of big data and Internet-of-Things (IoT), ubiquitous smart devices continuously sense the environment and generate large amount of data. So that inference and learning techniques play key roles in IoT, which can create knowledge and unveil important information from the sensory data. An IoT network with only centralized processor or cloud-based remote server may lead to heavy data traffic and long system latency. Such unfavorable consequences have been gradually becoming major concerns which hinder further applications of IoT technologies. By applying the concept of edge computing, computation and data storage resources are distributed closer to the end nodes in the IoT networks. These intelligent end nodes can be organized into distributed or decentralized local networks that support cooperative inference and learning, which is promising to reduce the data volume to be uploaded to the central processor, shorten the response time, and improve robustness and scalability. However, in typical IoT networks, resources like computing power, communication bandwidth and energy budget is rather limited. Therefore it is essential to explore approaches for the inference and learning that can make the most of local resources and reduce unnecessary consumption.

In this thesis, we study cooperative inference and learning schemes for the IoT networks with only limited resources. We address this topic by developing appropriate distributed least-mean-square (LMS) algorithms for inference and learning from linear models and cooperative deep learning-based face recognition pipeline for inference and learning from nonlinear models. To be more specific, this thesis covers the following aspects:

- Firstly, we consider a wireless sensor network (WSN) in the context of IoT where every nodes have a common learning objective, performing distributed LMS estimation cooperatively by means of exchanging local intermediate

estimates. To preserve the merits of local cooperation and save energy budget as well as communication resources at the same time, a diffusion strategy with event-based communication policy is proposed, in which nodes exchange information with their neighbors only when a criterion is satisfied. By doing so, the diffusion network can significantly reduce communication overhead while being able to achieve satisfactory steady-state mean-squared deviation (MSD) performance.

- Secondly, we consider a WSN consists of multiple clusters of nodes where only the nodes belong to the same cluster share the same learning objective. In this scenario, inappropriate cooperation among nodes belongs to different clusters may worsen learning accuracy and waste the limited resources. To avoid this issue, we propose a multitask diffusion strategy whose mean stability is independent of the inter-cluster cooperation, based on which we also develop optimization schemes in order to allow every nodes to optimize their inter-cluster cooperation with neighbors in different clusters. As such, the entire diffusion network are able to achieve better learning performance compared to the case where no inter-cluster cooperation is utilized among clusters.
- Finally, in addition to online learning from linear data model, cooperative learning from nonlinear model is also studied. To be specific, we designed a face recognition pipeline for IoT surveillance application by using convolutional neural network (CNN)-based deep learning techniques. The pipeline is implementable with low-cost embedded IoT devices, and is able to achieve short-enough image-to-result latency. We showed that local cooperation among these low-cost embedded devices can lead to enhanced learning performance. The results imply that deep learning tasks can be realized in local IoT network with limited resources instead of uploading raw data to a power-hungry central processor resides remotely or on the cloud by spending a lot communication bandwidth, which could also reduce the system latency at the same time.

List of Figures

1-1	Topology diagram shows typical IoT architectures for the edge/fog computing: distributed (left) and decentralized (right).	6
2-1	Illustrative example of a single-task diffusion network for adaptive inference and learning.	14
2-2	Example of a clustered multitask network.	20
3-1	An example of an adaptive network where each node is associated with a cost function to minimize.	28
3-2	Block diagram illustrates the algorithm flow of the proposed EB-ATC for node k	34
3-3	(a) The network topology. The numbers next to the nodes are node indices. (b) Data and noise profile of each node. From top to bottom: i) noise power ii) input signal power.	43
3-4	Simulation results for a stationary environment (a) the MSD performance comparison, (b) The ENTR comparison of EB-ATC with different triggering thresholds.	44
3-5	Performance comparison among other algorithms.	45
3-6	Non-stationary case where parameter changes at certain time instant (a) MSD performance comparison. (b) The resulting ENTR curves.	47
3-7	Cooperation benefit against the ACOS.	48
3-8	Non-stationary case where parameters vary continuously at every iteration.	49
4-1	An example of a clustered multitask network with three clusters depicted in red, blue, and green circles, respectively.	61

4-2	(a) Data and noise profile of each node. (b) The network topology. The numbers next to the nodes are node indices.	81
4-3	Distributions of the cluster parameters $\{\mathbf{w}_{C_p}^o\}_{p=1}^3$ under two different settings of the coefficient s_v	82
4-4	(a) Network MSD performance comparison amongst different strate- gies (i)-(vi). (b) Plots of simulated MSDs, theoretical MSDs, and approximate theoretical MSDs.	84
4-5	Network MSD comparison in the non-stationary case.	85
4-6	Network MSD of clusters (a) C_2 and (b) C_3 , when the regularization weight of MDLMS is $\eta = 5$	87
4-7	Network MSD (a) and comparison of steady-state network MSD gains with respect to ATC with no inter-cluster cooperation (b) when the regularization weight of MDLMS is $\eta = 5$	88
4-8	Network MSD performance when differences between parameters are small (a) and large (b).	89
5-1	Schematic overview of the proposed pipeline. There are three on-line testing stages: stage 1, 2 and 3, and one off-line stage.	99
5-2	Illustration of camera configuration used in cooperative face recog- nition: three cameras monitor same region from different angles and geometry positions.	111
5-3	Example of how testing image triplets are selected. From top to bottom: three identities $i = 1, 2, 3$, face left camera, frontal camera, and right camera, respectively.	113

List of Tables

3.1	Computational Complexity Comparison for Node k per Iteration . .	35
4.1	Definitions of Commonly Used Symbols.	69
5.1	Comparison of Processing Time of Different Face Detection Methods	103
5.2	Comparison Among CNN Models	104
5.3	Comparison Among Variants of MobileNets	105
5.4	Summary of Training Hyper-parameters	107
5.5	A Summary of Experiment Setups	114
5.6	The Test Results on Recognition Accuracy	115
5.7	Comparison with Cloud-based Solution	116
5.8	Test Results on Processing Latency	117

Chapter 1

Introduction

1.1 The Emerging Internet-of-Things

As technology keeps evolving and advancing, the cost of computing power, data and energy storage declines continuously, which makes embedded devices with on-board sensing, processing and communication abilities more and more ubiquitous. Through various of device-to-device networking solutions such as Bluetooth, Wi-Fi, ZigBee, NB-IoT, LoRa, GSM, and the emerging 5G technologies [1–3] smart devices can be interconnected seamlessly and organized into the paradigm of the Internet-of-Things (IoT), in which every nodes in the sensor networks can be connected to the Internet for information exchanging as they collect and process data from the physical world [4–7]. The number of connected smart devices in the IoT is expected to increase dramatically in the near future. According to some latest study, it is believed that there would be more than 50 billion devices connected to the IoT by the year of 2020 [7].

Through the platform of IoT, connected sensing, computing, and communication resources will be distributed around almost every corner of the world, which offers capabilities like location sensing, environment sensing, remote controlling, ad hoc networking and secure communication [4]. The IoT technologies can considerably improve the quality, boost the efficiency and even making revolutionary transformations in the fields like industry manufacturing, agriculture, logistics, transportation, power grids, environmental protection, smart home and Health care, etc [4, 8–14]. Meanwhile, vast number of IoT nodes generate huge amount of

data in rapid velocity. Therefore, it is essential to develop appropriate inference and learning techniques for IoT applications, so as to convert raw data into meaningful information or knowledge efficiently [7, 15], and realize the enhancement for individuals' daily life, companies' decision making, and industries' operations by utilizing the extracted insights [6].

1.2 Inference and Learning for IoT

In this thesis, we use the terminology *inference and learning for IoT* to refer to analyzing noisy real-time sensor data streams collected in the context of IoT by signal processing or machine learning techniques in order to assist the understanding of the data and making decisions or predictions. To be specific, we consider two typical categories of inference and learning problems and related solutions in the context of IoT.

1.2.1 Adaptive Filtering with LMS

The first category of inference and learning problem for IoT is real-time estimation of unknown parameters from a linear regression data model. Parameter estimation problems widely exist in various IoT applications. For example, the parameter of interest may represent the environment indicators such as temperature, humidity or pollutant density in smart city application, or represent the location of an object such as a piece of inventory or an AGV in logistics or manufacturing process, or represent the parameters of power system in smart grids, or the available channel information for cognitive radio which is also an important application in IoT context [16–20].

In this thesis, we will address real-time parameter estimation by adaptive filtering techniques. From machine learning perspective, adaptive filtering can also be referred to as the *online learning* which relies on stochastic gradient descent (SGD) method to handle real-time data processing task [16, 21]. Compared to the inference and learning techniques like the maximum likelihood estimation, the maximum a posterior estimation, Kalman filtering, or expectation-maximization, adaptive filtering algorithms do not require the *a priori* probabilities or statistics

of involved random variables, which makes them very convenient to apply [15, 21].

According to the specific cost function, there are two major families of adaptive filtering algorithms, namely the least-mean-squares (LMS) and recursive least-squares (RLS). The former one minimizes the mean-square error (MSE) cost

$$J(w) = \mathbb{E}(\mathbf{d} - \mathbf{u}^\top w)^2 \quad (1.1)$$

to find the optimal estimate of parameter $w \in \mathbb{R}^M$, where $\mathbf{d} \in \mathbb{R}$ and $\mathbf{u} \in \mathbb{R}^M$ are zero-mean jointly distributed random variables and related with each other via model

$$\mathbf{d} = \mathbf{u}^\top w + \mathbf{v} \quad (1.2)$$

where $\mathbf{v}$ denotes zero-mean noise variable. while the RLS algorithm minimize the least-squares cost

$$J(w) = \sum_{i=1}^n (d(i) - u(i)^\top w)^2, \quad (1.3)$$

or regularized version of (1.3), where $\{d(i), u(i)\}, i = 1, 2, \dots, n$ are n sets of observed realizations of random variable $\mathbf{d}$ and $\mathbf{u}$ that satisfy the model in (1.1).

Compared to the LMS, RLS converges faster, but RLS is less numerically stable and more computationally intensive. To be specific, the computational complexity of LMS is only $O(M)$ which could be much less than RLS whose complexity is of order $O(M^2)$ (M is the dimension of unknown parameter) [21]. In summary, LMS algorithm is able to adaptively learn and track the unknown parameter and is also very computational efficient, which needs little memory and storage space. These merits makes LMS an ideal choice for inference and learning in IoT applications especially for big data context [21–23].

1.2.2 Face Recognition with CNN

The adaptive filtering such as LMS algorithm can be regarded as a supervised online learning process where optimal parameter is estimated or learned from training data streams $\{d(i), u(i)\}$, here $\{u(i)\}$ are input data (or samples) and $\{d(i)\}$ are desired output response (or labels).

In addition to the online learning from data streams generated by linear models like (1.2), the second category of inference and learning problem for the IoT is classification using feature vectors extracted by nonlinear model such as the deep convolutional neural networks (CNN) [24, 25]. In particular, we consider the face recognition, one of the fundamental techniques for visual security and surveillance which is an important and trending application in the context of smart city, smart building or smart home [26–29]. Technically, face recognition is a special case of classification task that belongs to supervised learning generally, in which the algorithm and system need to associate a correct identity label (i.e., the desired output response) to the facial features extracted from video frames (the input data). The development of CNNs has remarkably improved the performance of computer vision tasks like classification, detection and recognition, since CNNs are able to learn robust and discriminative deep features [30–33], which makes CNNs the most suitable and the most widely utilized deep learning model for diverse computer vision problem [25, 34, 35] as well as one of the foundational technique for addressing vision-based big data analytics in the IoT context [36].

1.3 The Need of Local Intelligence and Cooperation

Considering the large number of IoT nodes, the rapid data generation velocity, and the places where data is generated are usually distributed over large geographical space, using a pure centralized architecture to perform inference and learning from real-time stream data may not be the suitable choice, and inevitably exposes to some critical limitations. Since there is only one central fusion center which is possibly on the cloud takes the role of data processing, it is difficult to meet the following requirements for the real-time IoT applications [37].

- *Short latency.* For examples, in the context of smart manufacturing, real-time adjustment and control is of a basic requirement, the latencies incurred in the loop from sensing data to decision making are desirable to be kept as short as milliseconds. For the formation coordination and fleet control of drones or automated guided vehicles, position estimation and driving commands have to be made in real time, it is preferred that the system can be operated with latencies less than tens of milliseconds [38].
- *Communication bandwidths and energy constraints.* Vast number of nodes can generate enormous amount of data, sending all these data to the central fusion center requires prohibitively large bandwidth and consume a lot of energy at the same time, especially in the smart visual surveillance applications, transmitting video streams to remote processor is not only impractical but also unnecessary. Besides, many typical IoT devices are small battery-powered embedded smart sensors nodes which cannot sustain to transmit high volume of data over long distance and keep low error rate the same time. Moreover, the limited availability of bandwidth can even worse the system latency.

To deal with the above challenges, it is advantageous to use edge computing to endow IoT end nodes or edge devices with local intelligence as well as resources like communication, data storage [7, 38–41], which can significantly reduce network latency, save bandwidth and energy budget. In addition, new nodes can also be integrated to the existing sensor network easily by local ad-hoc networking to increase the covering area or replace malfunctioned nodes, so that system robustness and scalability can also be improved. As such, central fusion center can be configured to perform higher-level delay-insensitive and more computation-intensive big data analysis tasks. For example, to extract useful insights from historical records collected periodically from IoT end nodes or edge devices, and then use this information to adjust the settings or optimize the behavior of IoT end nodes or edge nodes for a more effective utilization of limited resources.

On the other hand, the information collected by a single node is usually limited by its noise power and location, and hence may not be representative enough to achieve ideal inference and learning performance. So that, some nodes could per-

Figure 1-1: Topology diagram shows typical IoT architectures for the edge/fog computing: distributed (left) and decentralized (right).

form significantly worse than other nodes. Therefore, it is beneficial in general for IoT nodes to share information and collaborate with each other via device-to-device communication. As such, information and data available to each node is enriched, inference and learning can then be processed through local cooperation among end nodes or between end nodes and nearest edge devices, which could enhance the learning performance of individual node as well as the network performance compared to the case where each node only process its own data individually and does not share information with each other. When the nodes share a common objective (for example, the scenarios in Chapter 3 and Chapter 5), it is natural to expect positive performance gain to be obtained via cooperation. On the other hand, when the nodes in an IoT network exhibits heterogeneity (as will be discussed in Chapter 4), it could also be beneficial to cooperate among nodes associated with different learning task if their objective is correlated in some way.

In such cooperative schemes, the related local network architectures may be fully distributed, decentralized, or hybrid ones [37,39] which depends on the specific inference and learning applications. Examples of a distributed and a decentralized architecture are illustrated in the Fig. 1-1. In a distributed architecture, nodes cooperate among each other to solve inference and learning tasks by sharing local communication to peer neighbors. While in a decentralized architecture, there are additional nodes act as the IoT edge gateways (blue squares), end nodes cooperate with gateway node to handle learning tasks. However, they may also maintain communication with peer neighboring nodes whenever necessary (for example, the

edge gateways may be temporarily inaccessible due to networking issues or power limit). In this thesis, we shall focus on the distributed cooperative schemes since a distributed topology can recover to a decentralized one whenever only certain nodes need to maintain the inference ability to act as the IoT edge gateways.

1.4 Motivations

Although in edge/fog computing, IoT end nodes are supposed to equip with necessary computation, communication, and data storage to support cooperative inference and learning, the amount of these resources is still quite limited. It is therefore imperative that the algorithms, strategies, or techniques for cooperative inference and learning be appropriately designed so as to reduce the resource consumption, improve the learning performance by making the most of constrained resources. To be more specific, in this thesis, we are motivated to address the following problems of cooperative inference and learning for IoT with limited resources:

- (1) In distributed cooperative schemes, information exchange among nodes is beneficial for the network to gain considerable performance improvements. However, for sensor networks consist of low-cost embedded sensing platforms powered by batteries, too frequent communication over wireless channels could consume a lot bandwidth and deplete the energy budget very quickly. It is of practical meaning to develop energy-efficient distributed strategies for cooperative inference, aiming at achieve reasonable reduction on communication overhead and energy consumption without deteriorate the performance significantly.
- (2) Considering sensor networks consist of heterogeneous nodes whose learning objective may differ from each other. In this scenario, there may exists certain relationship among the different inference objectives, so that local cooperation can still be beneficial for distributed inference and learning. However, blind cooperation may result in a worse performance compared with the case where nodes independently solve their own inference and learning problems. This not only makes the network lose the advantages of cooperation, but also causes the waste of computing and communication resources. To avoid this

undesirable phenomenon, it is necessary to optimize the cooperation among nodes associated with different inference or learning objectives.

- (3) In IoT networks, it is also interested to apply complex nonlinear inference and learning such as face recognition for security and surveillance purpose, which can be realized by deep learning-based computer vision (CV) techniques. However, sending large amount of continuously generated video data directly to the remote central fusion center will lead to huge communication overhead and undesired long response loop and system latency. Moreover, running the entire face recognition pipeline in a single low cost resource-limited IoT end node may not offer satisfactory accuracy and robustness. Therefore, it calls for a suitable cooperative scheme for such CV application so that large communication overhead can be avoided, and learning performance of local IoT end nodes can be improved.

1.5 Contributions and Outline of the Thesis

In this section, we summarize our contributions towards fulfilling the motivations and objectives mentioned above.

- We consider LMS estimation problem for single-task diffusion networks (see Section 2.1.2 and related figure in the next chapter for detail). In order to reduce communication among neighboring nodes while preserve the advantages of typical diffusion strategies, we propose a diffusion LMS adaptation strategy that adopts an event-based adapt-then-combine scheme (EB-ATC). Different from the conventional adapt-then-combine (ATC) diffusion strategies where nodes keep exchanging local information at every iteration. In the EB-ATC, each node in the network examines the difference between the newly obtained update of local intermediate estimate and the most recent intermediate estimate transmitted to its neighbors. Transmission of the local update is triggered only if this difference is sufficiently large. The proposed EB-ATC strategy is simple to implement, has same order of computational complexity with ATC. Meanwhile, the proposed approach is able to achieve

substantial reduction on the communication rate and resultant energy consumption while being more accurate than other methods in the literature in terms of steady-state MSD when similar amount of energy saving is achieved.

- We also consider a distributed LMS problem for estimating the realizations of slowly-varying random parameters through a clustered multitask network (see Section 2.2 of Chapter 2 for descriptions and illustration therein). We propose a multitask diffusion strategy where an additional intermediate inter-cluster diffusion combination step is incorporated between the local adaptation and the intra-cluster diffusion combination steps which are the two standard computation for the ATC-type diffusion strategies. Unlike multitask diffusion strategies in the literatures that use separate regularization parameter and a normalized inter-cluster cooperation weights to promote and control the inter-cluster cooperation, the proposed multitask diffusion strategy merges these two quantities into one parameter. By doing so, the network MSD performance is affected by the inter-cluster cooperation weights exclusively, so that tuning the regularization parameter is not needed any longer. Based on this feature, we further propose optimization schemes, include a centralized formulation, a distributed formulation, and an online adaptive implementation, which enable every nodes adjust their inter-cluster cooperation for the improvement of overall network performance, so as to avoid performance deterioration caused by cooperating with inter-cluster neighbors in a naive fashion.
- In trending research works in the area of deep learning and related face recognition techniques, people are mostly interested in competing the sheer learning accuracy. From a practical deployment point of view, we instead consider how to realize a CNN-based face recognition pipeline which is implementable with limited resources like low-cost embedded IoT devices. To this end, we identified suitable face detection method, suitable datasets for training and testing purposes, and trained an efficient CNN model that is able to extract deep feature on the low-cost embedded platform in real time. Moreover, through the proposed cooperative scheme, the accuracy can be improved to a acceptable level with short latency. By the proposed pipeline, data traffic

for IoT face recognition application can also be drastically reduced.

Apart from the introduction presented in this chapter, the rest of this thesis is organized as follows: In Chapter 2, we first summarize the background and review related state-of-the-art works for each area we have worked on. In Chapter 3, the proposed EB-ATC strategy and related performance analysis is presented. Besides, we develop a novel multitask diffusion strategy, analyze its performance and discuss how to optimize inter-cluster cooperation in Chapter 4. Then in Chapter 5, we introduce the methodology adopted towards building an efficient testbed for realizing deep learning face recognition pipeline. Finally, in Chapter 6 we provide concluding remarks for this thesis, and briefly outlook some possible future directions to explore.

1.6 Notations

Throughout this thesis, $\mathbb{N}$, $\mathbb{R}$ stand for the set of natural numbers and the set of real numbers respectively. Moreover, all vectors are column vector. We use boldface characters for random variables, and plain characters for realizations of the corresponding random variables as well as deterministic quantities. Besides, we use upper-case characters for matrices and lower-case ones for vectors and scalars. For ease of comparison, we also adopt similar notations used in [16, 42].

The notation $\mathbf{1}_N$ represents an $N \times 1$ vector with all entries being one, and I_N is an $N \times N$ identity matrix. The vector $\mathbf{0}_M$ is an $M \times 1$ vector with all entries being zero. $A^\top$ is the transpose of the matrix A , $\text{Tr}(A)$ denotes the trace of A , if A is a symmetric square matrix, we use $\lambda_{\max}(A)$ to denote its largest eigenvalue and $\rho(A)$ its spectral radius. The operation $A \otimes B$ denotes the Kronecker product of the two matrices A and B . The notation $\|\cdot\|$ is the Euclidean norm, $\|\cdot\|_{b,\infty}$ denotes the block maximum norm [16], while $\|\cdot\|_\infty$ denotes the maximum absolute row sum of its argument, and we use $\|x\|_\Sigma^2 \triangleq x^\top \Sigma x$ for any column vector x and non-negative definite matrix Σ . We use $(x_i)_{i=1}^n$ to represent the sequence of vectors or scalars $x_1, \dots, x_n$, while $(A_i)_{i=1}^n$ represents the sequence of matrices $A_1, \dots, A_n$. We also use $[x_i]_{i=1}^n$ to denote the matrix $[x_1, \dots, x_n]$. We use $\text{diag}\{\cdot\}$ to denote a matrix whose main diagonal is given by its arguments, and $\text{col}\{\cdot\}$ to denote a

column vector formed by its arguments. The notation $\text{vec}(\cdot)$ represents a column vector consisting of the columns of its matrix argument stacked on top of each other. If $\sigma = \text{vec}(\Sigma)$, we let $\|\cdot\|_\sigma = \|\cdot\|_\Sigma$, and use either notations interchangeably. In addition, we use $x \geq y$ to represent entry-wise inequality between vectors x and y , and $|\cdot|$ is the number of the elements of its set argument. The notation $\langle x, y \rangle$ denotes the inner product of vector x and y . Finally, we write $\alpha = O(u)$ if $|\alpha| < c \cdot |u|$ for some positive constant c .

Chapter 2

Background and Related Works

In this thesis, we first study the distributed cooperative online learning methods for diffusion networks that perform LMS estimation, we consider the energy-efficient strategy for a single-task network and also investigate how to optimize the inter-cluster cooperation for a clustered multitask network. Then, we realize a CNN-based face recognition pipeline by distributed cooperative scheme, and evaluates the proposed pipeline on real-world embedded IoT devices. This chapter provides the background and review related state-of-the-art works.

2.1 Diffusion LMS Strategies for Single-task Networks

2.1.1 Fundamentals of Single-task Networks and Distributed LMS

In Chapter 1.2 we have briefly introduced the LMS algorithm and its basic formulation for a stand-alone case. It is more important for the IoT network that every nodes can collaborate with each other via a distributed topology. To be specific, we consider a multi-agent distributed learning network that consists of N nodes which are labeled as $k = 1, 2, \dots, N$ (see Fig. 2-1). The network is represented by a graph where the vertices denote the nodes, any two nodes are said to be connected so that they can exchange information with each other if there is an edge between them. The neighborhood of any particular node k is denoted by the notation $\mathcal{N}_k$

Figure 2-1: Illustrative example of a single-task diffusion network for adaptive inference and learning.

which consists of all the nodes that are connected to node k , including k itself. This thesis focuses on undirected network, which means if node k is a neighbor of node ℓ , then so is node ℓ to node k . In addition, throughout this thesis, we assume all the distributed networks are connected networks. That is to say, there always exist a path links them in both directions between any two different nodes [16, 43, 44].

At every time instant, every nodes in the network are able to collect streaming data $\{\mathbf{d}_k(i), \mathbf{u}_k(i)\}$, the random variable $\mathbf{d}_k(i) \in \mathbb{R}$ and random vector $\mathbf{u}_k(i) \in \mathbb{R}^M$ obeys the linear regression model

$$\mathbf{d}_k(i) = \mathbf{u}_k^T(i)w_k^\circ + \mathbf{v}_k(i), \text{ for all } i \geq 0, \text{ and } k = 1, 2, \dots, N \quad (2.1)$$

where the vector $w^\circ \in \mathbb{R}^M$ denotes the unknown parameter, and the scalar random variable $\mathbf{v}_k(i)$ is a zero-mean additive observation noise. The random process $\{\mathbf{d}_k(i), \mathbf{u}_k(i)\}$ are assumed to be jointly wide-sense stationary.

Assumption 1. *in order to facilitate the performance analysis of such network in Chapter 3 and Chapter 4, $\{\mathbf{d}_k(i), \mathbf{u}_k(i), \mathbf{v}_k(i)\}$ are also assumed to satisfy the following conditions.*

- (a) *The regression process $\{\mathbf{u}_k(i)\}$ is zero-mean, spatially independent and temporally white. The regressor $\mathbf{u}_k(i)$ has positive-definite symmetric covariance matrix $R_{u,k} = \mathbb{E} [\mathbf{u}_k(i)\mathbf{u}_k^T(i)]$.*

(b) The noise process $\{\mathbf{v}_k(i)\}$ is spatially independent and temporally white with variance $\sigma_{v,k}^2$.

(c) The regression and noise are independent of each other for all $\{k, \ell, i, j\}$.

In the above distributed learning networks, each node k seeks to estimate its parameter of interest w_k° by minimizing the following MSE cost

$$J_k(w) = \frac{1}{2} \mathbb{E} |\mathbf{d}_k(i) - \mathbf{u}_k(i)^\top w|^2, \quad (2.2)$$

which is strongly convex and twice differentiable. In a single task network, every nodes have a same learning objective, which is to estimate a common unknown parameter

$$w^\circ = w_k^\circ, \quad \text{for all } k = 1, 2, \dots, N. \quad (2.3)$$

Moreover, the nodes are not only interested in minimizing its own cost $J_k(w)$, but also cooperate with each other by means of information exchange to minimize the following aggregate cost

$$J^{\text{glob}} \triangleq \sum_{k=1}^N J_k(w). \quad (2.4)$$

The above network learning objective is interested to every nodes in the network because in the procedure of minimizing the collective learning objective (2.4), nodes can obtain more information shared by their neighbors, which is beneficial for improving the overall learning performance of the entire network measured in network MSD

$$\text{MSD}^{\text{network}} = \lim_{i \rightarrow \infty} \frac{1}{N} \sum_{k=1}^N \|\tilde{\mathbf{w}}_k(i)\|^2 \quad (2.5)$$

where $\mathbf{w}_k(i)$ is the estimate made by node k at time instant i and

$$\tilde{\mathbf{w}}_k(i) = w^\circ - \mathbf{w}_k(i) \quad (2.6)$$

represents the error vector of node k . By cooperating with local neighbors, the dis-

tributed learning network can obtain a MSD performance that is even comparable to the centralized scheme [44, 45] while being more robust and scalable and less energy-consuming, which is desirable for IoT application with limited resources.

Several useful distributed solutions have been proposed in the literature in order to fulfill inference and learning tasks over sensor networks, such as the incremental strategies [17, 46–48] and the consensus strategies [49–53].

In the incremental implementation, nodes equipped with local processing abilities continuously make local estimates in response to the streaming-data, and exchange the local intermediate estimate with their predefined neighbors. In this scheme, information is passed from one node to the successor one along a cyclic path until all agents are visited and the process is then repeated [44, 54]. However, this category of distributed strategies requires pre-determined paths for message passing, which is prone to node or communication link failure, and restrict participating nodes with only one precursor from which information is received and only one successor to which information is passed.

Unlike the incremental strategies, the consensus-type strategies allow information exchange among all local neighboring nodes. Each node sends its local information to neighboring nodes (or neighbors for short), collecting and combining information received from its neighbors at the same time, so as to update its local inference. The consensus schemes were originally proposed to enable nodes in a network to reach agreement on the value averaged across the network [51, 55, 56], which is very useful and commonly adopted to approximate the global average value by only local cooperation and processing. Some consensus-based inference algorithms like [57–60] adopt two time scales, one for data collection and inference making and the other for reaching consensus among all nodes. These two time-scale approaches hinder the adaptation ability to streaming data of the network for real-time applications, which is improved in [52, 61–64], by combining the adaptation and information exchange with neighbors into single time scale.

Compared with the previous two categories, diffusion strategies [16, 45, 65–73] are particularly advantageous for continuous online adaptation, especially the ATC diffusion strategies and its variants (in this thesis, we will use diffusion strategies to refer the ATC-like diffusion strategy since ATC diffusion strategy has superior per-

formance). To be specific, diffusion networks do not need to maintain a predefined information passing sequence in comparison with incremental strategies [44], which promotes the plug-and-play cooperation and network scalability. Besides, unlike single time-scale consensus-based counterparts where time-dependent or decaying step-sizes are usually adopted for reaching agreement among nodes, by using constant step-sizes, diffusion strategies enhance the adaptability to parameter values that may changing over time. Particularly, for adaptive inference and learning from streaming data under constant step-sizes, diffusion strategies can not only achieve a lower steady-state network MSD than consensus-type strategies [68], but also enable both mean and mean-square stability to be insensitive to the weights govern the local cooperation [45]. Whereas, the networks apply consensus-based strategies can lose stability even if every individual nodes are stable on their own [16, 45]. Thanks to these merits, diffusion strategies have attracted a lot of research interest in recent years for both single-task scenarios where nodes share a common parameter of interest [74–80], and multitask networks where parameters of interest differ among nodes or groups of nodes [42, 81–84].

2.1.2 Prior Works in Energy Efficient Strategies for Single-task Networks

As mentioned above, in a single task network, every node seeks the same learning objective (see Fig. 2-1). The nodes cooperate with each other by means of information exchange, for example the local data, local gradient information, or the local intermediate iterate. The information received by each node is associated with non-negative combination weights used to scale the degree of trust for its neighbors. It is shown in [16, 45, 67, 68] that the choices of these weights will not influence the convergence of diffusion strategies, however they can significantly affect the steady-state MSD performance of a single task diffusion network. How to choose these combination weights were studied earlier in [16, 45, 67, 68], and in [85] for a heterogeneous network settings.

Given the combination weights selected, the advantages of diffusion strategies are rooted in communicating local information to neighboring nodes at each iteration. However, in typical networks of IoT applications, devices or nodes usually

have limited resources like energy budget and communication bandwidth, and are expected to last as long as possible without intervention, which make frequent information exchange with neighboring nodes over wireless networks unfordable. Several methods towards improving energy efficiency for diffusion networks have been proposed in the literature, the reported approaches can be categorized into two major directions: to reduce the number of neighbors to cooperate with [86–88]; and to reduce the amount of the local information to be communicated [89–91]. These works either rely on additional optimization procedures, or need selection or projection matrices to determine the set of cooperation neighbors or to pick entries to be transmitted. Although these methods are able to reduce energy consumed by communication, they require more computation resources and information to implement. In [92], the author proposed a diffusion strategy where the network only switch on communication at certain time instants with a fixed period, and remains silent at the rest of time. However, the performance of this strategy fluctuates periodically from a diffusion networks to a networks consists of non-cooperative nodes. A similar approach was proposed in [93], however the data is reserved so that continuous adaptation is lost between any two consecutive active time instants.

In contrast to time-driven communication where nodes share local information with each other at every iteration, event-based communication mechanisms permit nodes to trigger transmission of information to neighbors only when certain meaningful events take place. Due to the occurrence of such events is intermittent and even sparse over time, therefore the event-based communication can significantly reduce energy consumption by avoiding unnecessary information exchange especially when the system has reached steady-state. Moreover, by allowing every node in the network to access the limited bandwidth resource alternately, channel efficiency can also be improved.

Such mechanisms have been developed for applications such as state estimation and Kalman filtering [94–99], control and consensus problems [100–106] where information needs to be transmitted over wireless networks with constrained resources, so that communication burden and energy consumption can be reduced. Comprehensive overview and surveys on how event-based communication strategies are applied in these areas can be found in [107, 108]. Moreover, the concept of

event-based communication is also considered in the IoT-related research, to cut off unnecessary data transmission between local device and the cloud as well as among local IoT devices [109, 110].

Even though event-based mechanism is promising on reducing communication overhead for IoT and WSN, it has not been fully investigated in the context of ATC-type diffusion adaptation. Unlike most of literatures mentioned above where communication of the newly obtained local information is turned on if it significantly deviates from the previously transmitted one, the earlier work [111] on distributed normalized LMS estimation adopt a set-membership test approach where each node will broadcast its new data or update its local intermediate estimates only when the data is sufficiently informative for solving the underlying estimation problem. Moreover, different from ATC-type diffusion strategies, nodes in [111] carry out an sequential update to utilize data or local estimate received across its neighborhood, rather than using weighted averaging as is in the ATC strategies. In [112, 113], is adopted for cooperative spectrum sensing problems, each node in a wireless network can decide if or not to update its local estimate with new observation by resorting to similar set-membership technique in [111], then a node will communicate its local statistics with neighbors for further decision making only when its local estimate is updated. Likewise, in the work [114], every node selectively update its local intermediate estimate, by evaluating if the newly observed data can help with decreasing the MSD, communication at a node is then triggered only if it has updated the local estimate.

In [115], the author proposes a diffusion strategy where every entry of the local intermediate estimates are quantized into values of multiple levels before being transmitted to neighbors. Communication of entries in a local estimate vector is triggered once quantized local information of these entries goes through a quantization level crossing. The performance of this method is affected largely by the precision of selected quantization scheme. However, quantizing every entry of the estimate vector at every iteration before the needs additional computation time. Besides, choosing a good quantization scheme with favorable precision, and requiring every node being aware of this same quantization scheme is of practical difficulty for online adaptive inference and learning where parameters of interest

Figure 2-2: Example of a clustered multitask network.

and sensing environment may vary over time.

2.2 Diffusion Strategies for Multitask Networks

Most of the above-mentioned works focus on the single-task networks where that all the nodes in the network share a single common parameter of interest. However, in practical multi-agent networks, different nodes or clusters of nodes may have different parameters of interest.

To be more specific, in some applications the unknown parameters that each node is estimating may have same dimension, however their entires can be totally different while being related with each other in some manner. For example, nodes may be grouped into different clusters, and each cluster of nodes is responsible for tracking a particular target. If these targets move in certain formation at the same time, there are relationships among the position of each target, so that different clusters of nodes may be benefit from the potential cooperation. This kind of scenario fits the model of clustered multitask networks.

2.2.1 Fundamentals of Clustered Multitask Networks

In the context of clustered multitask networks, nodes are grouped into P clusters, each belonging to a unique cluster (illustrated in Fig. 2-2). Each cluster C_p , $p \in$

$\{1, 2, \dots, P\}$, aims to estimate an unknown parameter vector $w_{C_p}^\circ \in \mathbb{R}^M$. For example in Fig. 2-2, there are three clusters C_1 , C_2 , and C_3 associated with different learning objective $w_{C_1}^\circ$, $w_{C_2}^\circ$, and $w_{C_3}^\circ$ respectively. Let $\mathcal{N}_{C,k}$ to denote the intra-cluster neighborhood consists of all the neighboring nodes that belong to the same cluster as node k , including node k itself. The inter-cluster neighborhood is denoted by $\mathcal{N}_{I,k}$ which consists of neighboring nodes of k that belong to clusters different from $\mathcal{C}(k)$. We have

$$\mathcal{N}_{C,k} = \mathcal{N}_k \cap \mathcal{C}(k), \quad (2.7)$$

$$\mathcal{N}_{I,k} = \mathcal{N}_k \setminus \mathcal{C}(k), \quad (2.8)$$

and

$$\mathcal{N}_{C,k} \cup \mathcal{N}_{I,k} = \mathcal{N}_k, \quad (2.9)$$

$$\mathcal{N}_{C,k} \cap \mathcal{N}_{I,k} = \emptyset. \quad (2.10)$$

Considering a node k belongs to a cluster C_p for some $p \in \{1, 2, \dots, P\}$, let $\mathcal{C}(k)$ to refer to this specific cluster, so that the parameter of interest of node k relates to that of cluster $\mathcal{C}(k)$ through

$$w_k^\circ = w_{\mathcal{C}(k)}^\circ = w_{C_p}^\circ. \quad (2.11)$$

To perform the LMS estimation of $w_{\mathcal{C}(k)}^\circ$, each node k is associated with the following MSE cost similar to (2.2),

$$J_k(w_{\mathcal{C}(k)}) = \mathbb{E} \left| \mathbf{d}_k(i) - \mathbf{u}_k(i)^\top w_{\mathcal{C}(k)} \right|^2, \quad (2.12)$$

where the data collected by each node $\{\mathbf{d}_k(i), \mathbf{u}_k(i)\}$ also satisfy linear regression model (2.1) and conditions in Assumption 1.

Since in clustered multitask networks, different clusters have different learning objectives, therefore a network-wide cost like (2.4) for the single-task networks is no longer suitable in this case. Instead, cluster C_p estimates $w_{C_p}^\circ$ by solving the

estimation problem defined in terms of Q Nash equilibrium problems [42, 82]

$$\min_{w_{C_p}} J(w_{C_p}, w_{-C_p}), \quad \text{for } p \in \mathcal{P}, \quad (2.13)$$

where the notation w_{-C_p} represents the collection of unknown parameters interested to clusters other than C_p , and the cost function $J(w_{C_p}, w_{-C_p})$ associated to cluster C_p takes the following form,

$$J(w_{C_p}, w_{-C_p}) = \sum_{k \in \mathcal{N}_{C,k}} \left(J_k(w_{\mathcal{C}(k)}) + \sum_{\ell \in \mathcal{N}_{I,k}} g_{\ell k} \|w_{\mathcal{C}(k)} - w_{\mathcal{C}(\ell)}\|^2 \right) \quad (2.14)$$

which is an MSE cost regularized by a ℓ_2 -norm used to promote the similarities as well as inter-cluster cooperation by the Euclidean distance between parameters estimated by two adjacent clusters.

2.2.2 Review of Prior Works

To address the estimation problems in the clustered multitask networks, cooperative strategies are studied in [42, 82, 116–119]. In [117], nodes are not aware of whether their neighbors perform the same task. To reduce the resulting bias on the estimates, an on-line combination weights adjustment approach by minimizing the transient MSD metric was proposed to cluster the nodes. It is observed that, by introducing cooperation among clusters with different tasks, nodes can achieve better performance than by operating independently of each other. This implies the possibility that the network performance may be further improved if the relationship between different unknown parameters is utilized properly. To this end, [42, 116] propose a multitask diffusion adaptation strategy that inherits the basic structure of the ATC strategy [16], and promotes cooperation among clusters by adding an additional ℓ_2 -norm regularization to the ATC objective function. In [82] the performance of multitask diffusion strategy for an asynchronous network model is analyzed where nodes are assumed to randomly lose the network connectivity or stop local adaptation due to malfunction, it is showed that the clusters can still improve their MSD performance against the ones obtained without inter-cluster cooperation in this kind of scenarios. Besides, a ℓ_1 -norm regularization is considered

in the work [118] to promote sparsity in the difference between cluster parameters, the results was further extended in [119] which shows that cooperation among the clusters will be beneficial if the number of similar entries is favorable, which fits the environments in the area like cognitive radio and remote sensing. In [120], the author models the relationship between different tasks by a set of linear equality constraints rather than regularization techniques. In this work, every node has its own parameter of interest different from each other, and the adjacency in the network topology between any two nodes is determined by if they involve in same equality constraints. So that the local diffusion cooperation for solving the global cost function take places only among nodes subject two same constraints.

Similar to the single-task diffusion networks, the MSD performance of multitask networks is additionally controlled by the regularization weight and the cooperation weights between clusters or nodes with different parameters. Taking Fig. 2-2 as an example, in the clustered multitask networks, two types of diffusion cooperation take places at the same time: 1) the intra-cluster cooperation (for example the blue arrows depicted among nodes of cluster C_1) as is in the typical single-task network whose combination weights is denoted by $\{a_{\ell k}\}$; 2) the inter-cluster cooperation, where some of the nodes exchange information with neighbors in the clusters different from itself (for examples, the purple arrows between cluster C_1 and C_2 , and green arrows between cluster C_1 and C_3), the related weights are denoted by $\{g_{\ell k}\}$. Although different multitask diffusion strategies have been developed in the above literatures, how to select these inter-cluster cooperation weights $\{g_{\ell k}\}$ was left as an open problem. A heuristic adaptive method for selecting the inter-cluster cooperation weights was proposed in [121]. Nevertheless, the problem of how to choose the regularization and cooperation weights in a rigorous optimal way have not been adequately addressed.

Apart from the clustered multitask networks, in some other applications, the parameters of interest at each node may partially overlaps with each other. For example, two nodes may have some entries of the unknown parameters in common, while the other entries and even total number of entries in the unknown parameter differs from each other. This category of learning problems is usually referred to as the *node-specific estimation*. The reference [17] considers an estima-

tion problem where each node estimates three categories of parameters: a global parameter which will be estimated by every nodes in the network, parameters of common interest to a subset of nodes, and parameters of local interest only. A similar nodes-specific distributed estimation problem is solved by a diffusion-based strategy in [81], and is studied from a coalition game theoretic perspective in [122]. In the work [123], the relationship between nodes-specific estimation tasks are assumed to be unknown, and each node infers which parameters their neighbors are estimating in order to adjust their range of cooperation. The application of node-specific distributed strategies can be found in the fields such as speech and video enhancement, active noise control, and cooperative spectrum sensing [124]. In the aforementioned works, each node only cooperates with other nodes that have common entries in the parameters of interest, this cooperation scheme could only enhance the learning of that common entries. Therefore in this thesis, we focus only on clustered multitask networks and related learning problem formulated in (2.13).

2.3 CNN-based Face Recognition Methods

As a widely adopted approach for person identification for video surveillance, face recognition is of great interest in the fields of CV in both research academy as well as industry. Since the publication of the pioneer work [24] in 2012, it has been witnessed that the accuracy of CV tasks can be significantly improved (about at least 10%) by make use of deep convolution neural networks (CNN, or deep learning for short, interchangeably). As a special application learning over CV, the performance of face recognition in unconstrained environments has also been boosted with the aid of deep learning techniques [125]. Trained on large-scale face datasets, some preeminent deep learning face recognition methods, such as DeepID series [126, 127], FaceNet [128], and VGG face descriptor (or VGG-Face) [31], are able to achieve face verification accuracies higher than 98% over public standard benchmark dataset, namely, the Labelled Faces in the Wild (LFW) [129].

The typical pipeline of face recognition consists of face detection, feature extraction (or face representation), and classification or feature matching. Among

these steps, the feature extraction is usually the most critical and also most computational costly one which largely determines the overall accuracy of the pipeline [125]. In the deep learning-based face recognition methods mentioned previously [31, 126–128, 130], the feature extraction is realized by passing raw input image through relevant CNN models. In the context of IoT with limited resources, we shall not only pay attention to the accuracy but also concern more about the model complexity and resources requirements for implementation.

In DeepID-1 [126], 25 pairs of RGB and grey patches are cropped from different positions of the input face image, and fed into several parallel 9-layer CNN model whose output is a feature vector of dimension 160. The final feature vector is then obtained by concatenating the output of every CNN model which has a dimension of $25 \times 2 \times 160 = 8000$. DeepID-2 [130] improves against the DeepID-1 by trained on combination of both verification and classification loss. DeepID-3 [127] further improves over DeepID-1,2 by using a much deeper CNN model with up to 15 convolutional layers and a much bigger training dataset. Although the performance of DeepID-2,3 is very powerful (99.15% and 99.53% on LFW for verification, respectively), their parallel architecture of multiple CNN models results in a huge size of the entire system model with total number of parameters as big as 100 million [125].

VGG-Face [31] is developed based on the popular VGG-16 deep CNN model which is the runner-up of ILSVRC 2014 classification challenges [131]. The model achieves a comparable face recognition performance on LFW with a verification accuracy of 98.95%, using much less training image (2.6 million images) than the other state-of-the-art counterparts. However, the core CNN model of VGG-Face, namely the VGG-16, is also very bulky. The basic spirit of VGG-16 is to increase the sheer depth of traditional CNN, it stacks 13 layers of convolutional filters which are followed by three fully-connected layers, so that a very deep CNN model is obtained. As a result, the VGG-16 ends up with 140 million parameters in total, the number of multiply and accumulate (MAC) operations in a single inference run can be as high as 1.5×10^{10} [132]. To make things worse, the memory size required to load the weights of a pre-trained VGG-16 model is approximately 500MB-large, which is formidable for low cost embedded IoT platforms (like Raspberry Pi series)

whose total on-board memory is usually less than 1GB.

Another preeminent deep learning face recognition model is the FaceNet [128]. It can be regarded as the face-recognition version of the GoogLeNet [133] trained instead on face images dataset (200 million images). Unlike the DeepID series and VGG-Face, which are designed under the focus of sheer accuracy, the GoogLeNet and FaceNet is built to improve computational efficiency, which leads to a very deep CNN model (22 layers) realized with much less parameters. The total number of parameters in the FaceNet is only 7.5 million which is about 18 times less than VGG-Face, and the number of MAC operations is about 1.6×10^9 which is roughly the one-tenth of VGG-Face. At the same time, the recognition (face verification) accuracy of FaceNet is almost the same with VGG-Face (98.87% on LFW).

Nevertheless, the above-mentioned face recognition models rely on either desktop-PC class server with dedicated graphic processing units (GPUs) to accelerate the inference procedure or the powerful artificial intelligence engines deployed on the cloud. Meanwhile, it is better for some IoT applications to have inference and learning processing near the end-user or near the sensors. But most of the smart sensor like embedded platforms available for IoT applications can only provide very limited computing power, memory size and storage space [134, 135].

To resolve this pain, and also to make deep learning more practical to adopt in daily life, the MobileNets series CNN models published in 2017 [136] stands out. Although these models are not designed especially for face recognition tasks, they are promising candidatures with much smaller size but very efficient learning ability compared with VGG-16 and GoogLeNet, which make them very attractive for IoT applications on embedded devices. With two adjustable hyper-parameters that control the model complexity and performance, MobileNets allow users to switch among 16 versions of implementations whose number of parameters and MAC operations range from 4.2 million to 0.5 million and 5.7×10^8 to 0.4×10^8 , respectively. The most complex version of MobileNets is about 32 times smaller and 27 times less compute intensive than the VGG-16 model, and one-third smaller and 2.5 times less complex than GoogLeNet while being able to achieve similar learning performance for classification on ImageNet database [137] compared to VGG-16 and GoogLeNet in terms of Top-1 accuracy.

Chapter 3

Distributed LMS Estimation with Event-based Diffusion Cooperation

3.1 Introduction

In this chapter, we consider a single-task diffusion network performs distributed LMS estimation. For the sake of saving communication overhead and reducing energy consumption, we modify the ATC diffusion strategy by introducing an event-based communication mechanism before the local combination step. Such that transmission of intermediate estimates from a sensor to its neighboring nodes is triggered only when a criterion is satisfied. We provide a sufficient condition for the mean error stability of our proposed event-based strategy, and derive an upper bound of its steady-state network MSD, we also obtain a sufficient condition under which this upper-bound holds. Numerical results demonstrate that the proposed event-based strategy can achieve similar steady-state network MSD as the ATC diffusion strategy but at a significantly lower communication rate.

The rest of this chapter is organized as follows. In Section 3.2, we introduce the network model, problem formulation, discuss prior works, and also describe our proposed EB-ATC strategy. In Section 3.3 we examine the performance of the proposed strategy from perspectives of mean and mean-square error behavior. Simulation results are demonstrated in Section 3.4 followed by concluding remarks

Figure 3-1: An example of an adaptive network where each node is associated with a cost function to minimize.

in Sections 3.5.

3.2 Data Models and Preliminaries

In this section, we first present our network and data model assumptions, then give a brief discussion of the ATC diffusion strategy, and finally present our proposed EB-ATC diffusion strategy.

3.2.1 Network and Data Model

In this chapter, we consider a single-task network consists of N nodes defined in Section 2.1.1. An example of such network model is shown in Fig. 3-1, where Node k minimizes its cost $J_k(w)$ cooperatively with neighboring nodes, i.e., node $\{m, \ell, n\}$, by exchanging local information $\bar{\psi}_k(i)$ with them. In a classical ATC diffusion network, transmitting and receiving information between two neighboring nodes happens simultaneously at every iteration. However, in this chapter, they may or may not take places at the same time, due to the event-based communication scheme.

Following the same network model detailed in Section 2.1.1. The neighborhood

of each node k is denoted by the notation $\mathcal{N}_k$ which represents the set contains both node k and every nodes connected to node k by an edge. In addition, every node in the network aims to estimate an common unknown parameter vector w° . At each time instant $i \geq 0$, each node k is able to observe a scalar random variable $\mathbf{d}_k(i)$ and a random vector $\mathbf{u}_k(i)$. The observed data $\{\mathbf{d}_k(i), \mathbf{u}_k(i)\}$ and parameter of interest w° is related through the linear regression model (2.1) introduced in 2.1.1 as shown below.

$$\mathbf{d}_k(i) = \mathbf{u}_k^\top(i)w^\circ + \mathbf{v}_k(i). \quad (3.1)$$

To facilitate the performance analysis in the sequel, we shall need the Assumption 1.

Remark 1. *It is noted that the independence condition for the regression data described in the Assumption 1 may not hold in general for different engineering applications. however, even if the independence assumption can not be satisfied the numerical results can still match the theoretical expressions when the step-sizes are sufficiently small [16, 138]. Literatures [78, 139] also showed that ATC diffusion strategy can still be stable even in the case where regressors and noise may be temporally correlated. As a commonly adopted approach in the literature [42] for simplifying the derivations of performance analysis, we shall proceed with Assumptio 1 in the following chapters of this thesis.*

3.2.2 ATC Diffusion Strategy for Distributed LMS Estimation

Recall that, in oder to estimate the parameter w° , the network minimizes the following network learning objective:

$$\min_w \sum_{k=1}^N J_k(w), \quad (3.2)$$

where each node k is associated with a local MSE cost as below,

$$\frac{1}{2}J_k(w) = \mathbb{E} |\mathbf{d}_k(i) - \mathbf{u}_k(i)^\top w|^2, \quad (3.3)$$

which is strongly convex and twice differentiable. Following the argument in [16], the cost function of network objective (4.3) can be further approximated by a local cost for each of the node k as below,

$$J_k^{\text{glob'}}(w) = J_k(w) + \sum_{\ell \in \mathcal{N}_k \setminus k} a_{\ell k} \|w - w^\circ\|^2. \quad (3.4)$$

Then using the observed data $\{\mathbf{d}_k(i), \mathbf{u}_k(i)\}$ to compute instantaneous approximation of true statistics

$$R_{u,k} \approx u_k(i)^\top u_k(i), \quad r_{du} \approx d_k(i) u_k(i), \quad (3.5)$$

and applying SGD to minimize local cost (3.4), we arrive at the ATC diffusion strategy [16, 21, 45, 66] which is a distributed adaptive filtering algorithm where every node k in the network attempts to solve (2.4) or (3.2) in a cooperative manner, by performing the following adaptive update of estimate and local convex combination at each time instant i :

$$\begin{aligned} \boldsymbol{\psi}_k(i) &= \mathbf{w}_k(i-1) + \mu_k \mathbf{u}_k(i) [\mathbf{d}_k(i) - \mathbf{u}_k(i)^\top \mathbf{w}_k(i-1)], \\ \mathbf{w}_k(i) &= \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\psi}_\ell(i), \end{aligned} \quad (3.6)$$

In the above equations, the procedure in the first line is referred to as the *adaptation* step and the second line is the *combination* step. Besides, the combination weights $\{a_{\ell k}\}$ are non-negative scalars and satisfy:

$$a_{\ell k} \geq 0, \quad \sum_{\ell=1}^N a_{\ell k} = 1, \quad a_{\ell k} = 0, \text{ if } \ell \notin \mathcal{N}_k. \quad (3.7)$$

The combination weights $\{a_{\ell k}\}$ scale the information node k receives from node $\ell \in \mathcal{N}_k$. Since the value of any weight obeys $a_{\ell k} \in [0, 1]$, the meaning of the weights can be interpreted as a degree of the confidence that node k assigns to its cooperation with node ℓ . It is shown in [16, 45, 67, 68] that the choices of $\{a_{\ell k}\}$ will not influence the convergence of diffusion strategies, however by assigning different weights the steady-state network MSD performance can be significantly affected. How to choose these combination weights is not treated in this thesis. The related

topic is studied in [16, 45, 67, 68], and in [85] for a heterogeneous network settings. For simplicity, we assume the *Metropolis rule* for choosing $\{a_{\ell k}\}$ in this chapter. Please see the section 3.4 for detailed implementation.

In [16, 66], it is showed that the local estimates $\mathbf{w}_k(i)$ in the ATC strategy converges in mean to the true parameter $\mathbf{w}^\circ$, if the following conditions are satisfied:

- a) The measured data $\{\mathbf{d}_k(i), \mathbf{u}_k(i)\}$ satisfy the model (2.1) and the Assumption 1.
- b) The combination weights satisfy (3.7) so that the matrix A whose (ℓ, k) -th entry is the weight $\{a_{\ell k}\}$ satisfy $A^\top \mathbf{1}_N = \mathbf{1}_N$.
- c) The step-sizes of each node k satisfy

$$\mu_k < \frac{2}{\lambda_{\max}(R_{u,k})}. \quad (3.8)$$

Moreover, if the step-sizes are chosen to be sufficiently small and also satisfy (3.8), then the network mean-square error $\frac{1}{N} \sum_{k=1}^N \|\tilde{\mathbf{w}}_k(i)\|^2$ converges as $i \rightarrow \infty$, and the network MSD can be approximately expressed as below

$$\text{MSD}^{\text{network}} = \frac{1}{N} \text{vec}(\mathcal{A}^\top \mathcal{M} \mathcal{S} \mathcal{M} \mathcal{A})^\top \cdot (I_{N^2 M^2} - \frac{1}{2} \mathcal{F})^{-1} \text{vec}(I_{NM}), \quad (3.9)$$

where the notation $\mathcal{A}$, $\mathcal{M}$, $\mathcal{S}$ and $\mathcal{F}$ will be define later in (3.35), (3.39), (3.62) and (3.68). The proof of mean convergence for the ATC strategy is provided in Appendix A.3. One can refer to [16] for the detailed proof of mean-square stability and the derivation steady-state network MSD, which is not revisited in this thesis due to tedious proof steps are involved.

3.2.3 Proposed EB-ATC Diffusion Strategy

Other than allowing every nodes to communicate with its neighbors at every iteration, we intend to modify the ATC strategy so that after the adaptation step the local intermediate estimate $\boldsymbol{\psi}_k(i)$ of each node k is only transmitted to its neighbors at certain triggering time instants s_k^n , $n \in \mathbb{N}$. Let $\bar{\boldsymbol{\psi}}_k(i)$ to denote the last local intermediate estimate node k sent to its neighbors at time instant i , then for

Algorithm 1 Event-based ATC Diffusion Strategy (EB-ATC)

1: **for** every node k at each time instant i **do**

2: ***Local Adaptation:***

3: Obtain intermediate estimate $\boldsymbol{\psi}_k(i)$ by performing

$$\boldsymbol{\psi}_k(i) = \boldsymbol{w}_k(i-1) + \mu_k \boldsymbol{u}_k(i) [\boldsymbol{d}_k(i) - \boldsymbol{u}_k(i)^\top \boldsymbol{w}_k(i-1)]$$

4: ***Event-based Triggering:***

5: Compute a prior gap $\boldsymbol{\epsilon}_k^-(i)$ and event triggering function $f(\boldsymbol{\epsilon}_k^-(i))$.

6: **if** $f(\boldsymbol{\epsilon}_k^-(i)) > \delta_k(i)$ **then**

7: (i) Trigger the communication, broadcast local update $\boldsymbol{\psi}_{k,i}$ to every neighbors $\ell \in \mathcal{N}_k$.

8: (ii) Mark $\gamma_k(i) = 1$, and update $\bar{\boldsymbol{\psi}}_\ell(i) = \boldsymbol{\psi}_\ell(i)$.

9: **else if** $f(\boldsymbol{\epsilon}_k^-(i)) \leq \delta_k(i)$ **then**

10: (i) Keep silent.

11: (ii) Mark $\gamma_k(i) = 0$, and update $\bar{\boldsymbol{\psi}}_\ell(i) = \bar{\boldsymbol{\psi}}_\ell(i-1)$.

12: **end if**

13: ***Diffusion Combination:***

14: Obtain new estimate by performing

$$\boldsymbol{w}_k(i) = a_{kk} \boldsymbol{\psi}_k(i) + \sum_{\ell \in \mathcal{N}_k \setminus k} a_{\ell k} \bar{\boldsymbol{\psi}}_\ell(i)$$

15: **end for**

any time instant j such that

$$j \in [s_k^n, s_k^{n+1}), \quad (3.10)$$

we let the following relationship hold,

$$\bar{\boldsymbol{\psi}}_k(j) = \boldsymbol{\psi}_k(s_k^n), \quad (3.11)$$

which means that for any neighbor of node k , the value of the intermediate estimate obtained from node k will remain the same until the next triggering of node k occurs.

To decide the next triggering of communication for each node k , we introduce $\boldsymbol{\epsilon}_k^-(i)$ as the *a priori* gap at each time instant i , which is given by

$$\boldsymbol{\epsilon}_k^-(i) = \boldsymbol{\psi}_k(i) - \bar{\boldsymbol{\psi}}_k(i-1). \quad (3.12)$$

Then, we leverage the following event triggering function,

$$f(\boldsymbol{\epsilon}_k^-(i)) = \|\boldsymbol{\epsilon}_k^-(i)\|^2. \quad (3.13)$$

Then for each node k , transmission of its newly obtained local intermediate estimate $\boldsymbol{\psi}_k(i)$ is triggered whenever

$$f(\boldsymbol{\epsilon}_k^-(i)) - \delta_k(i) > 0, \quad (3.14)$$

where $\delta_k(i)$ is the non-negative triggering threshold adopted by node k at time i . In general, the triggering thresholds $\{\delta_k(i)\}$ can be time-varying, so that we have the value of triggering threshold $\delta_k(i)$ of each node k is upper bounded by

$$\delta_k = \sup\{\delta_k(i) | i \geq 0\}. \quad (3.15)$$

In addition, we define axillary binary variables $\{\gamma_k(i)\}$ to indicate if node k trigger its communication at time instant i as below

$$\gamma_k(i) = \begin{cases} 1, & \text{if node } k \text{ trigger,} \\ 0, & \text{otherwise.} \end{cases} \quad (3.16)$$

Then, the sequence of triggering time instants

$$0 \leq s_k^1 \leq s_k^2 \leq \dots, \quad (3.17)$$

can be obtained recursively as

$$s_k^{n+1} = \min\{i \in \mathbb{N} | i > s_k^n, \gamma_k(i) = 1\}. \quad (3.18)$$

We run the EB-ATC strategy described in Algorithm 1 for all the nodes in the network. A succinct form of the EB-ATC can be summarized as the following

Figure 3-2: Block diagram illustrates the algorithm flow of the proposed EB-ATC for node k .

equations,

$$\boldsymbol{\psi}_k(i) = \mathbf{w}_k(i-1) + \mu_k \mathbf{u}_k(i) [\mathbf{d}_k(i) - \mathbf{u}_k(i)^\top \mathbf{w}_k(i-1)], \quad (3.19)$$

$$\mathbf{w}_k(i) = a_{kk} \boldsymbol{\psi}_k(i) + \sum_{\ell \in \mathcal{N}_k \setminus k} a_{\ell k} \bar{\boldsymbol{\psi}}_\ell(i). \quad (3.20)$$

Note that the combination step (3.20) of EB-ATC is different from that of ATC strategy. In EB-ATC each node k combines the intermediate local estimate $\bar{\boldsymbol{\psi}}_k(i)$ that its neighbor ℓ most recently sent to it. At the same time, each node k always combines its own intermediate estimate $\boldsymbol{\psi}_k(i)$ regardless of the triggering status.

To help with the demonstration of our proposed algorithm more clearly, a block diagram for the EB-ATC is provided in Fig. 3-2, where the shaded area in light green envelopes the event-based communication mechanism which is the major difference between the proposed EB-ATC and conventional ATC strategy. Although the EB-ATC needs additional computing operation and memory space to execute the even-triggering decision making and store the most recently received values, the memory size required does not impose additional burden for the modern embedded IoT devices, given that the information stored are only vectors (i.e., arrays of floating numbers which usually take up only a few KB in terms of computer data storage). In addition, computation required for EB-ATC is of same order of ATC for both multiplications and additions, which is only linearly proportional to the dimension of the unknown parameters. A comparison of computational complexity of the EB-ATC against ATC is shown in Table 3.1, where $n_k = |\mathcal{N}_k|$.

Table 3.1: Computational Complexity Comparison for Node k per Iteration

Algorithm	Multiplication	Addition
ATC	$(n_k + 2)M + 1$	$(n_k + 1)M$
EB-ATC	$(n_k + 3)M + 1$	$(n_k + 3)M - 1$

3.3 Performance Analysis

In this section, we analyze the performance of the proposed EB-ATC diffusion strategy, in terms of both mean and mean-square error behavior. We first provide the sufficient condition under which our EB-ATC is stable in the mean, and then show an upper bound of the steady-state MSD for the EB-ATC.

3.3.1 Network Error Recursions

To carry out the analysis of error behavior, we first define some necessary notations and derive the recursion models of errors across the network. To begin with, the error vectors of each node k at time instant i are given by

$$\tilde{\boldsymbol{\psi}}_k(i) = \boldsymbol{w}^\circ - \boldsymbol{\psi}_k(i), \quad (3.21)$$

$$\tilde{\boldsymbol{w}}_k(i) = \boldsymbol{w}^\circ - \boldsymbol{w}_k(i). \quad (3.22)$$

Since in previous Section 3.2.3, we stated that under EB-ATC every node only combines the local intermediate estimate $\{\bar{\boldsymbol{\psi}}_\ell(i) | \ell \in \mathcal{N}_k\}$ that most recently received from its neighbors. So, apart from the *a priori* gap given in (3.12), here we also introduce the *a posteriori* gap $\boldsymbol{\epsilon}_k(i)$ as

$$\boldsymbol{\epsilon}_k(i) = \boldsymbol{\psi}_k(i) - \bar{\boldsymbol{\psi}}_k(i), \quad (3.23)$$

in order to capture the discrepancy between the local intermediate estimate $\boldsymbol{\psi}_k(i)$ newly obtained at current time instant i , and the estimate $\bar{\boldsymbol{\psi}}_k(i)$ that is available at every neighboring nodes $\ell \in \mathcal{N}_k$. It is noteworthy that we have the following

hold,

$$\boldsymbol{\epsilon}_k(i) = \begin{cases} 0, & \text{if } \|\boldsymbol{\epsilon}_k^-(i)\|^2 > \delta_k(i), \\ \boldsymbol{\epsilon}_k^-(i), & \text{otherwise.} \end{cases} \quad (3.24)$$

Followed from (3.24), it is straightforward to have the result below,

$$\|\boldsymbol{\epsilon}_k(i)\| \leq (\delta_k)^{\frac{1}{2}} \quad (3.25)$$

Putting the iterates $\tilde{\boldsymbol{\psi}}_{k,i}$, $\tilde{\boldsymbol{w}}_{k,i}$, and $\boldsymbol{\epsilon}_k(i)$ across all nodes altogether into collective recursive vector, we have,

$$\tilde{\boldsymbol{\psi}}(i) = \text{col} \left\{ \left(\tilde{\boldsymbol{\psi}}_k(i) \right)_{k=1}^N \right\}, \quad (3.26)$$

$$\tilde{\boldsymbol{w}}(i) = \text{col} \left\{ \left(\tilde{\boldsymbol{w}}_k(i) \right)_{k=1}^N \right\}, \quad (3.27)$$

$$\boldsymbol{\epsilon}(i) = \text{col} \left\{ \left(\boldsymbol{\epsilon}_k(i) \right)_{k=1}^N \right\}. \quad (3.28)$$

Then, subtracting both sides of (3.19) from $\boldsymbol{w}^\circ$, and applying the data model (2.1), we obtain the error vector of local intermediate estimate for each node k as below:

$$\tilde{\boldsymbol{\psi}}_k(i) = [I_M - \mu_k \boldsymbol{u}_k(i) \boldsymbol{u}_k^\top(i)] \tilde{\boldsymbol{w}}_k(i) - \mu_k \boldsymbol{u}_k(i) \boldsymbol{v}_k(i). \quad (3.29)$$

Note that by substituting (3.23) into (3.20), the local combination step can be expressed as

$$\boldsymbol{w}_k(i) = a_{kk} \boldsymbol{\psi}_k(i) + \sum_{\ell \in \mathcal{N}_k \setminus k} a_{\ell k} (\boldsymbol{\psi}_\ell(i) - \boldsymbol{\epsilon}_\ell(i)), \quad (3.30)$$

or equivalently,

$$\boldsymbol{w}_k(i) = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\psi}_\ell(i) - \sum_{\ell \in \mathcal{N}_k \setminus k} a_{\ell k} \boldsymbol{\epsilon}_\ell(i). \quad (3.31)$$

Next, subtracting both sides of the above equation from $\boldsymbol{w}^\circ$ we obtain

$$\tilde{\boldsymbol{w}}_k(i) = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \tilde{\boldsymbol{\psi}}_\ell(i) + \sum_{\ell \in \mathcal{N}_k \setminus k} a_{\ell k} \boldsymbol{\epsilon}_\ell(i). \quad (3.32)$$

Let A be the matrix whose (ℓ, k) -th entry is the weight $a_{\ell k}$, and also introduce the matrix C as

$$C = A - \text{diag} \left\{ (a_{kk})_{k=1}^N \right\}, \quad (3.33)$$

then relating (3.27), (3.28), (3.29), and (3.32) yields the following recursion:

$$\tilde{\mathbf{w}}(i) = \mathcal{B}(i)\tilde{\mathbf{w}}(i-1) - \mathcal{A}^\top \mathcal{M} \mathbf{s}(i) + \mathcal{C}^\top \boldsymbol{\epsilon}(i), \quad (3.34)$$

where

$$\mathcal{A} = A \otimes I_M, \quad (3.35)$$

$$\mathcal{C} = C \otimes I_M, \quad (3.36)$$

$$\mathcal{B}(i) = \mathcal{A}^\top (I_{MN} - \mathcal{M} \mathcal{R}_u(i)), \quad (3.37)$$

$$\mathcal{R}_u(i) = \text{diag} \left\{ (\mathbf{u}_k(i) \mathbf{u}_k^\top(i))_{k=1}^N \right\}, \quad (3.38)$$

$$\mathcal{M} = \text{diag} \left\{ (\mu_k I_M)_{k=1}^N \right\}, \quad (3.39)$$

$$\mathbf{s}(i) = \text{col} \left\{ (\mathbf{u}_k(i) \mathbf{v}_k(i))_{k=1}^N \right\}. \quad (3.40)$$

3.3.2 Mean Error Analysis

Suppose Assumption 1 holds, then by taking expectation on the both sides of (3.34) we have the following recursive equation for the network mean error vector,

$$\mathbb{E}[\tilde{\mathbf{w}}(i)] = \mathcal{B} \mathbb{E}[\tilde{\mathbf{w}}(i-1)] + \mathcal{C}^\top \mathbb{E}[\boldsymbol{\epsilon}(i)], \quad (3.41)$$

where

$$\mathcal{B} = \mathbb{E}[\mathcal{B}] = \mathcal{A}^\top (I_{MN} - \mathcal{M} \mathcal{R}_u), \quad (3.42)$$

$$\mathcal{R}_u = \mathbb{E}[\mathcal{R}_u(i)] = \text{diag} \left\{ (R_{u,k})_{k=1}^N \right\}. \quad (3.43)$$

Furthermore, we have the following result on the asymptotic behavior of the network mean error.

Theorem 1. (Mean Error Stability) *Suppose that Assumption 1 holds. Then, the*

network mean error vector of EB-ATC, i.e., $\mathbb{E}[\tilde{\mathbf{w}}(i)]$, is bounded input bounded output (BIBO) stable in steady state if the step-size μ_k is chosen such that

$$\mu_k < \frac{2}{\lambda_{\max}(R_{u,k})}. \quad (3.44)$$

Moreover, the block maximum norm of the network mean error is upper-bounded by

$$\frac{\alpha}{1-\beta} \cdot \max_{1 \leq k \leq N} (\delta_k)^{\frac{1}{2}}, \quad (3.45)$$

where,

$$\alpha = \max_{1 \leq k \leq N} (1 - a_{kk}), \quad (3.46)$$

$$\beta = \|I_{MN} - \mathcal{MR}_u\|_{b,\infty}. \quad (3.47)$$

Proof. See Appendix 3.6.2 □

3.3.3 Mean-square Error Analysis

Thanks to the triggering mechanism and resulting *a posteriori* gap (3.28) correlates with the error vectors (3.26) and (3.27), so that explicitly characterizing the exact network MSD of EB-ATC is technically difficult. Instead, we investigate the upper bound of the network MSD in the sequel.

We shall first derive the MSD recursions as follows. From the recursion (3.34), we have the following for any compatible non-negative definite matrix Σ :

$$\begin{aligned} \|\tilde{\mathbf{w}}(i)\|_{\Sigma}^2 &= \tilde{\mathbf{w}}(i-1)^{\top} \mathbf{B}(i)^{\top} \Sigma \mathbf{B}(i) \tilde{\mathbf{w}}(i-1) \\ &\quad + \mathbf{s}(i)^{\top} \mathcal{M}^{\top} \mathcal{A} \Sigma \mathcal{A}^{\top} \mathcal{M} \mathbf{s}(i) + \boldsymbol{\epsilon}(i)^{\top} \mathcal{C} \Sigma \mathcal{C}^{\top} \boldsymbol{\epsilon}(i) \\ &\quad + 2\tilde{\mathbf{w}}(i-1)^{\top} \mathbf{B}(i)^{\top} \Sigma \mathcal{C}^{\top} \boldsymbol{\epsilon}(i) - 2\mathbf{s}(i)^{\top} \mathcal{M}^{\top} \mathcal{A} \Sigma \mathcal{C}^{\top} \boldsymbol{\epsilon}(i) \\ &\quad - 2\tilde{\mathbf{w}}(i-1)^{\top} \mathbf{B}(i)^{\top} \Sigma \mathcal{A}^{\top} \mathcal{M} \mathbf{s}(i). \end{aligned} \quad (3.48)$$

Taking expectation on both sides of the above expression, the last term evalu-

ates to zero under Assumption 1, and we obtain the following recursive equation,

$$\mathbb{E}\|\tilde{\mathbf{w}}(i)\|_{\Sigma}^2 = \mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|_{\Sigma'}^2 + t_2 + t_3 + 2t_4 - 2t_5, \quad (3.49)$$

where the weighting matrix Σ' is

$$\Sigma' = \mathbb{E} [\mathbf{B}(i)^{\top} \Sigma \mathbf{B}(i)], \quad (3.50)$$

and the last four terms in (3.49) are given as follows,

$$t_2 = \mathbb{E}[\mathbf{s}(i)^{\top} \mathcal{M} \mathcal{A} \Sigma \mathcal{A}^{\top} \mathcal{M} \mathbf{s}(i)], \quad (3.51)$$

$$t_3 = \mathbb{E}[\boldsymbol{\epsilon}(i)^{\top} \mathcal{C} \Sigma \mathcal{C}^{\top} \boldsymbol{\epsilon}(i)], \quad (3.52)$$

$$t_4 = \mathbb{E}[\tilde{\mathbf{w}}(i-1)^{\top} \mathbf{B}(i)^{\top} \Sigma \mathcal{C}^{\top} \boldsymbol{\epsilon}(i)], \quad (3.53)$$

$$t_5 = \mathbb{E}[\mathbf{s}(i)^{\top} \mathcal{M}^{\top} \mathcal{A} \Sigma \mathcal{C}^{\top} \boldsymbol{\epsilon}(i)]. \quad (3.54)$$

Further, let $\sigma = \text{vec}(\Sigma)$ and $\sigma' = \text{vec}(\Sigma')$, we then have

$$\sigma' = \mathcal{E} \sigma, \quad (3.55)$$

where

$$\begin{aligned} \mathcal{E} &= \mathbb{E} [\mathbf{B}(i)^{\top} \otimes \mathbf{B}(i)^{\top}] \\ &= [I_{M^2 N^2} - I_{MN} \otimes \mathcal{M} \mathcal{R}_u - \mathcal{M} \mathcal{R}_u \otimes I_{MN} + (\mathcal{M} \otimes \mathcal{M}) \mathbb{E} (\mathcal{R}_u(i) \otimes \mathcal{R}_u(i))] \mathcal{A} \otimes \mathcal{A}, \end{aligned} \quad (3.56)$$

so that (3.49) can be rewritten as,

$$\mathbb{E}\|\tilde{\mathbf{w}}(i)\|_{\sigma}^2 = \mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{E}\sigma}^2 + t_2 + t_3 + 2t_4 - 2t_5. \quad (3.57)$$

Moreover, we have the following inequality relationships for the terms $t_2, t_3, t_4,$

and t_5 (see Appendix 3.6.1 for detailed derivations),

$$t_2 = \text{vec}(\mathcal{A}^\top \mathcal{M} \mathcal{S} \mathcal{M} \mathcal{A})^\top \sigma, \quad (3.58)$$

$$t_3 \leq \Delta \cdot \text{vec}(\mathcal{C}^\top \mathcal{C}) \sigma, \quad (3.59)$$

$$2t_4 \leq \mathbb{E} \|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{E}\sigma}^2 + t_3, \quad (3.60)$$

$$t_5 = -\text{vec}(\mathcal{C}^\top \mathcal{G}(i) \mathcal{M} \mathcal{S} \mathcal{M} \mathcal{A})^\top \sigma, \quad (3.61)$$

where

$$\mathcal{S} = \text{diag} \{(\sigma_{v,k}^2 R_{u,k})_{k=1}^N\}, \quad (3.62)$$

$$\Delta = \sum_{k=1}^N \delta_k, \quad (3.63)$$

$$\mathcal{G}(i) = \mathbb{E} \text{diag} \{(\gamma_k(i) I_M)_{k=1}^N\} - I_{MN}. \quad (3.64)$$

Therefore, by substituting (3.58)-(3.61) into (3.57), we obtain the following inequality relationship for the network MSD,

$$\mathbb{E} \|\tilde{\mathbf{w}}(i)\|_\sigma^2 \leq \mathbb{E} \|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{D}\sigma}^2 + [f_1 + f_2 + f_3(i)]^\top \sigma, \quad (3.65)$$

where $\mathcal{D} = 2\mathcal{E}$, matrix $\mathcal{E}$ is given in (3.56), and

$$\begin{aligned} f_1 &= \text{vec}(\mathcal{A}^\top \mathcal{M} \mathcal{S} \mathcal{M} \mathcal{A}), \\ f_2 &= 2\Delta \cdot \text{vec}(\mathcal{C}^\top \mathcal{C}), \\ f_3(i) &= 2 \text{vec}(\mathcal{C}^\top \mathcal{G}(i) \mathcal{M} \mathcal{S} \mathcal{M} \mathcal{A}). \end{aligned} \quad (3.66)$$

Having the inequality relationship (3.65) and the above assumption, we further develop the following results.

Theorem 2. (Mean-square Error Behavior) *Suppose that Assumptions 1 holds. Then, as $i \rightarrow \infty$, the network MSD of EB-ATC, i.e., $\mathbb{E} \|\tilde{\mathbf{w}}(i)\|^2/N$, has a finite constant upper bound if the step sizes $\{\mu_k\}$ are chosen such that $\rho(\mathcal{D}) < 1$ is satisfied. In addition, it follows that*

$$\mathcal{D} = \mathcal{F} + O(\mathcal{M}^2), \quad (3.67)$$

where

$$\mathcal{F} = 2\mathcal{B}^\top \otimes \mathcal{B}^\top. \quad (3.68)$$

So that if $\{\mu_k\}$ also satisfy

$$\frac{1 - \frac{\sqrt{2}}{2}}{\lambda_{\min}(R_{u,k})} < \mu_k < \frac{1 + \frac{\sqrt{2}}{2}}{\lambda_{\max}(R_{u,k})}, \quad (3.69)$$

then an upper bound of the network MSD in steady state is given by

$$[(f_1 + f_2)^\top (I_{M^2N^2} - \mathcal{F})^{-1} + f_{3,\infty}] \sigma + O(\mu_{\max}^2), \quad (3.70)$$

where f_1, f_2 have been given in (3.66), and

$$\sigma = \frac{1}{N} \text{vec}(I_{MN}), \quad (3.71)$$

$$\mu_{\max} = \max_{1 \leq k \leq N} \{\mu_k\}, \quad (3.72)$$

$$f_{3,\infty} = \lim_{i \rightarrow \infty} \sum_{j=0}^{i-1} f_3(i-j)^\top \mathcal{F}^j. \quad (3.73)$$

Proof. See Appendix 3.6.3 □

Remark 2. For (3.69), to make the L.H.S. is smaller than the R.H.S. so that the set of μ_k is non-empty, we might need the condition number of $R_{u,k}$ no bigger than $\frac{2+\sqrt{2}}{2-\sqrt{2}}$. However, (3.69) is just a sufficient condition to guarantee the right hand side (R.H.S.) of (3.65) converges at steady state, so that the steady-state network MSD of EB-ATC will not diverge and has a finite upper bound in theory. Indeed, as can be observed from numerical results in Section 3.4, choosing a step size even smaller than left hand side (L.H.S.) of (3.69) will not render the network MSD of the EB-ATC at steady state divergent. This is because in stochastic gradient descent (which is used in the local adaptation step of EB-ATC), the smaller the step size, the lower the learning rate, and thus the smaller the MSD. Therefore, the L.H.S of (3.69) is no longer needed to ensure the stability of the network MSD.

Based on the rationale in Remark 2, it can be concluded that the network MSD

of EB-ATC will not diverge at the steady state, as long as $\{\mu_k\}$ satisfy

$$\mu_k < \frac{1 + \frac{\sqrt{2}}{2}}{\lambda_{\max}(R_{u,k})}. \quad (3.74)$$

3.4 Simulation Results

3.4.1 Simulation Setup

In this section, we shall illustrate the MSD performance and energy-efficiency of the proposed EB-ATC, and compare against ATC and the non-cooperative LMS algorithm. To this end, we consider numerical examples where the parameter of interest w° has a dimension of $M = 10$, and the network consists of $N = 60$ nodes is setup and depicted in Fig. 3-3a. The measurement noise powers $\{\sigma_{v,k}^2\}$ are generated from a uniform distribution over $[-25, -10]$ dB. Besides, we adopt covariance $R_{u,k} = \sigma_{u,k}^2 I_M$ for the zero-mean regressor $u_k(i)$, where the coefficients $\{\sigma_{u,k}^2\}$ are drawn uniformly from the interval $[1, 2]$. The data and noise profile across the network is shown in Fig. 3-3b.

For the ease of implementation, we adopt constant and uniform triggering thresholds $\delta_k(i) = \delta$ for every node. Moreover, we adopt the *Metropolis* rule for the diffusion combination (3.20) since it has low complexity and good performance in terms of both convergence rate and steady-state MSD for single-task diffusion networks [16]. To be specific, each node k computes the weights $\{a_{\ell k}\}$ for all its neighboring nodes by the following rule,

$$a_{\ell k} = \begin{cases} \frac{1}{\max\{|\mathcal{N}_k|, |\mathcal{N}_\ell|\}}, & \text{if } \ell \neq k \text{ and } \ell \in \mathcal{N}_k, \\ 1 - \sum_{\ell \in \mathcal{N}_k \setminus \{k\}} a_{\ell k}, & \text{if } \ell = k, \\ 0, & \text{otherwise.} \end{cases} \quad (3.75)$$

To demonstrate the communication reducing ability of EB-ATC, we define the metric of expected network triggering rate (ENTR) as follows:

$$\text{ENTR}(i) = \frac{1}{N} \sum_{k=1}^N \mathbb{E} \gamma_k(i). \quad (3.76)$$

Figure 3-3: (a) The network topology. The numbers next to the nodes are node indices. (b) Data and noise profile of each node. From top to bottom: i) noise power ii) input signal power.

The value of ENTR at time instant i captures how often communication is triggered by each node at that time instant i , on average. Therefore, ENTR is directly proportional to the average communication overhead incurred by the nodes in the network at each time instant. From (3.76), it is straightforward that $0 \leq \text{ENTR}(i) \leq 1$. So that a smaller value of $\text{ENTR}(i)$ means a lower energy consumption. Moreover, ATC strategy has $\text{ENTR}(i) = 1$ for all $i \geq 0$, since in ATC networks nodes keep exchanging information at every iteration. All the simulation results shown in the sequel are averaged over 200 Monte Carlo runs.

(a) MSD performance

(b) ENTR curves

Figure 3-4: Simulation results for a stationary environment (a) the MSD performance comparison, (b) The ENTR comparison of EB-ATC with different triggering thresholds.

3.4.2 Stationary Environment

In this case, we set step sizes $\mu = 0.02$ for all the nodes, the numerical results are shown in Fig. 3-4. It can be observed from Fig. 3-4(a) that MSDs of the proposed EB-ATC in steady-state are a few dBs higher than that of ATC strategy, but are still much lower compared with the non-cooperative LMS algorithm, which demonstrates that our proposed EB-ATC is able to preserve the advantages of diffusion cooperation. Meanwhile, the convergence of EB-ATC is relatively slower. This is due to in the transient phase, the event-based communication mechanism of EB-

Figure 3-5: Performance comparison among other algorithms.

ATC restricts nodes in the network from exchanging the newest local intermediate estimates $\{\psi_{k,i}\}$ too frequently, so as to save energy and reduce the occupation of limited communication resources which is particularly helpful in the scenarios like massive IoT networks. Nevertheless, this leads to inferior transient performance compared to ATC.

On the other hand, EB-ATC achieves significant communication overhead savings compared to ATC. From Fig. 3-4(b), we can see that the ENTR curves of EB-ATC decay rapidly over time during the transient phase, and for all the different triggering thresholds we tested, EB-ATC uses less than 30% of the communication overhead of ATC after the time instant $i \approx 200$, which is approximately the average time that the MSD of ATC drops within 95% of its steady-state value. This demonstrates that even though EB-ATC has not reached steady state (at $i \approx 600$), communication between nodes is not triggered very frequently since the intermediate estimates do not alter significantly after this time instant. Furthermore, in steady state, although each node maintains estimates that are close to the true parameter value, communication triggering does not completely stop. This is because occasional abrupt changes in the random noise and regressors can make the update of local estimate deviate significantly, which is in the same spirit of why MSD does not converge to zero.

We also provide comparison of MSD performance among state-of-the-art ATC-type diffusion LMS strategies including the Data-Reserved Periodic Diffusion LMS (DR-PDLMS) [93] and the Partial Diffusion LMS (P-DLMS) [89]. The comparison results are shown in Fig. 3-5, where the simulation environment is same with previous case except the step sizes are chose as $\mu = 0.05$ for each of the nodes. We select period value $p = 5$ for DR-PDLMS, and $L = 2$ for P-DLMS such that these two algorithm are able to achieve 80% savings of communication. To make a fair comparison, we use $\delta = 2 \times 10^{-3}$ for our EB-ATC to achieve a similar level of communication saving (about 81% on average over 1000 iterations). It is clear in Fig. 3-5 that our proposed EB-ATC outperforms both DR-PDLMS and P-DLMS in terms of steady-state MSD performance, while being slightly slower in the convergence.

3.4.3 Non-stationary Environment

In addition to the previous examples in stationary environments, we also tested our EB-ATC in non-stationary scenarios, in order to demonstrate the tracking performance as well as to compare energy saving ability against conventional ATC strategy in a more practical situation.

In the first non-stationary example we configure the unknown parameters to change randomly at time instants $i = 500m$, for $m = 1, 2, 3$. The step sizes are set to $\mu = 0.05$ for every node, all the other experiment setups remain unchanged with those in the previous stationary scenario. To obtain more comprehensive results, we tested seven different values of triggering thresholds, i.e.,

$$\delta = \{1.0, 1, 25, 1.5, 1.75, 2.0, 2.25, 2.5\} \times 10^{-3}. \quad (3.77)$$

In order to obtain figures which are clearer and more readable for showing the tracking ability as well as the resulting ENTR, we pick four thresholds from (3.77) and illustrate corresponding MSD and ENTR in Fig. 3-6. All the MSDs of EB-ATC with different thresholds converges within 500 iterations. As expected, from Fig. 3-6(a), it can be observed that MSDs of EB-ATC are higher than that of ATC by only a few dBs. At the same time, even the smallest triggering threshold we

(a) MSD performance

(b) ENTR curves

Figure 3-6: Non-stationary case where parameter changes at certain time instant (a) MSD performance comparison. (b) The resulting ENTR curves.

tested can achieve a communication overhead saving of about 70% (see Fig. 3-6(b)) at most of the time instants, without sacrificing the steady-state accuracy and convergence speed.

In practical applications, the unknown parameters usually change over time, which requires learning or estimation algorithms to keep continuous adaptation. Although ATC strategy converges relatively faster than the EB-ATC, nodes in a ATC network have to keep exchanging information in the steady state so as to maintain accurate estimates and track changes in the parameters. As has been mentioned earlier, communicating at every iteration leads to large overhead, which can be avoided by the proposed EB-ATC. To visualize this effect, in Fig. 3-7 we illustrate the trade offs between the cooperation benefits (measured in terms of steady-state MSD that EB-ATC improves against the non-cooperative LMS) and

Figure 3-7: Cooperation benefit against the ACOS.

the average communication overhead saving (ACOS) which is computed via

$$\text{ACOS} = 1 - \frac{1}{n} \sum_{i=1}^n \text{ENTR}(i), \quad (3.78)$$

where the number of iterations was $n = 2000$ in Fig. 3-6. From Fig. 3-7, it is clear that bigger triggering thresholds leads to higher ACOS value but fewer cooperation benefit that EB-ATC can achieve. What's more, all the tested triggering thresholds can achieve communication overhead savings more than 60% on average over all time instants. Among which the three largest thresholds, i.e. $\delta = \{2.0, 2.25, 2.5\} \times 10^{-3}$, lead to energy savings more than 80% while being less than 3 dB worse in terms of cooperation benefit compared to ATC strategy that achieves a cooperation benefit of 12.8 dB. However, this saving ability is almost the same with that obtained by the smallest triggering threshold.

In the second non-stationary test case, we allow the parameter to vary continuously at every iteration. To be specific, the parameter follows

$$\mathbf{w}^\circ(i) = \mathbf{w}^\circ(i-1) + \mathbf{z}(i), \quad (3.79)$$

where $\mathbf{z}(i)$ is the input driving vector drawn from a standard Gaussian distribution

Figure 3-8: Non-stationary case where parameters vary continuously at every iteration.

with covariance matrix $2 \times 10^{-6} I_M$. It can be seen from Fig. 3-8 that both ATC and EB-ATC are able to converge as was in stationary case, which indicates that although the parameter undergoes a slow continuous change at every time instants the algorithms can still maintain accurate tracking rather than diverging or being fluctuating in MSD. However, when compared to the first 500 iterations of previous non-stationary case where parameter does not change, the steady-state MSDs are much higher and the ACOSs also drop obviously, which is expected since continuously time-varying parameter is more difficult to track and will cause intermediate estimates deviate more significantly compared with stationary cases, which in turn leads to more frequent triggering of communication.

3.5 Summary

We have proposed an EB-ATC strategy for diffusion networks where every node communicate with its neighbors only when significant changes occur in the local updates. The proposed algorithm is not only able to significantly reduce communication overhead, but can still maintain satisfactory MSD performance at steady state compared with the conventional ATC diffusion strategy. In addition, we also inves-

tigated the error behavior of the proposed strategy in both mean and mean-square sense, and derived sufficient conditions to secure both mean and mean-squared stabilities.

3.6 Derivations and Proofs

3.6.1 Derivations of (3.58)-(3.61)

We derive the expressions and related inequality relationships for terms t_2 , t_3 , t_4 , and t_5 as below.

Term t_2

For this term, we have

$$\begin{aligned}
 t_2 &= \mathbb{E} [\text{Tr} (\mathcal{A}^\top \mathcal{M} \mathbf{s}(i) \mathbf{s}(i)^\top \mathcal{M} \mathcal{A} \Sigma)] \\
 &= \text{Tr} [\mathcal{A}^\top \mathcal{M} \mathbb{E} (\mathbf{s}(i) \mathbf{s}(i)^\top) \mathcal{M} \mathcal{A} \Sigma] \\
 &= \text{Tr} (\mathcal{A}^\top \mathcal{M} \mathcal{S} \mathcal{M} \mathcal{A} \Sigma) \\
 &= \text{vec} (\mathcal{A}^\top \mathcal{M} \mathcal{S} \mathcal{M} \mathcal{A})^\top \sigma,
 \end{aligned} \tag{3.80}$$

where the equality (3.80) follows from the identity

$$\text{Tr}(AB) = \text{vec}(A^T)^T \text{vec}(B), \tag{3.81}$$

and matrix $\mathcal{S}$ in (3.80) is given by

$$\mathcal{S} = \text{diag} \{ (\sigma_{v,k}^2 R_{u,k})_{k=1}^N \}. \tag{3.82}$$

Term t_3

Similarly, we have the following for t_3 ,

$$\begin{aligned}
 t_3 &= \text{Tr} [\mathcal{C}^\top \mathbb{E} (\boldsymbol{\epsilon}(i) \boldsymbol{\epsilon}(i)^\top) \mathcal{C} \Sigma] \\
 &= \text{vec}(\mathcal{C})^\top [\Sigma \otimes \mathbb{E} (\boldsymbol{\epsilon}(i) \boldsymbol{\epsilon}(i)^\top)] \text{vec}(\mathcal{C})
 \end{aligned} \tag{3.83}$$

Moreover, by applying linear algebra techniques, it can be verified that the relationship $yy^\top \leq y^\top y I_N$ holds for any vector $y \in \mathbb{R}^N$, and thus $\boldsymbol{\epsilon}(i) \boldsymbol{\epsilon}(i)^\top \leq \boldsymbol{\epsilon}(i)^\top \boldsymbol{\epsilon}(i) I_{MN}$

follows immediately. So that we have

$$\begin{aligned}
\mathbb{E}(\boldsymbol{\epsilon}(i)\boldsymbol{\epsilon}(i)^\top) &\leq \mathbb{E}[\boldsymbol{\epsilon}(i)^\top \boldsymbol{\epsilon}(i) I_{MN}] \\
&= \mathbb{E}\left[\sum_{k=1}^N \|\boldsymbol{\epsilon}_k(i)\|^2 I_{MN}\right] \\
&\leq \sum_{k=1}^N \delta_k I_{MN}.
\end{aligned} \tag{3.84}$$

The last inequality follows from (3.25). Letting

$$\Delta = \sum_{k=1}^N \delta_k, \tag{3.85}$$

due to $\Sigma \geq 0$ the following results follows,

$$\Sigma \otimes [\mathbb{E}(\boldsymbol{\epsilon}(i)\boldsymbol{\epsilon}(i)^\top) - \Delta I_{MN}] \leq 0, \tag{3.86}$$

and therefore,

$$\text{vec}(\mathcal{C})^\top \{\Sigma \otimes [\mathbb{E}(\boldsymbol{\epsilon}(i)\boldsymbol{\epsilon}(i)^\top) - \Delta I_{MN}]\} \text{vec}(\mathcal{C}) \leq 0, \tag{3.87}$$

which gives,

$$\begin{aligned}
&\text{vec}(\mathcal{C})^\top [\Sigma \otimes \mathbb{E}(\boldsymbol{\epsilon}(i)\boldsymbol{\epsilon}(i)^\top)] \text{vec}(\mathcal{C}) \\
&\leq \Delta \cdot \text{vec}(\mathcal{C})^\top (\Sigma \otimes I_{MN}) \text{vec}(\mathcal{C}) \\
&= \Delta \cdot \text{Tr}(\mathcal{C}^\top \mathcal{C} \Sigma),
\end{aligned} \tag{3.88}$$

by relating the above inequality to (3.83), we obtain

$$t_3 \leq \Delta \cdot \text{vec}(\mathcal{C}^\top \mathcal{C}) \sigma. \tag{3.89}$$

We have used the identity

$$\text{vec}(AXB) = (B^\top \otimes A) \text{vec}(X) \tag{3.90}$$

to obtain (3.88), and (3.89) follows from (3.81).

Term t_4

Since matrix $\Sigma \geq 0$, so that we have $\Sigma = \Theta\Theta^\top$. Then, let

$$\begin{aligned} P &= \tilde{\mathbf{w}}(i-1)^\top \mathbf{B}^\top(i) \Theta, \\ Q &= \boldsymbol{\epsilon}(i)^\top \mathcal{C} \Theta. \end{aligned} \quad (3.91)$$

From the fact $(P - Q)(P - Q)^\top \geq 0$ we have,

$$PQ^\top + QP^\top \leq PP^\top + QQ^\top. \quad (3.92)$$

Substituting (3.91) into the above inequality and taking expectation on the both sides gives,

$$\begin{aligned} 2t_4 &\leq \mathbb{E} [\tilde{\mathbf{w}}(i-1)^\top \mathbf{B}(i)^\top \Sigma \mathbf{B}(i) \tilde{\mathbf{w}}(i-1)] + \mathbb{E} [\boldsymbol{\epsilon}(i)^\top \mathcal{C} \Sigma \mathcal{C}^\top \boldsymbol{\epsilon}(i)] \\ &= \mathbb{E} \|\tilde{\mathbf{w}}(i-1)\|_{\Sigma'}^2 + t_3, \\ &= \mathbb{E} \|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{E}\sigma}^2 + t_3. \end{aligned} \quad (3.93)$$

Term t_5

Applying manipulations similar to t_3 , we obtain

$$\begin{aligned} t_5 &= \text{Tr} [\mathcal{C}^\top \mathbb{E} (\boldsymbol{\epsilon}(i) \mathbf{s}(i)^\top) \mathcal{M} \mathcal{A} \Sigma] \\ &= \text{vec} (\mathcal{C}^\top \mathbb{E} (\boldsymbol{\epsilon}(i) \mathbf{s}(i)^\top) \mathcal{M} \mathcal{A} \Sigma)^\top \sigma. \end{aligned} \quad (3.94)$$

In order to evaluate the covariance matrix $\mathbb{E} (\boldsymbol{\epsilon}(i) \mathbf{s}(i)^\top)$, we shall separately derive its (k, ℓ) -th block entry, i.e., $\mathbb{E} [\boldsymbol{\epsilon}_k(i) \mathbf{u}_\ell(i) \mathbf{v}_\ell(i)]$ which can be expressed by

$$\mathbb{E} [\boldsymbol{\epsilon}_k(i) \mathbf{u}_\ell(i) \mathbf{v}_\ell(i)] = \mathbb{E} [\boldsymbol{\psi}_k(i) \mathbf{u}_\ell(i) \mathbf{v}_\ell(i)] - \mathbb{E} [\bar{\boldsymbol{\psi}}_k(i) \mathbf{u}_\ell(i) \mathbf{v}_\ell(i)] \quad (3.95)$$

Substituting the data model (2.1) into the local adaptation (3.19), we can express the local intermediate estimate $\boldsymbol{\psi}_k(i)$ as follows,

$$\boldsymbol{\psi}_k(i) = \mathbf{w}_k(i-1) + \mu_k \mathbf{u}_k(i) \mathbf{u}_k(i)^\top \tilde{\mathbf{w}}_k(i-1) + \mu_k \mathbf{u}_k(i) \mathbf{v}_k(i), \quad (3.96)$$

so that we have

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\psi}_k(i)\mathbf{u}_\ell(i)\mathbf{v}_\ell(i)] &= \mathbb{E}[\mathbf{w}_k(i-1)\mathbf{u}_\ell(i)\mathbf{v}_\ell(i)] \\
&+ \mu_k \mathbb{E}[\mathbf{u}_k(i)\mathbf{u}_k(i)^\top \tilde{\mathbf{w}}_k(i-1)\mathbf{u}_\ell(i)\mathbf{v}_\ell(i)] \\
&+ \mu_k \mathbb{E}[\mathbf{u}_k(i)\mathbf{v}_k(i)\mathbf{u}_\ell(i)\mathbf{v}_\ell(i)].
\end{aligned} \tag{3.97}$$

Note that (3.97) evaluates to zero if $\ell \neq k$, and when $\ell = k$ the first two terms in (3.97) also evaluate to zero, and the last term equals $\mu_k \sigma_{v,k}^2 R_{u,k}$. Moreover, from (3.12) and (3.24), at particular time instant i , we have $\mathbb{E}[\overline{\boldsymbol{\psi}}_k(i)\mathbf{u}_\ell(i)\mathbf{v}_\ell(i)]$ equals $\mathbb{E}[\boldsymbol{\psi}_k(i)\mathbf{u}_\ell(i)\mathbf{v}_\ell(i)]$ if node k trigger the communication, and evaluates to zero, otherwise. Hence, we can conclude that

$$\mathbb{E}[\boldsymbol{\epsilon}_k(i)\mathbf{u}_\ell(i)\mathbf{v}_\ell(i)] = \begin{cases} 0, & \text{if } \ell \neq k, \\ \mu_k \sigma_{v,k}^2 R_{u,k}, & \text{if } \ell = k \text{ and } \gamma_k(i) = 0. \end{cases}$$

So that the term t_5 can be expressed as,

$$t_5 = -\text{vec}(\mathcal{C}^\top \mathcal{G}(i) \mathcal{M} \mathcal{S} \mathcal{M} \mathcal{A})^\top \sigma, \tag{3.98}$$

where matrix $\mathcal{S}$ is given in (3.62) and

$$\mathcal{G}(i) = \mathbb{E} \text{diag} \{ (\gamma_k(i) I_M)_{k=1}^N \} - I_{MN}. \tag{3.99}$$

3.6.2 Proof of the Theorem 1

Taking block maximum norm $\|\cdot\|_{b,\infty}$ to the vector $\mathbb{E}[\boldsymbol{\epsilon}(i)]$, due to every norm is a convex function of its argument, by Jensen's inequality and Lemma ??, we have

$$\|\mathbb{E}[\boldsymbol{\epsilon}(i)]\|_{b,\infty} \leq \mathbb{E} \left[\|\boldsymbol{\epsilon}(i)\|_{b,\infty} \right] \tag{3.100}$$

$$= \mathbb{E} \left[\max_{1 \leq k \leq N} \|\boldsymbol{\epsilon}_k(i)\| \right] \tag{3.101}$$

$$\leq \max_{1 \leq k \leq N} \left(\frac{\delta_k}{\lambda_{\min}(Y)} \right)^{\frac{1}{2}}, \tag{3.102}$$

where we have used the definition of the block maximum norm in [16] for the equality (3.101), and inequality (3.102) follows from the Lemma ???. The R.H.S. of (3.102) is a finite constant scalar, which implies that the input signal to the recursion (3.41), i.e., $\mathbb{E}[\epsilon(i)]$ is bounded. Therefore, the recursion (3.41) is BIBO stable if $\rho(\mathcal{B}) < 1$.

In addition, since matrix $\mathcal{A}^\top$ is left-stochastic, by applying the Lemma D.5 and Lemma D.6 in [16], we have the following from (3.42),

$$\rho(\mathcal{B}) = \rho(\mathcal{A}^\top (I_{MN} - \mathcal{M}\mathcal{R}_u)) \quad (3.103)$$

$$\leq \rho(I_{MN} - \mathcal{M}\mathcal{R}_u) \quad (3.104)$$

$$= \|I_{MN} - \mathcal{M}\mathcal{R}_u\|_{b,\infty}. \quad (3.105)$$

Therefore, we conclude that the network mean error is BIBO stable if

$$\|I_{MN} - \mathcal{M}\mathcal{R}_u\|_{b,\infty} < 1, \quad (3.106)$$

which further yields the condition (3.44).

To establish the upper bound (3.45), we iterate (3.41) from $i = 0$, which gives,

$$\mathbb{E}[\tilde{\mathbf{w}}(i)] = \mathcal{B}^i \mathbb{E}[\tilde{\mathbf{w}}(0)] + \sum_{j=0}^{i-1} \mathcal{B}^j \mathcal{C}^\top \mathbb{E}[\epsilon(i-j)]. \quad (3.107)$$

Then applying block maximum norm $\|\cdot\|_{b,\infty}$ on both sides of the above equation, by the properties of vector norms and induced matrix norms, it can be obtained that

$$\begin{aligned} \|\mathbb{E}[\tilde{\mathbf{w}}(i)]\|_{b,\infty} &\leq \|\mathcal{B}^i\|_{b,\infty} \|\mathbb{E}[\tilde{\mathbf{w}}(0)]\|_{b,\infty} + \sum_{j=0}^{i-1} \|\mathcal{B}^j\|_{b,\infty} \|\mathcal{C}^\top \mathbb{E}[\epsilon(i-j)]\|_{b,\infty} \\ &\leq \|\mathcal{A}^\top\|_{b,\infty}^i \|I_{MN} - \mathcal{M}\mathcal{R}_u\|_{b,\infty}^i \|\mathbb{E}[\tilde{\mathbf{w}}(0)]\|_{b,\infty} \\ &\quad + \sum_{j=0}^{i-1} \left(\|\mathcal{A}^\top\|_{b,\infty}^j \|I_{MN} - \mathcal{M}\mathcal{R}_u\|_{b,\infty}^j \|\mathcal{C}^\top\|_{b,\infty} \|\mathbb{E}[\epsilon(i-j)]\|_{b,\infty} \right). \end{aligned} \quad (3.108)$$

Let $\alpha = \|\mathcal{C}^\top\|_{b,\infty}$, from the Lemma D.3 of [16] we have

$$\alpha = \|\mathcal{C}^\top\|_\infty = \max_{1 \leq k \leq N} (1 - a_{kk}). \quad (3.109)$$

Moreover, since matrix $\mathcal{A}^\top$ is left-stochastic, so that we have $\|\mathcal{A}^\top\|_{b,\infty} = 1$ by the Lemma D.4 of [16]. Let

$$\beta = \|I_{MN} - \mathcal{MR}_u\|_{b,\infty}, \quad (3.110)$$

then substitute (3.102) into (3.108) we obtain the following,

$$\begin{aligned} \|\mathbb{E}[\tilde{\mathbf{w}}(i)]\|_{b,\infty} &\leq \|\mathbb{E}[\tilde{\mathbf{w}}(0)]\|_{b,\infty} \cdot \beta^i \\ &\quad + \alpha \cdot \max_{1 \leq k \leq N} (\delta_k)^{\frac{1}{2}} \cdot \sum_{j=0}^{i-1} \beta^j. \end{aligned} \quad (3.111)$$

If step size μ_k is chosen to satisfy $0 \leq \beta < 1$, then letting $i \rightarrow \infty$ on both sides of (3.111) we arrive at following inequality relationship

$$\lim_{i \rightarrow \infty} \|\mathbb{E}[\tilde{\mathbf{w}}(i)]\|_{b,\infty} \leq \frac{\alpha}{1 - \beta} \cdot \max_{1 \leq k \leq N} (\delta_k)^{\frac{1}{2}}, \quad (3.112)$$

and the proof is complete.

3.6.3 Proof of the Theorem 2

To obtain the upper bound of network MSD at steady state, iterating (3.65) from $i = 1$, we have

$$\mathbb{E}\|\tilde{\mathbf{w}}(i)\|_\sigma^2 \leq \mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{D}\sigma}^2 + (f_1 + f_2)^\top \left(\sum_{j=0}^{i-1} \mathcal{D}^j \right) \sigma + \left(\sum_{j=0}^{i-1} f_3(i-j)^\top \mathcal{D}^j \right) \sigma, \quad (3.113)$$

where the vectors f_1 , f_2 , and $f_3(i)$ are given previously in (3.66). Letting $i \rightarrow \infty$, the first term on the R.H.S. of the above inequality converges to zero, and the

second term converge to a finite value

$$(f_1 + f_2)^\top (I_{M^2 N^2} - \mathcal{D})^{-1} \sigma, \quad (3.114)$$

if and only if $\mathcal{D}^i \rightarrow 0$ as $i \rightarrow \infty$, i.e., $\rho(\mathcal{D}) < 1$. From (3.64) and (3.66) we have $f_3(i)$ is bounded due to every entry of matrix $\mathcal{G}(i)$ is bounded. Furthermore, if $\rho(\mathcal{D}) < 1$, there exists a norm $\|\cdot\|_\zeta$ such that $\|\mathcal{D}\|_\zeta < 1$, therefore we have

$$|f_3(i-j)^\top \mathcal{D}^j \sigma| \leq a \cdot \|\mathcal{D}\|_\zeta^j, \quad (3.115)$$

for some positive constant a . Since $\|\mathcal{D}\|_\zeta^j \rightarrow 0$ as $j \rightarrow \infty$, the series,

$$\sum_{j=0}^{i-1} |f_3(i-j)^\top \mathcal{D}^j \sigma| \quad (3.116)$$

converges as $i \rightarrow \infty$, which implies the absolute convergence of the third term of R.H.S of (3.113).

Besides, the matrix $\mathcal{F}$ given in (3.68) can be explicitly expressed as

$$\begin{aligned} \mathcal{F} &= 2\mathcal{B}^\top \otimes \mathcal{B}^\top \\ &= [I_{M^2 N^2} - I_{MN} \otimes \mathcal{M}\mathcal{R}_u - \mathcal{M}\mathcal{R}_u \otimes I_{MN} (\mathcal{M} \otimes \mathcal{M}) (\mathcal{R}_u \otimes \mathcal{R}_u)] \mathcal{A} \otimes \mathcal{A}. \end{aligned} \quad (3.117)$$

Substituting (3.56) in to $\mathcal{D} = 2\mathcal{E}$ and comparing the expression of matrix $\mathcal{D}$ with (3.117), we can obtain

$$\mathcal{D} = \mathcal{F} + O(\mathcal{M}^2), \quad (3.118)$$

where

$$O(\mathcal{M}^2) = (\mathcal{M} \otimes \mathcal{M}) \{ \mathbb{E} [\mathcal{R}_u(i) \otimes \mathcal{R}_u(i)] - \mathcal{R}_u \otimes \mathcal{R}_u \}. \quad (3.119)$$

So that substituting (3.118) into the R.H.S of (3.113) gives

$$\begin{aligned} \mathbb{E}\|\tilde{\mathbf{w}}(i)\|_{\sigma}^2 &\leq \mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{F}\sigma}^2 + (f_1 + f_2)^{\top} \left(\sum_{j=0}^{i-1} \mathcal{F}^j \right) \sigma + \left(\sum_{j=0}^{i-1} f_3(i-j)^{\top} \mathcal{F}^j \right) \sigma \\ &\quad + \mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|_{O(\mathcal{M}^2)\sigma}^2 + g(i)^{\top} O(\mathcal{M}^2)\sigma, \end{aligned} \quad (3.120)$$

where

$$g(i) = f_1 + f_2 + \sum_j^{i-1} f_3(j). \quad (3.121)$$

Due to the vector $g(i)$ is bounded, therefore, if $\rho(\mathcal{D}) < 1$ such that the mean-squared error term $\mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|_{\sigma}^2$ is bounded, then the last two terms on the R.H.S of (3.120) become $O(\mu_{\max}^2)$. So the inequality (3.120) can be further expressed as below

$$\begin{aligned} \mathbb{E}\|\tilde{\mathbf{w}}(i)\|_{\sigma}^2 &\leq \mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{F}\sigma}^2 + (f_1 + f_2)^{\top} \left(\sum_{j=0}^{i-1} \mathcal{F}^j \right) \sigma \\ &\quad + \left(\sum_{j=0}^{i-1} f_3(i-j)^{\top} \mathcal{F}^j \right) \sigma + O(\mu_{\max}^2). \end{aligned} \quad (3.122)$$

Choosing $\sigma = \frac{\text{vec}(I_{MN})}{N}$ and using arguments similar for (3.113), as $i \rightarrow \infty$ the R.H.S of (3.122) converges to

$$\frac{1}{N} [(f_1 + f_2) (I_{M^2N^2} - \mathcal{F})^{-1} + f_{3,\infty}] \text{vec}(I_{MN}) + O(\mu_{\max}^2), \quad (3.123)$$

if and only if $\mathcal{F}$ is stable, namely, $\rho(\mathcal{F}) < 1$, where $f_{3,\infty}$ is given in (3.73). Since $\rho(\mathcal{F}) = 2\rho(\mathcal{B})^2$, so that a sufficient condition to guarantee $\rho(\mathcal{F}) < 1$ is $\rho(\mathcal{B}) < \frac{\sqrt{2}}{2}$. In addition, by the Lemma D.5 in [16], we have

$$\rho(\mathcal{B}) \leq \max_{1 \leq k \leq N} \rho(I_{MN} - \mu_k R_{u,k}) \quad (3.124)$$

Therefore, we can obtain that $\rho(\mathcal{B}) < \frac{\sqrt{2}}{2}$ hold if step sizes is chosen to satisfy condition (3.69). The proof is complete.

Chapter 4

Diffusion Strategy for Multitask Networks with Optimized Inter-Cluster Cooperation

4.1 Introduction

In this chapter, we consider distributed LMS estimation for a multitask diffusion network where nodes are grouped into several interconnected clusters. Hereby, each cluster performs LMS estimation of the realizations of a different unknown random parameter. Different from the previous chapter 3 where we have concerned about the communication and energy efficiency, in this chapter we address the performance optimization of a multitask diffusion network. So that every nodes, clusters and the whole diffusion network can operate in a more cost-effective manner, and the potential waste of limited resources like communication bandwidth and energy budget caused by improper choice of cooperation weights can be avoided.

In clustered multitask diffusion networks [42], the inter-cluster cooperation plays an important role of bridging the cooperation between clusters associated with different learning objectives. Therefore, the inter-cluster weights and regularization strength should be appropriately selected to improve the overall network performance, rather than using naively selected values which may even worsen the network performance. However, there is no discussion and methodology mentioned in [42] on how to find the suitable inter-cluster cooperation weights and the

regularization strength, which motivates us to fill this gap.

Inspired by the elegance of the adapt-then-combine diffusion strategy, we propose a multitask diffusion strategy where an inter-clustered cooperation is only inserted right before the intra-cluster combination and is affected by a single factor. As such, the mean stability of the proposed strategy can be guaranteed as long as every individual node is stable in the mean on their own right, regardless of how the inter-cluster cooperation weights are chosen. This feature facilitates the free selection of the relevant cooperation weights, so that the optimization schemes can then be developed without concerning about the network stability.

In addition, performance-wise, the proposed estimation strategy is able to achieve an asymptotically unbiased estimation, when the parameters of all the clusters are same in the mean. To realize the objective of improving the network performance, we develop a centralized as well as a distributed inter-cluster cooperation weights selection schemes that enable every node in the network to optimally adjust its cooperation weights associated to the inter-cluster neighbors. Simulation results demonstrate that our approaches leads to an improved steady-state network MSD on average, compared with the case where clusters do not cooperate each other and selecting weights by several other commonly adopted methods in the literature.

The rest of this chapter is organized as follows. In Section 4.2, we introduce the network model involved in this chapter and the problem formulation of distributed LMS estimation in clustered multitask networks. We also briefly review the prior works, and introduce our proposed multitask strategy. In Section 4.3 we examine the convergence performance of the proposed strategy via mean and mean-square error behavior analysis. In Section 4.4, we present our optimization schemes for the selection of the inter-cluster cooperation weights and a related adaptive implementation. Finally, several numerical results are shown and the a summary of this chapter follow in Sections 4.5 and 4.6, respectively.

Figure 4-1: An example of a clustered multitask network with three clusters depicted in red, blue, and green circles, respectively.

4.2 Models, Formulations, and Multitask Diffusion

In this section, we first present network and data model of clustered multitask networks, briefly overview the multitask diffusion LMS strategy proposed by [42], and then present our proposed multitask diffusion strategy.

4.2.1 Models and Assumptions

In this chapter, we consider a clustered multitask network with N nodes (see an example depicted in Fig. 4-1). Similar to the single-task networks, here the multitask network is also represented by an undirected graph where the vertices denote the nodes, any two nodes are said to be connected if there is an edge between them. The neighborhood of any particular node k is denoted by $\mathcal{N}_k$, which include all the nodes that are connected to node k , and node k itself. Since the network is assumed to be undirected, if node k is a neighbor of node ℓ then node ℓ is also a neighbor of node k . Without loss of generality, we assume that the network is connected.

Following the same settings and definitions for the network model described in Section 2.2.1, let the clusters be indexed by $\mathcal{P} \triangleq \{1, 2, \dots, P\}$. In this chapter, we

suppose the unknown parameters interested to each cluster, namely $\mathbf{w}_{C_p}^\circ$, $p \in \mathcal{P}$, to be *random* and correlated with each other. This is different from most of the literatures [16, 42, 45, 83], which assume that parameters of interest are deterministic. Thus, our objective is then to characterize the expected network estimation performance, where the expectation is not only taken over the random realizations of the data and noise, but also the random parameters. Although [140] also assumes that parameters are random and follow a random walk model, that work considers a *single-task* network where every node in the network learn the same parameter of interest.

We let $\mathbf{w}_k^\circ = \mathbf{w}_{C_p}^\circ$ for any node $k \in C_p$. At each time instant $i \geq 1$, each node k observes a random variable $\mathbf{d}_k(i) \in \mathbb{R}$, and a random vector $\mathbf{u}_k(i) \in \mathbb{R}^{M \times 1}$. We assume these observations are related to $\mathbf{w}_k^\circ$ via the linear regression model similar to (2.1) introduced in 2.1.1 as shown below.

$$\mathbf{d}_k(i) = \mathbf{u}_k^\top(i) \mathbf{w}_k^\circ + \mathbf{v}_k(i). \quad (4.1)$$

Note that although $\mathbf{w}_k^\circ$ is random, for the steady-state analysis in the sequel, its realization does not vary over the time instants i . Besides, we need the following assumptions.

Assumption 2. *The parameters of interest $\{\mathbf{w}_k^\circ\}_{k=1}^N$ are independent of $\{\mathbf{u}_\ell(i), \mathbf{v}_\ell(i)\}$ for all $\{k, \ell, i\}$.*

Remark 3. *Along with the Assumption 1 described in Section 2.1.1, we also need the above Assumption 2 in order to facilitate the performance analysis in the sequel. The need of independent conditions in this chapter follows similar statement in Remark 1.*

Assumption 3. *All the cluster parameters take values from the same compact space $\mathcal{W}$, and we do not have any prior information on their distributions. It is further assumed that these cluster parameters are uniformly distributed over $\mathcal{W}$, and share the same mean value $\bar{\mathbf{w}} \in \mathbb{R}^M$. Let $\mathbf{w}^\circ = \text{col} \{(\mathbf{w}_k^\circ)_{k=1}^N\}$, we have $\mathbb{E} \mathbf{w}^\circ = \mathbf{1}_N \otimes \bar{\mathbf{w}}$. In addition, the second-order moment of $\mathbf{w}^\circ$ is*

$$\mathcal{R}_w = \mathbb{E} \mathbf{w}^\circ \mathbf{w}^{\circ\top}. \quad (4.2)$$

Remark 4. *An example of scenario described in the above assumption is when clusters are tasked to localize different targets that may appear randomly within a specified area. The assumption that cluster parameters have the same mean can also be used for modeling the case where the mean of each cluster parameter is a known a priori so that can be subtracted out from the data model (4.1). For instance, a cluster may be responsible for monitoring a specific sub-area, meanwhile a target might appear within that sub-area with a known distribution. Besides, the above (4.2) is needed because the parameters $\{\mathbf{w}_{C_p}^\circ\}_{p \in \mathcal{P}}$ may be correlated in general. For example, in the aforementioned target tracking problem, if each cluster track a different target and the targets are moving in tandem or formation, then these underlying unknown parameters can be correlated with each other. Similarly, when estimating the temperature of an region, the temperature field of each individual position can be different however correlated. It is thus beneficial to cooperate among clusters by allowing nodes in different clusters to exchange information with each other.*

To enhance such inter-cluster cooperation so that the network can make the most of limited resources, we shall develop inter-cluster cooperation schemes which can minimize the network MSD *on average* over realizations of the cluster parameters. As will become clear in the sequel, finding the optimal inter-cluster cooperation depends on $\mathcal{R}_w$. We shall first assume that the correlation matrix $\mathcal{R}_w$ is known a priori, and then provide an adaptive implementation where $\mathcal{R}_w$ is estimated in each iteration.

4.2.2 Multitask Diffusion Strategy with Normalized Inter-cluster Cooperation

In [42], each node cluster $C_p, p \in \{1, 2, \dots, N\}$ is associated with the following cost that is similar to (2.14) previously formulated in Section 2.2.1 :

$$J(w_{C_p}, w_{-C_p}) = \sum_{k \in \mathcal{N}_{C,k}} \left(\frac{1}{2} \mathbb{E} |\mathbf{d}_k(i) - \mathbf{u}_k(i)^\top w_{C(k)}|^2 + \eta \sum_{\ell \in \mathcal{N}_{I,k}} \rho_{\ell k} \|w_{C(k)} - w_{C(\ell)}\|^2 \right), \quad (4.3)$$

where $\mathcal{N}_{C,k}$ and $\mathcal{N}_{I,k}$ denotes the intra-cluster neighborhood and inter-cluster neighborhood, respectively, whose definition has been given in Section 2.2.1. Coefficient η and $\{\rho_{\ell k}\}$ are non-negative regularization parameter and inter-cluster cooperation weights, respectively. Following the similar treatment in [16], cost function (4.3) can be further decomposed to the following cost for each of the node k in cluster $\mathcal{C}(k)$,

$$J_{C(k)}(w_k) = \frac{1}{2} \mathbb{E} |\mathbf{d}_k(i) - \mathbf{u}_k(i)^\top w_{C(k)}|^2 + \eta \sum_{\ell \in \mathcal{N}_{I,k}} \rho_{\ell k} \|w_{C(k)} - w_{C(\ell)}\|^2 + \sum_{\ell \in \mathcal{N}_{C,k}} a_{\ell k} \|w_{C(k)} - w_{C(\ell)}^\circ\|^2, \quad (4.4)$$

then using the observed data $\{\mathbf{d}_k(i), \mathbf{u}_k(i)\}$ for instantaneous approximation of true statistics

$$R_{u,k} \approx u_k(i)^\top u_k(i), \quad r_{du} \approx d_k(i) u_k(i), \quad (4.5)$$

and applying SGD to minimize local cost (4.4), a multitask diffusion LMS strategy (MDLMS) which takes the following form is obtained in the work [42]:

$$\begin{cases} \boldsymbol{\psi}_k(i) &= \mathbf{w}_k(i-1) + \mu_k \mathbf{u}_k(i) [\mathbf{d}_k(i) - \mathbf{u}_k(i)^\top \mathbf{w}_k(i-1)] \\ &\quad + \mu_k \sum_{\ell \in \mathcal{N}_{I,k}} \eta \rho_{\ell k} [\mathbf{w}_\ell(i-1) - \mathbf{w}_k(i-1)] \\ \mathbf{w}_k(i) &= \sum_{\ell \in \mathcal{N}_{C,k}} a_{\ell k} \boldsymbol{\psi}_\ell(i) \end{cases} \quad (4.6)$$

where μ_k is a constant step size. Besides, the scalar intra-cluster combination weights $\{a_{\ell k}\}$ satisfy

$$a_{\ell k} \geq 0, \sum_{\ell \in \mathcal{N}_{C,k}} a_{\ell k} = 1, a_{\ell k} = 0, \text{ if } \ell \notin \mathcal{N}_{C,k}. \quad (4.7)$$

Likewise, the scalar inter-cluster cooperation weights $\{\rho_{\ell k}\}$ satisfy

$$\rho_{\ell k} \geq 0, \sum_{\ell \in \mathcal{N}_{I,k}} \rho_{\ell k} = 1, \rho_{\ell k} = 0, \text{ if } \ell \notin \mathcal{N}_{I,k}. \quad (4.8)$$

Compared with the local adaptation step of the ATC strategy [45], the additional third term on the right-hand side (R.H.S.) of the first equation of (4.6) is used to promote the cooperation among clusters by absorbing the estimates $\mathbf{w}_\ell(i-1)$ from neighboring nodes in different clusters. Thus, more information is utilized during diffusion-type adaptation, so that the network estimation performance could be improved by selecting η and $\{\rho_{\ell k}\}$ properly.

Note that the inter-cluster weights defined in (4.8) is subject to sum-to-one normalization [42], it is η that controls the overall regularization strength. Choosing $\eta = 0$ will make the MDLMS degrade to the normal ATC diffusion LMS within each cluster and no more inter-cluster cooperation exists. On the other hand, a large η will impose more similarity to the estimates over the entire network. However, for clusters have heterogeneous parameter of interest, a large η maybe improper and even deteriorates the network MSD performance. In [42], the author proposes to choose the value of η heuristically according the experienced knowledge of the application at hand, and chose inter-cluster cooperation weights by the following way,

$$\rho_{\ell k} = \begin{cases} |\mathcal{N}_{I,k}|^{-1}, & \text{if } \ell \in \mathcal{N}_{I,k}, \\ 0, & \text{otherwise,} \end{cases} \quad (4.9)$$

for the ease of implementation. We will refer to (4.9) as the *inter-cluster averaging rule* in the sequel.

Last but not least, the main results on mean and mean-square convergence properties of the above MDLMS established in [42] are revisited as below.

- a) If the measured data $\{\mathbf{d}_k(i), \mathbf{u}_k(i)\}$ satisfy the model (2.1) and the Assumption 1, and the combination weights satisfy (4.7) so that the matrix A satisfy $A^\top \mathbb{1}_N = \mathbb{1}_N$, then the MDLMS asymptotically converges in the mean and if the step-sizes of each node k satisfy

$$\mu_k < \frac{2}{\lambda_{\max}(R_{u,k}) + 2\eta}. \quad (4.10)$$

- b) The MDLMS is mean-square stable if the step sizes μ_k for all $k = 1, 2, \dots, N$ is chosen to be sufficiently small and also satisfy the above condition (4.10)

It is worth noting that, the condition on step-size (4.10) clearly shows the mean and also mean-square stability of MDLMS is affected by the regularization strength factor η , which indicates that given a pre-determined value of step size, one may not be able to choose η freely since it is likely to make the network unstable. To overcome this issue, and also to optimally select the inter-cluster cooperation weights, we propose a new multitask diffusion strategy detailed in the sequel.

4.2.3 The Multitask Adapt, Inter-cluster cooperate, and then Combine diffusion strategy

In this subsection, we propose a multitask diffusion strategy that performs adaptation before inter-cluster cooperation. We show in Section 4.3 that our proposed strategy can achieve asymptotically unbiased estimation and mean stability irrespective of the inter-cluster cooperation weights under the assumptions in Section 4.2.1.

Conditioned on $\mathbf{w}_{\mathcal{C}(k)} = w_{\mathcal{C}(k)}$ in the local cost (2.12) for all $k = 1, 2, \dots, N$,

$$J_k(w_{\mathcal{C}(k)}) = \frac{1}{2} \mathbb{E} \left[\left| \mathbf{d}_k(i) - \mathbf{u}_k(i)^\top \mathbf{w}_{\mathcal{C}(k)} \right|^2 \middle| \mathbf{w}_{\mathcal{C}(k)} = w_{\mathcal{C}(k)} \right] \quad (4.11)$$

we consider the same cluster cost function formulated in (2.14). Proceeding in a

similar derivation as [42], we can rewrite the first equation of (4.6) as:

$$\begin{cases} \boldsymbol{\psi}_k(i) &= \boldsymbol{w}_k(i-1) + \mu_k \boldsymbol{u}_k(i) [\boldsymbol{d}_k(i) - \boldsymbol{u}_k(i)^\top \boldsymbol{w}_k(i-1)], \\ \boldsymbol{\phi}_k(i) &= \boldsymbol{\psi}_k(i) - \mu_k \sum_{\ell \in \mathcal{N}_{I,k}} \eta \rho_{\ell k} [\boldsymbol{w}_k(i-1) - \boldsymbol{w}_\ell(i-1)]. \end{cases} \quad (4.12)$$

Since the value $\boldsymbol{\psi}_k(i)$ is the updated local estimate and thus a better estimate for $\boldsymbol{w}_k^\circ$ than $\boldsymbol{w}_k(i-1)$ [44], we use $\boldsymbol{\psi}_k(i)$ and $\boldsymbol{\psi}_\ell(i)$ to replace $\boldsymbol{w}_k(i-1)$ and $\boldsymbol{w}_\ell(i-1)$, respectively in the second equation of (4.12). Let

$$\begin{aligned} g_{\ell k} &= \mu_k \eta \rho_{\ell k}, \text{ if } \ell \in \mathcal{N}_{I,k}, \\ g_{kk} &= 1 - \mu_k \sum_{\ell \in \mathcal{N}_{I,k}} \eta \rho_{\ell k}. \end{aligned} \quad (4.13)$$

These eliminate the redundant degree of freedom offered by η which complicates the optimization of the inter-cluster cooperation weights $\{\rho_{\ell k}\}$ in (4.8). We obtain the following Multitask Adapt, Inter-cluster cooperate, and then Combine (MAIC) diffusion strategy:

$$\begin{cases} \boldsymbol{\psi}_k(i) &= \boldsymbol{w}_k(i-1) + \mu_k \boldsymbol{u}_k(i) [\boldsymbol{d}_k(i) - \boldsymbol{u}_k(i)^\top \boldsymbol{w}_k(i-1)] \\ \boldsymbol{\phi}_k(i) &= \sum_{\ell \in \mathcal{N}_{I,k}^+} g_{\ell k} \boldsymbol{\psi}_\ell(i) \\ \boldsymbol{w}_k(i) &= \sum_{\ell \in \mathcal{N}_{C,k}} a_{\ell k} \boldsymbol{\phi}_\ell(i) \end{cases} \quad (4.14)$$

where $\mathcal{N}_{I,k}^+ = \mathcal{N}_{I,k} \cup \{k\}$ and the inter-cluster weights $\{g_{\ell k}\}$ satisfy

$$g_{\ell k} \geq 0, \sum_{\ell \in \mathcal{N}_{I,k}^+} g_{\ell k} = 1, g_{\ell k} = 0, \text{ if } \ell \notin \mathcal{N}_{I,k}^+. \quad (4.15)$$

Let G be a matrix with the (ℓ, k) -th entry being $g_{\ell k}$. Then, from (4.13), the matrix G is left-stochastic, i.e., $G^\top \mathbf{1}_N = \mathbf{1}_N$.

In the proposed MAIC diffusion strategy (4.14), the first line of equations denotes the adaptation step, the second line is the *inter-cluster cooperation* step, and the third line is the local combination step. Comparing (4.14) to the ATC diffusion strategy shown in (3.6), it is clear that the proposed MAIC diffusion strategy is in

the same spirit as the ATC diffusion strategy, which performs adaptation at each node before combining information from neighboring nodes. Similarly, MAIC first performs adaptation, before cooperatively integrating information from neighboring nodes not within its cluster. Finally, it performs combination of information from neighboring nodes within its own cluster. As will be showed in the sequel, unlike the condition (4.10), convergence properties of the proposed MAIC will no longer depends on parameters affect the inter-cluster cooperation.

4.3 Convergence Analysis of the MAIC Strategy

In this section, we analyze the performance of the proposed MAIC diffusion strategy by studying its convergence behavior for both mean and mean-square error vectors. Then we derive sufficient conditions for stability in mean and mean-square sense, and also show the steady-state network MSD. For ease of reference, we put the commonly used symbols in Table 4.1.

4.3.1 Recursive Models of Network Errors

We first derive recursive models for several error vectors that will be used for our stability analysis. The error vectors at each node k for each time instant i are defined as

$$\begin{aligned}\tilde{\boldsymbol{\psi}}_k(i) &= \mathbf{w}_k^\circ - \boldsymbol{\psi}_k(i), \\ \tilde{\boldsymbol{\phi}}_k(i) &= \mathbf{w}_k^\circ - \boldsymbol{\phi}_k(i), \\ \tilde{\mathbf{w}}_k(i) &= \mathbf{w}_k^\circ - \mathbf{w}_k(i).\end{aligned}\tag{4.16}$$

The individual iterates $\{\tilde{\boldsymbol{\psi}}_k(i)\}$, $\{\tilde{\boldsymbol{\phi}}_k(i)\}$, and $\{\tilde{\mathbf{w}}_k(i)\}$ are further collected into network error iterates as below:

$$\begin{aligned}\tilde{\boldsymbol{\psi}}(i) &= \text{col} \left\{ (\tilde{\boldsymbol{\psi}}_k(i))_{k=1}^N \right\}, \\ \tilde{\boldsymbol{\phi}}(i) &= \text{col} \left\{ (\tilde{\boldsymbol{\phi}}_k(i))_{k=1}^N \right\}, \\ \tilde{\mathbf{w}}(i) &= \text{col} \left\{ (\tilde{\mathbf{w}}_k(i))_{k=1}^N \right\}.\end{aligned}\tag{4.17}$$

Table 4.1: Definitions of Commonly Used Symbols.

Symbol	Equation
$\mathcal{A} = A \otimes I_M$	(4.24)
$\mathcal{G} = G \otimes I_M$	(4.22)
$\mathcal{R}_u(i) = \text{diag} \{ (\mathbf{u}_k(i) \mathbf{u}_k(i)^\top)_{k=1}^N \}$	(4.28)
$\mathcal{R}_u = \mathbb{E} \mathcal{R}_u(i) = \text{diag} \{ (R_{u,k})_{k=1}^N \}$	(4.35)
$\mathcal{M} = \text{diag} \{ (\mu_k I_M)_{k=1}^N \}$	(4.29)
$\mathcal{S} = \text{diag} \{ (\sigma_{v,k}^2 R_{u,k})_{k=1}^N \}$	(4.41)
$\mathcal{B}(i) = \mathcal{A}^\top \mathcal{G}^\top (I_{MN} - \mathcal{M} \mathcal{R}_u(i))$	(4.27)
$\mathcal{B} = \mathbb{E} \mathcal{B}(i) = \mathcal{A}^\top \mathcal{G}^\top (I_{MN} - \mathcal{M} \mathcal{R}_u)$	(4.33)
$\mathcal{E} = \mathbb{E} [\mathcal{B}(i)^\top \otimes \mathcal{B}(i)^\top]$	(4.48)
$\mathcal{F} = \mathcal{B}^\top \otimes \mathcal{B}^\top$	(4.54)
$\mathcal{R}_w = \mathbb{E} \mathbf{w}^\circ \mathbf{w}^{\circ\top}$	(4.2)
$R_{w,\ell k} = \mathbb{E} \mathbf{w}_\ell^\circ \mathbf{w}_k^{\circ\top}$	(4.67)
$\Omega_{a,k} = \text{diag} \left\{ \left(\mu^2 \sigma_{v,\ell}^2 \text{Tr}(R_{u,\ell}) \right)_{\ell \in \mathcal{N}_{I,k}^+} \right\}$	(4.70)
$\Omega_{b,k} = \mathbb{E} [\mathbf{W}_k^\top \mathbf{W}_k]$	(4.71)
$\mathbf{W}_k = [\mathbf{w}_\ell^\circ]_{\ell \in \mathcal{N}_{I,k}^+}$	(4.72)
$n_k = \mathcal{N}_{I,k}^+ $	(4.73)

Subtracting both sides of the first equation of (4.14) from $\mathbf{w}_k^\circ$, and applying the data model (4.1), we obtain the following error recursion:

$$\tilde{\boldsymbol{\psi}}_k(i) = (I_M - \mu_k \mathbf{u}_k(i) \mathbf{u}_k(i)^\top) \tilde{\mathbf{w}}_k(i-1) - \mu_k \mathbf{u}_k(i) \mathbf{v}_k(i). \quad (4.18)$$

Note that the second equation of (4.14) can be expressed as,

$$\boldsymbol{\phi}_k(i) = g_{kk} \boldsymbol{\psi}_k(i) + \sum_{\ell \in \mathcal{N}_{I,k}} g_{\ell k} \boldsymbol{\psi}_\ell(i). \quad (4.19)$$

Similarly, subtracting both sides of the above equation from $\mathbf{w}_k^\circ$ leads to

$$\tilde{\phi}_k(i) = g_{kk}(\mathbf{w}_k^\circ - \psi_k(i)) + \sum_{\ell \in \mathcal{N}_{I,k}} g_{\ell k}(\mathbf{w}_\ell^\circ - \psi_\ell(i) + \mathbf{w}_k^\circ - \mathbf{w}_\ell^\circ) \quad (4.20)$$

$$= \sum_{\ell \in \mathcal{N}_{I,k}^+} g_{\ell k} \tilde{\psi}_\ell(i) + \sum_{\ell \in \mathcal{N}_{I,k}^+} g_{\ell k}(\mathbf{w}_k^\circ - \mathbf{w}_\ell^\circ)$$

Relating the above equation with the second line of (4.17) gives

$$\tilde{\phi}(i) = \mathcal{G}^\top \tilde{\psi}(i) + (I_{MN} - \mathcal{G}^\top) \mathbf{w}^\circ, \quad (4.21)$$

where

$$\mathcal{G} = G \otimes I_M. \quad (4.22)$$

Then, subtracting the third equation of (4.14) from $\mathbf{w}_k^\circ$ and using (4.17) we have

$$\tilde{\mathbf{w}}(i) = \mathcal{A}^\top \tilde{\phi}(i), \quad (4.23)$$

where

$$\mathcal{A} = A \otimes I_M. \quad (4.24)$$

Now, substituting (4.21) into the equation (4.23) yields

$$\tilde{\mathbf{w}}(i) = \mathcal{A}^\top \mathcal{G}^\top \tilde{\psi}(i) + \mathcal{A}^\top (I_{MN} - \mathcal{G}^\top) \mathbf{w}^\circ. \quad (4.25)$$

Finally, substituting (4.18) into the above expression we arrive at the following error recursion:

$$\tilde{\mathbf{w}}(i) = \mathcal{B}(i) \tilde{\mathbf{w}}(i-1) - \mathbf{s}(i) + \mathbf{r}, \quad (4.26)$$

where

$$\mathcal{B}(i) = \mathcal{A}^\top \mathcal{G}^\top (I_{MN} - \mathcal{M} \mathcal{R}_u(i)), \quad (4.27)$$

$$\mathcal{R}_u(i) = \text{diag} \{ (\mathbf{u}_k(i) \mathbf{u}_k(i)^\top)_{k=1}^N \}, \quad (4.28)$$

$$\mathcal{M} = \text{diag} \{ (\mu_k I_M)_{k=1}^N \}, \quad (4.29)$$

$$\mathbf{s}(i) = \mathcal{A}^\top \mathcal{G}^\top \mathcal{M} \text{col} \{ (\mathbf{u}_k(i) \mathbf{v}_k(i))_{k=1}^N \}, \quad (4.30)$$

$$\mathbf{r} = \mathcal{A}^\top (I_{MN} - \mathcal{G})^\top \mathbf{w}^\circ. \quad (4.31)$$

4.3.2 Mean Error Analysis

Suppose Assumptions 1 and 2 all hold, then by taking expectation on both sides of (4.26), we obtain:

$$\mathbb{E} \tilde{\mathbf{w}}(i) = \mathcal{B} \mathbb{E} \tilde{\mathbf{w}}(i-1) + \mathbf{r}, \quad (4.32)$$

where

$$\mathcal{B} = \mathbb{E} \mathcal{B}(i) = \mathcal{A}^\top \mathcal{G}^\top (I_{MN} - \mathcal{M} \mathcal{R}_u), \quad (4.33)$$

$$\mathbf{r} = \mathbb{E} \mathbf{r} = \mathcal{A}^\top (I_{MN} - \mathcal{G})^\top \mathbb{E} \mathbf{w}^\circ, \quad (4.34)$$

and

$$\mathcal{R}_u = \mathbb{E} \mathcal{R}_u(i) = \text{diag} \{ (R_{u,k})_{k=1}^N \}. \quad (4.35)$$

Theorem 3. (Mean stability) *Suppose that Assumptions 1-2 hold. Then, MAIC is stable in the mean if the step-size μ_k is chosen such that*

$$\mu_k < \frac{2}{\lambda_{\max}(R_{u,k})}. \quad (4.36)$$

Furthermore, if Assumption 3 also holds, then MAIC is asymptotically unbiased.

Proof. See Appendix 4.7.1. □

From Theorem 3, we see that condition (4.36) is also the one that ensures the mean stability of a single node in the non-cooperative case [44, 45, 68]. Therefore,

if individual nodes are stable in the mean, the network that applies MAIC will be stable in the mean as well. On the contrary, even though every individual node is stable in the mean in the non-cooperative case, MDLMS is not guaranteed to be mean stable since its mean stability depends on the regularization weight η [42].

4.3.3 Mean-Square Error Analysis

We now examine the mean-square stability of the proposed MAIC strategy, under the Assumptions 1-2. From the error recursion (4.26), for any compatible non-negative definite matrix Σ we have that,

$$\begin{aligned} \|\tilde{\mathbf{w}}(i)\|_{\Sigma}^2 &= \tilde{\mathbf{w}}^{\top}(i-1)\mathbf{B}^{\top}(i)\Sigma\mathbf{B}(i)\tilde{\mathbf{w}}(i-1) + \mathbf{s}^{\top}(i)\Sigma\mathbf{s}(i) + \mathbf{r}^{\top}\Sigma\mathbf{r} \\ &\quad + 2\mathbf{r}^{\top}\Sigma\mathbf{B}(i)\tilde{\mathbf{w}}(i-1) - 2\tilde{\mathbf{w}}^{\top}(i-1)\mathbf{B}(i)^{\top}\Sigma\mathbf{s}(i) - 2\mathbf{s}^{\top}(i)\Sigma\mathbf{r}, \end{aligned} \quad (4.37)$$

where matrix $\mathbf{B}(i)$ is given in (4.27). Taking expectation on both sides of the expression (4.37), the last two terms of (4.37) equal zero according to the Assumptions 1-2, therefore we obtain the following equality relationship:

$$\mathbb{E}\|\tilde{\mathbf{w}}(i)\|_{\Sigma}^2 = \mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|_{\Sigma'}^2 + \mathbb{E}\|\mathbf{s}(i)\|_{\Sigma}^2 + \mathbb{E}\|\mathbf{r}\|_{\Sigma}^2 + 2\mathbb{E}(\mathbf{r}^{\top}\Sigma\mathbf{B}(i)\tilde{\mathbf{w}}(i-1)), \quad (4.38)$$

where

$$\Sigma' = \mathbb{E} [\mathbf{B}^{\top}(i)\Sigma\mathbf{B}(i)]. \quad (4.39)$$

Letting $\sigma = \text{vec}(\Sigma)$ and evaluating the second term of the R.H.S. of (4.38), we have

$$\begin{aligned} \mathbb{E}\|\mathbf{s}(i)\|_{\Sigma}^2 &= \mathbb{E} \text{Tr} (\mathbf{s}(i)\mathbf{s}^{\top}(i)\Sigma) \\ &= \text{Tr} [(\mathbb{E}\mathbf{s}(i)\mathbf{s}^{\top}(i)) \Sigma] \\ &= \text{Tr} (\mathcal{A}^{\top}\mathcal{G}^{\top}\mathcal{M}\mathcal{S}\mathcal{M}\mathcal{G}\mathcal{A}\Sigma) \\ &= \text{vec} (\mathcal{A}^{\top}\mathcal{G}^{\top}\mathcal{M}\mathcal{S}\mathcal{M}\mathcal{G}\mathcal{A})^{\top} \sigma, \end{aligned} \quad (4.40)$$

where

$$\mathcal{S} = \text{diag} \{ (\sigma_{v,k}^2 R_{u,k})_{k=1}^N \}, \quad (4.41)$$

and the equality (4.40) follows from the identity $\text{Tr}(AB) = \text{vec}(A^\top)^\top \text{vec}(B)$. Using a similar argument, we also have

$$\mathbb{E} \|\mathbf{r}\|_\Sigma^2 = \text{vec} [\mathcal{A}^\top (I_{MN} - \mathcal{G})^\top \mathcal{R}_w (I_{MN} - \mathcal{G}) \mathcal{A}]^\top \sigma. \quad (4.42)$$

As for the last term of (4.38), we obtain

$$\begin{aligned} \mathbb{E} [\mathbf{r}^\top \Sigma \mathcal{B}(i) \tilde{\mathbf{w}}(i-1)] &= \mathbb{E} [\text{vec} (\mathbf{r}^\top \Sigma \mathcal{B}(i) \tilde{\mathbf{w}}(i-1))] \\ &= \mathbb{E} [(\mathcal{B}(i) \tilde{\mathbf{w}}(i-1))^\top \otimes \mathbf{r}^\top] \sigma \end{aligned} \quad (4.43)$$

$$\begin{aligned} &= \mathbb{E} [(\mathcal{B}(i) \tilde{\mathbf{w}}(i-1)) \otimes (I_{MN} \cdot \mathbf{r})]^\top \sigma \\ &= [\mathbb{E}(\mathcal{B}(i) \otimes I_{MN}) \mathbb{E}(\tilde{\mathbf{w}}(i-1) \otimes \mathbf{r})]^\top \sigma \end{aligned} \quad (4.44)$$

$$= [\mathcal{B}_I \mathbb{E}(\tilde{\mathbf{w}}(i-1) \otimes \mathbf{r})]^\top \sigma, \quad (4.45)$$

where

$$\mathcal{B}_I = \mathbb{E}(\mathcal{B}(i) \otimes I_{MN}) = \mathcal{B} \otimes I_{MN}. \quad (4.46)$$

We have used the identity $\text{vec}(AXB) = (B^\top \otimes A) \text{vec}(X)$ to obtain (4.43), and (4.44) follows from identity $(AC) \otimes (BD) = (A \otimes C)(B \otimes D)$. Let $\sigma' = \text{vec}(\Sigma')$, so that from (4.39), we have

$$\sigma' = \mathcal{E} \sigma, \quad (4.47)$$

where

$$\begin{aligned} \mathcal{E} &= \mathbb{E} [\mathcal{B}(i)^\top \otimes \mathcal{B}(i)^\top] \\ &= [I_{M^2 N^2} - I_{MN} \otimes \mathcal{M} \mathcal{R}_u - \mathcal{M} \mathcal{R}_u \otimes I_{MN} \\ &\quad + (\mathcal{M} \otimes \mathcal{M}) \mathbb{E}(\mathcal{R}_u(i) \otimes \mathcal{R}_u(i))] \mathcal{G} \mathcal{A} \otimes \mathcal{G} \mathcal{A}. \end{aligned} \quad (4.48)$$

From (4.38), (4.40), (4.42), (4.45), and (4.47), we obtain

$$\mathbb{E} \|\tilde{\mathbf{w}}(i)\|_\sigma^2 = \mathbb{E} \|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{E} \sigma}^2 + [f_a + f_b + f_c(i-1)]^\top \sigma, \quad (4.49)$$

where

$$f_a = \text{vec} \left(\mathcal{A}^\top \mathcal{G}^\top \mathcal{M} \mathcal{S} \mathcal{M} \mathcal{G} \mathcal{A} \right), \quad (4.50)$$

$$f_b = \text{vec} \left[\mathcal{A}^\top (I_{MN} - \mathcal{G})^\top \mathcal{R}_w (I_{MN} - \mathcal{G}) \mathcal{A} \right], \quad (4.51)$$

$$f_c(i-1) = 2\mathcal{B}_I \mathbb{E}(\tilde{\mathbf{w}}(i-1) \otimes \mathbf{r}). \quad (4.52)$$

The following result follows almost immediately.

Lemma 1. *For the time varying vector term, $\mathbb{E}(\tilde{\mathbf{w}}(i-1) \otimes \mathbf{r})$, which appears on the R.H.S. of (4.52), we have the following,*

$$\mathbb{E}(\tilde{\mathbf{w}}(\infty) \otimes \mathbf{r}) = [(I_{MN} - \mathcal{B})^{-1} \otimes I_{MN}] f_b. \quad (4.53)$$

Theorem 4. (Mean-Square Stability) *Suppose that Assumptions 1-2 hold, and the matrix $\mathcal{B}$ is stable, namely, $\rho(\mathcal{B}) < 1$. Then, MAIC is stable in the mean-square sense if and only if the matrix $\mathcal{E}$ is stable, or equivalently, $\rho(\mathcal{E}) < 1$.*

Proof. See Appendix 4.7.2. □

To proceed the steady-state network MSD derivation with matrix $\mathcal{E}$ directly makes the analysis intractable. Therefore, in the following, we find a reasonable alternative for the $\mathcal{E}$ which will be used to derive an approximation for the steady-state network MSD. This approach follows the practice that is commonly adopted in various works like [16, 42, 66] when analyzing mean-square stability of diffusion strategies. Recalling (4.33), we let

$$\begin{aligned} \mathcal{F} &= \mathcal{B}^\top \otimes \mathcal{B}^\top \\ &= [I_{M^2 N^2} - I_{MN} \otimes \mathcal{M} \mathcal{R}_u - \mathcal{M} \mathcal{R}_u \otimes I_{MN} \\ &\quad + (\mathcal{M} \otimes \mathcal{M}) (\mathcal{R}_u \otimes \mathcal{R}_u)] \mathcal{G} \mathcal{A} \otimes \mathcal{G} \mathcal{A}. \end{aligned} \quad (4.54)$$

Comparing (4.48) and (4.54), we see that $\mathcal{E} - \mathcal{F} = \mathcal{O}(\mathcal{M}^2)$ and

$$\mathbb{E} \|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{E}\sigma}^2 = \mathbb{E} \|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{F}\sigma}^2 + \mathcal{O}(\|\mathcal{M}\|^2) \mathbb{E} \|\tilde{\mathbf{w}}(i-1)\|_{\sigma}^2. \quad (4.55)$$

When the recursion (4.49) is stable, i.e., MAIC is mean-square stable, the last

term in (4.55) is negligible when the step-sizes $\{\mu_k\}_{k=1}^N$ are chosen to be sufficiently small since $\mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|^2$ is bounded. Therefore, by adopting the approximation $\mathcal{E} \approx \mathcal{F}$, we have $\mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{E}\sigma}^2 \approx \mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{F}\sigma}^2$, and the following approximate recursion relationship

$$\mathbb{E}\|\tilde{\mathbf{w}}(i)\|_{\sigma}^2 = \mathbb{E}\|\tilde{\mathbf{w}}(i-1)\|_{\mathcal{F}\sigma}^2 + [f_a + f_b + f_c(i-1)]^{\top} \sigma. \quad (4.56)$$

The above recursion (4.56) is mean-square stable if and only if $\mathcal{F}$ is stable, which is achieved if and only if $\mathcal{B}$ is stable as $\rho(\mathcal{F}) = \rho(\mathcal{B})^2$. We have from (4.56),

$$\lim_{i \rightarrow \infty} \mathbb{E}\|\tilde{\mathbf{w}}(i)\|_{(I_{M^2N^2} - \mathcal{F})\sigma}^2 = [f_a + f_b + f_c(\infty)]^{\top} \sigma, \quad (4.57)$$

where

$$f_c(\infty) = 2 [\mathcal{B}(I_{MN} - \mathcal{B})^{-1} \otimes I_{MN}] f_b, \quad (4.58)$$

which is obtained by applying Lemma 1 to (4.52). By choosing

$$\sigma = \frac{1}{N} (I_{M^2N^2} - \mathcal{F})^{-1} \text{vec}(I_{MN}), \quad (4.59)$$

the steady-state network MSD,

$$\zeta = \lim_{i \rightarrow \infty} \frac{1}{N} \mathbb{E}\|\tilde{\mathbf{w}}(i)\|^2, \quad (4.60)$$

can be approximated as

$$\zeta \approx \frac{1}{N} [f_a + f_b + f_c(\infty)]^{\top} (I_{M^2N^2} - \mathcal{F})^{-1} \text{vec}(I_{MN}). \quad (4.61)$$

4.4 Optimizing The Inter-cluster Cooperation

In this section, we formulate optimization problems to obtain the inter-cluster cooperation weights that minimize an upper bound of (4.61). For simplicity, we assume uniform step-sizes, i.e., $\mu_k = \mu$ for all k , throughout this section. We show that this problem is decomposable into local optimization problems at each node,

and then develop an adaptive implementation for our approach.

The averaging rule (4.9) assigns identical weights for inter-cluster cooperation irrespective of the noise and data profiles across the nodes as well as the relationships such as correlation between cluster parameters. Although the MSD performance of clusters whose parameters are highly correlated can be improved by cooperation under large weights, it might not be a good choice for clusters with less correlated parameters to use weights of same values, and is probable to cause performance deterioration (see Section 4.5.2 or examples) which will worsen the network MSD performance in the end. Therefore, our objective is to optimize the inter-cluster cooperation weights in (4.15), so that cooperation among nodes from different clusters can be tailored to improve the network MSD performance.

4.4.1 Formulations of Inter-cluster Weights Optimization

The general idea of our inter-cluster weights selection scheme is to adjust the weights via optimization of (4.61), by taking into account both the noise and data profile of each node and the correlation between different parameters. Due to directly optimizing (4.61) is computationally difficult, we alternatively seek the inter-cluster cooperation weights that minimize a proxy of (4.61). Technically speaking, we drop the term of $f_c(\infty)$ appeared in (4.61), and consider

$$\hat{\zeta} = \frac{1}{N} (f_a + f_b)^\top (I_{M^2N^2} - \mathcal{F})^{-1} \text{vec}(I_{MN}) \quad (4.62)$$

instead. Numerical results in Section 4.5 suggest that this approximation is reasonable since it does not deviate from the actual MSD (4.61) greatly.

It is not difficult to find that when a uniform step-size is applied, i.e., $\mu_k = \mu$ for all k , an upper bound for the $\hat{\zeta}$ in (4.62) is given by

$$c \cdot \text{Tr} \left[\mu^2 \mathcal{A}^\top \mathcal{G}^\top \mathcal{S} \mathcal{G} \mathcal{A} + \mathcal{A}^\top (I_{MN} - \mathcal{G})^\top \mathcal{R}_w (I_{MN} - \mathcal{G}) \mathcal{A} \right]. \quad (4.63)$$

where c is a positive constant (see Appendix 4.7.4 for the derivation). By focusing on the terms that are dependent on the inter-cluster cooperation weights matrix G , (4.63) can be re-written as

$$c \cdot [\text{vec}(\mathcal{G})^\top \mathcal{K} \text{vec}(G) - \kappa^\top \text{vec}(\mathcal{G})], \quad (4.64)$$

where

$$\begin{aligned}\mathcal{K} &= [\mathcal{A}\mathcal{A}^\top \otimes (\mu^2 \mathcal{S} + \mathcal{R}_w)], \\ \kappa &= 2 \text{vec} [\mathcal{R}_w \mathcal{A}\mathcal{A}^\top].\end{aligned}$$

Let $y = \text{vec}(\mathcal{G})$, the optimal inter-cluster cooperation weights can be obtained by solving the following centralized optimization problem, which is a quadratic programming problem [141]:

$$\begin{aligned}(\text{P1}) \quad & \min_y \quad y^\top \mathcal{K} y - \kappa^\top y \\ \text{s.t.} \quad & g_{\ell k} \geq 0, \quad \sum_{\ell \in \mathcal{N}_{I,k}^+} g_{\ell k} = 1, \quad g_{\ell k} = 0, \text{ if } \ell \notin \mathcal{N}_{I,k}^+.\end{aligned}$$

Although the centralized method can lead to better optimization results in general, having a centralized controller to frequently collect network-wise information and then compute the inter-cluster cooperation weights could be less practical for some IoT networks. Therefore, in the sequel, we turn to find a distributed optimization procedure based on a different upper bound of (4.61) (see Appendix 4.7.4):

$$\bar{c} \cdot \text{Tr} [\mu^2 \mathcal{G}^\top \mathcal{S} \mathcal{G} + (I_{MN} - \mathcal{G})^\top \mathcal{R}_w (I_{MN} - \mathcal{G})], \quad (4.65)$$

where $\bar{c}$ is a positive constant. To obtain the bound (4.65), we assume that the intra-cluster combination matrix A is symmetric and doubly-stochastic¹. Although this additional assumption looks restrictive, combination rules like the *Metropolis rule* (see Section 4.5) [44] which leads to symmetric and doubly-stochastic combination matrices are computationally convenient and usually deliver satisfactory performance compared with non-cooperative strategies [45]. Ignoring all terms in (4.65) independent of the inter-cluster weights $\{g_{\ell k}\}$, we obtain

$$\mu^2 \sum_{k=1}^N \left\{ \sum_{\ell=1}^N \left[\mu^2 g_{\ell k}^2 \sigma_{v,\ell}^2 \text{Tr}(R_{u,\ell}) - 2g_{\ell k} \text{Tr}(R_{w,\ell k}) + g_{\ell k} \sum_{m=1}^N g_{mk} \text{Tr}(R_{w,\ell m}) \right] \right\}, \quad (4.66)$$

where

$$R_{w,\ell k} = \mathbb{E} \mathbf{w}_\ell^\circ \mathbf{w}_k^{\circ\top}, \quad (4.67)$$

¹A doubly stochastic matrix X satisfies $X^\top \mathbf{1}_N = X \mathbf{1}_N = \mathbf{1}_N$.

which is the (ℓ, k) -th block of matrix $\mathcal{R}_w$. To optimize (4.66), we can decompose it into N separate local optimization problems for each node k as follows:

$$\begin{aligned} \text{(P2)} \quad & \min_{q_k} \quad q_k^\top (\Omega_{a,k} + \Omega_{b,k}) q_k - 2h_k^\top q_k \\ \text{s.t.} \quad & \mathbf{1}_{n_k}^\top q_k = 1, \quad q_k \geq 0_{n_k}. \end{aligned}$$

$$q_k = \text{col} \left\{ (g_{\ell k})_{\ell \in \mathcal{N}_{I,k}^+} \right\}, \quad (4.68)$$

$$h_k = \text{col} \left\{ (\text{Tr}(R_{w,\ell k}))_{\ell \in \mathcal{N}_{I,k}^+} \right\}, \quad (4.69)$$

$$\Omega_{a,k} = \text{diag} \left\{ (\mu^2 \sigma_{v,\ell}^2 \text{Tr}(R_{u,\ell}))_{\ell \in \mathcal{N}_{I,k}^+} \right\}, \quad (4.70)$$

$$\Omega_{b,k} = \mathbb{E}[\mathbf{W}_k^\top \mathbf{W}_k], \quad (4.71)$$

$$\mathbf{W}_k = [\mathbf{w}_\ell^\circ]_{\ell \in \mathcal{N}_{I,k}^+}, \quad (4.72)$$

$$n_k = |\mathcal{N}_{I,k}^+|, \quad (4.73)$$

where $[\mathbf{w}_\ell^\circ]_{\ell \in \mathcal{N}_{I,k}^+}$ denotes a matrix whose columns are $\mathbf{w}_\ell^\circ$ for $\ell \in \mathcal{N}_{I,k}^+$. The problem (P2) is a quadratic programming problem, which can be solved independently using standard techniques by each node k [141].

4.4.2 Adaptive Implementation of the Distributed Method

In Section 4.4.1, we have implicitly assumed that the data profiles $\sigma_{v,k}^2 \text{Tr}(R_{u,k})$, for $k = 1, \dots, N$, are known, which may not be the case in practical applications. We now present an online estimation procedure similar to those in [44, 67, 142, 143] to *adaptively* estimate these statistics and update the inter-cluster cooperation weights. For each $\ell \in \mathcal{N}_{C,k}$, let $\hat{\mathbf{x}}_{\ell k}(i)$ be the estimate of $\mu^2 \sigma_{v,\ell}^2 \text{Tr}(R_{u,\ell})$ made by node k at time instant i . Then $\hat{\mathbf{x}}_{\ell k}(i)$ can be updated recursively by the following moving-average method:

$$\hat{\mathbf{x}}_{\ell k}(i) = \alpha \cdot \hat{\mathbf{x}}_{\ell k}(i-1) + (1-\alpha) \cdot \|\boldsymbol{\psi}_\ell(i) - \mathbf{w}_k(i-1)\|^2, \quad (4.74)$$

where the coefficient α is chosen from $[0, 1]$.

Although in some applications the statistics $\mathcal{R}_w$ can be obtained by probing the underlying environment before the network is deployed, we are interested in

Algorithm 2 MAIC with adaptive update of inter-cluster cooperation weights

- 1: Initialized with
 $w_k(0) = 0_M$, $\hat{\mathbf{x}}_{\ell k}(0) = 0$ for all k , weights $\hat{g}_{\ell k}(0) = 0$.
 - 2: **for** every node k at each time instant $i \geq 1$ **do**
 - 3: $\boldsymbol{\psi}_k(i) = \mathbf{w}_k(i-1) + \mu_k \mathbf{u}_k(i) [\mathbf{d}_k(i) - \mathbf{u}_k(i)^\top \mathbf{w}_k(i-1)]$
 - 4: Receive $\boldsymbol{\psi}_\ell(i)$ from all $\ell \in \mathcal{N}_{C,k}$ and compute:
 $\hat{\mathbf{x}}_{\ell k}(i) = \alpha \cdot \hat{\mathbf{x}}_{\ell k}(i-1) + (1-\alpha) \cdot \|\boldsymbol{\psi}_\ell(i) - \mathbf{w}_k(i-1)\|^2$
 - 5: Receive $\mathbf{w}_\ell(i-1)$ from all $\ell \in \mathcal{N}_{I,k}^+$ and update:
 $\hat{\boldsymbol{\tau}}_{\ell k}(i) = \mathbf{w}_\ell(i-1)^\top \mathbf{w}_k(i-1)$
 $\widehat{\mathbf{W}}_k(i) = [\mathbf{w}_\ell(i-1)]_{\ell \in \mathcal{N}_{I,k}^+}$
 - 6: Update the following quantities:
 $\hat{\mathbf{h}}_k(i) = \text{col} \left\{ (\hat{\boldsymbol{\tau}}_{\ell k}(i))_{\ell \in \mathcal{N}_{I,k}^+} \right\}$
 $\hat{\boldsymbol{\Omega}}_{a,k}(i) = \text{diag} \left\{ (\hat{\mathbf{x}}_{\ell k}(i))_{\ell \in \mathcal{N}_{I,k}^+} \right\}$
 $\hat{\boldsymbol{\Omega}}_{b,k}(i) = \widehat{\mathbf{W}}_k(i)^\top \widehat{\mathbf{W}}_k(i)$
 - 7: Solve (P2) with quantities
 h_k , $\Omega_{a,k}$, and $\Omega_{b,k}$ replaced by
 $\hat{\mathbf{h}}_k(i)$, $\hat{\boldsymbol{\Omega}}_{a,k}(i)$, and $\hat{\boldsymbol{\Omega}}_{b,k}(i)$, respectively
 to obtain the solution $\hat{q}_k(i) = \text{col} \left\{ (\hat{g}_{\ell k}(i))_{\ell \in \mathcal{N}_{I,k}^+} \right\}$
 - 8: Perform inter-cluster cooperation:
 $\boldsymbol{\phi}_k(i) = \sum_{\ell \in \mathcal{N}_{I,k}^+} \hat{g}_{\ell k}(i) \boldsymbol{\psi}_\ell(i)$
 - 9: Perform intra-cluster combination:
 $\mathbf{w}_k(i) = \sum_{\ell \in \mathcal{N}_{C,k}} a_{\ell k} \boldsymbol{\phi}_\ell(i)$
 - 10: **end for**
-

the cases that the $\mathcal{R}_w$ is not accessible and therefore both $R_{w,\ell k}$ and $\text{Tr}(R_{w,\ell k})$ are unknown for all ℓ and k . This is more practical because the correlation information across the parameters may not be time-invariant, hence the network should be able to adapt to changes in $\mathcal{R}_w$. For realizing the adaptability over $\mathcal{R}_w$, at time instant i , each node uses

$$\hat{\boldsymbol{\tau}}_{\ell k}(i) = \mathbf{w}_\ell(i-1)^\top \mathbf{w}_k(i-1) \quad (4.75)$$

as an instantaneous approximation for $\text{Tr}(R_{w,\ell k})$. Finally, the matrix $\Omega_{b,k}$ can be approximated at each time instant i by

$$\hat{\boldsymbol{\Omega}}_{b,k}(i) = \widehat{\mathbf{W}}_k(i)^\top \widehat{\mathbf{W}}_k(i), \quad (4.76)$$

where $\widehat{\mathbf{W}}_k(i) = [\mathbf{w}_\ell(i-1)]_{\ell \in \mathcal{N}_{I,k}^+}$. Having $\hat{\mathbf{x}}_{\ell k}(i)$, $\hat{\boldsymbol{\tau}}_{\ell k}(i)$, and $\hat{\boldsymbol{\Omega}}_{b,k}(i)$ obtained by

(4.74), (4.75), and (4.76) respectively, the distributed weights optimization method proposed in Section 4.4.1 can then be implemented adaptively as summarized in Algorithm 2.

The computational complexity towards solving (P2) will depend on the specific solver used, which is difficult to characterize explicitly. However, even for a rough estimate, it is still worth studying the computational complexity of Algorithm 2 without considering the procedure of solving (P2). Let n_k be the number of neighbors of node k in its inter-cluster neighborhood as defined in (4.73), and m_k be the number of neighbors of node k in its intra-cluster neighborhood, the number of multiplications and additions involved in Algorithm 2 is $(n_k^2 + 5n_k + m_k + 2)M + 1$ and $(n_k^2 + 5n_k + m_k)M - n_k^2 - 2n_k$ respectively.

4.5 Simulation Results

In this section, we provide examples to compare the network MSD performance of MAIC to those of some other strategies in the literature. The network we tested consists of 10 nodes as depicted in Fig. 4-2a. The clusters C_1 , C_2 , and C_3 are denoted by pink squares, green circles, and blue triangles, respectively. The *Metropolis rule* is adopted because it results in a symmetric doubly-stochastic intra-cluster combination matrix, and has low complexity and good performance in terms of both convergence rate and steady-state MSD for single-task diffusion networks [16]. The *Metropolis rule* has intra-cluster combination weights given by

$$a_{\ell k} = \begin{cases} \frac{1}{\max\{|\mathcal{N}_{C,k}|, |\mathcal{N}_{C,\ell}|\}}, & \text{if } k \neq \ell \text{ and } \ell \in \mathcal{N}_{C,k}, \\ 1 - \sum_{\ell \in \mathcal{N}_{C,k} \setminus \{k\}} a_{\ell k}, & \text{if } k = \ell, \\ 0, & \text{otherwise.} \end{cases} \quad (4.77)$$

All the experiment results are averaged over 10,000 Monte-Carlo runs.

The performance of the following strategies will be compared in three different experiments in the sequel:

- (i) MAIC with the inter-cluster cooperation weights optimized by the centralized method in Section 4.4.1 by solving (P1).

(a) Topology of tested network

(b) Data profile of nodes

Figure 4-2: (a) Data and noise profile of each node. (b) The network topology. The numbers next to the nodes are node indices.

- (ii) MAIC with the inter-cluster cooperation weights optimized by the distributed method in Section 4.4.1 by solving (P2).
- (iii) MAIC with inter-cluster cooperation weights selected adaptively by Algorithm 2.
- (iv) MDLMS proposed by [42] with the cooperation weights selected by the averaging rule (4.9).
- (v) MDLMS with adaptive regularization proposed in [121].
- (vi) The conventional ATC strategy without inter-cluster cooperation. In this case, each cluster acts as an independent subnetwork that performs ATC

(a) $s_v = 0.01^2$

(b) $s_v = 0.03^2$

Figure 4-3: Distributions of the cluster parameters $\{\mathbf{w}_{C_p}^\circ\}_{p=1}^3$ under two different settings of the coefficient s_v .

using the Metropolis combination rule. In [144], a clustering strategy using adaptive adjustment of the inter-cluster cooperation weights was proposed, which leads to no cooperation between clusters with different parameters if the cluster parameters are known a priori and sensor measurements are noiseless. This comparison benchmark is then equivalent to the ideal case in [144], and is used to avoid including errors introduced by the clustering strategy.

4.5.1 Illustrative Examples

The cluster parameters are chosen to be $M \times 1$ random parameters with $M = 2$. The mean of each random cluster parameter is set to be $\bar{w} = 0.7 \times \mathbf{1}_M$, and the correlation matrix of the cluster parameters is given by

$$\begin{bmatrix} I_M & 0.9I_M & 0.5I_M \\ 0.9I_M & I_M & 0.5I_M \\ 0.5I_M & 0.5I_M & I_M \end{bmatrix}. \quad (4.78)$$

The zero-mean regressor $\mathbf{u}_k(i)$ has covariance $R_{u,k} = \sigma_{u,k}^2 I_M$. Fig. 4-2(a) shows $\sigma_{u,k}^2$ and the noise variances $\sigma_{v,k}^2$. The covariance matrix of each parameter $\mathbf{w}_k^\circ$ is $s_v \sigma_{w,k}^2 I_M$. We vary s_v for different experiments while keeping $\sigma_{w,k}^2$ fixed as shown in Fig. 4-2(a). We can compute the (ℓ, k) -th block of matrix $\mathcal{R}_w$, i.e., $\mathcal{R}_{w,\ell k} = s_v \gamma_{\ell k} \sigma_{w,\ell} \sigma_{w,k} I_M + \bar{w} \bar{w}^\top$, where $\gamma_{\ell k}$ is the correlation coefficient between $\mathbf{w}_\ell^\circ$ and $\mathbf{w}_k^\circ$, shown as the scalar multiplier in the (ℓ, k) block entry in (4.78).

In this simulation, we set $s_v = 0.01^2$, and the realizations of the cluster parameters $\{\mathbf{w}_{C_p}^\circ\}_{p=1}^3$ are shown in Fig. 4-3(a), in which the center red bars and black crosses are the medians and means of realizations of each parameter respectively, the boxes' upper and lower edges envelope 50 percent of realizations of each parameter, red crosses outside the upper and lower black bars represent outliers. The step-size is set to $\mu = 0.05$ for all the nodes. The learning coefficient is set to $\alpha = 0.7$ for Algorithm 2. To make a fair comparison, we make the assumption that the cluster information is known a priori for the adaptive regularization method in [121] such that node clustering is not needed and the regularization is now imposed on the inter-cluster information exchange only. Fig. 4-4(a) shows that the MAIC strategies with optimized inter-cluster cooperation weights (i.e., strategy (i), (ii), and (iii)) achieve lower steady-state MSDs than MDLMS with cooperation weights selected by the averaging rule given in (4.9), and MDLMS with adaptive regularization. The regularization weight used is chosen to be $\eta = 1$, which minimizes the steady-state network MSD of MDLMS. In Fig. 4-4(a) and (b), we can also see that the approximate theoretical steady-state network MSDs computed using (4.62) with inter-cluster cooperation weights obtained by the centralized optimization (P1) and the distributed optimization (P2) match well with the simulated

(a) Network MSD performance comparison

(b) MSD model validation

Figure 4-4: (a) Network MSD performance comparison amongst different strategies (i)-(vi). (b) Plots of simulated MSDs, theoretical MSDs, and approximate theoretical MSDs.

MSDs, and do not differ significantly from the theoretical steady-state MSDs obtained from (4.61). Particularly, in Fig. 4-4(b) two experiment settings of s_v were used: comparison with $s_v = 0.01^2$ is shown in the upper subplots and that with $s_v = 0.03^2$ in the lower subplots.

In addition, we also compare the tracking performance of each strategy in a non-stationary environment where the means and correlations of the cluster parameters change at the time instants $i = 250m$, for $m = 1, 2, 3$. Specifically, we set $\bar{w} = 0.8$

Figure 4-5: Network MSD comparison in the non-stationary case.

and (4.78) as the correlation matrix for $1 \leq i < 250$ and $750 \leq i \leq 1000$, and

$$\begin{cases} \bar{w} = 0.6, \gamma_{12} = \gamma_{23} = 0.5, \gamma_{13} = 0.1, & \text{for } 250 \leq i < 500, \\ \bar{w} = 1.2, \gamma_{12} = \gamma_{23} = \gamma_{13} = 0.1, & \text{for } 500 \leq i < 750. \end{cases}$$

The regularization weight is now set to $\eta = 12$ for both MDLMS with averaging rule and MDLMS with adaptive regularization. From Fig. 4-5 we see that although MDLMS with adaptive regularization improves over its counterpart with stationary weights chosen by the averaging rule, the proposed MAIC strategies with optimized weights still achieve lower steady-state network MSDs. This is because the adaptive regularization method still needs to tune the weight η that governs the total degree of inter-cluster cooperation as in the conventional MDLMS [42]. Therefore the improvement by MDLMS with adaptive regularization over MDLMS with averaging rule is limited.

4.5.2 Benefits of Optimized Inter-cluster Cooperation

In this simulation, we set $M = 1$ and the means of the cluster parameters $\{\mathbf{w}_{C_p}^o\}_{p=1}^3$ are $\bar{w} = 1$. The correlation matrix of the cluster parameters is

$$\begin{bmatrix} 1 & \gamma_{12} & 0.5 \\ \gamma_{21} & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix},$$

where the correlation coefficients $\gamma_{12} = \gamma_{21}$. The noise variances $\sigma_{v,k}^2$ across all nodes are drawn uniformly from $[-15, -5]$ dB. Here, we use $s_v = 0.03^2$, step-size $\mu = 0.1$, $\eta = 5$, and all other conditions remain unchanged as in Section 4.5.1. We first fix $\gamma_{12} = 0.9$. From Fig. 4-6, it can be observed that MDLMS using the averaging rule as well as MDLMS with adaptive regularization lead to a deterioration in the steady-state MSD performance of C_2 and C_3 compared to the no inter-cluster cooperation ATC, although the overall network performance is improved (see Fig. 4-7(a)). However, the MAIC strategies with optimized weights are able to achieve better steady-state MSDs for all clusters compared to the no inter-cluster cooperation ATC. This clearly shows the benefit of the inter-cluster weights selection scheme proposed in Section 4.4. Next, we let the correlation coefficient γ_{12} range from 0.1 to 0.9 to examine how much improvement is achieved against ATC without inter-cluster cooperation by different strategies. As shown in Fig. 4-7, as γ_{12} increases, i.e., the parameters $\mathbf{w}_{C_1}^\circ$ and $\mathbf{w}_{C_2}^\circ$ become more correlated, the MSD gains of the MAIC strategies with optimized weights become larger, whereas the MSD gain of MDLMS with the averaging rule and adaptive regularization do not vary significantly.

4.5.3 Performance Comparison under Different Means

In the next simulation, we compare the performance of each strategy when different cluster parameters have different means. We set $s_v = 0.03^2$ and keep all the other settings the same as in Section 4.5.1. Choose a $\delta_{\bar{\mathbf{w}}} \in (0, 1)$. The mean vectors of the three clusters are set to be $\bar{\mathbf{w}}_{C_1} = (1 - \delta_{\bar{\mathbf{w}}}) \times \mathbf{1}_M$, $\bar{\mathbf{w}}_{C_2} = \mathbf{1}_M$, and $\bar{\mathbf{w}}_{C_3} = (1 + \delta_{\bar{\mathbf{w}}}) \times \mathbf{1}_M$. When $\delta_{\bar{\mathbf{w}}} = 0.06$ is small, it can be observed from Fig. 4-8(a) that the MSD of each multitask diffusion strategy is still better than that for ATC without inter-cluster cooperation. When $\delta_{\bar{\mathbf{w}}} = 0.3$, Fig. 4-8(b) shows that the MSDs for all strategies except MAIC with weights obtained by centralized optimization, are worse than the MSD for ATC without inter-cluster cooperation. This is expected since in this case the cluster parameters are on average more different from each other, and inter-cluster cooperation introduces errors into the diffusion process instead of improving the MSD performance. However for MAIC with centralized optimization, the main diagonal entries of G become dominant and the off-diagonal entries tend to zero

(a) Performance of cluster C_2

(b) Performance of cluster C_3

Figure 4-6: Network MSD of clusters (a) C_2 and (b) C_3 , when the regularization weight of MDLMS is $\eta = 5$.

(a) Network performance

(b) Network MSD gains

Figure 4-7: Network MSD (a) and comparison of steady-state network MSD gains with respect to ATC with no inter-cluster cooperation (b) when the regularization weight of MDLMS is $\eta = 5$.

(a) Network MSD under small mean differences

(b) Network MSD under large mean differences

Figure 4-8: Network MSD performance when differences between parameters are small (a) and large (b).

as $\delta_{\bar{w}}$ becomes large. This allows MAIC to decrease the degree of inter-cluster cooperation to avoid estimation bias so that its performance is similar to that of ATC without inter-cluster cooperation.

4.6 Summary

We have proposed a multitask diffusion strategy that performs adaptation before inter-cluster cooperation. We showed by error behavior analysis that this new strategy has the desirable property that mean stability can be achieved regardless of the inter-cluster cooperation weights, which allows these weights to be adjusted without compromising the network stability. We also proposed a centralized as well as distributed method to optimize the inter-cluster cooperation weights in order to improve the steady-state network MSD. Finally, we presented an adaptive implementation, which enables the network to update the inter-cluster cooperation weights according to changes in the noise and data profile as well as the cluster parameter statistics.

4.7 Proofs

4.7.1 Proof of Theorem 3

Iterating (4.32) from the time instant $i = 1$ and then letting $i \rightarrow \infty$ on both sides gives

$$\lim_{i \rightarrow \infty} \mathbb{E} \tilde{\mathbf{w}}(i) = \lim_{i \rightarrow \infty} \mathcal{B}^i \mathbb{E} \tilde{\mathbf{w}}(0) + \lim_{i \rightarrow \infty} \left(\sum_{j=0}^{i-1} \mathcal{B}^j \right) r \quad (4.79)$$

if and only if the spectral radius of the matrix $\mathcal{B}$, $\rho(\mathcal{B})$, is less than one. We have

$$\begin{aligned} \|\mathcal{B}\|_{b,\infty} &\leq \|\mathcal{A}^\top \mathcal{G}^\top\|_{b,\infty} \cdot \|I_{MN} - \mu \mathcal{R}_u\|_{b,\infty} \\ &= \|I_{MN} - \mu \mathcal{R}_u\|_{b,\infty}, \end{aligned} \quad (4.80)$$

where the last equality follows since both A and G are left stochastic matrices and $\|\mathcal{A}^\top \mathcal{G}^\top\|_{b,\infty} = \|\mathcal{A}^\top G^\top\|_\infty = 1$ from Lemma D.4 of [16]. Therefore, if

$$\|I_{MN} - \mu \mathcal{R}_u\|_{b,\infty} < 1, \quad (4.81)$$

then $\rho(\mathcal{B}) \leq \|\mathcal{B}\|_{b,\infty} < 1$, and MAIC achieves mean-stability. The step-size condition (4.36) now follows from (4.81).

In addition, when Assumption 3 is satisfied, from (4.34) we have $r = \mathbb{E} \mathbf{r} = 0_{NM}$. From (4.79), we then readily obtain $\lim_{i \rightarrow \infty} \mathbb{E} \tilde{\mathbf{w}}(i) = 0_{NM}$ if $\mathcal{B}$ is stable, which implies that MAIC achieves an asymptotically unbiased estimation, and the proof is complete.

Note that the mean stability condition (4.81) and the resulting condition (4.36) are also valid in the case where the cluster parameters are deterministic. Moreover, if the parameters are deterministic and are the same for every cluster, then it is easy to verify that $r = 0$, and both the proposed MAIC and the multitask diffusion strategy proposed in [42] reduce to the conventional single-task ATC strategy where only one cluster (which is the whole network) exists.

4.7.2 Proof of Theorem 4

Iterating the variance relation (4.49) from $i = 1$ we have

$$\mathbb{E}\|\tilde{\mathbf{w}}(i)\|_{\sigma}^2 = \mathbb{E}\|\tilde{\mathbf{w}}(0)\|_{\mathcal{E}^i\sigma}^2 + (f_a + f_b)^{\top} \left(\sum_{j=0}^{i-1} \mathcal{E}^j \right) \sigma + \sum_{j=0}^{i-1} [f_c(i-j-1)]^{\top} \mathcal{E}^j \sigma. \quad (4.82)$$

Letting $i \rightarrow \infty$, the first and second terms on the R.H.S. of the above equation converges to zero and a finite value, respectively, if and only if $\mathcal{E}^i \rightarrow 0$ as $i \rightarrow \infty$, i.e., the matrix $\mathcal{E}$ is stable. From (4.52), we can rewrite the third term of (4.82) as

$$2 \sum_{j=0}^{i-1} [\mathcal{B}_I \mathbb{E}(\tilde{\mathbf{w}}(i-j-1) \otimes \mathbf{r})]^{\top} \mathcal{E}^j \sigma. \quad (4.83)$$

The matrix $\mathcal{B}_I$ and vector σ in (4.83) have finite entries. In Appendix 4.7.3, we show that $\mathbb{E}(\tilde{\mathbf{w}}(i-1) \otimes \mathbf{r})$ is uniformly bounded if matrix $\mathcal{B}$ is stable. In addition, we have

$$c_{\xi} = \|\mathcal{E}\|_{\xi} < 1, \quad (4.84)$$

for some norm $\|\cdot\|_{\xi}$, if matrix $\mathcal{E}$ is stable. Therefore, by applying the Cauchy–Schwarz inequality to each term of the sum in (4.83), and by norm equivalence we have

$$\left| [\mathcal{B}_I \mathbb{E}(\tilde{\mathbf{w}}(i-j-1) \otimes \mathbf{r})]^{\top} \mathcal{E}^j \sigma \right| \leq a \cdot c_{\xi}^j, \quad (4.85)$$

for some positive constant a . Since $c_{\xi}^j \rightarrow 0$ as $j \rightarrow \infty$, the series,

$$\sum_{j=0}^{i-1} \left| [\mathcal{B}_I \mathbb{E}(\tilde{\mathbf{w}}(i-j-1) \otimes \mathbf{r})]^{\top} \mathcal{E}^j \sigma \right| \quad (4.86)$$

converges as $i \rightarrow \infty$, which implies the absolute convergence of (4.83), and the proof is complete.

4.7.3 Proof of Lemma 1

From the error recursion (4.26), applying the Kronecker product with $\mathbf{r}$ on both sides we have:

$$\tilde{\mathbf{w}}(i) \otimes \mathbf{r} = (\mathcal{B}(i)\tilde{\mathbf{w}}(i-1)) \otimes \mathbf{r} - \mathbf{s}(i) \otimes \mathbf{r} + \mathbf{r} \otimes \mathbf{r}. \quad (4.87)$$

Note that

$$(\mathcal{B}(i)\tilde{\mathbf{w}}(i-1)) \otimes \mathbf{r} = (\mathcal{B}(i) \otimes I_{MN})(\tilde{\mathbf{w}}(i-1) \otimes \mathbf{r}), \quad (4.88)$$

hence taking expectation on the both sides of (4.87) we obtain

$$\mathbb{E}(\tilde{\mathbf{w}}(i) \otimes \mathbf{r}) = \mathcal{B}_I \mathbb{E}(\tilde{\mathbf{w}}(i-1) \otimes \mathbf{r}) + \mathbb{E}(\mathbf{r} \otimes \mathbf{r}). \quad (4.89)$$

Recalling (4.31) and (4.51), and applying the identity $a \otimes a = \text{vec}(aa^\top)$, we find that the last term on the R.H.S. of (4.89) equals to

$$f_b = \text{vec} [\mathcal{A}^\top (I_{MN} - \mathcal{G})^\top \mathcal{R}_w (I_{MN} - \mathcal{G}) \mathcal{A}].$$

Since $\mathcal{B}_I = \mathcal{B} \otimes I_{MN}$, thus $\mathcal{B}_I$ is stable if matrix $\mathcal{B}$ is stable. Therefore, when $\mathcal{B}$ is stable, (4.89) is a BIBO stable recursion with bounded driving term f_b . As a result, $\mathbb{E}(\tilde{\mathbf{w}}(i) \otimes \mathbf{r})$ converges and thus is uniformly bounded. Letting $i \rightarrow \infty$, it is easy to obtain from (4.89) that

$$\mathbb{E}(\tilde{\mathbf{w}}(\infty) \otimes \mathbf{r}) = [(I_{MN} - \mathcal{B})^{-1} \otimes I_{MN}] f_b. \quad (4.90)$$

4.7.4 Upper bounds of Steady-state Network MSD

Since $\mathcal{F}$ is required to be stable to ensure the mean-square stability of (4.56), we have

$$(I_{M^2N^2} - \mathcal{F})^{-1} = \sum_{j=0}^{\infty} \mathcal{F}^j, \quad (4.91)$$

Then substituting (4.54) and (4.91) into (4.62), and applying identities $\text{vec}(AXB) = (B^\top \otimes A) \text{vec}(X)$ and $\text{Tr}(AB) = \text{vec}(A^\top)^\top \text{vec}(B)$, we have

$$\hat{\zeta} = \frac{1}{N} \sum_{j=0}^{\infty} \text{Tr} \left(\mathcal{B}^j \mathcal{X} \mathcal{B}^{\top j} \right), \quad (4.92)$$

where matrix $\mathcal{X}$ is given by

$$\begin{aligned} \mathcal{X} &= \mu^2 \mathcal{A}^\top \mathcal{G}^\top \mathcal{S} \mathcal{G} \mathcal{A} \\ &\quad + \mathcal{A}^\top (I_{MN} - \mathcal{G})^\top \mathcal{R}_w (I_{MN} - \mathcal{G}) \mathcal{A}. \end{aligned} \quad (4.93)$$

From Lemma D.2 of [16], we obtain

$$\begin{aligned} \frac{1}{N} \sum_{j=0}^{\infty} \text{Tr} \left(\mathcal{B}^j \mathcal{X} \mathcal{B}^{\top j} \right) &\leq \frac{b}{N} \cdot \sum_{j=0}^{\infty} \|\mathcal{B}^j\|_{b,\infty} \cdot \|\mathcal{X}\|_{b,\infty} \|\mathcal{B}^{\top j}\|_{b,\infty} \\ &\leq b \cdot \left(\sum_{j=0}^{\infty} \|\mathcal{B}\|_{b,\infty}^{2j} \|\mathcal{X}\|_{b,\infty} \right) \\ &\leq b \cdot \left(\sum_{j=0}^{\infty} \beta^{2j} \|\mathcal{X}\|_{b,\infty} \right) \\ &\leq \frac{b}{1 - \beta^2} \cdot \|\mathcal{X}\|_{b,\infty} \end{aligned} \quad (4.94)$$

$$\leq c \cdot \text{Tr}(\mathcal{X}), \quad (4.95)$$

where we use (4.80) in the third inequality, b and c are positive constants, and $\beta = \|I_{MN} - \mu \mathcal{R}_u\|_{b,\infty} < 1$. The R.H.S. of (4.95) is the bound (4.63) in Section 4.4.1.

Next, we assume that the matrix A is symmetric so that $\mathcal{A}$ is as well. From

the (4.94), we obtain

$$\begin{aligned}
\frac{1}{N} \sum_{j=0}^{\infty} \text{Tr} \left(\mathcal{B}^j \mathcal{X} \mathcal{B}^{\top j} \right) &\leq \frac{b}{1 - \beta^2} \cdot \|\mathcal{X}\|_{b,\infty} \\
&\leq \frac{b}{1 - \beta^2} \cdot \|\mathcal{A}^\top\|_{b,\infty}^2 \cdot \|\mathcal{Y}\|_{b,\infty} \\
&= \frac{b}{1 - \beta^2} \cdot \|\mathcal{Y}\|_{b,\infty} \\
&\leq \bar{c} \cdot \text{Tr}(\mathcal{Y}), \tag{4.96}
\end{aligned}$$

where $\bar{c}$ is a positive constant. In the third inequality we use the fact $\|\mathcal{A}^\top\|_{b,\infty} = 1$, and the matrix $\mathcal{Y}$ is given by

$$\mathcal{Y} = \mu^2 \mathcal{G}^\top \mathcal{S} \mathcal{G} + (I_{MN} - \mathcal{G})^\top \mathcal{R}_w (I_{MN} - \mathcal{G}). \tag{4.97}$$

This is the bound (4.65) in Section 4.4.1.

Chapter 5

Cooperative CNN-based Face Recognition using Low-cost Embedded IoT Devices

5.1 Introduction

In previous Chapter 3 and Chapter 4, we have showed two distributed LMS estimation strategies, the first is able to reduce communication overhead and energy consumption, the second can improve the learning performance and resource utilization efficiency for the IoT networks. LMS algorithms discussed in the previous two chapters can be interpreted as online learning from linear regression models, which is useful for predicating future trends or finding input-output causal relationships such as system identification problems.

In addition to the techniques of learning from linear models, inference and learning using complex nonlinear model such as deep CNNs is another important category of techniques for the IoT applications as mentioned earlier in Section 1.2. Particularly, in this chapter, we will focus on the CNN-based face recognition for the applications like smart building where individuals need to be identified in environments such as classrooms or offices, either for the purpose of security surveillance or being used as the first step for a human behavior understanding system.¹

¹The work in this chapter is part of the results obtained for a research project in the Delta-

Different from works that have been done in the area of CNN-based face recognition which mainly focus on proposing novel CNN architectures or competing accuracy, our objective is realizing end-to-end cooperative CNN-based face recognition pipeline on resource constrained IoT networks that consists of several low-cost smart camera (SC) nodes, each of which is equipped with a embedded computing platform and a camera sensor.

In the proposed cooperative face recognition pipeline, deep features are extracted from light-weight efficient CNN deployed on the SC nodes and then shared to the other peer SCs for fusion and inference. By doing so, the learning performance can be improved compared to the case where each SC perform end-to-end CNN-based face recognition pipeline individually. At the meanwhile, by the proposed cooperative face recognition pipeline, the conventional heavy-duty GPU-based or cloud-based central processor is replaced by low-cost IoT devices, the local cooperation between SCs can avoid the need of sending vast amount of continuous raw image data (video frames) to the remote central processor via wired or wireless communication, and therefore a lot communication bandwidth can be saved, image-to-result response time can be shortened as well.

In the rest of this chapter, we first introduce the overall design of our pipeline in Section 5.2, in Section 5.3 we describe how each module of the whole pipeline is established, in Section 5.4 we show the methodology of implementing the CNN model training and final face recognition, then we demonstrate and discuss the evaluation test results in the Section 5.5 which is followed by a brief summary in Section 5.6.

5.2 Problem Statement and Overall Design

In this chapter, we focus on identifying a group of people (such as a group of students in a class room or a group of staff in an office) by using deep learning-based face recognition techniques. In the underlying face recognition scenario, we assume there are N identities to recognize which are indexed by $i \in \mathcal{I} = \{1, 2, \dots, N\}$, where $\mathcal{I}$ denotes the set of identities of interests. We use M smart camera (SC)

NTU Corporate Lab for Cyber-Physical Systems. This project is still on going.

Figure 5-1: Schematic overview of the proposed pipeline. There are three on-line testing stages: stage 1, 2 and 3, and one off-line stage.

nodes indexed by $k = 1, 2, \dots, M$ to capture the video frames that contains facial images of same identity simultaneously. Each SC node has a camera sensor that is able to capture 1080P (1920×1080) high resolution video frames. Similar to the typical face recognition systems [145], we also use a pipeline that consists of three modules, i.e., face detection, feature extraction, and recognition. Let $y_k = \phi(x_k)$ denote the feature vector extracted from facial images x_k that is cropped from video frames captured by each SC node, where the function $\phi(\cdot)$ denotes the nonlinear mapping from input image space $\mathbb{R}^{m \times m}$ to output feature space $\mathbb{R}^d$ carried out by the selected CNN. To facilitate the face recognition procedure, we incorporate an off-line stage to collect up to p_i facial image data $\{x_{i,j}^b\}$ of each identity i , and store extracted features $\{y_{i,j}^b = \phi(x_{i,j}^b)\}$ for all $j = 1, 2, \dots, p_i$ and $i \in \mathcal{I}$, to setup the feature database $\mathcal{D} = \{Y_i^b | i \in \mathcal{I}\}$, where $Y_i^b = \{y_{i,j}^b | j = 1, 2, \dots, p_i\}$ is the collection of features belongs to i -th identity that are stored in the database (see Fig. 5-1 for the illustration).

The stored features $\{y_{i,j}^b\}$ are obtained by face detection and feature extraction techniques same with on-line stage. During the online stage, recognizing the

identity appears in the video frames is to assign an identity label $i^\circ \in \mathcal{I}$ to the extracted feature y_k such that

$$i^\circ = \arg \max_{i \in \mathcal{I}} s_i(y_k, Y_i^b), \quad (5.1)$$

where $s_i(y_k, Y_i^b)$ denotes similarity among the online extracted feature y_k and features of identity i stored in the database. The similarity can be computed by method like euclidean distance, the specific choice of similarity score will be detailed in the sequel.

In this chapter, we shall not assume a dedicated device to be used as an IoT edge gateway device (EGD). That is to say, any SC node can also play the role of the EGD. We use this setting to enhance the flexibility and robustness of the IoT network for two reasons. First, this setting allow each SC node and also the sensor network to maintain inference ability when the dedicated EGD temporarily out of access as mentioned earlier in the Chapter 1. Second, in some cases, for example SCs are a group of mobile robots carrying cameras for surveillance patrol, it is important that every SC nodes have their own on-board inference ability. For the purpose of saving energy, SC nodes can take turns to act as the EGD, so that the network recovers to a decentralized topology from a distributed topology.

Considering that the field of view of each SC node could be constrained by its geographic locations and the direction that its camera is facing, therefore recognizing the identities of interest by relying solely on the data obtained by each single SC may not provide robust and accurate inference results. This calls for the cooperation among SC nodes to enhance the inference performance. Inspired by spirit of ATC diffusion strategy and related local cooperation among peer neighbors that has been discussed in previous two chapters, we consider to let each SC node process its own collected video frames and extract features, and then share the obtained features with other SC nodes for a feature fusion operation which resembles the combination step of ATC strategy where each node share and combine intermediate estimate from other peer neighbors.

Unlike traditional video analysis pipelines where continuous video streams are directly transmitted to remote central analytic engine, in our pipeline, each SC only transmit one feature vector extracted from the input video frame at a time

via wireless local area network. Typically, a single 1080P RGB image can be as large as a few MBs, whereas a feature vector has dimension less than 1000 is of only a few of KBs normally. Therefore, by utilizing the local computing power and cooperation schemes, the communication resources required for face recognition inference can be significantly reduced compared to centralized solutions.

Moreover, for the real-world implementation, we use Raspberry Pi 3 Model B which has a 4-core 1.2GHz CPU and only 1GB on-board RAM and Logitech c920 webcam as camera sensors to setup our SC nodes. Compared with some other choices like the Nivida Jetson TX series equipped dedicated on-board GPU unit, the cost of a Raspberry Pi-like device is about ten times cheaper, which is more suitable for massive deployment in IoT applications.

To evaluate the impact of computing power on the inference latency, we consider both Raspberry Pi and ordinary desktop PC, to represent and compare the case where features are fused by SC nodes and inference are made by local cooperation, and the case where dedicated powerful EGD is available to fuse features extracted by SC nodes and perform inference, which is only for comparison study.

5.3 Design of the Pipeline Modules

Although the adopted edge/fog computing-based architecture is helpful for reducing the communication overhead, it is essential to select appropriate face detection algorithm and CNN model so as to reduce the latency at each stage and achieve a real-time inference performance. To this end, we evaluate several commonly used methods for face detection as well as some candidate CNN models for face recognition. All the tests were carried out on two different computing platforms:

- a) A desktop PC with a 4-core 3.40GHz Intel i7 CPU and 16GB RAM,
- b) A Raspberry Pi 3 Model B with a 4-core 1.2GHz Broadcom BCM2837 ARM Cortex-A53 CPU and 1GB RAM.

5.3.1 Evaluation of Face Detection Methods

Face detection is the very first step of the entire pipeline, which is critical to the overall performance. There are two mostly applied off-the-shelf program libraries

providing face detection toolkits, namely the OpenCV [146,147] and the Dlib [148]. OpenCV provides a cascade classifiers based on Haar feature and related earlier works [149,150]. In this method, each weak classifier computes multiple Haar features by moving a window whose size is same with the input image, then Haar features are compared to a threshold for obtaining classification results. The final output is the weighted sum of the results from every weak classifiers. The most appealing advantage of Haar feature-based detectors is the fast processing speed since they are insensitive to the image size. On the other hand, Dlib implements face detection function based on the Histogram of Oriented Gradients (HOG) descriptor [151] that is more accurate than Haar feature-based methods. Through local intensity gradients or directions of edges, the HOG features can describe the shape of an object and summarize the histogram of different orientations of gradients within a region on the image [152]. Usually, a SVM classifier will be applied after the HOG features are obtained. However, the HOG-based face detection requires a window to slide over the image repeatedly, from top to bottom and from left to right, to search the most likely area that represents a face. To make things worse, this sliding window scheme can lead to rather long latency if the underlying image is big in size.

We have tested both two methods, using 1080P live video stream as input. Besides, we also found that Dlib is slightly more robust to pose variations and occlusions than OpenCV. However, OpenCV is much faster than Dlib which resulted in a latency even longer than 60 seconds on Raspberry Pi to detect faces over a stationary 1080P image. Thus, no meaningful test results of detection time for Dlib was recorded on Raspberry Pi. Although OpenCV is fast enough for real-time implementation on platforms like Raspberry Pi, it suffers from a less robust detection performance and a relative narrow field of view according to our real-world experiments. That is to say, when the yaw angle of the face in the frame is large, its detection performance will drop substantially.

Therefore, we turned attention to the state-of-the-art CNN-based face detection methods such as [153–155], among which the MTCNN proposed in [155] is most attractive. Inspired by both cascaded classifiers like [149] and accuracy of CNN models for CV applications, MTCNN uses three cascaded light-weight CNNs

Table 5.1: Comparison of Processing Time of Different Face Detection Methods

Method	On PC	On Raspberry Pi
OpenCV	0.03s	0.36s
Dlib	0.25s	N.A.
MTCNN	0.05s	0.44s

for multitask learning of face landmarks and face bounding boxes. As a result, MTCNN is not only more computationally efficient than other CNN-based face detection methods but also can provide joint face detection and alignment with impressive performance. We have also tested MTCNN using open-source python scripts. The results show that it can achieve a similar robustness with Dlib in terms of pose variations, it is even more robust to occlusions than Dlib. Besides, MTCNN is indeed fast enough to be run in real time on Raspberry Pi-like low cost embedded computing platforms. The comparison among OpenCV, Dlib, and MTCNN for processing time per image is summarized in the Table 5.1. To speed up the detection in real time, we have used a scaling factor of 0.25 to shrink camera frames to the one fourth of their original size (recall that under a resolution of 1080P, camera frames are as large as 1920×1080 pixels). In addition, all three methods we tested can provide an effective range² about 4 meters.

5.3.2 Selection of CNN Model

To determine a suitable deep CNN model for performing feature extraction on our SC nodes, we have used stationary image of size 224×224 have been used as the inputs, and tested three representative CNN models to evaluate their processing latencies:

- (i) VGG-16 [131] used by VGG-Face [31];
- (ii) Inception-V3 [156], an evolved version of GoogLeNet [133] used by FaceNet [128]. The Inception-V3 improves against original GoogLeNet about 7.8% in terms of Top-1 error³ of classification tasks, however, at the expense of about

²The effective range means the maximum distance beyond which the rate of successful detection drops significantly

³The term top-1 error rate is used to benchmark the performance of machine learning models in the ImageNet classification competition [24]. The tested model is said correctly classify a given image if the target label coincide with the prediction with highest probability that model has made.

Table 5.2: Comparison Among CNN Models

Model	Top-1 Accuracy	Billion MAC	On PC	On Raspberry Pi
VGG-16	71.5%	15.3	0.37s	N.A.
Inception-V3	69.8%	4.8	0.13s	1.41s
1.0 MobileNet-224	70.6%	0.6	0.08s	0.69s

three times more complex and computation-costly;

- (iii) 1.0 MobileNet-224 reported in [136], it is the most complex and most accurate version among all MobileNets family members.

As mentioned earlier in the Section 2.3, MobileNets are a series of light-weight and highly efficient CNN models, so it is natural to consider MobileNets as the target model to use. However, both FaceNet and VGG Face achieve very good performance for face verification tasks. Besides, they are widely adopted by other researchers in this area [157,158], and there is few literatures show whether Inception-V3 or VGG-16 can also be run smoothly on devices like the Raspberry Pi. Hence, we were motivated to test and compare these models. All three models we tested have been pre-trained by their developers, and we implemented these tests by using Keras [159] which is a popular python library for implementing deep learning pipelines based on Google’s Tensorflow [160] frameworks.

During the tests, it was noticed that the model of VGG-16 is too large in terms of memory size (approximately 550MB) to be implementable on our Raspberry Pi which only has a limited 1GB RAM. The Top-1 accuracy, number of multiply-accumulate (MAC) operations (as are reported in the literatures), as well as our testing results on average latency over 500 forward processing are summarized in the Table 5.2.

From the comparisons, it is clear that the MobileNets are most efficient and suitable to be implemented on embedded systems like Raspberry Pi. However, as is reported in [136], there are two hyper-parameters affect the trade-off between performance and model complexity: one parameter $\alpha \in (0, 1]$ uniformly shrinks the size of input channel of every layer of the model, another parameter $\rho \in \{128, 160, 192, 224\}$ controls the input image resolution. Typically, there are 16 options for the choice of $\{\alpha, \rho\}$ which lead to 16 different variants of MobileNets. Each of the variants has different computational cost and ability of learning deep

Table 5.3: Comparison Among Variants of MobileNets

Model (α, ρ)	Top-1 Acc.	Million MAC	Million Para.	APO Rank.	APP Rank.
(0.25, 128)	41.3%	14	0.47	1	4
(0.25, 160)	46.0%	21	0.47	2	3
(0.25, 192)	49.0%	34	0.47	3	2
(0.25, 224)	50.6%	41	0.47	4	1
(0.5, 128)	56.2%	49	1.34	5	8
(0.5, 160)	59.9%	77	1.34	6	7
(0.75, 128)	61.8%	104	2.59	7	12
(0.5, 192)	62.1%	110	1.34	8	6
(0.5, 224)	64.0%	150	1.34	9	5
(0.75, 160)	65.2%	162	2.59	10	11
(1.0, 224)	70.7%	569	4.24	16	16

features.

To evaluate the model efficiency of achieving relevant Top-1 accuracy, we applied two metrics to each variant of MobileNets: accuracy per operation (APO) and accuracy per parameter (APP) which are given by

$$\begin{aligned}
 \text{APO} &= \frac{\text{Top-1 Accuracy}}{\text{Number of MAC incurred}}, \\
 \text{APP} &= \frac{\text{Top-1 Accuracy}}{\text{Number of Parameters}}.
 \end{aligned} \tag{5.2}$$

The models with top 10 APO ranking together with their accuracy on ImageNet dataset [24], model complexity and APP ranking are given in the Table 5.3.

Recall that our objective is to build a pipeline that can complete the data-to-result processing in real time (i.e., to achieve a throughput around 1.0 FPS). Nevertheless, only the face detection module itself can lead to latencies about 0.5 seconds. Thus, the model 1.0 MobileNet-224 we have tested may not be the best choice. Due to it takes a Raspberry Pi 0.7 seconds to extract a feature vector, on average.

From Table 5.3, we can see that the 0.5 MobileNet-224 is about half computationally complex and three times smaller compared with the baseline variant 1.0 MobileNet-224 (the most accurate one), which means a much faster forward processing speed as well as a less training time. Among all the models listed above, 0.5 MobileNet-224 is the most accurate among the models whose number of operations is less than 150 million and number of parameters is less than 2 million. On

the other hand, it is the second least computationally complex one among all the listed models while being more accurate than the other simpler variants.

Therefore, 0.5 MobileNet-224 has been selected as our target CNN model. It worth mentioning that, since the CNN model functions as the feature extractor in our face recognition pipeline, so we manually removed the last two layers of the 0.5 MobileNet-224, i.e., the fully connected (FC) layer and softmax classifier, and used the output of the last global average pooling layer as the feature vector. Thanks to the removal of FC layer which contains dense parameters, the number of parameter of the resulting model is only 0.83 million which is 38% fewer than the original. We will refer to this resulting CNN model as the 0.5 MobileNet-224-FE.

5.4 The Proposed CNN-based Cooperative Face Recognition

After selected both face detection technique and CNN model for feature extraction, in this section we first show how the 0.5 MobileNet-224-FE was trained as the feature extractor of our face recognition application, then introduce the method adopted for the fusion of features from different SC nodes as well as the algorithm for recognition process.

5.4.1 Training the CNN Model

In order to obtain a well-trained model, it is essential to choose a suitable training dataset by considering both availability and quality. Therefore, the first dataset to consider is the VGG-Face dataset [31] that is constructed by the development team of VGG-Face. This dataset covers 2622 identities, each with 1000 face images. Being trained on this dataset by using softmax loss and triplet-like loss, VGG Face claims an accuracy of 98.95% for face verification over LFW tests. Another quality-reliable public dataset is the CASIA-WebFace [161] which contains 494,414 images for 10,575 identities. It has been used for training a face recognition CNN model [161] that is able to achieve a performance comparable to some other famous deep learning face recognition methods like the DeepFace [162] and DeepID-2 [126].

In comparison, the VGG Face dataset is much deeper since it has much more

Table 5.4: Summary of Training Hyper-parameters

Optimizer	SGD with Nesterov momentum of 0.9
Loss	Cross-entropy loss (Softmax)
Initial learning rate	$\mu(0) = 0.1$
Learning rate scheduling	Fold every 20 epochs
Number of epochs	100
Batch size	64

images for each identity. On the other hand, the CASIA-WebFace is wider due to it covers three times more identities than the VGG dataset, while being four times fewer in terms of total images. Except for the formidable time cost of training on VGG Face dataset with limited GPU and hardware resources, recently some researchers [163] also suggest that using a wider dataset can lead to better robustness and face recognition performance especially when the CNN model being trained is not very deep compared with those super deep models like the ResNet [164]. Given that the models of MobileNets family are relatively shallower and simpler than those gigantic ones, so we have chosen CASIA-WebFace as our training dataset. Moreover, CASIA-WebFace does not overlap with the *de facto* testing dataset LFW, so that it is convenient to implementing the entire training-testing pipeline with “CASIA-WebFace+LFW” framework.

Considering that training a deep model usually takes several days even with powerful GPUs, therefore we firstly used a small portion of the original dataset to tune the hyper-parameters used for the training procedure, so as to shorten the period of trial-and-errors. The small dataset we have used contains 4051 images and 20 identities which are randomly picked from the CASIA-WebFace. After several iterations of trials-and-errors on the small dataset, we figured out a set of proper settings of hyper-parameters (see Table 5.4) to be used for the training on the entire dataset.

5.4.2 Recognition Method

To recognize the identity appears in the scene, we adopted a SoftMax operator proposed in [157] to calculate the similarity score $s_i(y_k, Y_i^b)$ associated with i -th

identity, which is obtained by:

$$s_i(y_k, Y_i^b) = \frac{1}{P} \sum_{\beta=0}^P s_{i,\beta}(y_k, Y_i^b), \quad (5.3)$$

where

$$s_{i,\beta}(y_k, Y_i^b) = \frac{\sum_{j=1}^{p_i} s(y_k, y_{i,j}^b) \exp \{ \beta s(y_k, y_{i,j}^b) \}}{\sum_{j=1}^{p_i} \exp \{ \beta s(y_k, y_{i,j}^b) \}}. \quad (5.4)$$

In the above equation, β is a non-negative hyper-parameter controls the trade-off between the effect of averaging the similarity scores or taking the $\max(\cdot)$ (or $\min(\cdot)$) over them [157]. Multiple values of β will be used to compute $\{s_{i,\beta}(y_k, Y_i^b)\}$, and the final similarity score is averaged over all choice of β . In (5.4), the $s(y_k, y_{i,j}^b)$ is the pairwise similarity score between two features y_k and $y_{i,j}^b$, it is computed by

$$s(y_k, y_{i,j}^b) = \frac{\langle y_k, y_{i,j}^b \rangle}{\|y_k\| \|y_{i,j}^b\|}, \quad (5.5)$$

which also represents the cosine angle between two feature vectors. Besides, the SoftMax operator (5.4) can be regarded as a weighted average where the weight associated to each pairwise score $s(y_k, y_{i,j}^b)$ is proportional to the score itself, i.e., the higher the score $s(y_k, y_{i,j}^b)$, the bigger the weight. The recognition algorithm we adopted using the SoftMax operator-based similarity score is summarized in the Algorithm 3.

Moreover, we choose the above SoftMax operator-based similarity score because the resulting recognition method only needs a relatively small number of features (e.g., 20-30 features) for each identity to be collected before the actual deployment. So that it does not need additional training stage which is usually required by more complex classifiers like the support vector machine or softmax classifiers (with FC layers). As such, once a new identity appears and needs to be recognized, the SC nodes can capture only tens of images of this person in real time, and then append newly obtained features to the existing database. This is particularly convenient for adaptation to dynamic environments in the context of IoT, since training a classifier not only requires sufficiently large data samples to avoid over fitting but also needs more computation resources as well as additional re-training time, which

Algorithm 3 Face Recognition using SoftMax Similarity Score

- 1: **for** each identity class i in the feature library **do**
- 2: **for** each specific hyper-parameter β **do**
- 3: **for** each feature of identity i stored in the database, i.e., $y_{i,j}^b$ **do**
- 4: Calculate pairwise similarity score $s(y^f, y_{i,j}^b)$ using (5.5).
- 5: **end for**
- 6: Calculate similarity score $s_{i,\beta}(y^f, Y_i^b)$ among input feature y^f and the database features of i -th identity using (5.4).
- 7: **end for**
- 8: Find the final similarity score $s_i(y^f, Y_i^b)$ of the input feature associated with identity i using (5.3).
- 9: **end for**
- 10: Return the recognized identity label $i \in \mathcal{I}$ which gives the highest similarity score,

$$i = \arg \max_{i \in \mathcal{I}} s_i(y^f, Y_i^b)$$

makes it lacks on-line adaptation flexibility.

5.4.3 Cooperative Face Recognition by Multi-view Feature Fusion

Recall that each SC node will simultaneously monitor the environment and extract features from camera frames. In the cooperative scheme mentioned in the Section 5.2, these independently extracted features will be sent to other SC nodes for a fusion process prior to the further recognition inference. In the literature, fusion schemes that are commonly adopted for multi-view computer vision scenarios includes the max fusion, averaging fusion, and concatenation fusion [135, 165, 166]. Although concatenation fusion can preserve more information about the object, naively concatenating multiple features without knowing their spatial relationship may worsen the performance. Hence, it calls for on-line pose estimation process to enable carefully designed concatenation methods. However, on-line pose estimation can incur additional processing time, which will further increase the pipeline latency. Therefore, in this chapter, we use both max fusion and averaging fusion via local cooperation among SC nodes and compare their effects on the pipeline performance in Section 5.5. To be more specific, under the max fusion scheme, each SC node $k = 1, 2, \dots, M$ fuse its own feature y_k and features shared by its

neighbors $\{y_\ell | \ell \in \mathcal{N}_k\}$ via the following way

$$y_{k,m}^{f,\max} = \max_{\ell \in \mathcal{N}_k} y_{\ell,m}, \quad (5.6)$$

where $y_{k,m}^{f,\max}$ is the m -th entry of the fused feature vector obtained at SC node k , and $y_{\ell,m}$ is the m -th entry of feature from ℓ -th neighbor of SC node k . Alternatively, the averaging fusion is given as below,

$$y_k^{f,\text{avg}} = \frac{1}{|\mathcal{N}_k|} \sum_{\ell \in \mathcal{N}_k} y_\ell. \quad (5.7)$$

To reduce the dimension of features vectors so that communication overhead can be cut down and inference latency can be shortened, dimension reduction techniques such as principal component analysis (PCA) can be applied to the feature before transmitting them over wireless channels. The effect of applying PCA on the inference performance will be shown in the following sections.

With the feature fusion method and local cooperation introduced, we the resultant cooperative face recognition algorithm is then summarized in Algorithm 3.

5.5 Evaluation and Results

5.5.1 Experiment Setup

For evaluating the performance of the proposed cooperative face recognition pipeline, the orientation of cameras is assumed such that each camera can observe the same person from a different angle like depicted in Fig. 5-2.

We have used the LFW [129] to implement all the tests. Images in the LFW testing dataset are pre-aligned with deep funneling [167]. Moreover, the yaw angles of faces in the LFW images are distributed between -50° and 50° largely [158]. So, in this chapter, we assume that there are $M = 3$ SC nodes for the experiments, and each of which is able to obtain facial image x_k for a same identity such that the yaw angle of the face appears in x_k belongs to a set of interval

Figure 5-2: Illustration of camera configuration used in cooperative face recognition: three cameras monitor same region from different angles and geometry positions.

$\Theta = \{\theta_{+2}, \theta_{+1}, \theta_0, \theta_{-1}, \theta_{-2}\}$, where we set

$$\begin{cases} \theta_{-2} &= [-60^\circ, 30^\circ), \\ \theta_{-1} &= [-30^\circ, 15^\circ), \\ \theta_0 &= [-15^\circ, 15^\circ], \\ \theta_1 &= (15^\circ, 30^\circ], \\ \theta_2 &= (30^\circ, 60^\circ). \end{cases} \quad (5.8)$$

We noticed that in the real-world scenarios, the yaw angle of a face in x_k may exceed 60° or -60° . In this situation, it is difficult for face detection to perform consistently, so that the SC node with that camera might not be able to extract feature sometimes, due to no face in the camera frame can be detected. Because we adopt a cooperative face recognition scheme where multiple cameras are placed at different geometry position and configured to monitor same region, so that even some of SC nodes fail to provide features, the rest can still work normally to guarantee the robustness of the entire face recognition pipeline by sharing the feature vectors via local cooperation, which clearly shows the advantages of such

cooperative scheme. Nevertheless, in this chapter, for the ease of evaluation, we shall proceed our experiments with the assumption that each SC node can observe a face with yaw angle falls in any element in Θ . In addition, since there is only three SC nodes, we assume a fully-connected distributed topology where each SC node has access to features obtained by the other SC nodes.

Let $\mathcal{X}$ be the set of input frames or images, and let $g : \mathcal{X} \mapsto \Theta$ map an input frame or image to the interval into where the yaw angle of the underlying face exhibits. With the above settings, we further supposed that given a triplet of facial images $\{x_1, x_2, x_3\}$ of same identity, the triplet $\{g(x_k)|k = 1, 2, 3\}$ is either $\Theta^{\text{left}} = \{\theta_0, \theta_{-1}, \theta_{-2}\}$, or $\Theta^{\text{frontal}} = \{\theta_{+1}, \theta_0, \theta_{-1}\}$, or $\Theta^{\text{right}} = \{\theta_{+2}, \theta_{+1}, \theta_0\}$, which corresponds to the cases where the identity faces to the camera on the left, in the front, and on the right, respectively.

Moreover, the LFW dataset [167] contains 5749 identities and 13,233 images of size 250×250 pixels in total. In order to setup our off-line feature database $\mathcal{D}$, the original LFW dataset needs to be split into two disjoint parts $\mathcal{P}$ and $\mathcal{Q}$, where $\mathcal{P}$ is the gallery contains images for on-line testing and $\mathcal{Q}$ is the gallery contains images retained for off-line database. In total, p_i images will be picked from identity i for the $\mathcal{Q}$. We have used a uniform scheme where $p_i = p, \forall i \in \mathcal{I}$ for all experiments in this chapter. However, most of the identities in LFW only have a few number of images (less than ten images), which makes it difficult to split the dataset. Therefore, we manually filtered the original LFW such that identities have a number of images less than 30 were excluded. After this manipulation, we obtained a dataset contains 34 identity and 2,370 images. Although the number of identities is much fewer than the original, it is in line with the number of staffs or students that one can expect in environments like classrooms or offices, which still well fits our problem background.

To be specific, we randomly picked $p = 20$ images from every identity for the gallery $\mathcal{Q}$ to setup our off-line feature database $\mathcal{D}$, and use the reset of images for the gallery $\mathcal{P}$. Similar to the cross validation method usually used for model training procedure, we repeat this data split process for ten times, so that ten random splits $\{\mathcal{P}_n, \mathcal{Q}_n|n = 1, 2, \dots, 10\}$ were obtained for ten independent experiments, and $\{\mathcal{P}_n, \mathcal{Q}_n\}$ differ from $\{\mathcal{P}_m, \mathcal{Q}_m\}$ for any $n \neq m$ and $n, m = 1, 2, \dots, 10$.

Figure 5-3: Example of how testing image triplets are selected. From top to bottom: three identities $i = 1, 2, 3$, face left camera, frontal camera, and right camera, respectively.

Table 5.5: A Summary of Experiment Setups

Num. of identity	$N = 34$
Num. of features per id. in database	$p = 20$
Num. of testing samples per experiment	500 image triplets (1500 images)
Num. of independent experiments	10

To better simulate the real-world scenarios, we also used OpenCV to apply pose estimation to $\{\mathcal{P}_n\}$ for all $n = 1, 2, \dots, 10$, to classify images of every identity into five categories according to the yaw angle intervals in Θ . Then, we randomly generated 500 tests for every $\mathcal{P}_n, n = 1, 2, \dots, 10$. For each of the tests, we randomly picked an image triplet $\{x_{i,1}, x_{i,2}, x_{i,3}\}$ from same identity i in $\mathcal{P}_n$, such that $\{g(x_{i,1}), g(x_{i,2}), g(x_{i,3})\}$ is either Θ^{left} , or Θ^{frontal} , or Θ^{right} . That is to say, the image triplet was chosen as if the identity in the images faces either the left, frontal, or right camera (see Fig. 5-3).

5.5.2 Recognition Accuracy

The experiment setups described in the previous section are summarized in the Table 5.5. In the experiments, once a face was detected by MTCNN, a patch of size 224×224 were then cropped from the bounding box of the face and sent into 0.5 MobileNet-224-FE to extract the feature vector of dimension $d = 512$.

We ran through all the ten independent experiments by using split testing datasets $\{\mathcal{P}_n, \mathcal{Q}_n | n = 1, 2, \dots, 10\}$. For each individual run, we used 500 testing samples, i.e., 500 image triplets of same identity that were randomly drawn from $\mathcal{P}_n$. The recognition accuracy was computed as the ratio of number of correct recognition against the total number of tests which is same to the definition of Top-1 accuracy. The final results were then averaged over all ten experiments, the resulting recognition accuracies are given in the Table 5.6 where we use the term standard pipeline to refer to the one consists of MTCNN face detection, 0.5 MobileNet-224-FE, and Algorithm 3 without dimension reduction.

We also carried out testings for stand alone case for comparison where each SC node perform the inference individually without sharing feature vectors. In this setting, the inference accuracy is only 54.1% on average and is lower than the Top-1 classification accuracy of 0.5 MobileNet-224, which is expected due to in stand

Table 5.6: The Test Results on Recognition Accuracy

Tested method	Max Fusion	Averaging Fusion
Standard pipeline	64.8%	60.9%
Standard pipeline with PCA ($d = 128$)	65.4%	61.7%
Standard pipeline with PCA ($d = 256$)	66.9%	62.2%

alone case without cooperation by sharing features, it is relatively hard for a single SC to recognize an identity accurately when side faces are present to the camera (i.e. the yaw angle is very large). On the contrary, from the Table 5.6, we can see that no matter which fusion method is used and no matter if PCA is applied or not, the accuracies obtained by cooperative face recognition approach are higher than that obtained in the stand alone case, which clearly shows the value of the local cooperation by sharing features among SC nodes. Moreover, the max fusion (5.6) outperforms averaging fusion in accuracy(5.7). This is because max fusion might keep some of the most important features so that the fused feature can be more discriminative although the averaging fusion may be able to reduce the noise embedded in the feature vectors. Besides, it is interesting that using PCA for dimension reduction can not only reduce the communication overhead but is also beneficial for improving the recognition accuracy. The highest accuracy achieved among all the different settings was 66.9% which resulted from the max fusion and dimension $d = 128$, which can save 75% communication overhead compared to the case where dimension of features is $d = 512$ without using PCA.

However, it could be expected that the recognition accuracies obtained in Table 5.6 cannot rival those obtained by sophisticated CNN-based face verification pipelines like VGG Face, DeepIDs, and FaceNet, etc., thanks to our focus in this chapter is to address real-time inference problem using only light-weight CNN models and low-cost IoT devices with limited computing and storage resources rather than developing performance-centric complex CNN models and using memory-hungry expensive GPUs. Besides, it is noticed that the original 0.5 MobileNet-224 model achieves a Top-1 classification accuracy of 64.4% on the ImageNet, which also shows the trade-off between deep learning ability and computation efficiency.

Besides, in Table 5.7, identification accuracy, and data volume that would be incurred for face recognition is compared between the proposed pipeline and a

Table 5.7: Comparison with Cloud-based Solution

Pipeline method	Accuracy(LFW)	Data Volume
Cloud (VGG-based)	74.1% [33]	4055KB(per raw image)
Proposed ($d = 256$)	66.9%	4.59KB(per feature vector)

cloud-based solution where VGG Face is selected since the memory size of its model is too big to be deployed on embedded IoT devices like Raspberry Pi. From Table 5.7 we can see that the accuracy of our pipeline is still comparable to VGG, considering that the training dataset used by VGG Face is nearly 5 times larger than the one used to train our CNN model. On the other hand, for each image to be processed for recognition task, the data volume which could be incurred in our pipeline is about 900 times less than that in the cloud solution where 1080P raw images are uploaded from front-end camera sensor, which shows that our pipeline has great potential for reducing communication overhead.

5.5.3 Pipeline Latency

In addition to the recognition accuracy, processing latency of the proposed pipeline is also evaluated. Just like what we have done for other processing speed evaluations in previous sections, face recognition inference has also been implemented on both SC node (i.e., the Raspberry Pi) and desktop PC to evaluate and compare the resultant latency. We have tested several different settings for implementing the face detection module, feature extraction module, and recognition module. The results on processing latencies of tested implementations are summarized in the Table 5.8, where the time for face detection refers to the results reported in Section 5.3.1 which are obtained using live video streams as input.

By combining processing time of each module, the latency of the entire pipeline can then be obtained. From Table 5.8, the shortest image-to-feature latency on Raspberry Pi is 0.727 seconds, which was achieved by using OpenCV as face detection method. However, it can be expected that OpenCV is less robust to pose variations so that the overall pipeline performance would be worse than using MTCNN which is only 0.075 seconds slower than OpenCV on Raspberry Pi.

On the other hand, it is obvious that PCA is also quite effective for reducing computational complexity and processing latency. When applying PCA with $d =$

Table 5.8: Test Results on Processing Latency

Pipeline Module	Method Setting	PC	Raspberry Pi
Face detection	OpenCV	0.033s	0.361s
Face detection	MTCNN	0.050s	0.436s
Feature extraction	0.5 MobileNet-224-FE	0.071s	0.366s
Recognition	Algorithm 3 no PCA	0.067s	0.311s
Recognition	Algorithm 3 PCA ($d = 256$)	0.022s	0.231s
Recognition	Algorithm 3 PCA ($d = 128$)	0.020s	0.195s

128, a latency reduction of 0.047 seconds can be achieved on our PC platform. Even though the improvement in processing speed become marginal when comparing PCA with $d = 128$ and $d = 256$ for PC, it is still beneficial to adopt PCA techniques for realizing a higher throughput on low-cost embedded devices whose hardware specifications are similar to the Raspberry Pi.

To conclude, using MTCNN and max fusion scheme with PCA of $d = 256$ can enable the entire pipeline to achieve the highest accuracy of 66.9% among all different settings. More importantly, under this particular configuration, the complete data-to-result processing can be finished in 0.822 seconds and 1.033 seconds, when implementing the recognition module on the PC and on the Raspberry Pi respectively, which shows that implementing the pipeline merely using SC nodes is able to achieve satisfactory real-time performance which is even comparable to the case where dedicated more powerful EGD is involved for fusion and inference.

5.6 Summary

In this chapter, we have proposed a cooperative face recognition pipeline where deep face features are extracted from light-weight CNN models deployed on SC nodes which are made up of low-cost embedded devices. To realize the proposed pipeline, we evaluated and compared the performance and flexibility of several face detection methods and state-of-the-art CNN models, we also identified suitable face datasets and trained a CNN model as the feature extractor. Besides, we adopted a softmax operator based algorithm for face recognition, in which similarity scores between input face feature and features pre-obtained in an off-line stage are computed and compared. Finally, we evaluated the accuracy and processing latency of the implemented pipeline. The test results show that our pipeline is able to run in real

time and provide acceptable recognition accuracy using only low-cost IoT devices. In the proposed cooperative face recognition scheme, every SC nodes cooperate with each other to solve the face recognition problem by sharing features among other peer SC nodes. It is showed that local cooperation by feature fusion can lead to improved recognition accuracy compared to the stand alone case where SC nodes perform the entire face recognition pipeline individually. Moreover, the experimental results reveal that dimension reduction by PCA is not only uesful for reducing communication overhead and computational complexity but is also beneficial for improving the inference accuracy as well, which is desirable for IoT networks with only limited resources.

Chapter 6

Conclusions and Future Work

With booming development of embedded computing and wireless networking technologies, IoT applications have been gradually becoming ubiquitous in various fields of daily life and industries. By leveraging the distributed cooperative inference and learning schemes, real-time processing and analysis of big data can be realized using only limited computation, communication, and data storage resources. Insights of data can also be obtained in real time at places nearer to the data source, so as to reduce communication overhead and shorten latency.

6.1 Conclusion

In this thesis, distributed inference and learning problems were addressed for resource-constrained IoT networks, with focuses on saving communication overhead, reducing energy consumption, and improving performance as well as resource utilization efficiency. Specifically, we investigated diffusion strategies for distributed LMS online learning in the context of both single-task and multitask IoT networks, we also proposed a cooperative deep learning pipeline for face recognition application that is implementable with real-world low cost IoT devices. The contributions of each chapter are summarized as below.

In Chapter 3, we studied the problem of reducing unnecessary wireless communication and saving energy budget for single-task diffusion networks where nodes perform distributed cooperative LMS estimation to learn a common unknown parameter. We proposed a modification to the conventional ATC diffusion strategy

by incorporating an event-based communication scheme, such that each node communicates its local intermediate estimate to neighboring nodes only if the local estimate has changed significantly compared with the one obtained in previous iteration. This simple-to-implement criterion and resulting event-based communication scheme can achieve remarkable reduction of communication overhead. Therefore, nodes in the underlying IoT networks are able to maintain adaptive tracking ability with much reduced energy consumption, and prolong their battery life considerably. Meanwhile, the resulting steady-state estimation accuracy can still be comparable to that obtained by the ATC strategy.

In Chapter 4, we also studied how to optimize the efficiency of inter-cluster cooperation for clustered multitask diffusion networks. We proposed a strategy, namely the MAIC, for multitask diffusion LMS estimation of parameters that may vary randomly over time. With the proposed MAIC, the inter-cluster cooperation does not impact the network stability, so that the weights govern the inter-cluster cooperation can be adjusted to improve the steady-state network MSD performance. We developed centralized as well as distributed schemes to optimize the inter-cluster cooperation weights for every nodes in the network. By doing so, each cluster in the network is able to benefit from the inter-cluster cooperation rather than using heuristically selected weights which may deteriorate the clusters' MSD performance and thus waste computation as well as energy resources.

At last, other than linear regression models studied in Chapter 3 and Chapter 4, In Chapter 5 we realized a deep learning-based face recognition pipeline that can be ran on low-cost embedded devices to perform real-time inference. To achieve this goal, we carried out extensive evaluations for selecting suitable face detection method and deep CNN model for feature extraction. Then we trained the selected CNN model, and applied a softmax operator-based algorithm for recognizing the identity captured in the images or camera frames. The obtained pipeline is able to identify a person from tens of candidates with short latency and acceptable accuracy. Through distributed cooperative scheme, our proposed pipeline could be more robust and more accurate than the case where each smart camera node relies solely on its own data. More importantly, our pipeline can save a lot of communication resources since only a few feature vectors are transmitted over

wireless channels instead of spending a lot bandwidth to send raw video streams to remote analytics server.

6.2 Future Work

Except for the current results demonstrated in each chapter, some interesting directions to consider for the future works are summarized as follows.

A natural extension for the work in Chapter 3 would be exploring modifications or variants of the current EB-ATC. For example, it might be useful to consider the information theoretic metrics as the triggering criteria. To be specific, each node may examine if the entropy or mutual information obtained from the newly made local update change significantly compared with that obtained from intermediate estimate previously sent. On the other hand, rather than combining the intermediate estimate most recently received from neighbors passively, each node may adopt a local estimation approach to infer how the latest intermediate estimates at neighbors will change relative to the ones received last time. A similar concept has been adopted in [168], it would also be interesting to design a proper estimation or inference scheme for EB-ATC, and investigate whether using the inferred values for diffusion combination can lead to an improved performance.

Besides, although in Chapter 3 the algorithm, performance analysis and numerical results have been shown, we have yet to characterize the exact MSD expression and the rate of convergence, as well as their relationships with the triggering threshold. As a next step in the future, it would be interesting as well to close this gap, for either the current EB-ATC or any of its potential variants. As such, a theoretical guide for choosing the triggering threshold may be obtained. Besides, a potential optimization of the performance over the threshold can also be formulated to improve the steady-state MSD and rate of convergence jointly.

In multitask diffusion strategies such as the MAIC, quite a lot wireless communication among nodes is incurred. Since the event-based mechanism has been found very helpful in the context of WSNs or IoT, it would be interesting to develop suitable event-based approaches for either inter-cluster or intra-cluster cooperation, in order to reduce unnecessary communication overhead and preserve the benefits of

both two types of diffusion cooperation. As suggested in [43, 169], acquiring more information in each iteration of the learning process may not always lead to better steady-state MSD performance. Thus, it might be interesting to examine if a better cluster and network performance can be obtained by limiting the frequency of inter-cluster cooperation where information being exchanged is from heterogeneous nodes that seek different learning objective.

In Chapter 5, although we have obtained some preliminary but promising results of using embedded devices to implement complex deep learning tasks, there are several ways to improve and complement the current pipeline. First of all, instead of using the cross-entropy loss that is more suitable for general classification tasks, it is of great interest to use face recognition-specialized loss function to train the CNN-based feature extractor. For example, the triplet loss used in [128], the center loss proposed by [170] and the recently proposed large margin cosine loss [32] which achieve impressive accuracy on face recognition or verification. Secondly, it would be interesting to explore a better recognition algorithm as well, by making use of state-of-the-art unsupervised learning or semi-supervised learning. To further improve the accuracy of the pipeline, it is of interest to leverage multi-view or multimodal learning techniques [171, 172] to design a more suitable feature fusion method. Last but not the least, applying the distributed estimation algorithm proposed in Chapter 3 would be a natural extension to this thesis and interesting research direction to explore. Finding a suitable and meaningful formulation is the key to this direction.

Appendix A

Definitions, Lemmas and Proofs in References

A.1 Definition of BIBO Stability

According to the definition of BIBO stability given in [173], in a discrete-time linear time-invariant system as shown in the following

$$\begin{aligned}x(k+1) &= Ax(k) + Bu(k), \\y(k) &= Cx(k) + Du(k),\end{aligned}\tag{A.1}$$

the input sequence $u(k)$ is said to be bounded if the following condition is satisfied,

$$|u(k)| < u_{\max} < \infty.\tag{A.2}$$

Then the system (A.1) is said to be BIBO stable if each eigenvalue of matrix A has a magnitude less than 1.

A.2 Definition of the Block Maximum Norm and Related Lemmas

According to [16], for any $N \times 1$ block column vector $x = \text{col}\{x_1, x_2, \dots, x_N\}$, its block maximum norm is defined as

$$\|x\|_{b,\infty} \triangleq \max_{1 \leq k \leq N} \|x_k\|. \quad (\text{A.3})$$

Then for any $N \times N$ block matrix $\mathcal{A}$, its induced block maximum norm is defined as

$$\|\mathcal{A}\|_{b,\infty} \triangleq \max_{x \neq 0} \frac{\|\mathcal{A}x\|}{\|x\|}. \quad (\text{A.4})$$

Furthermore, regarding the block maximum norm, there are some important lemmas introduced in [16] which are briefly reproduced below (please refer to [16] for the proofs).

First, according to the Lemma D.2, for any Hermitian and nonnegative-definite matrix $\mathcal{A}$, it has the following property,

$$c_1 \cdot \text{Tr}(\mathcal{A}) \leq \|\mathcal{A}\|_{b,\infty} \leq c_2 \cdot \text{Tr}(\mathcal{A}), \quad (\text{A.5})$$

where c_1, c_2 are positive constants.

Second, according to the Lemma D.3, for any matrix $\mathcal{A} = A \otimes I_M$, the following holds

$$\|\mathcal{A}\|_{b,\infty} = \|A\|_\infty, \text{ and } \|\mathcal{A}^\text{T}\|_{b,\infty} \leq N \cdot \|\mathcal{A}\|_{b,\infty}. \quad (\text{A.6})$$

Third, the Lemma D.4 says that for any left-stochastic matrix A (namely $A^\text{T} \mathbf{1}_N = \mathbf{1}_N$), let $\mathcal{A}^\text{T} = A^\text{T} \otimes I_M$, then the following holds,

$$\|\mathcal{A}\|_{b,\infty} = 1. \quad (\text{A.7})$$

Forth, the Lemma D.5 says that for an $N \times N$ Hermitian block diagonal matrix $\mathcal{D} = \text{diag}\{D_1, D_2, \dots, D_N\}$ whose block entry $D_k, k = 1, 2, \dots, N$ is also Hermi-

tion, then the following holds,

$$\rho(\mathcal{D}) = \max_{1 \leq k \leq N} \rho(D_k) = \|\mathcal{D}\|_{b,\infty}. \quad (\text{A.8})$$

Finally, regarding the Lemma D.6, we have that for an $N \times N$ Hermitian block diagonal matrix $\mathcal{D} = \text{diag}\{D_1, D_2, \dots, D_N\}$ whose block entry $D_k, k = 1, 2, \dots, N$ is also Hermitian, if A_1, A_2 are both left-stochastic matrices and $\mathcal{A}_1^\top = A_1^\top \otimes I_M$, $\mathcal{A}_2^\top = A_2^\top \otimes I_M$, then we have

$$\rho(\mathcal{A}_1 \cdot \mathcal{D} \cdot \mathcal{A}_2) \leq \rho(\mathcal{D}). \quad (\text{A.9})$$

A.3 Proof of the Mean Convergence of ATC Strategy

According to [16], the mean error recursion model of the ATC strategy can be given as below,

$$\mathbb{E}[\tilde{\mathbf{w}}(i)] = \mathcal{B}\mathbb{E}[\tilde{\mathbf{w}}(i-1)] \quad (\text{A.10})$$

where the definition of the error vector follows (3.27) and matrix $\mathcal{B}$ has been given in (3.42). Iterating the above equation from the time instant $i = 1$ and then letting $i \rightarrow \infty$ on both sides gives

$$\lim_{i \rightarrow \infty} \mathbb{E}\tilde{\mathbf{w}}(i) = \lim_{i \rightarrow \infty} \mathcal{B}^i \mathbb{E}\tilde{\mathbf{w}}(0) \quad (\text{A.11})$$

if and only if the spectral radius of the matrix $\mathcal{B}$ is less than one, i.e. $\rho(\mathcal{B}) < 1$. Because we have

$$\begin{aligned} \|\mathcal{B}\|_{b,\infty} &\leq \|\mathcal{A}^\top\|_{b,\infty} \cdot \|I_{MN} - \mu\mathcal{R}_u\|_{b,\infty} \\ &= \|I_{MN} - \mu\mathcal{R}_u\|_{b,\infty}, \end{aligned} \quad (\text{A.12})$$

where $\mathcal{R}_u$ has been given in (3.43), and the last equality follows since A is left-stochastic matrix and $\|\mathcal{A}^\top\|_{b,\infty} = \|A^\top\|_\infty = 1$ from Lemma D.4 in [16]. Therefore,

if

$$\|I_{MN} - \mu\mathcal{R}_u\|_{b,\infty} < 1, \tag{A.13}$$

then $\rho(\mathcal{B}) \leq \|\mathcal{B}\|_{b,\infty} < 1$, so that ATC strategy achieves its mean-stability. The step-size condition (3.8) just follows from (4.81).

Appendix B

List of Acronyms

IoT	Internet-of-Things
WSN	Wireless sensor network
LMS	Least mean-squares
MSD	Mean-squared deviation
D2D	Device-to-device
AI	Artificial Intelligence
CV	Computer vision
GPU	Graphics processing unit
ATC	Adapt-then-combine
EB-ATC	Event-based adapt-then-combine
BIBO	bounded input bounded output
SGD	Stochastic gradient descent
ENTR	Expected network triggering rate
ACOS	Average communication overhead saving
MAIC	Multitask adapt, inter-cluster cooperate, and then combine
MDLMS	Multitask diffusion LMS strategy
CNN	Convolutional neural network
LFW	Labelled faces in the wild
EGD	Edge gateway device
SC	Smart camera
FPS	Frame per second
HOG	Histogram of oriented gradients

MAC	Multiply–accumulate
FC	Fully connected
APO	Accuracy per operation
APP	Accuracy per parameter
PCA	Principal component analysis

Appendix C

List of Publications

Journals

1. **Y. Wang**, W. P. Tay, and W. Hu, “A multitask diffusion strategy with optimized inter-cluster cooperation,” *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 3, pp. 504–517, April 2017.
2. **Y. Wang**, W. P. Tay, C. C. Wang and W. Hu, “Distributed Estimation with Event-based Diffusion Cooperation,” under preparation.

Conferences

1. **Y. Wang**, W. P. Tay, and W. Hu, “An energy-efficient diffusion strategy over adaptive networks,” *Proc. Int. Conf. on Info. Comm. and Signal Process.*, Singapore, Jul. 2015.
2. **Y. Wang**, W. P. Tay, and W. Hu, “Multitask diffusion LMS with optimized inter-cluster cooperation,” *Proc. IEEE Statistical Signal Process. Workshop*, Palma de Mallorca, Spain, Jun. 2016.
3. **Y. Wang**, W. P. Tay, and W. Hu, “An event-based diffusion LMS strategy,” *Proc. IEEE Sensor Array and Multichannel Signal Process. Workshop*, Sheffield, United Kingdom, Jul. 2018.
4. C. Wang, W. P. Tay, **Y. Wang** and Y. Wei, “A privacy-preserving diffusion strategy over multitask networks,” *Proc. Int. Conf. on Acoustics, Speech and Signal Process.*, Brighton, United Kingdom, May 2019.

Patents

1. **Y. Wang**, W. P. Tay, J. S. Kee, K. Thangamariappan, “Wireless sensor network and parameter optimization method thereof, and warehouse system”, US Patent Filled, Nov. 2018.

Bibliography

- [1] J. Qiao, X. S. Shen, J. W. Mark, Q. Shen, Y. He, and L. Lei, “Enabling device-to-device communications in millimeter-wave 5G cellular networks,” *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 209–215, January 2015.
- [2] W. Ejaz, A. Anpalagan, M. A. Imran, M. Jo, M. Naeem, S. B. Qaisar, and W. Wang, “Internet of Things (IoT) in 5G wireless communications,” *IEEE Access*, vol. 4, 2016.
- [3] N. Saxena, A. Roy, B. J. R. Sahu, and H. Kim, “Efficient IoT gateway over 5g wireless: A new design with prototype and implementation results,” *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 97–105, February 2017.
- [4] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, “A vision of IoT: Applications, challenges, and opportunities with china perspective,” *IEEE Internet Things J.*, vol. 1, no. 4, pp. 349–359, Aug 2014.
- [5] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A survey on enabling technologies, protocols, and applications,” *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015.
- [6] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqua, and I. Yaqoob, “Big IoT data analytics: Architecture, opportunities, and open research challenges,” *IEEE Access*, vol. 5, pp. 5247–5261, 2017.
- [7] S. K. Sharma and X. Wang, “Live data analytics with collaborative edge and cloud processing in wireless IoT networks,” *IEEE Access*, vol. 5, pp. 4621–4635, July 2017.
- [8] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of Things for smart cities,” *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb 2014.
- [9] Z. Bi, L. D. Xu, and C. Wang, “Internet of Things for enterprise systems of modern manufacturing,” *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1537–1546, May 2014.
- [10] L. D. Xu, W. He, and S. Li, “Internet of Things in industries: A survey,” *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov 2014.
- [11] C. Perera, C. H. Liu, and S. Jayawardena, “The emerging Internet of Things marketplace from an industrial perspective: A survey,” *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 4, pp. 585–598, Dec 2015.

- [12] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. S. Kwak, "The Internet of Things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.
- [13] T. N. Pham, M. F. Tsai, D. B. Nguyen, C. R. Dow, and D. J. Deng, "A cloud-based smart-parking system based on Internet-of-Things technologies," *IEEE Access*, vol. 3, pp. 1581–1591, 2015.
- [14] S. H. Shah and I. Yaqoob, "A survey: Internet of Things (IoT) technologies, applications and challenges," in *Proc. IEEE Smart Energy Grid Eng.*, Aug 2016, pp. 381–385.
- [15] J. A. Stankovic, "Research directions for the Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 3–9, Feb 2014.
- [16] A. H. Sayed, "Diffusion adaptation over networks," in *Academic Press Library in Signal Process.* Elsevier, 2014, vol. 3, pp. 323 – 453.
- [17] N. Bogdanović, J. Plata-Chaves, and K. Berberidis, "Distributed incremental-based LMS for node-specific adaptive parameter estimation," *IEEE Trans. Signal Process.*, vol. 62, no. 20, pp. 5382–5397, Oct 2014.
- [18] I. Nevat, G. W. Peters, K. Avnit, F. Septier, and L. Clavier, "Location of Things: Geospatial tagging for IoT using time-of-arrival," *IEEE Trans. on Signal and Inf. Process. over Netw.*, vol. 2, no. 2, pp. 174–185, 2016.
- [19] M. Rana, "Least means squares fourth based microgrid state estimation algorithm using the internet of things technology," *PLoS ONE*, vol. 12, no. 5, pp. 1–13, 05 2017. [Online]. Available: <https://doi.org/10.1371/journal.pone.0176099>
- [20] M. Wu, N. N. Xiong, and L. Tan, "An intelligent adaptive algorithm for environment parameter estimation in smart cities," *IEEE Access*, vol. 6, pp. 23 325–23 337, 2018.
- [21] S. Theodoridis, *Machine learning: a Bayesian and optimization perspective*. Academic Press, 2015.
- [22] M. S. Mahdavinejad, M. Rezvan, M. Barekatin, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for Internet of Things data analysis: A survey," *Digital Commun. and Netw.*, vol. 4, no. 3, pp. 161–175, 2018.
- [23] V. Solo, "LMS: Past, present and future," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, May 2019, pp. 7740–7744.
- [24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances in Neural Info. Process. Syst.*, 2012, pp. 1097–1105.
- [25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

- [26] OpenFog Consortium *et al.*, “Openfog reference architecture for computing,” *Architecture Working Group*, 2017.
- [27] T. Kim, C. Ramos, and S. Mohammed, “Smart city and IoT,” *Future Gen. Comput. Syst.*, vol. 76, pp. 159 – 162, 2017.
- [28] C. G. García, D. Meana-Llorián, B. C. P. G-Bustelo, J. M. C. Lovelle, and N. Garcia-Fernandez, “Midgar: Detection of people through computer vision in the Internet of Things scenarios to improve the security in smart cities, smart towns, and smart homes,” *Future Gen. Comput. Syst.*, vol. 76, pp. 301–313, 2017.
- [29] V. A. Memos, K. E. Psannis, Y. Ishibashi, B.-G. Kim, and B. B. Gupta, “An efficient algorithm for media-based surveillance system (EAMSuS) in IoT smart city framework,” *Future Gen. Comput. Syst.*, vol. 83, pp. 619–628, 2018.
- [30] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN features off-the-shelf: An astounding baseline for recognition,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition Workshops*, June 2014, pp. 512–519.
- [31] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition,” in *Proc. British Machine Vision Conf.*, 2015.
- [32] H. Wang, Y. Wang, Z. Zhou *et al.*, “Cosface: Large margin cosine loss for deep face recognition,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, 2018.
- [33] X. Wu, R. He, Z. Sun, and T. Tan, “A light CNN for deep face representation with noisy labels,” *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2884–2896, Nov 2018.
- [34] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M.S.Lew, “Deep learning for visual understanding: A review,” *Neurocomputing*, vol. 187, pp. 27–48, 2016.
- [35] W. Rawat and Z. Wang, “Deep convolutional neural networks for image classification: A comprehensive review,” *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 2017.
- [36] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, “Deep learning for IoT big data and streaming analytics: A survey,” *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 2018.
- [37] Y. Chen, S. Kar, and J. M. F. Moura, “The Internet of Things: secure distributed inference,” *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 64–75, Sep. 2018.
- [38] M. Chiang and T. Zhang, “Fog and IoT: An overview of research opportunities,” *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec 2016.

- [39] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec 2016.
- [40] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct 2017.
- [41] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for Internet of Things: a primer," *Digital Commun. and Netw.*, vol. 4, no. 2, pp. 77–86, 2018.
- [42] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4129–4144, Aug 2014.
- [43] A. H. Sayed *et al.*, "Adaptation, learning, and optimization over networks," *Foundations and Trends® in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.
- [44] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, no. 4, pp. 460–497, April 2014.
- [45] A. H. Sayed, S. Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 155–171, May 2013.
- [46] D. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM J. Optim.*, vol. 7, no. 4, pp. 913–926, 1997.
- [47] M. G. Rabbat and R. D. Nowak, "Quantized incremental algorithms for distributed optimization," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 798–808, April 2005.
- [48] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Process.*, vol. 55, no. 8, pp. 4064–4077, Aug 2007.
- [49] R. Olfati-Saber, "Distributed kalman filter with embedded consensus filters," in *Proc. Conf. Decision and Control*, Dec 2005, pp. 8179–8184.
- [50] L. Xiao, S. Boyd, and S. Lall, "A space-time diffusion scheme for peer-to-peer least-squares estimation," in *Proc. Int. Conf. on Info. Process. in Sensor Networks*, 2006, pp. 168–176.
- [51] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan 2007.

- [52] A. Nedic, A. Ozdaglar, and P. A. Parrilo, “Constrained consensus and optimization in multi-agent networks,” *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 922–938, April 2010.
- [53] K. Srivastava and A. Nedic, “Distributed asynchronous constrained stochastic optimization,” *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 772–790, Aug 2011.
- [54] J. B. Predd, S. B. Kulkarni, and H. V. Poor, “Distributed learning in wireless sensor networks,” *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, July 2006.
- [55] R. L. Berger, “A necessary and sufficient condition for reaching a consensus using degroot’s method,” *J. American Statistical Assoc.*, vol. 76, no. 374, pp. 415–418, 1981.
- [56] J. Tsitsiklis and M. Athans, “Convergence and asymptotic agreement in distributed decision problems,” *IEEE Trans. Autom. Control*, vol. 29, no. 1, pp. 42–50, Jan 1984.
- [57] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, “Distributed kalman filtering using consensus strategies,” in *Proc. Conf. Decision and Control*, Dec 2007, pp. 5486–5491.
- [58] S. Kar, J. M. F. Moura, and K. Ramanan, “Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication,” *IEEE Trans. Inf. Theory*, vol. 58, no. 6, pp. 3575–3605, June 2012.
- [59] S. Zhu, Y. C. Soh, and L. Xie, “Distributed parameter estimation with quantized communication via running average,” *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4634–4646, Sept 2015.
- [60] G. Soatti, M. Nicoli, S. Savazzi, and U. Spagnolini, “Consensus-based algorithms for distributed network-state estimation and localization,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. PP, no. 99, pp. 1–1, 2016.
- [61] I. D. Schizas, G. Mateos, and G. B. Giannakis, “Distributed LMS for consensus-based in-network adaptive processing,” *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2365–2382, June 2009.
- [62] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan 2009.
- [63] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, “Gossip algorithms for distributed signal processing,” *Proc. IEEE*, vol. 98, no. 11, pp. 1847–1864, Nov 2010.
- [64] S. S. Stankovic, M. S. Stankovic, and D. M. Stipanovic, “Decentralized parameter estimation by consensus based stochastic approximation,” *IEEE Trans. Autom. Control*, vol. 56, no. 3, pp. 531–543, March 2011.

- [65] C. G. Lopes and A. H. Sayed, “Diffusion least-mean squares over adaptive networks: Formulation and performance analysis,” *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, July 2008.
- [66] F. S. Cattivelli and A. H. Sayed, “Diffusion LMS strategies for distributed estimation,” *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1035–1048, March 2010.
- [67] X. Zhao and A. H. Sayed, “Performance limits for distributed estimation over LMS adaptive networks,” *IEEE Trans. Signal Process.*, vol. 60, no. 10, pp. 5107–5124, Oct 2012.
- [68] S. Y. Tu and A. H. Sayed, “Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks,” *IEEE Trans. Signal Process.*, vol. 60, no. 12, pp. 6217–6234, Dec 2012.
- [69] J. Chen and A. H. Sayed, “On the learning behavior of adaptive networks—part I: Transient analysis,” *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3487–3517, June 2015.
- [70] —, “On the learning behavior of adaptive networks—part II: Performance analysis,” *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3518–3548, June 2015.
- [71] X. Zhao and A. H. Sayed, “Asynchronous adaptation and learning over networks—part I: Modeling and stability analysis,” *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 811–826, Feb 2015.
- [72] —, “Asynchronous adaptation and learning over networks—part II: Performance analysis,” *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 827–842, Feb 2015.
- [73] —, “Asynchronous adaptation and learning over networks—part III: Comparison analysis,” *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 843–858, Feb 2015.
- [74] W. Hu and W. P. Tay, “Multi-hop diffusion LMS for energy-constrained distributed estimation,” *IEEE Trans. Signal Process.*, vol. 63, no. 15, pp. 4022–4036, Aug 2015.
- [75] Y. Zhang, C. Wang, L. Zhao, and J. A. Chambers, “A spatial diffusion strategy for tap-length estimation over adaptive networks,” *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4487–4501, Sept 2015.
- [76] R. Abdolee and B. Champagne, “Diffusion LMS strategies in sensor networks with noisy input data,” *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 3–14, Feb 2016.
- [77] S. Ghazanfari-Rad and F. Labeau, “Formulation and analysis of LMS adaptive networks for distributed estimation in the presence of transmission errors,” *IEEE Internet Things J.*, vol. 3, no. 2, pp. 146–160, April 2016.

- [78] M. J. Piggott and V. Solo, "Diffusion LMS with correlated regressors I: Realization-wise stability," *IEEE Trans. Signal Process.*, vol. 64, no. 21, pp. 5473–5484, Nov 2016.
- [79] K. Ntemos, J. Plata-Chaves, N. Kolokotronis, N. Kalouptsidis, and M. Moonen, "Secure information sharing in adversarial adaptive diffusion networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. PP, no. 99, pp. 1–1, 2017.
- [80] C. Wang, Y. Zhang, B. Ying, and A. H. Sayed, "Coordinate-descent diffusion learning by networked agents," *IEEE Trans. Signal Process.*, vol. 66, no. 2, pp. 352–367, Jan 2018.
- [81] J. Plata-Chaves, N. Bogdanović, and K. Berberidis, "Distributed diffusion-based LMS for node-specific adaptive parameter estimation," *IEEE Trans. Signal Process.*, vol. 63, no. 13, pp. 3448–3460, July 2015.
- [82] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, "Multitask diffusion adaptation over asynchronous networks," *IEEE Trans. Signal Process.*, vol. 64, no. 11, pp. 2835–2850, June 2016.
- [83] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks with common latent representations," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 3, pp. 563–579, April 2017.
- [84] Y. Wang, W. P. Tay, and W. Hu, "A multitask diffusion strategy with optimized inter-cluster cooperation," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 3, pp. 504–517, April 2017.
- [85] J. Fernandez-Bes, J. Arenas-García, M. T. M. Silva, and L. A. Azpicueta-Ruiz, "Adaptive diffusion schemes for heterogeneous networks," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5661–5674, Nov 2017.
- [86] X. Zhao and A. H. Sayed, "Single-link diffusion strategies over adaptive networks," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, March 2012, pp. 3749–3752.
- [87] R. Arablouei, S. Werner, K. Doğançay, and Y. Huang, "Analysis of a reduced-communication diffusion LMS algorithm," *Signal Process.*, vol. 117, pp. 355–361, 2015.
- [88] W. Huang, X. Yang, and G. Shen, "Communication-reducing diffusion LMS algorithm over multitask networks," *Inf. Sciences*, vol. 382, pp. 115–134, 2017.
- [89] R. Arablouei, S. Werner, Y. F. Huang, and K. Doğançay, "Distributed least mean-square estimation with partial diffusion," *IEEE Trans. Signal Process.*, vol. 62, no. 2, pp. 472–484, Jan 2014.
- [90] M. O. Sayin and S. S. Kozat, "Compressive diffusion strategies over distributed networks for reduced communication load," *IEEE Trans. Signal Process.*, vol. 62, no. 20, pp. 5308–5323, Oct 2014.

- [91] I. E. K. Harrane, R. Flamary, and C. Richard, “Doubly compressed diffusion LMS over adaptive networks,” in *Proc. 50-th Asilomar Conf. on Signals, Sys. and Comp.*, Nov 2016, pp. 987–991.
- [92] Y. Wang, W. P. Tay, and W. Hu, “An energy-efficient diffusion strategy over adaptive networks,” in *Proc. 10th Int. Conf. on Info., Commun. and Signal Process.*, Dec 2015, pp. 1–5.
- [93] J. Lee, J. Kong, W. Song, and S. Kim, “Data-reserved periodic diffusion LMS with low communication cost over networks,” *IEEE Access*, vol. 6, pp. 54 636–54 650, 2018.
- [94] J. Sijs and M. Lazar, “Event based state estimation with time synchronous updates,” *IEEE Trans. Autom. Control*, vol. 57, no. 10, pp. 2650–2655, Oct 2012.
- [95] J. Wu, Q. S. Jia, K. H. Johansson, and L. Shi, “Event-based sensor data scheduling: Trade-off between communication rate and estimation quality,” *IEEE Trans. Autom. Control*, vol. 58, no. 4, pp. 1041–1046, April 2013.
- [96] D. Shi, T. Chen, and L. Shi, “Event-triggered maximum likelihood state estimation,” *Automatica*, vol. 50, no. 1, pp. 247 – 254, 2014.
- [97] Q. Liu, Z. Wang, X. He, and D. H. Zhou, “Event-based recursive distributed filtering over wireless sensor networks,” *IEEE Trans. Autom. Control*, vol. 60, no. 9, pp. 2470–2475, Sept 2015.
- [98] S. Weerakkody, Y. Mo, B. Sinopoli, D. Han, and L. Shi, “Multi-sensor scheduling for state estimation with event-based, stochastic triggers,” *IEEE Trans. Autom. Control*, vol. 61, no. 9, pp. 2695–2701, Sept 2016.
- [99] A. Mohammadi and K. N. Plataniotis, “Event-based estimation with information-based triggering and adaptive update,” *IEEE Trans. Signal Process.*, vol. 65, no. 18, pp. 4924–4939, Sept 2017.
- [100] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson, “Event-based broadcasting for multi-agent average consensus,” *Automatica*, vol. 49, no. 1, pp. 245–252, 2013.
- [101] E. Garcia, Y. Cao, and D. W. Casbeer, “Decentralized event-triggered consensus with general linear dynamics,” *Automatica*, vol. 50, no. 10, pp. 2633–2640, 2014.
- [102] D. Antunes and W. P. M. H. Heemels, “Rollout event-triggered control: Beyond periodic control performance,” *IEEE Trans. Autom. Control*, vol. 59, no. 12, pp. 3296–3311, Dec 2014.
- [103] G. Guo, L. Ding, and Q. Han, “A distributed event-triggered transmission strategy for sampled-data consensus of multi-agent systems,” *Automatica*, vol. 50, no. 5, pp. 1489–1496, 2014.

- [104] D. Ding, Z. Wang, B. Shen, and G. Wei, “Event-triggered consensus control for discrete-time stochastic multi-agent systems: the input-to-state stability in probability,” *Automatica*, vol. 62, pp. 284–291, 2015.
- [105] W. Hu, L. Liu, and G. Feng, “Consensus of linear multi-agent systems by distributed event-triggered strategy,” *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 148–157, Jan 2016.
- [106] L. Xing, C. Wen, F. Guo, Z. Liu, and H. Su, “Event-based consensus for linear multiagent systems without continuous communication,” *IEEE Trans. Cybern.*, vol. 47, no. 8, pp. 2132–2142, Aug 2017.
- [107] X. M. Zhang, Q. L. Han, and B. L. Zhang, “An overview and deep investigation on sampled-data-based event-triggered control and filtering for networked systems,” *IEEE Trans. Ind. Informat.*, vol. 13, no. 1, pp. 4–16, Feb 2017.
- [108] J. Qin, Q. Ma, Y. Shi, and L. Wang, “Recent advances in consensus of multi-agent systems: A brief survey,” *IEEE Trans. Ind. Electron.*, vol. 64, no. 6, pp. 4972–4983, June 2017.
- [109] P. Kolios, C. Panayiotou, G. Ellinas, and M. Polycarpou, “Data-driven event triggering for iot applications,” *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1146–1158, Dec 2016.
- [110] P. Kolios, G. Ellinas, C. Panayiotou, and M. Polycarpou, “Energy efficient event-based networking for the internet of things,” in *Proc. IEEE World Forum on Internet of Things*, Dec 2016, pp. 1–6.
- [111] S. Werner, Y. F. Huang, M. L. R. de Campos, and V. Koivunen, “Distributed parameter estimation with selective cooperation,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, April 2009, pp. 2849–2852.
- [112] F. C. Ribeiro, M. L. R. de Campos, and S. Werner, “Distributed cooperative spectrum sensing with selective updating,” in *Proc. Eur. Signal Process. Conf.*, Aug 2012, pp. 474–478.
- [113] I. Sobron, W. A. Martins, M. L. R. de Campos, and M. Velez, “Data-selective cooperative spectrum sensing based on imperfect information exchange,” in *Proc. IEEE Int. Symp. on Dynamic Spectr. Access Netw.*, April 2014, pp. 129–132.
- [114] J. Lee, S. Kim, and W. Song, “Data-selective diffusion LMS for reducing communication overhead,” *Signal Process.*, vol. 113, pp. 211–217, 2015.
- [115] I. Utlu, O. F. Kilic, and S. S. Kozat, “Resource-aware event triggered distributed estimation over adaptive networks,” *Digital Signal Process.*, vol. 68, pp. 127–137, 2017.
- [116] J. Chen, C. Richard, and A. H. Sayed, “Diffusion LMS for clustered multitask networks,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, May 2014, pp. 5487–5491.

- [117] X. Zhao and A. H. Sayed, “Distributed clustering and learning over networks,” *IEEE Trans. Signal Process.*, vol. 63, no. 13, pp. 3285–3300, July 2015.
- [118] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, “Multitask diffusion LMS with sparsity-based regularization,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process*, April 2015, pp. 3516–3520.
- [119] —, “Proximal multitask learning over networks with sparsity-inducing coregularization,” *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6329–6344, Dec 2016.
- [120] —, “Diffusion LMS for multitask problems with local linear equality constraints,” *IEEE Trans. Signal Process.*, vol. 65, no. 19, pp. 4979–4993, Oct 2017.
- [121] S. Monajemi, S. Sanei, S. H. Ong, and A. H. Sayed, “Adaptive regularized diffusion adaptation over multitask networks,” in *Proc. IEEE Int. Workshop on Machine Learning for Signal Process.*, Sept 2015, pp. 1–5.
- [122] N. Bogdanović, D. Ampeliotis, and K. Berberidis, “Coalitional game theoretic approach to distributed adaptive parameter estimation,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, April 2015, pp. 5793–5797.
- [123] J. Plata-Chaves, M. H. Bahari, M. Moonen, and A. Bertrand, “Unsupervised diffusion-based LMS for node-specific parameter estimation over wireless sensor networks,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, March 2016, pp. 4159–4163.
- [124] J. Plata-Chaves, A. Bertrand, M. Moonen, S. Theodoridis, and A. M. Zoubir, “Heterogeneous and multitask wireless sensor networks—algorithms, applications, and challenges,” *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 3, pp. 450–465, April 2017.
- [125] G. Hu, Y. Yang, D. Yi, J. Kittler, W. Christmas, S. Z. Li, and T. Hospedales, “When face recognition meets with deep learning: An evaluation of convolutional neural networks for face recognition,” in *Proc. Int. Conf. Comput. Vision Workshop (ICCVW)*, Dec 2015, pp. 384–392.
- [126] Y. Sun, X. Wang, and X. Tang, “Deep learning face representation from predicting 10,000 classes,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, 2014, pp. 1891–1898.
- [127] Y. Sun, D. Liang, X. Wang, and X. Tang, “Deepid3: Face recognition with very deep neural networks,” *arXiv preprint arXiv:1502.00873*, 2015.
- [128] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, June 2015.

- [129] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” Technical Report 07-49, University of Massachusetts, Amherst, Tech. Rep., 2007.
- [130] Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” in *Advances in neural information processing systems*, 2014, pp. 1988–1996.
- [131] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.
- [132] K. Grm, V. Štruc, A. Artiges, M. Caron, and H. K. Ekenel, “Strengths and weaknesses of deep learning models for face recognition against image degradations,” *arXiv preprint arXiv:1710.01494*, 2017.
- [133] C. Szegedy, W. Liu, T. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, 2015, pp. 1–9.
- [134] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, “Efficient processing of deep neural networks: A tutorial and survey,” *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec 2017.
- [135] S. Teerapittayanon, B. McDanel, and H. T. Kung, “Distributed deep neural networks over the cloud, the edge and end devices,” in *Proc. IEEE Int. Conf. on Distrib. Comput. Syst.*, June 2017, pp. 328–339.
- [136] 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,” *arXiv preprint arXiv:1704.04861*, 2017.
- [137] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, “Imagenet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.
- [138] A. H. Sayed, *Adaptive filters*. John Wiley & Sons, 2011.
- [139] M. J. Piggott and V. Solo, “Diffusion LMS with correlated regressors ii: Performance,” *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 3934–3947, Aug 2017.
- [140] X. Zhao, S. Y. Tu, and A. H. Sayed, “Diffusion adaptation over networks under imperfect information exchange and non-stationary data,” *IEEE Trans. Signal Process.*, vol. 60, no. 7, pp. 3460–3475, July 2012.
- [141] S. Boyd and L. Vandenberghe, “Convex optimization,” 2004.
- [142] C. K. Yu and A. H. Sayed, “A strategy for adjusting combination weights over adaptive networks,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, May 2013, pp. 4579–4583.

- [143] J. Fernandez-Bes, J. Arenas-García, and A. H. Sayed, “Adjustment of combination weights over adaptive diffusion networks,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, May 2014, pp. 6409–6413.
- [144] J. Chen, C. Richard, and A. H. Sayed, “Diffusion LMS over multitask networks,” *IEEE Trans. Signal Process.*, vol. 63, no. 11, pp. 2733–2748, June 2015.
- [145] M. Kasar, D. Bhattacharyya, and T. Kim, “Face recognition using neural network: a review,” *Int. J. of Security and Its Appl.*, vol. 10, no. 3, pp. 81–100, 2016.
- [146] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [147] A. Kaehler and G. Bradski, *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. O’Reilly Media, Inc., 2016.
- [148] E. K. Davis, “Dlib-ml: A machine learning toolkit,” *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.
- [149] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, vol. 1. IEEE, 2001, pp. I–I.
- [150] P. Viola and M. J. Jones, “Robust real-time face detection,” *Int. J. of Comput. Vision*, vol. 57, no. 2, pp. 137–154, 2004.
- [151] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, vol. 1. IEEE, 2005, pp. 886–893.
- [152] C. Shu, X. Ding, and C. Fang, “Histogram of the oriented gradient for face recognition,” *Tsinghua Science and Technology*, vol. 16, no. 2, pp. 216–224, April 2011.
- [153] S. Yang, P. Luo, C. Loy, and X. Tang, “From facial parts responses to face detection: A deep learning approach,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, 2015, pp. 3676–3684.
- [154] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, “A convolutional neural network cascade for face detection,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, 2015, pp. 5325–5334.
- [155] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitask cascaded convolutional networks,” *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, Oct 2016.
- [156] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, 2016, pp. 2818–2826.

- [157] I. Masi, A. T. Trãn, T. Hassner, J. T. Leksut, and G. Medioni, “Do we really need to collect millions of faces for effective face recognition?” in *Proc. European Conf. on Comput. Vision*, October 2016.
- [158] I. Masi, S. Rawls, G. Medioni, and P. Natarajan, “Pose-aware face recognition in the wild,” *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, pp. 4838–4846, 2016.
- [159] F. Chollet *et al.*, “Keras,” <https://keras.io>, 2015.
- [160] M. Abadi, P. Barham, J. Chen *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.
- [161] D. Yi, Z. Lei, S. Liao, and S. Z. Li, “Learning face representation from scratch,” *arXiv preprint arXiv:1411.7923*, 2014.
- [162] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, June 2014, pp. 1701–1708.
- [163] A. Bansal, C. Castillo, R. Ranjan, and R. Chellappa, “The do’s and don’ts for cnn-based face verification,” *arXiv preprint arXiv:1705.07426*, 2017.
- [164] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, 2016, pp. 770–778.
- [165] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Advances in Neural Inf. Process. Syst.*, 2014, pp. 568–576.
- [166] C. Feichtenhofer, A. Pinz, and A. Zisserman, “Convolutional two-stream network fusion for video action recognition,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, 2016, pp. 1933–1941.
- [167] G. B. Huang, M. Mattar, H. Lee, and E. Learned-Miller, “Learning to align from scratch,” in *NIPS*, 2012.
- [168] D. Han, Y. Mo, J. Wu, S. Weerakkody, B. Sinopoli, and L. Shi, “Stochastic event-triggered sensor schedule for remote state estimation,” *IEEE Trans. Autom. Control*, vol. 60, no. 10, pp. 2661–2675, Oct 2015.
- [169] A. H. Sayed, S. Tu, and J. Chen, “Online learning and adaptation over networks: More information is not necessarily better,” in *Proc. Info. Theory and Appl. Workshop*. IEEE, 2013, pp. 1–8.
- [170] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A discriminative feature learning approach for deep face recognition,” in *Proc. European Conf. on Comput. Vision*. Springer, 2016, pp. 499–515.
- [171] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, “Multimodal deep learning,” in *Proc. Int. Conf. on Machine Learning*, 2011, pp. 689–696.

- [172] S. Sun, “A survey of multi-view machine learning,” *Neural Comput. and Appl.*, vol. 23, no. 7-8, pp. 2031–2038, 2013.
- [173] C.-T. Chen, *Linear System Theory and Design*, 3rd ed. New York, NY, USA: Oxford University Press, Inc., 1998.