Project Title	Virtual Presence in Moving Objects through 5G
Project Acronym	PriMO-5G
Grant Agreement No	815191
Instrument	Research and Innovation Action
Topic	The PriMO-5G project addresses the area of “a) Focus on mmWave and super broadband services” in the call “EUK-02-2018: 5G” of the Horizon 2020 Work Program 2018-2020.
Start Date of Project	01.07.2018
Duration of Project	36 Months
Project Website	https://primo-5g.eu/

D4.3 - FINAL REPORT ON AI-ASSISTED NETWORKING AND EDGE COMPUTING

Work Package	WP4, AI-assisted Communications
Lead Author (Org)	KAIST
Contributing Author(s) (Org)	Arman Ahmadian (KAIST) [editor], Seongbae Jun (KAIST) [coeditor], Hyuncheol Park (KAIST), Byungjin Cho (AALTO), Woonghee Lee (EAB), Seonyong Kim (KAIST), Seongbae Jun (KAIST), Xiaolan Liu (KCL), Toktam Mahmoodi (KCL), Ki Won Sung (KTH), Joongheon Kim (KU), Soohyun Park (KU), Wonjoon Yun (KU), Seungeun Oh (YU), Seunghwan Kim (YU)
Due Date	12/31/2020
Date	12/01/2021
Version	5.11, submitted

Dissemination Level

X	PU: Public
	PP: Restricted to other programme participants (including the Commission)
	RE: Restricted to a group specified by the consortium (including the Commission)
	CO: Confidential, only for members of the consortium (including the Commission)

The work described in this document has been conducted within the project PriMO-5G. This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 815191. The project is also supported by the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00170, Virtual Presence in Moving Objects through 5G). The dissemination of results herein reflects only the author’s view, and the European Commission, IITP and MSIT are not responsible for any use that may be made of the information it contains.

Versioning and contribution history

Version	Date	Authors	Consolidated	Notes
1.00	2020-08-31	Arman Ahmadian (KAIST)	Yes	Draft ToC
1.01	2020-11-17	Byungjin Cho (AALTO)	No	Input added.
1.02	2020-11-19	Jose Costa-Requena (CMC)	No	Input added.
1.03	2020-11-19	Ki Won Sung (KTH), Woonghee Lee (EAB)	No	Input added.
1.04	2020-11-24	SangwooPark (KAIST)	No	Input added.
2.00	2020-11-24	Arman Ahmadian (KAIST)	Yes	Work added from several partners.
2.01	2020-11-25	Joongheon Kim, Soohyun Park, and Wonjoon Yun (KU)	No	Input added.
2.02	2020-11-27	Seonyong Kim (KAIST)	No	Input added.
2.10	2020-11-26	Arman Ahmadian (KAIST), Jose Costa-Requena (CMC)	Yes	Work added from several other partners.
2.20	2020-11-27	Arman Ahmadian (KAIST)	Yes	Work added from KAIST and KU.
3.00	2020-12-01	Arman Ahmadian (KAIST)	Yes	Added versioning and contribution history, disclaimer, table of contents, list of tables, list of figures, list of acronyms and abbreviations. Figure numbers cross-referenced.
3.01	2020-12-01	Seonyong Kim (KAIST)	No	Editorial changes applied.
3.02	2020-12-02	Seungeun Oh (YU)	No	Content added from YU.
3.03	2020-12-02	Konstantinos Antonakoglou (KCL)	No	Content added from KCL.
3.04	2020-12-03	Byungjin Cho (Aalto)	No	Content added from KCL.
3.05	2020-12-04	Joongheon Kim, Soohyun Park, Wonjoon Yun (KU)	No	Editorial changes applied.
3.06	2020-12-04	Ki Won Sung (KTH), Woonghee Lee (EAB)	No	Editorial changes applied.
3.07	2020-11-26	Konstantinos Antonakoglou (KCL)	No	Input added on NEF.
4.00	2020-12-04	Arman Ahmadian (KAIST)	Yes	Consolidated version from versions v3.01, v3.02, v3.03, v3.04, v3.05, v3.06. Proofreading done by applying several grammatical and spelling improvements.

Version	Date	Authors	Consolidated	Notes
				The structure of the deliverable is significantly simplified. Several blank headers as well as the contribution from CMC deleted.
4.01	2020-12-09	Arman Ahmadian (KAIST)	No	Proofreading done by applying several grammatical and spelling improvements. The structure of the deliverable is significantly simplified. Various editorial comments provided.
4.02	2020-12-09	Seongbae Jun (KAIST)	No	Introduction added in some chapters. The introduction of each study is abridged. The content of the main text is organized. Some text added to the final chapter.
4.03	2020-12-09	Joongheon Kim, Soohyun Park (KU)	No	Content added form KU (3.3)
4.04	2020-12-10	Ki Won Sung (KTH)	No	Minor editorial changes applied
4.05	2020-12-11	Byuongjin Cho (Aalto)	No	Comments addressed.
4.06	2020-12-11	Byuongjin Cho (Aalto)	No	Comments addressed.
4.07	2020-12-11	Xiaolan Liu (KCL) Toktam Mahmoodi (KCL)	No	Contribution added from KCL
4.08	2020-12-15	Xiaolan Liu (KCL)	No	References added by KCL.
4.10	2020-12-17	Arman Ahmadian (KAIST)	Yes	Consolidated version from versions v4.01, v4.02, v4.03, v4.04, v4.05, v4.06, v4.07, and v4.8. Proofreading done by applying several grammatical and spelling improvements. Comments added throughput the document. Equation numbers added, modified or corrected. Abstracts and collusions moved to the beginning and end of each chapter. A To-Do-List added for better tractability of the tasks.

Version	Date	Authors	Consolidated	Notes
4.11	2020-12-18	Sangwoo Park (KAIST)	No	Offline algorithm in subsection 4.5 removed. Minor adjustments made to reflect to change.
4.12	2020-12-18	Byungjin Cho (Aalto), Zhu Chao (Aalto)	No	Comments in v4.10 addressed
4.13	2020-12-17	Seungeun Oh and Seunghwan Kim (YU)	No	Comments addressed.
4.20	2020-12-21	Arman Ahmadian (KAIST)	Yes	Consolidated version from versions v4.11, v412, and v4.13. Proofreading done on various parts, including sections 3.2, 3.3, 3.4, 4.3, 4.4, 5.2, 5.4, and 5.5. Table of tables and table of algorithms added. Equation labels removed and unified. Some mathematical notations unified.
4.21	2020-12-21	Byuongjin Cho (Aalto)	No	Minor changes applied.
4.22	2020-12-21	Seungeun Oh and Seunghwan Kim (YU)	No	Minor changes applied.
4.23	2020-12-23	Xiaolan Liu (KCL)	No	Reference formatting changed.
4.24	2020-12-28	Seongbae Jun (KAIST)	No	The content in chapter 3 is organized. Minor editorial changes applied, mainly in chapter 5.
4.25	2021-01-06	Seungeun Oh and Seunghwan Kim (YU)	No	Abstracts and conclusions written for 4 sections.
4.26	2021-01-06	Xiaolan Liu (KCL)	No	Abstract and conclusion written.
5.00	2021-01-08	Arman Ahmadian (KAIST)	Yes	Input (Resource allocation in WPCN using DDPG) added from KAIST. Two sections (“Wireless SL” and “UL resource allocation”) are moved to chapter 5. An overview and a summary is written for each chapter. There is, now a figure in the overview of each chapter which shows the topics discussed in the chapter. chapters 1 and 6 completed. Several abbreviations and acronyms added.

Version	Date	Authors	Consolidated	Notes
				The references and table of abbreviations and acronyms sorted alphabetically. Mathematical notations unified throughout the text. Various structural and linguistic improvements applied throughput the document.
5.01	2021-01-11	Seongbae Jun (KAIST)	No	List of Acronyms and Abbreviations updated. Editorial changes applied.
5.10	2021-01-11	Arman Ahmadian (KAIST)	Yes	Header capitalization unified throughput the document. Reference formatting unified throughput the document. More acronyms and abbreviations defined. All equation punctuations corrected. Versioning table completed for initial versions. Executive summary written. Various structural and linguistic improvements applied throughput the document.
5.11 (Final)	2021-01-12	Arman Ahmadian (KAIST), Edward Mutafungwa (AALTO)	Yes	Final check and edit prior to submission

Disclaimer

PriMO-5G has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 815191. The project is also supported by the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00170, Virtual Presence in Moving Objects through 5G). The dissemination of results herein reflects only the author's view, and the European Commission, IITP and MSIT are not responsible for any use that may be made of the information it contains.

Table of contents

List of tables

List of algorithms

List of figures

Executive summary

This document aims to bring together AI-assisted solutions, tools, and architectural components that can be used to make possible or enhance wireless communication technologies that support immersive video services for moving objects, or techniques and methodologies to deploy AI algorithms needed for objected detection and image classification of visual data possibly delivered from edge-enabled nodes to AR/VR terminals via 5G networks.

In this regard, surveys, reviews and studies already existing in this work package serve as a foundation on which the AI-assisted networking and edge computing solutions and technologies presented in this deliverable are built.

The document is organized as follows

- Chapter 1 provides an introduction and an overview of the deliverable. The structure of the document as well as its relationship to other project outcomes are highlighted in this chapter.
- In chapter 2, we discuss drone localization and coordination using AI technologies that are able to use image capturing devices as well as antennas for location estimation and tracking.
- Chapter 3, focusses on data and service. More specifically, it presents AI-assisted methodologies for offloading computation or handing over data to nearby APs for more efficient caching and storage purposes.
- In chapter 4, we examine wireless networking, scheduling, signalling and optimization of the wireless communication hardware and various ways the RF signals may be used to convey information or power to/from moving objects.
- Chapter 5 studies model training on the edge, where the purpose is to find new frameworks for training AI solutions on the edge in order to facilitate training and/or enhancing data privacy and security.
- Finally, conclusions and outlooks are given in chapter 6.

List of acronyms and abbreviations

Acronym or abbreviations	Definition
5G	5th generation
ABP	Auxiliary beam pair
ACMS	Autonomous coordination management system
AI	Artificial intelligence
AoA	Angle of arrival
AoD	Angle of departure
AP	Access point
AR	Augmented reality
AWGN	Additive white Gaussian noise
BS	Base station
CAVIA	Context adaptation via meta-learning
CE	Channel estimation
CL	Centralized learning
CNN	Convolutional neural network
CPU	Central processing unit
CS	Compressive sensing
CSI	Channel state information
CUE	Cellular user
D2D	Device-to-Device
DAE	Denoising autoencoder
DBS	Drone base station
DDPG	Deep deterministic policy gradient
DNN	Deep neural network
DQN	Deep Q-learning network
DRL	Deep reinforcement learning
DSP	Digital signal processor
DUE	D2d user equipment
EKF	Extended Kalman filter
EM	Expectation maximization
FD	Federated distillation
FDD	Frequency division duplexing
FL	Federated learning
FOMAML	First-order model-agnostic meta-learning
GMM	Gaussian mixture model
GPU	Graphics processing unit
GT	Ground terminal
HAP	Hybrid access point
HetSLAgg	Hetero-modal split learning with feature aggregation
HTT	Harvest then transmit
I/Q	In-phase/Quadrature
iid	Independent and identically distributed
IoT	Internet of things
IoV	Internet of vehicles
LL	Local learning
LOS	Line-of-sight
MAB	Multi-armed bandit
MABP	Modified auxiliary beam pair

Acronym or abbreviations	Definition
MACS	Multi-agent cooperation system
MADRL	Multi-agent deep reinforcement learning
MAML	Model-agnostic meta-learning
MSGD	Mini-batch stochastic gradient descent
MDP	Markov decision process
MIMO	Multiple-input multiple-output
ML	Machine learning
MMSE	Minimum mean squared error
mmWave	Millimetre wave
MNIST	Modified national institute of standards and technology
MS	Mobile station
MSE	Mean squared error
nlDAE	Noise learning denoising autoencoder
NN	Neural network
NOMA	Non-orthogonal multiple access
PriMO-5G	Virtual presence in moving objects through 5G
QoI	Quality of Information
ReLU	Rectified linear unit
RF	Radio frequency
RFID	Radio frequency identification
RL	Reinforcement learning
RSS	Received signal strength
RV	Random variable
SFL	SplitFed learning
SGD	Stochastic gradient descent
SL	Split learning
SNR	Signal-to-noise ratio
SVM	Support vector machine
TDMA	Time-division multiple access
ToA	Time of arrival
TWH	Transmit while harvesting
UAV	Unmanned aerial vehicle
UE	User equipment
ULA	Uniform linear array
VFC	Vehicular fog computing
VFN	Vehicular fog node
VR	Virtual reality
WP	Work package
WPCN	Wireless-powered communication network
WSN	Wireless sensor network
16-QAM	16-ary quadrature amplitude modulation

1 Introduction

1.1 Scope of the document

With its focus on AI-assisted technologies in wireless communication, WP4 is tasked to demonstrate how AI can be used to improve the communication and networking capabilities of 5G wireless systems or how wireless edges may serve as a platform to run various AI algorithms and solutions in highly mobile scenarios such as in vehicular and drone (UAV) settings. The title of deliverable 4.3 is "Final report on AI-assisted networking and edge computing" in which the purpose is to exploit AI technologies to enhance the network slice orchestration with self-awareness, self-optimization, and self-configuration for supporting the demanding requirements of verticals as well as to investigate a different format of edge computing/caching for object detection, image captioning, and image classification that are needed to enhance visual images possibly delivered to AR/VR terminals via 5G networks.

1.2 Structure of the document

We divide Edge Intelligence into *AI for edge* and *AI on edge* [DZW+20]. AI for edge provides a better solution to optimization problems in edge computing with the help of AI. In other words, it studies how to leverage AI to provide more optimal solutions to problems in edge computing. Hence, it can also be called Intelligence-enabled edge computing. On the other hand, AI on edge focuses on how to build AI models on the edge. It is a scheme for running, training and inference of AI models on the edge. Its purpose is to learn from experience using distributed edge data. Therefore, it can also be understood as artificial intelligence on edge.

Based on these definitions, chapters 2 through 4 discusses AI for edge, in which the purpose is to run edge computing algorithms and it is achieved using AI algorithms and methodologies. More specifically, chapter 2 discusses UAV positioning, localization and coordination, chapter 3 deals with data and service distribution, and chapter 4 investigates the topology of edge and the techniques and methodologies to improve it using AI. On the other hand, the title of chapter 5 is AI on edge where the purpose is to run AI algorithms using the edge computing platform. The scopes of chapters 2 through 5 is demonstrated in Table 1.1 with regards to AI on edge and AI for edge.

Table 1.1 The scopes of chapters 2 through 5 with regards to AI on edge and AI for edge.

Chapters	**Area**	**What**	**How**
Chapter 2, 3, and 4	AI for Edge	Edge Computing	Artificial Intelligence
Chapter 5	AI on Edge	Artificial Intelligence	Edge Computing

1.2.1 Notation

Unless otherwise stated, we use the following notation throughout this deliverable. We use bold lower-case and upper-case letters for column vectors and matrices respectively and non-bold lower or upper case letters for scalars. $(\cdot)^T$ and $(\cdot)^*$ denote the transpose and conjugate transpose, respectively and $\mathbb{E}[\cdot]$ denotes the statistical expectation operator. $|\chi|$ denotes the cardinality of the set χ, and $\mathbb{R}^+$ represents the set of nonnegative real numbers. We write $V \sim \mathrm{Unif}(a,b)$, $W \sim \mathcal{N}(\mu,\sigma^2)$, $X \sim \mathcal{CN}(\mu,\sigma^2)$, $Y \sim \mathrm{Exp}(m)$, and $Z \sim \mathrm{Beta}(\alpha,\beta), \alpha,\beta > 0$ to show that V is distributed uniformly between a and b, W is normally distributed with mean and variance equal to μ and σ^2, X is a complex Gaussian RV with mean and variance equal to μ and σ^2, and Y is exponentially distributed with mean m, and Z is beta-

distributed with parameters α and β. Finally, $\mathcal{R}e\{z\}$ and $\mathcal{I}m\{z\}$ denote the real and imaginary parts of the complex number z.

1.3 Relationship to other project outcomes

As the name suggests, a substantial portion of this deliverable is concerned with AI technologies that enhance the networking capabilities of edges, which, in the particular case of PriMO-5G are firefighting drones and robots. In this regard, the algorithms and technologies developed in this deliverable are related to technologies formed or perfected in WP3. In particular, AI can complement, assist or even replace the mm-wave transmission technologies. On the other hand, the methodologies discussed in chapter 5, titled AI on edge, can be closely related to WP1. More specifically, the manner in which AI algorithms are used in chapter 5 is specified and is highly dependent on the scenarios and uses cases demonstrated in WP1.

2 Drone localization and coordination

2.1 Overview

As the primary focus of PriMO-5G is firefighting using drones, drone localization and coordination is probably the most central technology to consider. Hence, this chapter introduces key technologies required for UAV localization and coordination that improves performance by adopting AI. Figure 2.1 is a schematic illustrating a fleet of UAVs moving in a dynamic environment which uses antennas and cameras to find its path towards the fire location.

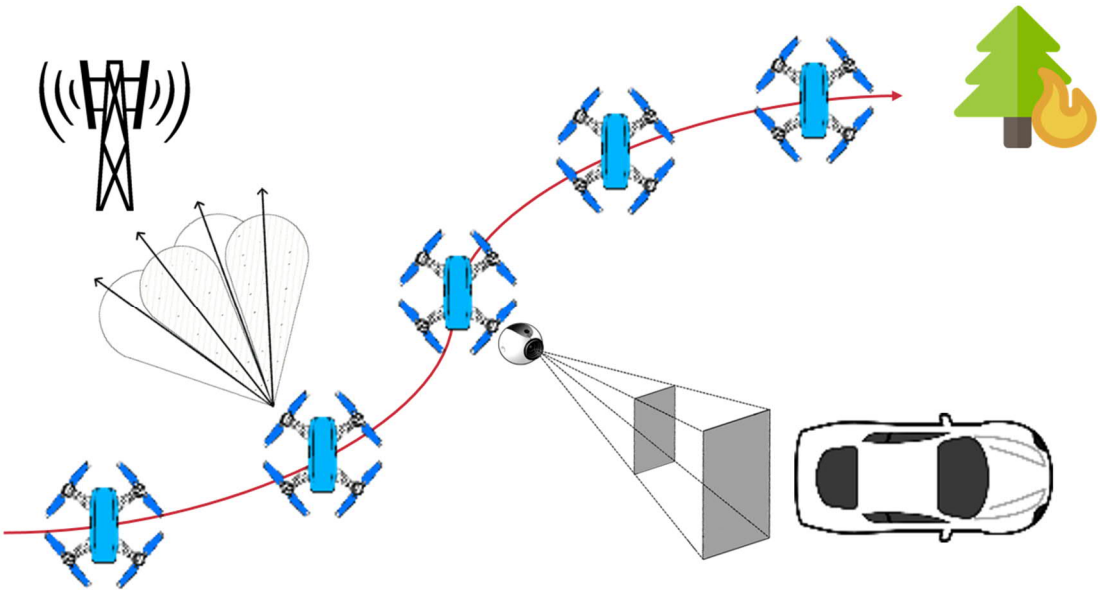


Figure 2.1 Schematic of drone localization and coordination using antennas and antennas for localization and coordination.

In subsequent sections of this chapter we design of a novel noise learning based DAE to improve the performance of DAE, Q-learning based low complexity beam tracking, and autonomous drone coordination based on a MADRL framework.

In section 2.2 we propose a new denoiser that modifies the structure of DAE, namely nlDAE. The proposed nlDAE learns the noise instead of the original data. Then, the denoising is performed by subtracting the regenerated noise from the noisy input. Hence, nlDAE is more effective than DAE when the noise is simpler to regenerate than the original data. We validate the effectiveness of nlDAE with the case study of UAV swarm-based UE localization.

In section 2.3, we propose a low complexity beam tracking algorithm combining model-free Q-learning for practical mobile mmWave MIMO systems. Compared to existing ABP-based algorithm, the proposed algorithm requires only a few beam searches with low overhead. In addition, the proposed algorithm is capable of high resolution angle estimation. The simulation result shows that the proposed beam tracking algorithm performs better than the existing algorithm without the knowledge of the model.

In section 2.4, we develop a MADLR-based management scheme for network resilience. The core idea we employ is that DBSes autonomously replenish the deficient network requirements with communication. Simulation results corroborate that the DBSes achieve a desirable performance in

terms of network communication reliability.

2.2 Design of a new denoiser based on neural networks: noise learning based denoising autoencoder

2.2.1 Overview

ML has recently received much attention as a key enabler for future wireless networks [CRT20, CCS+20, AOC19]. Among ML algorithms, DAE is widely utilized to improve the performance of applications in wireless networks by denoising observed data that is a superposition of the original data and noise [SPZ+19]. DAE is a NN model for unsupervised learning of a representation based on the construction of the learned representations robust to the addition of noise to the input samples [BYA+13]. The mechanism of DAE can be represented by two parts: 1) optimizing the NN by training a noisy training dataset towards the true input dataset, 2) denoising the test dataset using the optimized NN. The representative feature of DAE is that the dimension of the latent space is smaller than the size of the input vector. It means that the NN model is capable of encoding and decoding through a smaller dimension where the data can be represented.

The objective of this section is to improve the performance of DAE with a simple modification of its structure. Consider a noisy observation Y which consists of the original data X and the noise N, i.e., $Y = X + N$. From the information theoretical perspective, DAE attempts to minimize the expected reconstruction error by maximizing a lower bound on mutual information $I(X;Y)$. In other words, although Y is a function of the noisy input, it should capture the information of X as much as possible. Additionally, from the manifold learning perspective, DAE can be seen as a way to find a manifold where Y represents the data in a low dimensional latent space corresponding to X. However, we often face the problem that the stochastic feature of X to be restored is too complex to regenerate or represent. This is called the curse of dimensionality, i.e., the dimension of latent space for X is still too high in many cases.

What can we do if N is simpler to regenerate than X? It will be more effective to learn N and subtract it from Y instead of learning X directly. In this light, we propose a new denoising framework, named nlDAE. The main advantage of nlDAE is that it can maximize the efficiency of the ML approach (e.g., the required dimension of the latent space or number of training dataset) for wireless communications where N is typically easier to regenerate than X owing to their stochastic characteristics. To verify the advantage of nlDAE over the conventional DAE, we provide a practical application as a case study: AI-assisted UAV Swarm-based UE localization.

The following notations will be used throughout this section.

- X, N, Y: the RVs for the original data, the noise, and the noisy observations, respectively.
- $x, n, y \in RP$: the realization vector of X, N, Y, respectively, whose dimensions are P.
- $P' (< P)$: the dimension of the latent space.
- $\boldsymbol{W} \in RP'XP, \boldsymbol{W'} \in RPXP'$: the weight matrices for encoding and decoding, respectively.
- $b \in RP', b' \in RP$: the bias vectors for encoding and decoding, respectively.
- S: the sigmoid activation function for NNs, i.e. $S(a) = 1/(1 + e^{-a})$,
- and $S(a) = \left(S(a(1)), \dots, S(a(P))\right)^T$ where $a \in RP$ is an arbitrary input vector.
- f_θ: the encoding function where the parameter θ is $\{W, b\}$, i.e. $f_\theta(\boldsymbol{y}) = S(Wy + b)$.
- $g_{\theta'}$: the decoding function where the parameter θ' is $\{W', b'\}$, i.e. $g_{\theta'}(f_\theta(\boldsymbol{y})) = S\left(W'\left(f_\theta(\boldsymbol{y})\right) + b'\right)$.
- M: the number of training dataset.
- L: the number of test dataset.

2.2.2 Method of nlDAE

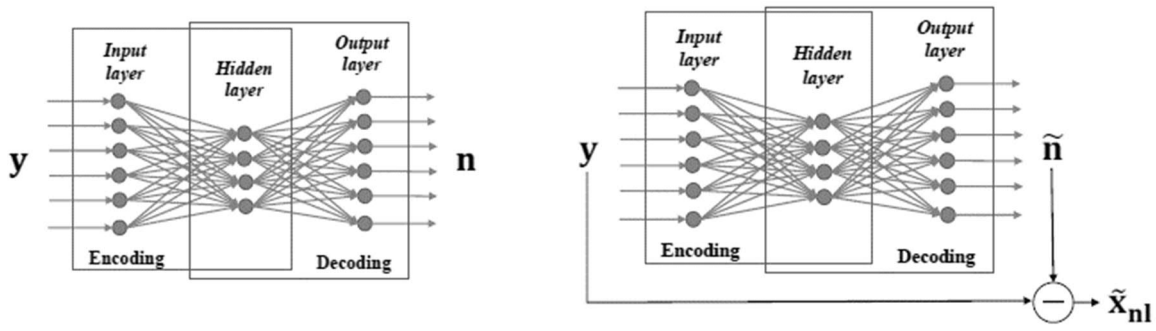
We first look into the mechanism of DAE to build NNs. Recall that DAE attempts to regenerate the original data $\boldsymbol{x}$ from the noisy observation $\boldsymbol{y}$ via training the NN. Thus, the parameters of a DAE model can be optimized by minimizing the average reconstruction error in the training phase as follows2

$$\theta^*, \theta'^* = \arg\min \frac{1}{M} \sum_{i=1}^{M} \text{Loss}\left(\boldsymbol{x}^{(i)}, \text{g}_{\theta'}\left(f_\theta\left(\mathbf{y}^{(i)}\right)\right)\right), \tag{2-1}$$

where Loss is a loss function such as squared difference between two inputs. Then, the $j\text{-}th$ regenerated data $\tilde{x}^{(j)}$ from $\boldsymbol{y}^{(j)}$ in test phase can be obtained as follows for all $j \in \{1, \dots, L\}$

$$\tilde{\boldsymbol{x}}^{(j)} = g_{\theta'^*}\left(f_{\theta^*}\left(\boldsymbol{y}^{(j)}\right)\right). \tag{2-2}$$

It is noteworthy that the sigmoid function S is a nonlinear operation, included in both encoding and decoding functions in the NN. Thus, the commutative law does not hold in this operation. From this perspective, we can hypothesize that learning N, instead of X, from Y can be beneficial in some cases even if the objective is still to reconstruct X. This is the fundamental idea of nlDAE.



(a) Training phase (b) Test phase

Figure 2.2 Illustration of concept of nlDAE.

The training and test phases of nlDAE are depicted in Figure 2.2. The parameters of nlDAE model can be optimized as follows for all $i \in \{1, \dots, M\}$

$$\theta^*, \theta'^* = \arg\min \frac{1}{M} \sum_{i=1}^{M} \text{Loss}\left(\boldsymbol{n}^{(j)}, g_{\theta'}\left(f_\theta\left(\boldsymbol{y}^{(j)}\right)\right)\right). \tag{2-3}$$

Let $\tilde{\boldsymbol{x}}_{nl}^{(j)}$ denote the $j\text{-}th$ regenerated data based on nlDAE, which can be represented as follows for all $j \in \{1, \dots, L\}$

$$\tilde{\boldsymbol{x}}_{nl}^{(j)} = \boldsymbol{y}^{(j)} - \tilde{\boldsymbol{n}}^{(j)} = \boldsymbol{y}^{(j)} - g_{\theta'^*}\left(f_{\theta^*}\left(\boldsymbol{y}^{(j)}\right)\right). \tag{2-4}$$

As shown in Figure 2.2 (a), when training the NN using the proposed nlDAE model, the noisy observation $\boldsymbol{y}$ is fed as the input to the encoder and the noise $\boldsymbol{n} = \boldsymbol{y} - \boldsymbol{x}$ is fed as the output of the decoder. This is different from the conventional DAE model, where the original data $\boldsymbol{x}$ is fed as the output of the decoder. As illustrated in Figure 2.2 (b), when denoising a test data set using the proposed nlDAE model, we first regenerate the noise ($\tilde{\boldsymbol{n}}$) using the trained nlDAE based NN, then, the denoised data ($\tilde{\boldsymbol{x}}_{nl}$) is obtained by subtracting the regenerated noise from the noisy data $\boldsymbol{y} - \tilde{\boldsymbol{n}}$. In summary, (2-3) and (2-4) contain the core idea of this method.

To provide the readers with insight into nlDAE, we address two simple examples where $Y = X + N$ as

follows:

- First, the objective is to reconstruct X from Y according to the variation of σN where $X \sim \text{Unif}(0, 2\sqrt{3})$, i.e. σX is 1, and $N \sim \mathcal{N}(0, \sigma^2 N^2)$.
- Second, $X \sim \text{Exp}(1)$, i.e. σX is 1, and all settings are equal to the first example.

As mentioned earlier, the probability distribution of X is fixed, and the standard deviation of N varies in these two scenarios. Figure 2.3 describes the performance comparison between DAE and nlDAE in terms of MSE for the two examples. Throughout this method, the squared error and the scaled conjugate gradient are applied as the loss function and the optimization method, respectively. Here, we set $P = 12$, $P' = 9$, $M = 10000$, and $L = 5000$.

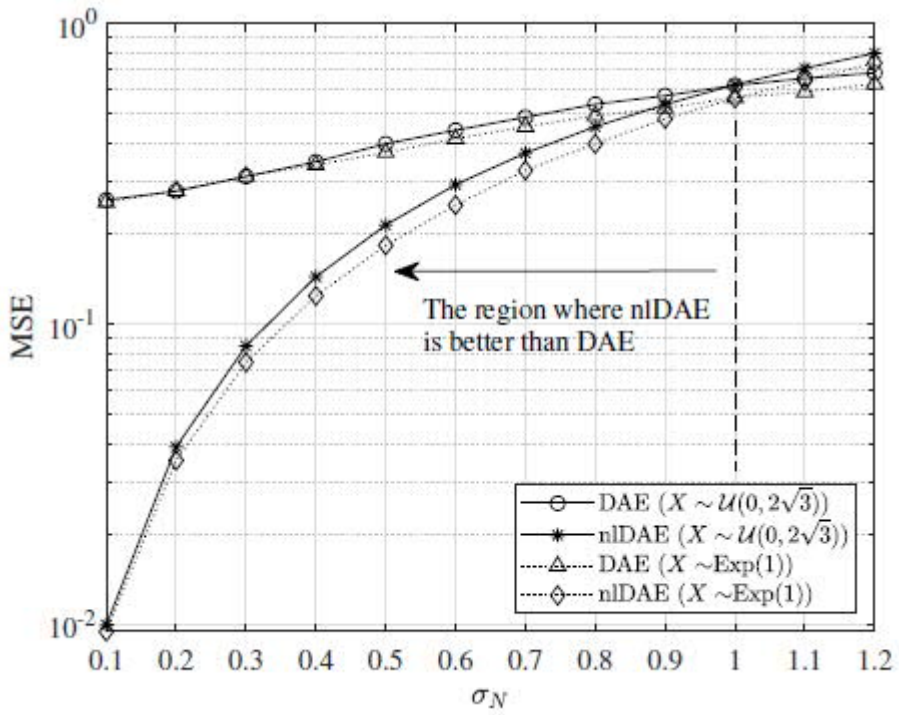


Figure 2.3 A simple example of comparison between DAE and nlDAE: reconstruction error according to σ_N.

It is observed that nlDAE is superior to DAE when σ_N is smaller than σ_X. This implies that the standard deviation is an important factor when we choose the denoiser between DAE and nlDAE.

2.2.3 Case study: AI-assisted UAV swarm-based UE localization

To validate the advantage of nlDAE over the conventional DAE in practical problems, we provide an application in wireless networks: AI-assisted UAV Swarm-based UE localization. The objective of this case study is to improve the localization quality through denoising the measured distance which is represented by the quantized value of the mixture of the true distance and error factors.

Consider a 3-D localization where P reference nodes and a single target node are randomly distributed. We estimate the position of the target node with the knowledge of the locations of P reference nodes. For this, let $\boldsymbol{x} \in \mathbb{R}^P$ denote the vector of true distances from P reference nodes to the target node where x denotes the distance between two random points in a 3-D space. We consider three types of RVs for

the noise added to the true distance as follows

- N_N: ranging error dependent on signal quality.
- N_U: ranging error due to clock asynchronization.
- N_B: ranging error dependent on signal quality.

We assume that N_N, N_U, N_B follow the normal, uniform, and Bernoulli distributions, respectively. Hence, we can define the RV for the noise N as follows

$$N = N_N + N_U + R_{\mathrm{NLoS}} N_B,$$

where R_{NLoS} is the distance bias in the event of NLoS. Besides, we assume that the distance is measured by ToA. Thus, we define the quantization function Q_B to represent the measured distance with the resolution of B, e.g., $Q_{10}(23) = 20$. In addition, the localization method based on multilateration is utilized to estimate the position of the target node.

In this case study, we consider the discrete values quantized by the function Q_B. Here, $\tilde{\mathbf{x}}_{nl}^{(j)}$ can be represented as follows

$$\tilde{\mathbf{x}}_{nl}^{(j)} = Q_B\left(\mathbf{y}^{(j)}\right) - g_{\theta'^*}\left(f_{\theta^*}\left(Q_B\left(\mathbf{y}^{(j)}\right)\right)\right),$$

where

$$\theta^*, \theta'^* = \arg\min \frac{1}{M} \sum_{i=1}^{M} \mathrm{Loss}\left(Q_B(\mathbf{n}^{(j)}), g_{\theta'}\left(f_\theta\left(Q_B(\mathbf{y}^{(j)})\right)\right)\right).$$

Thus, $\tilde{\boldsymbol{x}}_{nl}$ is utilized for the estimation of the target node position in nlDAE-assisted multilateration-based localization.

The performance of the proposed nlDAE is evaluated via $L = 5000$. In this simulation, 12 reference nodes and one target node are uniformly distributed in a $100 \times 100 \times 100$ cubic. We assume that $\mathrm{N_N} \sim \mathcal{N}(0, 20^2)$, $\mathrm{N_U} \sim \mathrm{Unif}(0, 20)$, $\mathrm{N_B} \sim \mathrm{Ber}(0.2)$, and $\mathrm{R_{NLoS}} = 50$. The distance resolution B is set to 10 for the quantization function Q_B. For a performance comparison, we select DAE as a conventional denoiser. We also provide the result of non-ML (i.e., only multilateration-based localization).

Figure 2.4 (a) shows localization error with respect to P', where $M = 10000$. The performance of nlDAE is better than non-ML and DAE for all values of P'. Moreover, nlDAE results in higher efficiency compared to DAE in terms of the required dimension of the latent space.

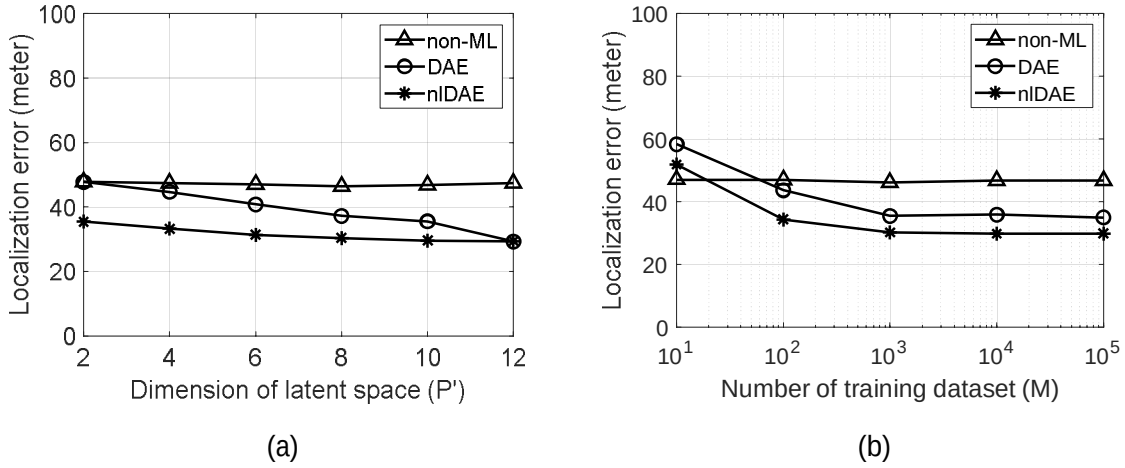


Figure 2.4 Localization error.

Figure 2.4 (b) shows localization error with respect to M, where P' = 9. Both DAE and nlDAE show better performance than non-ML after $M = 100$. In addition, nlDAE shows better performance than DAE regardless of M. Furthermore, the localization error converges almost completely for nlDAE when $M =$ 100. Hence, we can verify that nlDAE requires less training data than DAE for the localization error.

2.3 Q-learning-based low complexity beam tracking for mmWave beamforming system

2.3.1 Overview

The mmWave system is a promising candidate for next-generation wireless communication systems, and vehicular networks [UNM+19, KP20], because it can support high data rates using a huge amount of frequency resources. However, mmWave systems have disadvantage of high path-loss. To overcome this problem, a large number of antennas must be used for a sharp beam with a high beamforming gain, which has a significant impact on the performance of the system by providing accurate channel direction information. In addition, due to the mobility of MS, beam misalignments which cause the considerable performance degradation may occur. Therefore, accurate beam steering angle tracking algorithm is required to prevent the mmWave link from being disconnected when the channel rapidly varies.

There have been several previous works on estimation and tracking algorithm to get accurate beam or steering angle for mmWave communication systems. In [ZCH17], Zhu *et al*. proposed ABP-based angle estimation algorithm which utilizes adjacent two DFT beams with largest RSS by full beam search, and it works well in high SNR region. Unfortunately, the full beam search results in a large overhead. Also, ABP for angle estimation can be incorrectly selected due to noise effect, and serious performance loss may occur. On the other hand, there have been algorithms based on Kalman filter and its variants. In [VVH16], Va *et al*. proposed the conditional beam tracking algorithm based on EKF in mobile mmWave communication systems. The algorithm needs the received signal which is used as the measurement of EKF to track the beam steering angle. In our previous work [KHK+19], beam tracking algorithm combining the ABP with EKF was proposed. Specifically, the modified ABP structure to overcome incorrect ABP selection was presented, and the metric calculated in the estimation process was used as the measurement of EKF.

However, there is a critical limitation that these algorithms require the information of dynamic model

such as state-transition model, covariance of process and observation noise to perform Kalman filtering. Therefore, these algorithms are unsuitable in realistic environments. In [CCR+20], the Q-learning approach for beam tracking was proposed. Q-learning is a model-free reinforcement learning algorithm which utilizes experience, measurement, and reward from the environment by interacting its agent with the environment. However, inevitable performance loss occurs in Q-learning due to the discretized action space for tracking the beam steering angle.

In this section, to overcome limitation of prior works, we propose a model-free beam tracking algorithm which combines Q-learning with modified ABP for practical mobile environments. The proposed algorithm works in a practical situation without information of the dynamic model. Furthermore, tracking the continuous beam steering angle is possible by combining high-resolution angle estimator. Compared to the existing ABP-based angle estimator with full search for ABP selection, the proposed algorithm has low overhead because it steers only a few beams to perform angle estimation after Q-learning-based beam tracking. Simulation result shows that the proposed algorithm more accurately estimates the continuously changing angles than the existing algorithms.

The rest of this section is organized as follows. In subsection 2.3.2, we describe the system model and time-varying channel model. In subsection 2.3.3, we present an overview of the MABP method, on which our proposed method relies. In subsection 2.3.4, we propose our Q-learning-based beam tracking algorithm and finally, in subsection 2.3.5, we present the simulation results.

2.3.2 System model

This subsection describes the mmWave MIMO system model of UL channel in mobile environments, and its state-space model representation.

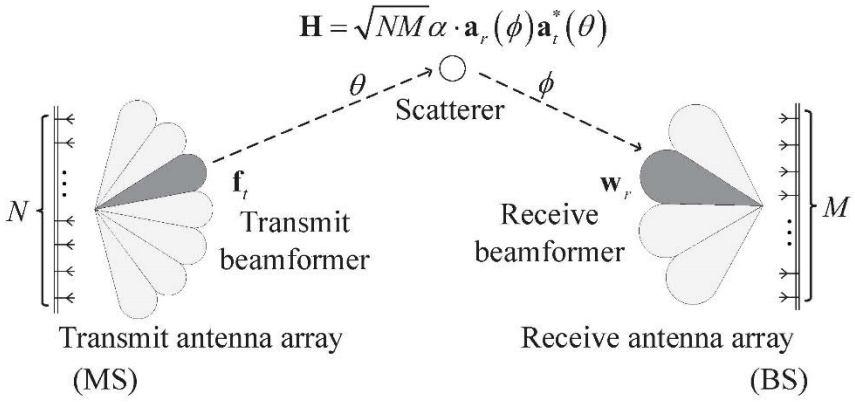


Figure 2.5 The mmWave MIMO system with analogue beamforming transceiver structure using single RF chain.

We consider the narrow band MIMO systems with beamforming transceiver structure using a single RF chain for easy implementation as shown in Figure 2.5.**Error! Reference source not found.** The MS and the BS are equipped with N antennas and M antennas, respectively. In mmWave MIMO channel, the paths are limited in sparse direction due to large path loss and highly directional property of high frequency signals. In this subsection, we consider the mmWave MIMO channel parameterized with path gain, the AoD at the MS, and the AoA at the BS, at time k, as follows

$$\mathbf{H}[k] = \sqrt{\frac{NM}{L}} \sum_{l=1}^{L} \alpha_l[k] \boldsymbol{a}_r(\phi_l[k]) \boldsymbol{a}_t^*(\theta_l[k]), \tag{2-5}$$

where L is the number of paths, $\alpha_l[k]$ is the path gain distributed $\mathcal{CN}(0,1)$, $\phi_l[k]$ and $\theta_l[k]$ are the AoA and AoD of the l-th path, respectively. $\boldsymbol{a}_t(\cdot) \in \mathbb{C}^{N\times 1}$ and $\boldsymbol{a}_r(\cdot) \in \mathbb{C}^{M\times 1}$ are the transmit and receive array response vectors, respectively. We assume a 2D model so that we only consider ULAs with antenna spacing of half wavelength at both the transmitter and receiver. The array response vector of ULAs at transmitter is expressed as

$$\boldsymbol{a}_t(\theta) = \frac{1}{\sqrt{N}} \left[1, e^{j\frac{2\pi}{\lambda} d_t \sin(\theta)}, \dots, e^{j\frac{2\pi}{\lambda}(N-1) d_t \sin(\theta)}\right]^{\mathrm{T}}, \tag{2-6}$$

where λ is a wavelength corresponding to the operating carrier frequency, and d_t is a distance between the adjacent antenna elements at transmitter. We can rewrite the array response vector at transmitter by using transmit spatial frequency $\mu \triangleq \frac{2\pi}{\lambda} d_t \sin(\theta)$ as follows,

$$\boldsymbol{a}_t(\mu) = \frac{1}{\sqrt{N}} \left[1, e^{j\mu}, \dots, e^{j(N-1)\mu}\right]^{\mathrm{T}}. \tag{2-7}$$

Similarly, the array response vector at receiver can be expressed by using the AoA ϕ and receive spatial frequency $\psi \triangleq \frac{2\pi}{\lambda} d_r \sin(\phi)$.

Considering an MS with high mobility such as a UAV in the aerial network, we assume that the channel between BS and MS is dominated by LOS path [SB98]. Therefore, we consider the single path mmWave channel, i.e., $L = 1$, as follows [KP20, ZCH17, JMC+17],

$$\mathbf{H}[k] = \sqrt{NM} \alpha[k] \boldsymbol{a}_r(\psi[k]) \boldsymbol{a}_t^*(\mu[k]), \tag{2-8}$$

where the path index is omitted.

Similar to [JMC+17], we adopt a Gaussian process noise model for the AoA evolution over the time, given by

$$\phi[k+1] = \phi[k] + \xi[k], \tag{2-9}$$

where $\xi[k] \sim \mathcal{N}(0, \sigma_\phi^2)$, and σ_ϕ^2 denotes the AoA variation in one time slot. The evolution model for path gain is given by the first-order Gauss-Markov model

$$\alpha[k+1] = \rho \alpha[k] + \varepsilon[k], \tag{2-10}$$

where ρ is the correlation coefficient, and $\varepsilon[k] \sim \mathcal{CN}(0, (1-\rho)^2)$. The received signal after combining can be expressed as

$$y[k] = \alpha[k] \cdot w_r^* a_r(\psi[k]) a_t^*(\mu[k]) f_t \cdot x[k] + w_r^* \boldsymbol{n}, \tag{2-11}$$

where $x[k]$ is a transmit symbol such that $\mathbb{E}[|x[k]|]^2 = 1$, $f_t \in \mathbb{C}^{N\times 1}$ is transmit beamformer such that $\|f_t\|_2^2 = 1$, $w_r \in \mathbb{C}^{M\times 1}$ is receive beamformer, $\boldsymbol{n} \in \mathbb{C}^{M\times 1}$ is the noise vector where its components n_i are independent and $n_i \sim \mathcal{CN}(0, \sigma^2)$. We define the SNR as $\mathrm{SNR} \triangleq 1/\sigma^2$.

$$\boldsymbol{n} \sim \mathcal{CN}(\mathbf{0}, \sigma^2 \boldsymbol{I})$$

2.3.3 An overview of MABP-based angle estimation

The ABP-based angle estimation is explained as follows [ZCH17]. Assuming a narrow band single-path channel with ULAs equipped at both the transmitter and receiver, each ABP consist of two continuous DFT-based analogue beams. The full search for all the beams are required by the receiver to simultaneously estimate the single path AoA. Firstly, the RSSs are calculated for all the beams, from which two consecutive beams are selected. One has the largest RSS and another has the second largest RSS. The received signals the two selected beams are derived as

$$y_m^{\Delta} = \alpha \boldsymbol{a}_r^*(\eta_m - \delta_r)\boldsymbol{a}_r(\psi)\boldsymbol{a}_t^*(\mu)\boldsymbol{f}_t x + \boldsymbol{a}_r^*(\eta_m - \delta_r)\boldsymbol{n}, \tag{2-12}$$

and

$$y_m^{\Sigma} = \alpha \boldsymbol{a}_r^*(\eta_m + \delta_r)\boldsymbol{a}_r(\psi)\boldsymbol{a}_t^*(\mu)\boldsymbol{f}_t x + \boldsymbol{a}_r^*(\eta_m + \delta_r)\boldsymbol{n}, \tag{2-13}$$

where η_m is the boresight angle of the m-th receive ABP, and $\delta_r = \pi / N_{\text{tot}}$ is approximates the half of half-power beamwidth for the receiver as shown Figure 2.6.

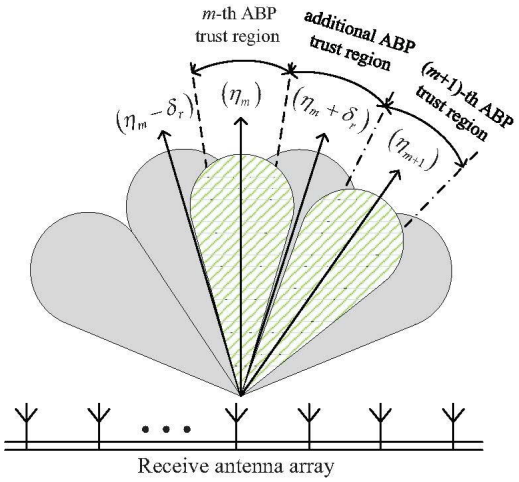


Figure 2.6 The mmWave MIMO system with analogue beamforming transceiver structure using single RF chain.

The RSSs calculated from the received signals of the two beams are derived as

$$\chi_m^{\Delta} = (y_m^{\Delta})^* y_m^{\Delta} \quad and \quad \chi_m^{\Sigma} = (y_m^{\Sigma})^* y_m^{\Sigma}. \tag{2-14}$$

The ratio metric is defined as

$$\zeta = \frac{\chi_m^{\Delta} - \chi_m^{\Sigma}}{\chi_m^{\Delta} + \chi_m^{\Sigma}} = -\frac{\sin(\psi - \eta_m)\sin(\delta_r)}{1 - \cos(\psi - \eta_m)\cos(\delta_r)}. \tag{2-15}$$

In [ZCH17], it was proved that ζ is a monotonically decreasing function of ψ, and thus there exists an inverse function. The estimated value of ψ can be recovered by using the inverse function. Although

ABP-based algorithm has good AoA estimation performance in high SNR region, if the ABP for estimating angles is incorrectly selected due to the effects of noise, the estimation procedure with fail. This mismatch occurs frequently at low SNR.

To prevent ABP wrong selection for estimating the AoA near the boundary of ABP probing range, we use additional ABP in Figure 2.6. Prior to describing additional ABP, we define the estimation trust region of ABP using the following characteristics of the ratio metric. Therefore, the estimation trust region of ABP is defined based on the value of ratio metric excluding the boundary region of probing range as follows,

$$\Omega_{tr} = \{\zeta \mid \zeta_{\text{low}} \leq \zeta \leq \zeta_{\text{up}}\}, \tag{2-16}$$

where $\zeta_{\text{low}} = -\zeta_{\text{up}} = -\frac{\sin\left(\frac{\delta_r}{2}\right)\sin(\delta_r)}{1-\cos\left(\frac{\delta_r}{2}\right)\cos(\delta_r)}$.

If the value of ratio metric is not in the estimation trust region, i.e., $\zeta < \zeta_{\text{low}}$ or $\zeta_{\text{up}} < \zeta$, additional ABP, whose boresight angle is shifted, should be used to estimate the AoA. The conditions for changing the boresight angle of ABP are as follows

$$\eta = \begin{cases} \eta_m + \delta_r & \text{if} \quad \zeta < \zeta_{\text{low}} \\ \eta_m - \delta_r & \text{if} \quad \zeta_{\text{up}} < \zeta \end{cases}. \tag{2-17}$$

By using the modified ABP structure with the condition in (2-17), the AoA to be estimated almost always exists in the estimation trust region, we can avoid the ABP wrong selection. Therefore, although it may be necessary to obtain one more RSS of a modified ABP based on the trust region, the performance degradation of the existing ABP algorithm can be overcome.

Our previous work [KHK+19] is modified ABP structure combined with EKF. However, this algorithm is based on EKF, and hence it still needs to have information about the dynamic model. Therefore, we apply the Q-learning to proposed algorithm instead of EKF.

2.3.4 Proposed Q-learning-based beam tracking algorithm

The components of Q-learning for beam tracking are as follows [CCR+20]. The state space is defined as $\mathcal{S} = \{(f_n, w_m), \forall n, m\}$, where $n \in \{1, \dots, |\mathcal{F}|\}$, and $m \in \{1, \dots, |\mathcal{W}|\}$ are transmit and receive beam indices, respectively. The codebooks of transmit and receive beam are denoted by $\mathcal{F}$ and $\mathcal{W}$, respectively. The action space is defined as $\mathcal{A} = \{\text{transmit beam index} \pm 1, \text{receive beam index} \pm 1\}$ which means that the transmit (or receive) beam is changed to adjacent beam. The reward function is defined as

$$R_{k+1} = \begin{cases} 1, & \text{if} \quad \dfrac{|y(n_f^{(k+1)}, n_w^{(k+1)})|^2}{\left|y\left(n_f^k, n_w^k\right)\right|^2} > c_u \\ 0, & \text{if} \quad c_l < \dfrac{|y(n_f^{(k+1)}, n_w^{(k+1)})|^2}{\left|y\left(n_f^k, n_w^k\right)\right|^2} \leq c_u \\ -1, & \text{otherwise} \end{cases}, \tag{2-18}$$

where $\left(n_f^k, n_w^k\right)$, $\left(n_f^{(k+1)}, n_w^{(k+1)}\right)$ are the transmit and receive beam index pair used at time k and $k+1$, respectively. c_u and c_l are the predefined parameters for calculating reward. And received signal

$y\left(n_f^k, n_w^k\right)$ is observation at time k which can be obtained by probing the transmit beam $\boldsymbol{f}_{n_f^k}$ and receive beam $\boldsymbol{w}_{n_w^k}$ as follows

$$y\left(n_f^k, n_w^k\right) = \boldsymbol{w}_{n_w^k}^* \boldsymbol{H} \boldsymbol{f}_{n_f^k} + \boldsymbol{n}. \tag{2-19}$$

In (2-18), for example, reward equal to 1 means that the action is taken in the direction of the optimal beam.

Finally, the process of updating Q-value is given below [SB98]

1. Select an action A_k from an action set $\mathcal{A}$,
2. Go to the next state S_{k+1},
3. Observe a reward R_{k+1} by using observations at time k and $k+1$, and then
4. Update the Q-value as follows

$$Q(S_k, A_k) \leftarrow (1-p) \underbrace{Q(S_k, A_k)}_{\text{old value}} + p \underbrace{\left[R_{k+1} + q \max_{a \in A} Q(S_{k+1}, a)\right]}_{\text{new information}}, \tag{2-20}$$

where $0 < p < 1$ is the learning rate (or step size), $0 < q < 1$ is the discount factor which determines the importance of future rewards. The Q-value update can be calculated as a weighted summation between old value and new information in Q-table of which dimension is $|\mathcal{S}| \times |\mathcal{A}|$.

2.3.4.1 Overall beam tracking algorithm

The overall algorithm to track the beam consists of two phases, and its pseudocode is shown in Algorithm 2.1. Initially, the beam near the optimal channel direction is tracked by using Q-learning. Then the modified ABP-based angle estimator is used to enable super-resolution estimation of beam steering angle.

- **Phase 1**: Find the beam by using model-free Q-learning-based beam tracking.
- **Phase 2**: Explore D beams on each side of the beam selected in Phase 1, and then select the beam pair with the largest RSS, and perform modified ABP-based angle estimation.

where the D is predefined parameter for additional beam searching.

Although **Phase1** can work without information of model in practical scenario, the limitations of the discretized action and state space which are components of Q-learning, cause the tracking performance loss. For example, when channels are static, inevitable changes of transmit or receive beam by design of action space result in poor tracking performance. Furthermore, in a situation when two beam or more beams have to switch with high mobility, only one beam index switching based on action space cannot track channel changes. To deal with this, modified ABP-based estimator is used with some additional overhead in **Phase2**. Thus, the proposed algorithm can overcome limitation of action space and track the continuous beam steering angle with high resolution.

Algorithm 2.1 The proposed beam tracking algorithm.

Assumption: Initial transmit and receive beam pair S_0 is known.
Initialize Q-table
$k = 0$
for $t = 1$: the number of episodes
Phase 1: Find the beams near the optimal angles of channel by using Q-learning
  for $n = 1$: the number of steps N_s
    Choose A_k and go to $S_{k+1} = (\mathbf{f}_{k+1}, \mathbf{w}_{k+1})$
    Obtain reward R_{k+1} according to the observations using (8)
    Update $Q(S_k, A_k)$
    Update $S_Q = (\mathbf{f}_{Q,k+1}, \mathbf{w}_{Q,k+1})$ according to the observations
    $k = k + 1$
  end step
Phase 2: Calculate the Estimated AoD, AoA by using modified ABP estimation
  Select the modified ABPs for AoD and AoA by comparing the RSSs of D beams on both sides of the S_Q as results of **Phase 1**
  $\zeta \leftarrow$ (14), Estimate the AoD, AoA $\leftarrow$ (16) and [3, eq(14)]
end episode
Return: Estimated AoD, and AoA

Comparing overhead between the proposed algorithm and existing algorithms [ZCH17, CCR+20] is as follows. The overhead is defined as the number of beam steering times required to track beam steering angle of one side (e.g., AoA). Existing Q-learning-based algorithm [CCR+20] requires (N_s) overheads that make up one episode for beam tracking. On the other hand, the proposed algorithm finds near the optimal beam based on Q-learning algorithm and performs additional high resolution angle estimation by using $2D$ beams, so $(N_s + 2D)$ overheads are required. However, ABP-based algorithm [ZCH17] requires $(|\mathcal{W}|)$ overheads that is size of codebook due to full beam search.

2.3.5 Simulation results

In this subsection, we evaluate the tracking performance of the proposed algorithm in mobile mmWave systems. The operating frequency is f_c = 28GHz, and both BS and MS employ ULAs with antenna spacing $d_t = d_r = \lambda/2$, and the number of antenna at the transmitter and receiver are set to be $N = 16$ and $M = 16$, respectively. Additionally, we use codebooks of transmit and receive beam with sizes of $|\mathcal{F}| = 16$ and $|\mathcal{W}| = 16$, respectively. The Q-learning parameters are discount factor $q = 0.5$, and learning rate $p = 0.5$, and the number of time steps per episode $N_s = 4$, and predefined parameter for additional beam searching $D = 3$. We consider that time-slots is 45ms for beam training period in outdoor mmWave channel model [RSP+14]. Therefore, the mobility 0.5° per time-slot is equal to $11.12^{\circ}/s$, as increasing distance between MS and BS, and it can be modelled as a high mobility environments. Suppose the distance MS and BS is 200m. This model can then be considered to move the MS at 133.3 km/h, approximately.

We compare the tracking performance between proposed algorithm and Q-learning-based algorithm [CCR+20]. In Figure 2.7, we provide the snapshots of the actual angle variation and tracking algorithms. The actual angle of channel varies according to the aforementioned Gaussian process noise model with $\sigma_{\phi}^{2} = (0.5)^{2}$ at SNR = 20dB. The modified ABP-EKF algorithm [KHK+19] is not applicable because it needs to know the dynamic model. The existing Q-learning-based beam tracking algorithm can track the beam near the angle of time-varying channel, but it cannot guarantee that the optimum beam is always tracked due to the limitation of the discretized action space. Whereas, the proposed algorithm using modified ABP structure tracks well the actual angle variation. As a numerical example of $N_s = 4$, $D = 3$ and $|\mathcal{W}| = 16$, the overheads to track beam steering angle of Q-learning-based algorithm, proposed algorithm, and ABP-based algorithm are 4, 10, and 16, respectively.

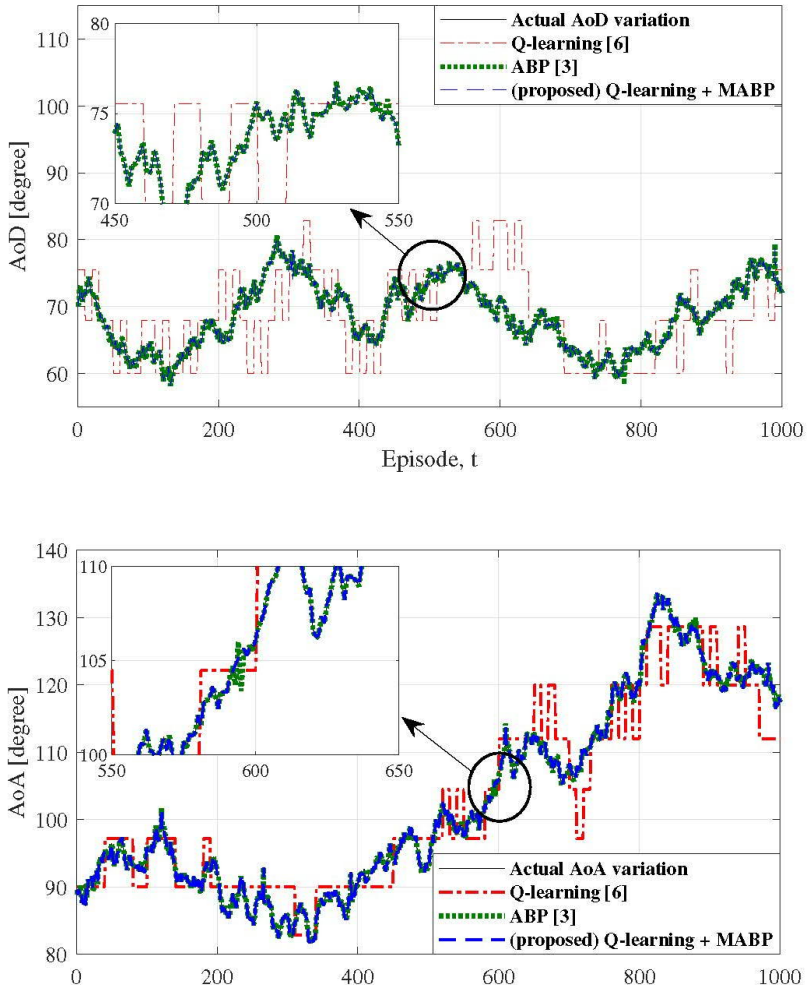


Figure 2.7 The mmWave MIMO system with analog beamforming transceiver structure using single RF chain. The snapshots of actual angle variation, Q-learning-based beam tracking, ABP-based beam tracking, and proposed beam tracking. (a) AoD. (b) AoA.

2.4 Autonomous drone coordination for network service resilience: a multi-agent deep reinforcement learning framework

2.4.1 Overview

Unpredictable demands of reliable network services have fuelled the need for flexible network communication supporting systems. Due to recent advances, UAVs, or drones, has begun to be considered as UE and as mobile BSes supplementing wireless connection to UEs. Taking advantage of the mobility feature of UAVs, it has been shown that BSes are able to complement the limitation of terrestrial cellular infrastructures deployed to fixed static locations [OKB+171, SKL192]. BSes facilitate their mobility to establish LOS communication link with users, ensuring reliable network services. The on-demand deployments of BSes allows them to continuously change their positions to optimize connection strength. In particular, when the low transmit power of UEs necessitates closer-ranged wireless connection, BSes can be effectively deployed to increase network capacity, replenish the communication coverage, and supplement the surging demands. Moreover, due to the relative low cost and the possibility of a broad range of applications, BSes are one promising solution in both academia and industry. However, UAVs are generally insufficiently maintained, causing BSe operations to face higher safety risks. For example, the engine shutdowns due to collisions with manned aircraft or terrain [OAE16] can damage the wireless BS's hardware. Moreover, the on-board power of BSes is jointly consumed by their mobility and communication support functions. To prevent unexpected battery problems, such as low battery, the power consumption of communication and mobility needs to be monitored regularly.

In light of this, autonomous BSes management system is crucial in imbuing more robust and resilient services into BS-based network systems. It is essential to conduct a joint optimization of the energy consumption and enhance the reliability of the network services despite the uncertain behaviour of UEs and neighbouring BSes. Recently, a considerable amount of literature has been published on the optimization of deployment of Bses for cellular services, including the optimization-based coverage control for transmit power reduction and deployment of BSes considering the uncertainties (e.g., demands changes of UEs) [MSB+16], optimal path-planning for multiple BSes to provide wireless services to the cell edge user based on the convex relaxation technique [CZL+18], the coverage maximization problem with minimum number of DBS [KYY16].

Even though these previous works show reasonable performance in terms of their objectives, the solution approaches are all centralized optimization problems. These approaches are impossible to yield an online (computational) solution for highly dynamic and distributed BS-enabled networks. To solve the given problem in a distributed manner, ML based approaches are effective under distributed systems settings. To handle high dynamics of BSes-enabled network (e.g., surging demands of UEs, malfunction of neighboring BSes, etc.) with high uncertainty and dynamic updates, a new multi-agent (for distributed computation over BSes) DRL scheme is designed in this study that considers the UEs and multiple BSes.

Compared to the aforementioned conventional optimization approaches, many ML techniques have been applied to improve the performance of BS-based communications, including an ML-based approach for autonomous trajectory optimization of BSes [CYG17] and the optimization of UAV location in a DL system with a joint K-means and EM algorithm based on GMM [LYC+20], dynamic optimization of the locations of UAVs in a VLC-enabled UAV based network for minimizing the transmit power [WCY+19].

Many studies have also applied DRL methods to BSes network systems, including the meta-reinforcement learning-based path-planning for BSes in dynamic and unknown wireless network environments [HCS+20] a Q-learning method-based dynamic location planning of UAVs in a NOMA based wireless network [LQC+19], and the optimization for UAV optimal energy consumption control considering communication coverage, fairness, and connectivity [LCT+18].

However, these general ML approaches have limitations, and cannot be applied to deterministic multi BSes decision-making under uncertain environment, thereby leading to undesired outputs. This is because the aforementioned studies do not consider the BS-based network system's partially observable multi-agent environment, since each agent has different information, and due to the

presence of UEs and BSes that cannot fully exchange information.

This underlines the need for further research. In this section, we consider the communication capabilities between BSes, and the presence of BSes that do not fully exchange information, while taking into account the uncertainty of deployment environment (especially with respect to BSes malfunctioning).

Figure 2.8 shows a schematic system that consists of two units (i.e., BSes, UEs). Among BSes, some of BSes cooperatively exchange information for reliable network services, while uncooperative BSes do not engage in any exchange of information. Under the uncertainty of BSes operations, such as unexpected out of power and shutdown, BSes need to autonomously complement the malfunction of network service. To cope with this problem, it is essential to configure a system where BSes automatically induce the optimal trajectories and coverage. In this process, the autonomous optimization system needs to take into account the characteristics of BSes (e.g., on-board battery).

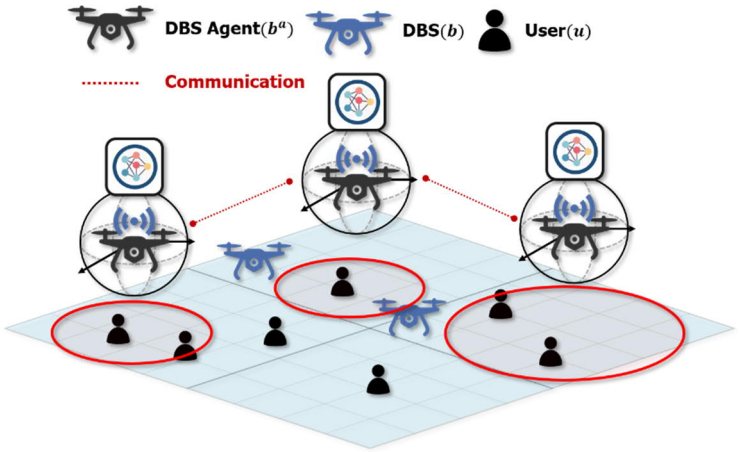


Figure 2.8 CommNet-based ACMS.

In this section, we configure the autonomous network resilience system based on a MADRL scheme, called CommNet [SSF+16]. The objective of the proposed scheme is to optimize the energy consumption of BSes and trajectories that strengthen connectivity. The policy has been trained by adjusting the threshold of coverage overlapped area. We verified that the threshold value is optimal for network service resilience. To the best of our knowledge, this is the first piece of BS-based autonomous network management research based on communication-based multi-agent DRL. The main contributions of this study are as follows

- We propose a new MADRL algorithm that achieves autonomous BSes cooperation in a distributed BSes context for conceiving an algorithm to estimate the uncertainty of the environment and to optimize the energy consumption and the overall network reliability and operation.
- To the best of our knowledge, the autonomous BSes cooperative management system (under the consideration of BSes that do not fully exchange information) via MADRL has not been studied yet. Thus the proposed scheme will guide dynamic BS-based autonomous network resilience studies in the future.
- Through performance evaluation of the proposed CommNet based autonomous BSes management scheme, we show that the proposed method can succeed in cooperatively managing distributed BSes.

2.4.2 Background on deep reinforcement learning

In this subsection, we first present a background on the conventional and modern reinforcement learning algorithms and their extensions to multi-agent systems. Then, we present the limitations of the predecessor work in terms of learning in distributed multi-agent cooperation.

In MADRL systems, the agent needs to concurrently observe the effect of its own actions as well as the behaviour of other agents. This characteristic of the multi-agent system constantly reshapes the environment and leads to non-stationarity (i.e., it becomes a non-stationary problem). As a result, the convergence theory of the predecessors is generally not guaranteed in a multi-agent system. Therefore, the information collecting and processing method should not affect the convergence stability of the agents in multi-agent systems.

2.4.3 Autonomous drone coordination

2.4.3.1 System description

Suppose that the users, DBSes, and multi-agent cooperative agents are deployed in our considered target network. We assume a leader DBS agent for handling communications between DBS agents. The leader DBS agent receives pieces of information for multi-agent cooperation (i.e., mean operation and distribution) as shown in Figure 2.9.

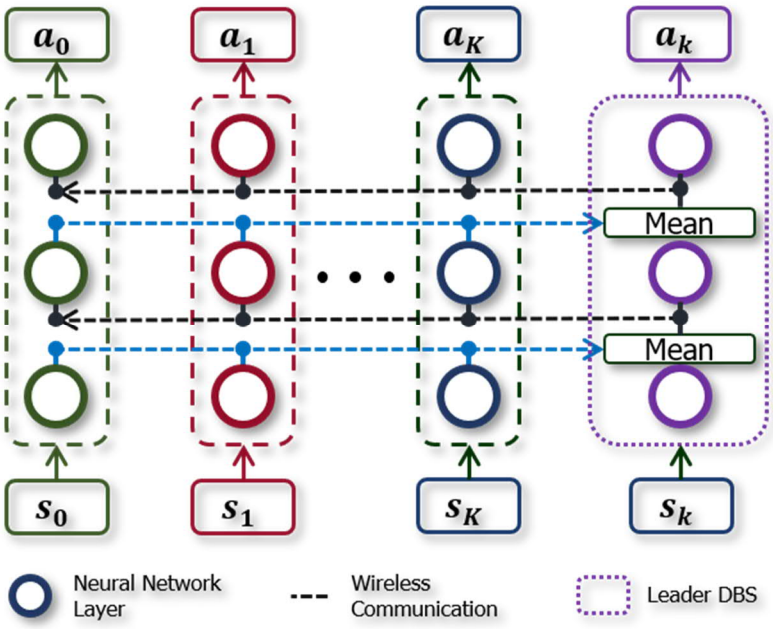


Figure 2.9 Structure of CommNet.

We assume that every user is associated with only one DBS and each DBS can be associated with multiple users. Each DBS has MACS to autonomously manage its own communication coverage. The objective of MACS of DBS is to increase the service population under uncertainty. Our proposed algorithm should be able to reliably work and achieve network service resilience even in unexpected situations, e.g., when DBSes malfunction, or when they are dropped, or become energy-exhausted [ZPR+16]. At the same time, each DBS tries to optimize energy consumption, in order to combat the power-hungry nature in DBS.

Each DBS has a maximum battery capacity. Moreover, each DBS has a maximum coverage range. Energy consumption relatively varies depending on the current coverage range. Since the communication coverage depends on the energy consumption, each user communicates with the nearest drone in order to save its own energy. Each user, by communicating with the nearest DBS, can achieve low-latency and high-speed communication. The locations of users are time-varying.

2.4.3.1.1 State space

The state space of the ACMS for DBSes consists of location information, status information (e.g., the amounts of energy consumption, communication coverage range, etc.), and relative position information with other users/DBSes.

It should be noted that each DBS agent can observe only the past and current pieces of information, whereas future information is not observable. Moreover, the observations of other DBS agents are not necessary for cooperation in this proposed MADRL scheme. Thus each DBS agent learns how to cooperate using the partially observable information and past experiences in unknown future states.

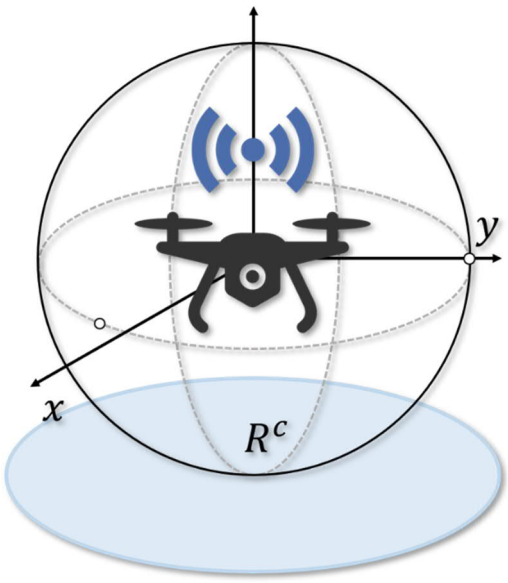


Figure 2.10 Drone Coordination.

2.4.3.1.2 Action space

The action space of the ACMS consists of eleven discrete actions, i.e., eight actions are for moving actions $\pm\alpha$ in x direction, $\pm\alpha$ in y direction, or $\pm\sigma$ in both x and y direction, and the other three actions are for controlling the coverage range $R^C \pm \beta$, where current position is (x, y) and communication coverage range is R^C. The illustrative description for this action space is as shown in Figure 2.10.

2.4.3.1.3 Reward

The reward space of the ACMS is classified into two groups, i.e., DBS rewards and cooperation rewards, respectively.

- **DBS Reward.** For defining the rewards in DBSes, we have to address three different parameters, i.e., energy consumption, battery discharge, and the number of users. The DBS agent has the power-hungry nature; thus the energy consumption model of DBS agent is required. The energy consumption of the DBS consists of two parts, the propulsion energy consumption and the communication related energy consumption. First, we define the energy consumption model, which includes the propulsion and communication energy, of DBS agents as negative reward. Seconds, the negative reward for the battery discharge r_b^k is defined as the aviation status of DBS agent, which depends on its remaining energy. Finally, the reward for the utilization rate is defined as the ratio of users in the current coverage to the number of users when the coverage is maximized.
- **Cooperation Reward.** In order to define the rewards for the cooperation among DBSes, we have to address the following three different cases, i.e., the overlapped area among DBSes, and the number of users. We propose the overlapped threshold, ω_{th}, for reward of the overlapped area among DBSes. The reward of the total utilization rate is defined as the ratio of the number of users who use the service and total number of users.

2.4.3.2 Algorithm for learning cooperation

The DBS agents of MACS are connected to users and other DBSes. Through communications with DBSes and other DBS agents, the autonomous DBS coordination scheme tries to learn how to maintain the network services under uncertainty. The proposed MACS scheme considers multi-agent system, so *CommNet*, a representative communication based MADLR algorithm, is applied. In *CommNet*, each agent uses only its observable state as shown in Figure 2.9. Note that the communication structure of *CommNet* makes convergence stable, even in a multi-agent system. *CommNet* maps the states of all agents to their actions. Note that conventional *CommNet* considers a central server that collects pieces of information of agents and distributes processed information. That is, each agent has access to a central server to share information. However, we randomly select one of the DBS agents as the leader DBS. This leader collects and distribute information. Each DBS agent sends its embedded state information to the leader DBS. The leader DBS collects the embedded information, and averages all of the received embedded messages. After that, the averaged message is taken as the input of the next layer. For other DBS agents, the leader DBS sends the averaged message to other DBS agents. The final output layer determines the agent's action. Here, $z \in \{0, .., Z\}$, where z is the number of communication steps. For each hidden layer, the activation function takes two input vectors for each DBS agent/the hidden state h_k^z and the communication vector. The softmax activation function is placed at the output layer. We can, then, interpret the output of the softmax function as the probability that each action is taken when the DBS agent observes state. We use the actor and critic reinforcement learning model based on CommNet.

2.4.4 Performance evaluation

2.4.4.1 Simulation setup

In this subsection, we numerically analyse the performance of the proposed ACMS scheme.

By default, we consider a 2 -dimensional 16 × 16 grid map, 25 randomly distributed users, 3 uncooperative DBSes, and 4 multi-agents system based DBS agents. We assume each episode has a total of 256 time steps. We configure a two-layered NN structures of ACMS as follows. In the first layer, the number of nodes is 512. The second layer also has 512 nodes. The hyperbolic-tangent function and ReLU function are used as each layer's activation function, respectively. A Xavier initializer is used for weight initialization. We use Adam optimizer. In the training procedure, ϵ-greedy method is used to make the DBS agents experience a variety of actions.

In ACMS dynamic environment, the network connectivity of users continuously changes as the status of uncooperative DBSes changes unpredictably (e.g., drop, coverage reduction, etc.). We investigated

the convergence of ACMS and its benchmark schemes. We assumed that the DBSes, but not the agents, have all malfunctioned, or are energy-exhausted. The DBSes transfer the latest locations. Every episode starts with randomly spreading agent DBSes on the grid. Each DBS is then randomly assigned a specific area of interest which is the last location left by DBS. In order to provide an optimal network service, which is the common goal of agent DBSes, they must provide the best network for each area. In order to observe how the overlapping ratio of the coverage affects the entire network, experiments were conducted on $\omega_{th} = 0.1$, $\omega_{th} = 0.2$, $\omega_{th} = 0.3$, $\omega_{th} = 0.4$, $\omega_{th} = 0.5$, and $\omega_{th} = 0.6$, where ω is the overlap ratio for coverage.

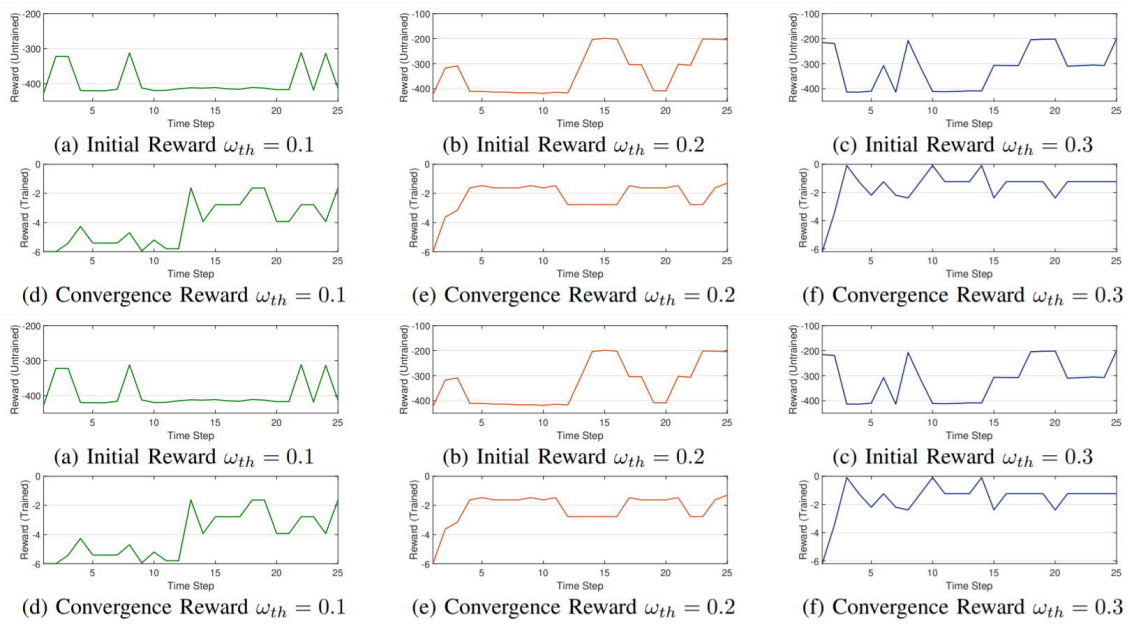


Figure 2.11 Reward tendency of untrained and trained DBS agents in autonomous DBS cooperation scheme. Reward is sum of four DBS agents' reward. ω_{th} affects reward which agents receives during training procedure.

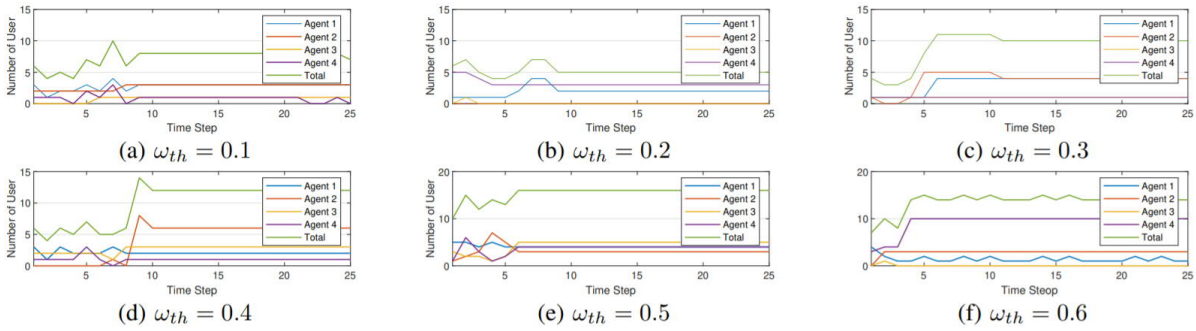


Figure 2.12 The number of connected users served by the trained DBS agents. Users are randomly distributed and move uncertainly, requiring the DBS agents to cooperatively find the optimal position and coverage range for reliable connectivity. There are four DBS agents with 15 randomly distributed users. Each line shows the summation of four DBS agents' rewards.

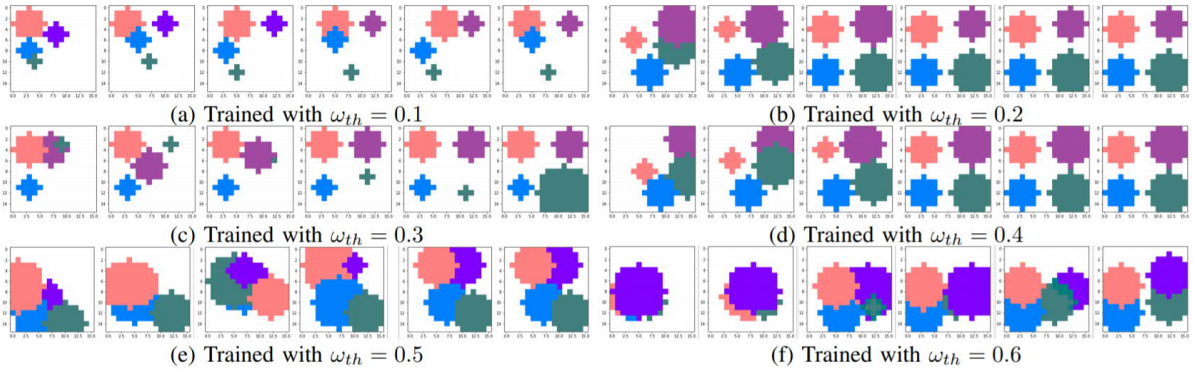


Figure 2.13 DBS agents behaviours. ω_{th} (overlapped threshold) affects the behaviour of DBS agents during the training procedure; circles depict the coverage range of each DBS agent and the centres of circles represent the position of each DBS agent

2.4.4.2 Simulation results

In the following, we investigate the reward convergence, network resilience, and trained behaviour of the proposed schemes, for a different *overlapped threshold*.

Reward convergence. We study the impact of *overlapped threshold* on the reward convergence. Figure 2.11 shows the reward flow for one episode of trained or untrained DBS agents. With ω_{th} = 0.1, the proposed autonomous DBS cooperation scheme achieves an average of about −4.5 rewards per episode. With ω_{th} = 0.2, the proposed scheme achieves an average reward of about −4 units per episode. Similarly, with ω_{th} = 0.3 the proposed scheme achieves an average reward of −3.5 units per episode. The results of these experiments achieve a relatively low average value compared to the rewards shown in Figure 2.11 (g--l). The *overlapped threshold* ω_{th} leads DBS agents to learn behaviour which reduces the overlapped coverage range. So the trained DBS agents mainly consider overlapped coverage range, not the number of reliably connected users. As shown in Figure 2.13 (a--c), the DBS agents keep as much distance as possible and avoid overlapping coverage ranges. Therefore, during an experiment episode, an average of −4 for three experiments is achieved. With ω_{th} = 0.4, the proposed scheme achieves an average reward of about −0.7 units per episode. With ω_{th} = 0.5, the proposed scheme achieves an average reward of about −0.5 units per episode. With ω_{th} = 0.6, the proposed scheme achieves average reward of about −0.2 units per episode. The relatively high *overlapped threshold* ω_{th} makes DBS agents learn behaviour which mainly focus on the number of reliably connected users. As such, the trained DBS agents does not focus on the overlapped area. As shown in Figure 2.13 (d--f), the DBS agents select a location where it can communicate with a large number of users rather than maintaining a distance from each other, and select a coverage range that can provide communication services to as many users as possible. Figure 2.13 shows that the optimal position and coverage cannot be determined, which shows that the reward continuously repeats up and down randomly, in the case of the non-trained model.

Network resilience (User connectivity). We study the impact of ω_{th}(*overlapped threshold*) on the number of users connected to DBS agents for network resilience. The DBS agents find the optimal position and coverage range for users who do not receive the network services. Figure 2.12 shows the number of users who provide services while adjusting optimal position and coverage during one episode by trained DBS agents. When ω_{th} = 0.6, the DBS agents move and control the coverage range to provide services to a total of 16 users, to 13 users when ω_{th} = 0.5, and to 9 users when ω_{th} = 0.4. This is the result of actions that focus on the number of users connected to DBS agents, rather than considering the degree of overlap of coverage between DBS agents when the *overlapped threshold* ω_{th} value is large. As shown in Figure 2.13, these characteristics of DBS agents are clearly shown

when ω_{th} = 0.4 or higher. When ω_{th} = 0.3, the DBS agents move and control the coverage range to provide services to a total of 10 users. When ω_{th} = 0.2 DBS agents move and control coverage range to provide services to a total of 5 users. When ω_{th} = 0.1 DBS agents move and control the coverage range to provide services to a total of 7 users. Compared to the ω_{th} = 0.4, 0.5, and 0.6 relatively small ω_{th} values lead DBS agents to focus on avoiding overlap of each other's coverage range. As a result, even when DBS agents can provide network services to more users, the agents maintain a minute gap between DBS agents while not increasing their coverage range.

Behaviour pattern analysis of DBS agents. Figure 2.13 shows how the trained DBS agents adjust the optimal location and coverage over time. When ω_{th} is set to 0.1, DBS agents move to the position where there is a relatively small overlapped area. At the same time, DBS agents provide services to as many users as possible and control the coverage range with the smallest overlap. As shown in Figure 2.13 (a), the DBS agents maintain the smallest average radius size among other cases. Similarly, when ω_{th} = 0.2, DBS agents move and control the coverage range for reducing the overlapped area. As shown in Figure 2.13 (b), DBS agents maintain a wider coverage than the case in Figure 2.13 (a). When ω_{th} = 0.3, DBS agents also find the position where the agents can control the coverage range for reducing the overlapped area. However, as shown in Figure 2.13 (c), the DBS agents allow some overlap between the green coverage and the purple coverage. When ω_{th} = 0.5, the DBS agents allow some overlap in the coverage range. Instead, DBS agents focus more on the number of connected users, so the rewards DBS agents receive is relatively higher than the aforementioned cases (i.e., ω_{th} = 0.1, 0.2, 0.3, and 0.4). When ω_{th} = 0.6, the DBS agents, unlike the previous cases, have a coverage area that overlaps about half of the coverage range. However, as shown in Figure 2.11, the DBS agents offset the rewards lost due to the overlapping coverage by the number of connected users. The DBS agents do not focus on reducing the overlapped area, but the position that increases the number of connected users.

2.5 Summary

This chapter studied the technologies, algorithms, and methodologies used in UAL localization and coordination. More specifically, we studied three problems in this area, namely, the nlDAE, Q-learning based beam tracking, and autonomous drone coordination based on MADRL.

In section 2.2, we introduced a new denoiser framework based on the NN, namely nlDAE. nlDAE is a modification of DAE in that it learns the noise instead of the original data. The fundamental idea of nlDAE is that learning noise can provide a better performance depending on the stochastic characteristics (e.g., standard deviation) of the original data and noise. We applied the proposed mechanism to the problems in wireless networks such as symbol demodulation and precise localization. The numerical results support that nlDAE is more effective than DAE in terms of the required dimension of the latent space and the number of training dataset. Applicability of nlDAE to other domains, e.g., image inpainting, remains as a future work. Furthermore, information theoretical criteria of decision making for the selection between DAE and nlDAE is an interesting further research.

In this section 2.3, we proposed the model-free beam tracking algorithm for practical environments without information of dynamic model. Compared to Kalman filter-based beam tracking algorithms, which require the information of the dynamic model, the proposed algorithm is based on model-free Q-learning. Also, we confirm that the proposed algorithm tracks the time-varying directional angle of channel with high resolution. In addition, compared to the conventional ABP-based estimation scheme that requires full beam search, the proposed algorithm requires only a few beam searches with low overhead.

Finally, we proposed an autonomous DBSes cooperation scheme based on communication DRL in section 2.4. The proposed scheme offers a promising solution to find optimal trajectories in the operating area and network coverage control of DBSes that can cover as many users as possible. Simulations

have shown that the proposed algorithm outperforms the single DRL schemes. Through the data-intensive evaluation, it is confirmed that the proposed method achieves desirable performance.

3 Data and service

3.1 Overview

One of the most important class of solutions that AI-based edge computing encompasses are complex AI algorithms that deal with and solve the problem of scarcity of computational and/or communication resources through relying on external computational or data cash sources. We refer to the first case, where computation is offloaded as the *service*, and the second case, where data is handed over as the *data*. Figure 3.1 shows the schematic of a drone which relies on external data sources and computational power for its operation.

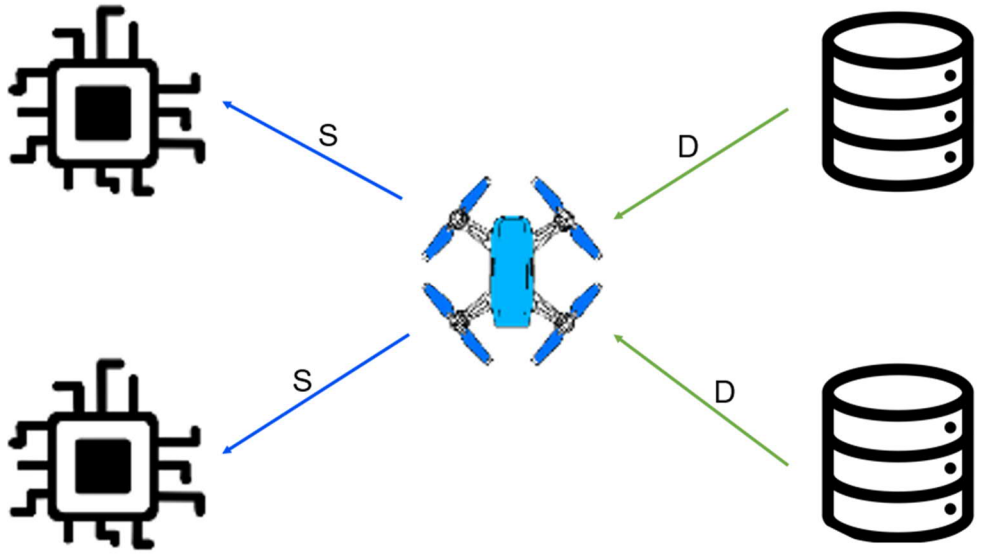


Figure 3.1 Schematic of a UAV offloading computation and relying on external caches for data.

In subsequent sections of this chapter we design computation offloading and task allocation schemes. First, we consider a centralized approach for firefighting scenarios, and then a decentralized one in adversarial environments. After that, we study dynamic video delivery in wireless caching networks using MDP.

In general, there are two different approaches that can be adopted for computation offloading, a centralized approach and decentralized one. The centralized approach relies on a single central decision-making engine where intelligence is set up using massive data obtained from heterogeneous sources. The decision is determined in a centralized manner after analysing the data, and communicated back to the source nodes for the required action. On the other hand, the decentralized approach allows the associated algorithm to run on the edge devices, i.e., autonomous vehicles or drones, meaning that the operations of data processing, knowledge extraction and decision-making are locally executed. In the following, two works are introduced, the first, called FlexSensing, is a centralized task offloading decision making algorithm presented in section 3.2, while the second is about decentralized task offloading decision making which is introduced in section 3.3.

Finally, section 3.4 addresses an IoV network that utilizes a D2D underlaid cellular system, where distributed caching at each vehicle is available and the video streaming service is provided via D2D links. Since wireless link activation for video delivery could introduce delays, node association is

determined in a larger timescale compared to power allocations. We jointly optimize these delivery decisions by maximizing the average video quality under the constraints on the playback delays of streaming users and the data rate guarantees for cellular vehicles. For each cache-enabled vehicle, the expected cost is obtained from the stochastic shortest path problem that is solved by DRL without the knowledge of global CSI. Specifically, the DDPG algorithm is adopted for dealing with the very large state space, i.e., time-varying channel states. Simulation results verify that the proposed video delivery algorithm achieves all the given goals, i.e., average video quality, smooth playback, and reliable data rates for cellular vehicles.

3.2 FlexSensing: QoI and latency aware task allocation for firefighting

3.2.1 Overview

In intelligent fire-fighting scenario, drone-based visual crowdsourcing has become an emerging computing paradigm that utilizes the images/video collected from drones to monitor the fire [ZPXY+18]. In a fire fighting scenario, the locations and the spread of fire in the building can be determined through visual-based crowdsourcing. To accurately determine the spread of fire in a timely manner, a widely adopted approach is to frequently collect high-quality visual data from dense measurement points. However, collecting and processing a large amount of visual data from moving drones requires a tremendous amount of communication and computing resources. In addition, latency constraints on time-sensitive information extraction require moving the computing resources closer to where the data is generated.

In order to solve these problems, we propose FlexSensing, a task allocation scheme that jointly optimizes the QoI and the processing latency in the case of drone-based visual crowdsourcing. Here, the QoI measures the amount of information extracted from the collected data and depends on the context of the data collection, such as the *position*, *orientation*, and *velocity* of the cameras. In this section, we take object detection as an example of a visual-crowdsourcing-based application and calculate the QoI as the number of pixels covering the targeted objects in each image.

In the following, we defined some useful terms

1. **VFNs**: The computing nodes carried by moving firetrucks with vehicle-to-everything (V2X) capacities. In FlexSensing, the VFNs are responsible for visual data processing.
2. **Service zones**: We divide a fire site into service zones of the same size based on the locations of the BSes. The BS located in the centre of each zone is selected to coordinate all the VFNs within the zone. We call the coordinator the *zone head*. With the existing cellular registration mechanisms, drones and VFNs always inform the zone head when they enter or leave the zone. In addition, VFNs periodically report their moving directions, locations, and available computing and communication resources to the zone head.
3. **Data collectors**: In FlexSensing, we assume that all drones are equipped with dash cameras and V2X modules. In a fire-fighting scenario, the severity of fire can be monitored by collecting and analysing the visual data captured by cameras installed on the drones. We consider any drones whose view covers a specific target of interest to be a data collector.
4. **Crowdsourcing tasks**: We define two types of crowdsourcing tasks in FlexSensing: visual data collection and real-time data processing. To monitor the fire, the zone head periodically assigns crowdsourcing tasks to VFNs within the service zone. More specifically, the visual data captured by data collectors within the service zone are transferred to the selected VFNs through V2V connections and processed there in real time.
5. **VFN workload**: The workload of a VFN can be estimated by the number of drones within the communication range.
6. **Processing latency**: The larger the amount of data and the higher the rate of data collection is, the greater the number of computing resources required to process the data. We assume that multiple processes can be run in parallel on each VFN and that the number of parallel

processes depends on the number of tasks to be processed simultaneously. Furthermore, the parallel processes are allowed to share CPUs and other system resources. Therefore, with a limited number of computing resources, the larger the number of processes running on the VFN is, the smaller the number of computing resources allocated to each process and the longer the execution time for each process. In FlexSensing, we use *processing latency* to denote the average executing time of processes running on the VFN.

7. **Data collection rate and QoI of data**: To accurately sense the targets of interest in a timely manner, a widely adopted approach is to frequently collect high-quality visual data from dense measurement points. In FlexSensing, we evaluate the frame rate selected for transferring the visual data as a metric of the data collection rate. Depending on the velocity, orientation and position of the drones in question, the QoI, which depends on the quality and content of the collected visual data, varies. The definition of QoI is application specific. In this work, we choose object detection as a typical computational task. In this case, we propose measuring the QoI of crowdsourced visual data by using the number of pixels covering the targets of interest in each image frame. In practice, when the camera is closer to the target, or when the resolution is higher, the number of pixels covering the targets of interest is expected to increase.
8. **System reward**: In FlexSensing, the QoI is highly likely to increase when more data is collected, whereas the processing latency will also increase with the data collection rate. To address the balance, we define a system reward, which decays along with the processing latency and increases with the QoI. In FlexSensing, the system reward unifies the processing latency and the QoI and can be tuned according to application-specific requirements. For example, if an application running on VFNs is more latency sensitive, the system reward will be more heavily weighted in favour of the processing latency, and vice versa.

3.2.2 VFN System

To provide low-latency processing for the collected visual data, we propose applying the concept of VFC [XZ+17]. More specifically, we propose turning fire trucks into VFNs equipped with CPU/GPU and V2X modules capable of handling computation and communication and utilizing these nodes for processing the visual data collected from drones within the range of a single hop. We select fire trucks as carriers of VFNs thanks to their large size and sufficient power supply.

The core idea of FlexSensing is to minimize the data collection rate for each sensing drone in the target area and to optimize the task distribution among VFNs to reduce the processing latency without degrading the QoI. We leverage the double DQN to solve the problem as described below.

First, we evenly divide a fire site into *service zones* with a cellular BS in the centre [ZPYLY+18]. The BS is configured as the coordinator of the service zone and runs a DQN agent. Second, we simulate the workload of VFNs based on the spatiotemporal distribution of surrounding drones, with the assumption that the VFN workload is proportional to the density of surrounding drones. In the initial stage, the DQN agents assign processing tasks to VFNs and select the data collection rate for each sensing drone (i.e., data collector) in a random manner. Based on the learned variation pattern in the workload of VFNs and the observed results of past decisions, the DQN agents gradually learn and update the task allocation strategies (i.e., the frame rate of data collection for each sensing drone and the computing tasks assigned to each VFN) toward specific application demands. Notably, the learning process is purely based on experience, without any predefined rules.

3.2.3 DQN approach

Owing to the complexity of the fire-fighting scenario (e.g. high mobility of drones), achieving complete knowledge of the drones and CFNs is impractical. In recent years, double DQN has shown substantial potential in terms of supporting a broad range of complex compelling applications [VGD+15]. To maximize the QoI and reduce the processing latency on VFNs, we adopt the DQN framework. In this

subsection, we first introduce the basic components of DQN. Then, we illustrate the methodology for seeking the optimal task allocation strategy via the DQN.

1. **State space**: We consider the practical online scenario where data collectors and VFNs enter and leave a specific service zone dynamically. Within a specific decision epoch j, we assume that the numbers of VFNs and drones involved remain the same. The state of an individual data collector includes the geographical information (e.g., location, speed and direction) and the configuration of its dash camera (e.g., effective range and FoV). Similarly, the state of a VFN consists of geographical information as well as the number of neighbouring drones. We use (x_t, y_t) to denote the location of a potential are of interest. If we allow the system to simultaneously assign data collectors, the action space increases exponentially with the increase of the data collectors. In such a case, it is extremely difficult for the model to converge during training. Hence, we make the state information set disjoint by splitting the data collector set. Specifically, we divide the state at a decision epoch into substates, where one and only one data collector's information is included in a substate.
2. **Action space**: As earlier stated, the zone head makes a joint control command (c_w, h_w) for the data collector at decision epoch j, in which $c_w = v$ indicates that the data collector w transfers its visual data to VFN v under a frame rate h_w. With a higher frame rate, the number of pixels covering the targets of interest in the collected images will increase, and the computational cost will grow.
3. **Task allocation strategy**: A task allocation strategy ϕ is defined as a mapping. More specifically, the zone head determines a joint control command for a data collector according to ϕ after observing the network state at the beginning of each decision epoch j.
4. **System reward**: When applying joint actions to the state, an immediate system reward at epoch j is received to quantify the task allocation experience for the system. We have two objectives, i.e., maximizing the QoI (in terms of the total number of pixels covering the targets of interest during object detection) while reducing the average processing latency. The reward is defined as follow

$$u = \sigma Q + (1 - \sigma)P,$$

where Q refers to the QoI and P refers to the processing latency, and $\sigma \in [0,1]$ is a scalar weight.

3.2.3.1 Methodology

In our scenario, given continuous values of the geographical information of drones and vehicles (e.g., location, velocity and direction) and VFN computing resource usage, there is an infinite number of state-action pairs [ZCXJ+20]; thus, the double DQN based algorithm approaches the optimal task allocation strategy by storing the state-action pairs as the adjustable parameters of the NN σ. As shown in Figure 3.2**Error! Reference source not found.**, σ denotes a vector of parameters associated with the DQN.

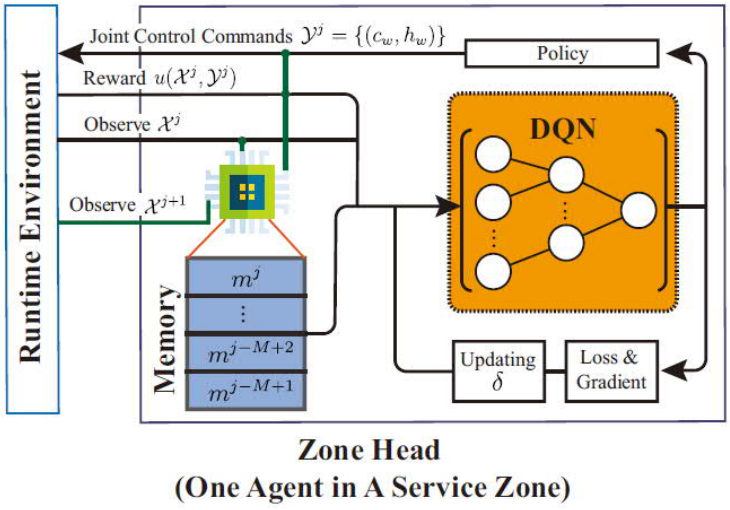


Figure 3.2 DQN Approach.

Figure 3.3 illustrates the flow of data in DQN algorithm. The agent in the zone headfirst observes its current state, and performs a set of joint actions. Then, the agent observes the subsequent state and receives an immediate utility. The value of Q-function would be adjusted using a time-varying learning rate.

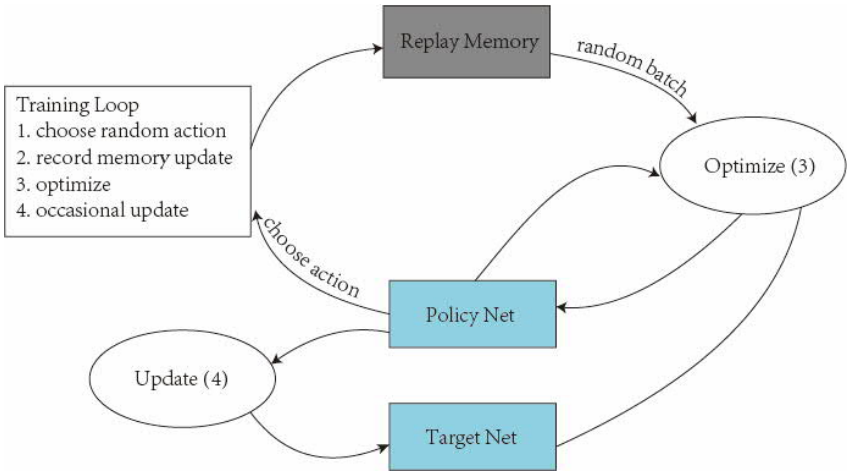


Figure 3.3 Flowchart of DQN algorithm.

To store the experience state, we assume that each zone head is equipped with a replay buffer of finite size M. As illustrated in Figure 3.3, we use m^j to denote the transition between two adjacent decision epochs j and $j+1$, and the set M^j denotes the experience pool at decision epoch j during the learning process. A *policy DQN* and a *target DQN* are maintained in the zone head, where σ^j represents the parameters at decision epoch j

$$L(\delta^j) = \mathbb{E}_{(X,Y,u(X,Y),X')\in\widehat{M}^j}\left[\left((1-\gamma)\cdot u(X,Y) + \gamma\cdot Q\left(X', \max_{Y'}\Omega\left(X',Y';\delta^j\right);\hat{\delta}^j\right) - Q(X,Y;\delta^k)\right)^2\right]. \quad (3\text{-}1)$$

At each decision epoch, according to the experience replay technique, the zone head randomly samples

a batch from the experience pool to train the DQN. In the training process, a loss function is defined in (3-1). By deriving the loss function with respect to σ^j, the parameters σ^j in the policy DQN are updated toward minimizing the mean-squared measure of equation error as shown in (3-1) at decision epoch j. The agent in the zone head regularly resets the target DQN parameters with the updated parameters in the policy DQN.

3.2.4 FlexSensing

The FlexSensing procedure is illustrated in Figure 3.4. Once a crowdsourcing task arrives, the zone head first collects the information of the data collectors and VFNs within the service zone. The geographic information of the involved drones, such as the location, velocity and moving direction, are collected and gathered at the zone head. The VFNs are also required to report their workload information.

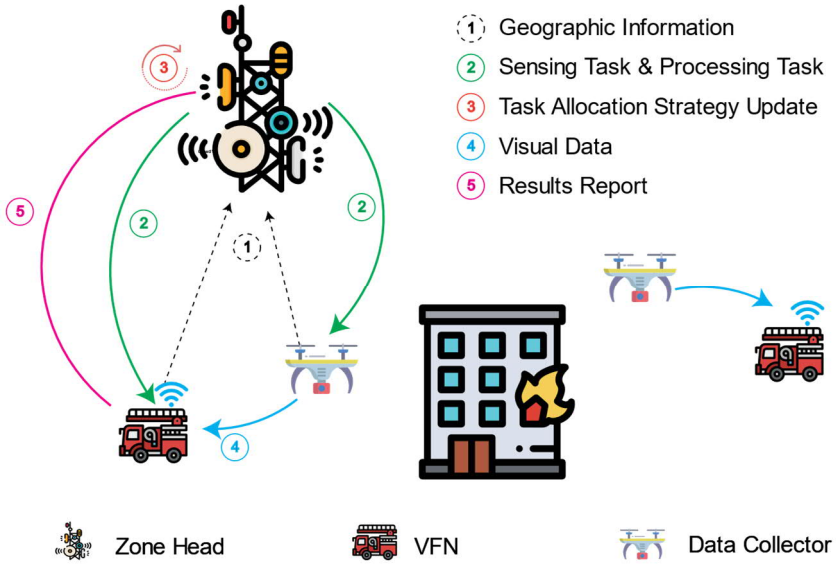


Figure 3.4 FlexSensing in a fire-fighting scenario.

Based on the collected information, the zone head learns and updates the task allocation strategy, in which each data collector in the service zone is asked to transmit the camera-captured visual data to a selected VFN with a proper frame rate. Although desirable, it is challenging to achieve an optimal task allocation strategy due to the massive scale of data collectors, diversified variation of the VFN workload, and uncertain movement of drones. To this end, we make effective use of the double deep DQN to tackle the key challenges therein. The zone head maintains a DQN agent, which has no prior knowledge in the beginning, but progressively learns to optimize the task allocation strategy based on the experienced performance.

Once the task allocation strategy is confirmed, each data collector tries to build a V2V connection with the selected VFN and starts to transmit the visual data at a specific frame rate. Once a frame is completely transferred, the VFNs process it in real time using object detection technologies (e.g., CNN).

Due to the mobility of vehicles, VFNs may leave the current service zone and enter another one before a frame is completely processed. In this case, the VFNs report the results to the new zone head, who then hands over the results to the previous one through the X2 interface in LTE/LTE-A.

3.3 Decentralized offloading decision making in adversarial environment

3.3.1 Overview

In disaster scenarios, network coverage constitutes a fundamental requirement for rescue operations. In such scenarios, however, connectivity might not be available due to damage to the terrestrial infrastructure or an excessive signalling load. Many emergency scenarios are characterized by the absence of an entity performing a data collection process and making the optimal decision in decision-making contexts. In this sense, it is necessary to design a distributed decision making algorithm which enables to adapt to dynamic and uncertain environments through interaction among nodes nearby. To do this, available computing resources of moving nodes can be supplemented to act as components of fog computing networks [XZ+17]. As the integration of fog computing and moving nodes, moving fog computing makes use of moving nodes' resources to promote the computational capability as well as further lower the delay of fog computing, so that the nodes are not only resource-consumers but also resource-providers. VFC forms a network where various types of drones with enough resources and good connectivity may become fog nodes to handle computational task offloading requests generated by neighbouring drones.

There are numerous distributed VFNs specifications, location, availability, and reputation and thus the service may have diverse preferences towards them. Due to non-random channel fluctuations and unpredictable variability of the system in VFC-inherited heterogeneous and dynamic environment, stochastic-based assumption makes the computational task offloading decisions less practical, i.e., obtained target states no longer exist, either static, in the fixed, smoothly or abruptly varied mean. Moreover, exchanging system state information between the task offloading requester and fog nodes causes high signalling overhead, and such a state is difficult to predict, so that the clients have no idea in prior which fog node perform the best performance and the advantage of VFC is nullified.

In view of the above, a drastic change is needed towards adaptive learning-based decision-making scheme. This work is to design a task offloading strategy addressing the challenges of computation-intensive applications in dynamic and uncertain environments, e.g., immersive video service or image recognition in firefighting scene, or infrastructure-less offloading service in future IoV, such as emergency D2D based offloading service.

3.3.2 Volatile scenario

We consider a VFC system, where two different types of vehicles are involved in V2V task offloading: resource requesting vehicle and VFC node. The resource requesting vehicle generates tasks and requests resource for the offloading service, while VFC nodes have sufficient computing resource and provide computing service. The resource requester interacts with accessible VFC nodes and may offload a task to a promising VFC node in a candidate set within its access region. Due to inherent mobility, a candidate VFC set may be different in different time instance. For example, a VFC node appears in a candidate set at different time instance, or leaves a candidate set temporarily but return into the set in a finite number of rounds.

3.3.3 Adversarial MAB approach

Considering joint perturbations in two sub-process, i) assessment stage and ii) choice stage, in adversarial environment would achieve improved exploitation-exploration balance [CL+06], resulting in better individual performance. The former is about *capability domain*, i.e., how to design a learning rule such that each client uses it to gain knowledge on service capability, while the latter is about the *suitability domain*, i.e., how to make a fog node selection better suited for the next task with time-varying size. Each resource requester assesses and learns the service capability of candidate resource providers, i.e., the estimation of service cost, in order to optimize the expected service cost. The service capability of a fog node is represented by the cost per bit for the previous task.

However, the potential service capability may not ensure its envisioned suitability for the upcoming task, due to time-varying input data. For example, when the input data size for the next task is small enough, one may try to explore more. Because, even if a poorly performed fog node is selected, the cost for the task with a small size is not excessive. One may try to exploit more with large size. In this sense, more suitable resource selection could be achieved by considering the input data size for the upcoming task, which would address the resource demand dependent exploration-exploitation trade-off. For this reason, one may consider soft-max process [AL+17] in the choice rule of learning scheme with a weight factor normalizing input data size.

Note that the normalized input data size value in the soft-max process plays an important role in making the high selection probability higher and the low selection probability lower, i.e. exploiting more for the larger task workload, while exploring more for the smaller one. One issue is that the soft-max process does not consider the uncertainty of the estimated cost and thus may draw sub-optimal arms too much, which may cause undesired outcomes, i.e., unbalanced exploration-exploitation trade-off and highly unstable/inaccurate individual performance. We circumvent the issue by considering algorithmic robustness for which bounded properties on estimation are used.

3.4 MDP for dynamic video delivery in wireless caching networks

3.4.1 Markov decision process

MDP is an optimization model for decision making under uncertainty [B57, P94]. It describes a stochastic decision process of an agent interacting with an environment or system. At each decision time, the system stays in a certain state s and the agent chooses an action a that is available at this state. After the action is performed, the agent receives an immediate reward R and the system transits to a new state s' according to the transition probability [AHN+15]. The MDP model is used for solving various design and resource management issues in WSNs. For WSNs, the MDP is used to model the interaction between a wireless sensor node (i.e., an agent) and their surrounding environment (i.e., a system) to achieve some objectives [AHN+15].

The MDP is defined by a tuple (S, A, P, R, T) where, S is a finite set of states, A is a finite set of actions, P is a transition probability function from state s to state s' after action a is taken, R is the immediate reward obtained after action a is made, and F is the set of decision epoch. Policy π is a mapping from a state to an action. The goal of MDP is to find an optimal policy to maximize or minimize a certain objective function. The MDP can be finite or infinite time horizon. For the finite time horizon MDP, an optimal policy π^* to maximize the expected total reward is defined as follows

$$\max V_\pi(s) = \mathbb{E}_{\pi,s}\left[\sum_{t=1}^{T} R\left(s_t' | s_t, \pi(a_t)\right)\right],$$

where s_t and a_t are the state and action at time t, respectively. For the infinite time horizon MDP, the objective can be to maximize the expected discounted total reward or to maximize the average reward. The former is defined as follows

$$\max V_\pi(s) = \mathbb{E}_{\pi,s}\left[\sum_{t=1}^{T} \gamma^t R\left(s_t' | s_t, \pi(a_t)\right)\right],$$

while the latter is expressed as follows

$$\max V_\pi(s) = \lim_{T\to\infty} \inf \frac{1}{T} \mathbb{E}_{\pi,s}\left[\sum_{t=1}^{T} \gamma^t R\left(s_t' | s_t, \pi(a_t)\right)\right].$$

Here, γ is the discounting factor.

3.4.2 Contributions

Using the features of the MDP, we solve the problem of video delivery in D2D environment. In this section, we jointly optimize these delivery decisions by maximizing the average video quality under the constraints on the playback delays of streaming users and the data rate guarantees for cellular vehicles. Depending on the channel and queue states of the users, the decision on the cache-enabled vehicle for video delivery is adaptively made based on the frame-based Lyapunov optimization theory by comparing the expected costs of vehicles. For each cache-enabled vehicle, the expected cost is obtained from the stochastic shortest path problem that is solved by DRL without the knowledge of the global CSI. MDP applies in this part. A framework of the compromising characteristics of the D2D underlaid cellular system, the vehicular network, and the wireless caching network is presented. For such a network, the joint optimization problem for association with the cache-enabled vehicle for delivering multimedia contents (e.g., video files) is formulated. The optimization problem maximizes the time-average video quality under the constraints on the limited playback delay of the DUE and the minimum data rate of the CUEs.

The problem of dynamic power allocations for both cellular and D2D links sharing the identical spectrum without the knowledge of global CSI is formulated based on a MDP and solved using the DRL approach. In contrast to the approaches in [RLL+15] and [LLX17], the proposed approach dynamically changes power allocations to control interference and to limit the playback delay based on channel statistics and queue states. In order to achieve efficient and improved learning of the delivery policy as well as to deal with the very large state space, we adopt a DDPG-based method because the state space is continuous and massively large. Considering the interference between the CUEs and DUEs, the decision on the cache-enabled vehicle that will deliver the content, i.e., the D2D transmitter, is made under the frame-based Lyapunov optimization theory [NS13], in larger timescale than power allocations of cellular and D2D links. With the help of the DRL-based power allocations determined in a smaller timescale, the node associations for content delivery can be also completed without global CSI.

3.4.3 Overall structure

D2D underlaid vehicular caching network where a certain vehicle user can request a video file form one of the cache-enabled vehicles in its vicinity while some CUEs are communicating with the BS, as shown in Figure 3.5.

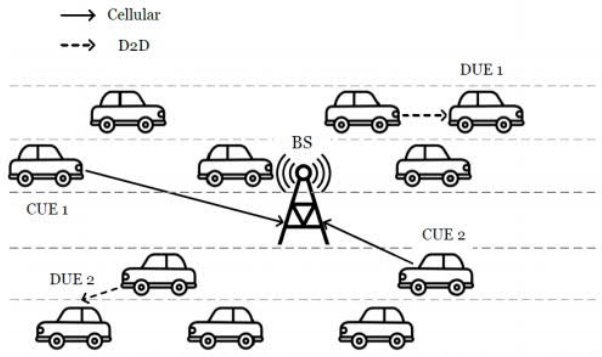


Figure 3.5 D2D underlaid cache-enabled vehicular network.

3.4.3.1 User queue model

Since a video file consists of many sequential chunks, the user receives the file from the cache-enabled vehicle and processes data for video streaming services in chunks. The quality of each chunk can differ in dynamic streaming and playback delay occurs when the chunk to be played has not yet arrived in the queue. Therefore, we focus on limiting the queueing delay by dynamically adjusting the queue backlogs of the streaming user. The queue dynamics of the DUE in each time slot can be represented as follow

$$Q(t+1) = \max\{Q(t+1) + \mu(t) - c, 0\}.$$

The arrival process is

$$\mu(t) = \left\lfloor \frac{R_d(\alpha(t), P_c(t), P_d(t), t) \cdot t_c}{S\left(q(\alpha(t))\right)} \right\rfloor.$$

3.4.3.2 Channel model

A Rayleigh fading channel is assumed for the wireless link from all vehicles to infrastructures and vehicles. We denote the channel gain by $h = \sqrt{X}\beta g$, where $X = A/d^{\gamma}$ controls path loss with d, A, and γ being the server-user distance, the path loss component, and the decay exponent. In addition, β is a long-normal shadowing RV with the standard deviation ξ, and $g \sim \mathcal{CN}(0,1)$ represents the fast fading component.

3.4.4 Dynamic node associations

Timescales of decisions on node association and power controls are different and formulates the optimization problem that maximizes the average video quality with the constraints of limited playback latency for DUEs and data rate guarantees for CUEs. Based on the average video quality, the playback delay of the streaming user, and the data rate of the CUE, the spectrum of which is reused by the streaming user, we can formulate the optimization problem that maximizes the long-term average video quality constrained by the need to avert queue emptiness and guarantee the data rate of the CUE.

3.4.5 Decision on cache-enabled vehicle for video delivery

3.4.5.1 Modelling of Markov decision process

In order to prevent the queue from becoming empty, the optimization is solved based on the Lyapunov optimization theory. According to the Lyapunov function, the drift-plus-penalty algorithm of the $k\text{-}th$ frame can be formulated as follows

$$\{P_{d,k}^{*}, P_{d,k}^{*}\} = \arg\min \mathcal{D}\left(\alpha_k, \Theta(t_k), P_{d,k}, P_{c,k}\right)$$

$$s.t. \quad 0 \le P_d \le P_0^d \\ 0 \le P_c \le P_0^c.$$

The above problem can be modelled by a MDP. The MDP is defined as $\mathcal{M} = \{S, A, T, r\}$, where S denotes the state space, A denotes the action space, T denotes the transition model and r denotes the reward structure. The queue backlog set of $\Theta(t_k)$ represents the current state that satisfies the Markov property. The state space is $\mathbb{R}^+$, where $\mathcal{Z}(t) \in \mathcal{Z} = \{0,1,\dots,Q\}$ and $W(t) \in \mathbb{R}^+$. The action set consists

of power allocations for the D2D transmitter and the CUE, i.e., $P_d(t)$ and $P_c(t)$ for $t \in \mathcal{T}_k$.

Denote the action at slot t by $\Xi(t) = [P_c(t), P_d(t)]$. By letting the action space be $\mathcal{A} \in [0, P_0^d] \times [0, P_0^c]$, the constraints are satisfied. Let power allocations for both the CUE and the DUE be uniformly discretized into $N_A + 1$ levels, and the finite action space be represented by

$$\mathcal{A} = \left\{0, \frac{P_0^d}{N_A}, \dots, \frac{(N_A-1)P_0^d}{N_A}\right\} \times \left\{0, \frac{P_0^C}{N_A}, \dots, \frac{(N_A-1)P_0^C}{N_A}\right\}$$

The action decisions are made over the $k\text{-}th$ frame, i.e., $t \in \mathcal{T}_k$, and the reward (i.e., incurred cost with the negative sign) at each slot $t \in \mathcal{T}_k$ is represented by

$$r\big(\Theta(t_k), \Xi(t)\big) = Z(t)\left(c - \left\lfloor \frac{t_c R_d(\alpha_k, P_d, P_c, t)}{S(q(\alpha_k))} \right\rfloor\right) + \gamma W(t)\big(\eta_c - R_c(P_c, P_d, t)\big) - VT \cdot \mathcal{P}\big(q(\alpha_k)\big);$$

Therefore, the reward r is the cost multiplied by the negative sign. At every slot t, channel gains are randomly generated, and state transitions occur according to random network events and the current queue state of $\Theta(t)$. The transition from $\Theta(t)$, to $\Theta(t+1)$ is defined as

$$P_{s's}(\xi) = \Pr\{\Theta(t+1) = s' | \Theta(t) = s, \Xi(t) = \xi\}, \text{ for all states } s, s' \in S \text{ and } \xi \in A.$$

According to Bellman optimality equation, the minimum incurred cost at $\Theta(t_0) = s_0$ is given by

$$\min_{\Xi} \mathbb{E}\left[\sum_{t=t_0}^{T} r\big(\Theta(t), \Xi(t)\big)\Big|_{\Theta(t_0)=s_0}\right] = \min_{\Xi} \mathbb{E}[r(s_0, \xi) + (\Theta(t+1) | \Theta(t) = s_0, \Xi(t) = \xi)]$$

$$= \min_{\Xi} \mathbb{E}\left[r(s_0, \xi) + \sum_{s \in S} P_{s,s_0}(\xi) G(s) | \Theta(t_0) = s_0, \Xi(t_0) = \xi\right].$$

Note that the channel information is not known, and the state transition probabilities are not given; therefore, we solve the problem using a DRL algorithm. Based on the finite MDP, the goal of reinforcement learning is to train a policy $\pi \in \Pi: S \times A \to [0,1]$ which gives all action candidates at every state the probability values in [0,1]. Policy π maps the state of the environment to the action to maximize the expected reward. Denote the expected reward under the policy π by $\mathcal{J}(\pi)$. With finite T steps, $\mathcal{J}(\pi)$ can be described as the accumulation of the reward at each time step

$$\mathcal{J}(\pi) = \mathbb{E}\left[\sum_{t=0}^{T} \delta^t r\big(\Theta(t), \Xi(t)\big)\Big|_{\pi}\right],$$

where δ is a discount factor that adjusts the effect of future rewards to the current decision. The optimal policy π^* is $\pi^* = \arg\max_{\pi} \mathcal{J}(\pi)$. In DRL, the policy π is approximated by parameter θ. The state sequence of s that is generated according to the policy π_θ is the distribution. Then, the expected reward obtained by the state sequence s and the policy π_θ can be denoted as $\mathcal{J}\big(s, \pi_\theta(s)\big)$, and the objective reinforcement learning is formulated as: $\arg\max_{\theta} \mathbb{E}_{s\sim\pi_\theta}\big[\mathcal{J}\big(s, \pi_\theta(s)\big)\big]$.

3.4.6 Simulation results

To verify the advantages of the proposed dynamic video delivery policy, we compared the proposed

scheme with four other schemes: “Genie-aided”, “The scheme presented in [LLX17]”. “Nearest”, “Highest-Qual”.

A comparison of our scheme with the “Genie-aided” scheme and the scheme presented can provide specific insight into the effectiveness of DRL-based power allocations.

The performance comparison of the proposed cache-enabled vehicle association for video delivery based on frame-based Lyapunov optimization with the “Nearest” and “Highest-Qual” schemes shows its advantages.

Figure 4.6 shows the plots of these performance metrics (data rate of the CUE, playback delay at the DUE, time-average video quality) versus the transmit power budgets of both cellular and DUEs.

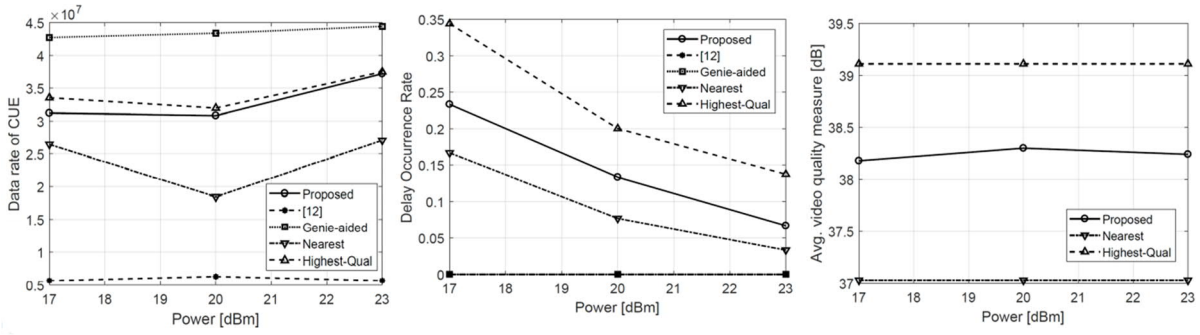


Figure 3.6 Data rate of CUE, delay occurrence rates, Average video quality (vs. P_0)

The method for the joint optimization of three decisions having different timescales in D2D underlaid cellular and vehicular caching networks was proposed

1. Association with a cache-enabled vehicle to allow video delivery
2. Power allocation for the DUE
3. Power allocation for the CUE.

The proposed algorithm maximizes the long-term time averaged video quality while limiting the playback delay and guaranteeing the data rate of the CUE, given the spectrum reuse policy. The decision on the cache-enabled vehicle to be used for video delivery is achieved by using the frame-based Lyapunov optimization theory under consideration of the interference signal from the CUE. The dynamic power allocations of both the CUEs and DUEs are obtained by using the DRL approach in the absence of knowledge of channel fast fading. Our intensive simulation results verify that the proposed algorithm effectively achieves a balanced trade-off between the data rate of the CUE, the playback delay occurrence of video streaming, and the average video quality.

3.5 Summary

This chapter studied data (handing over of data) and service (computation offloading) as yet another category of AI for edge. More specifically, we studied two computation offloading algorithms, namely, flexSensing as well as a decentralized task offloading algorithm. As well as dynamic video delivery in wireless caching networks.

In section 3.2, we proposed FlexSensing, a QoI- and processing-latency-aware task allocation scheme for drone-based visual crowdsourcing. This scheme aims at increasing the QoI while reducing the processing latency of crowdsourced visual data, taking into account the variation in the VFN workload

and drone mobility. We addressed the problem of seeking the optimal task allocation strategy via the formulation of an MDP and solve it through the DQN.

Section 3.3 is to propose adaptive learning based decentralized task offloading algorithm to optimize the average offloading cost. The proposed algorithm enables each agent to learn the service capability of each available fog node without frequent exchange of state information and achieve exploitation-exploration balance, which is further improved by considering input data size. At the same time, some properties could be satisfied, such as privacy by bandit feedback and security against oblivious attack by adversarial MAB, low complexity by sequential decision making, low signalling overhead by learning state information indirectly, instead of obtaining them from signal message.

In section 3.4, a method for the joint optimization of three decisions having different timescales in D2D underlaid cellular and vehicular caching networks were proposed: 1) association with a cache-enabled vehicle to allow video delivery, 2) power allocation for the DUE, and 3) power allocation for the CUE. The proposed algorithm maximizes the long-term time averaged video quality while limiting the playback delay and guaranteeing the data rate of the CUE, given the spectrum reuse policy. The decision on the cache-enabled vehicle to be used for video delivery is achieved by using the frame-based Lyapunov optimization theory under consideration of the interference signal from the CUE. The dynamic power allocations of both the CUEs and DUEs are obtained by using the DRL approach in the absence of knowledge of channel fast fading. Our intensive simulation results verify that the proposed algorithm effectively achieves a balanced trade-off between the data rate of the CUE, the playback delay occurrence of video streaming, and the average video quality.

4 Topology of edge

4.1 Overview

This chapter is studies the topology of edge which concerns wireless networking. Here topology refers to various ways the wireless communication signals are used to transfer information or power, their optimization techniques and methodologies, signalling, scheduling, wireless communication hardware and approaches to combat non-idealities thereof. Figure 4.1 is a schematic illustrating some of the problems discussed and some of the technologies demonstrated in this chapter.

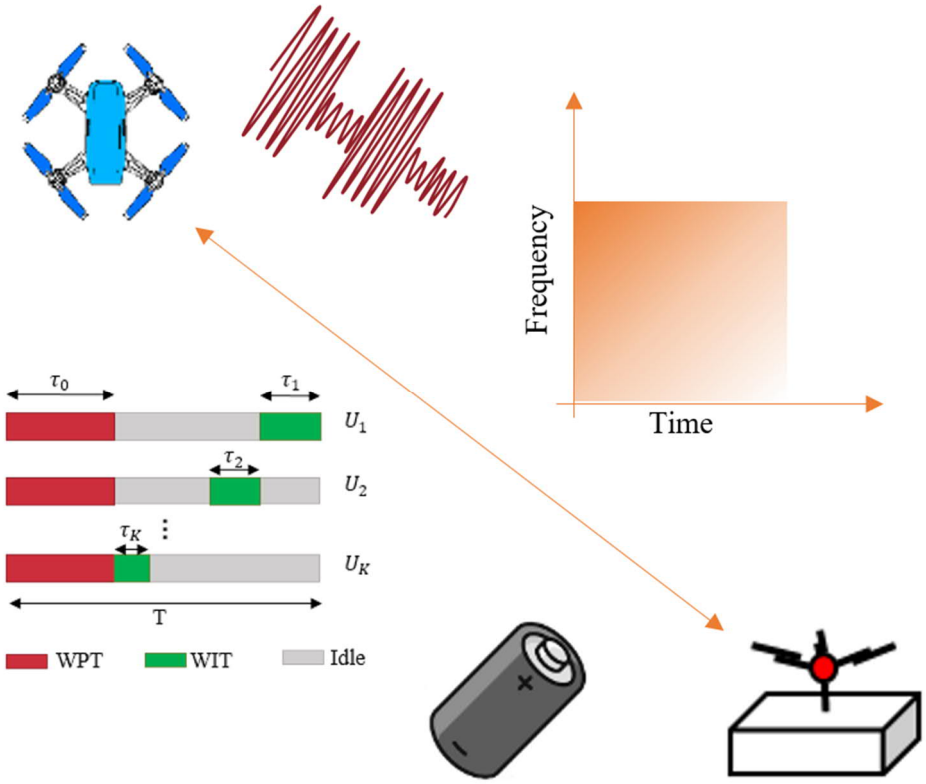


Figure 4.1 Schematic of a UAV and a sensor that jointly use AI to learn the dynamic wireless environment and optimize their operations accordingly.

In the following sections of this chapter, we design a meta-learning algorithm that can learn to demodulate from few pilots and a resource allocation procedure in wireless-powered communication networks using deep deterministic policy gradient.

Section 4.2 considers an IoT scenario in which devices sporadically transmit short packets with few pilot symbols over a fading channel. Devices are characterized by unique transmission non-idealities, such as I/Q imbalance. The number of pilots is generally insufficient to obtain an accurate estimate of the end-to-end channel, which includes the effects of fading and of the transmission side distortion. We propose to tackle this problem by using meta-learning. Accordingly, pilots from previous IoT transmissions are used as meta-training data in order to train a demodulator that is able to quickly adapt to new end-to-end channel conditions from few pilots. Various state-of-the-art meta-learning schemes are adapted to the problem at hand and evaluated, including MAML, FOMAML, REPTILE, and fast

CAVIA. An offline solution is developed. Numerical results validate the advantages of meta-learning as compared to training schemes that either do not leverage prior transmissions or apply a standard joint learning algorithms on previously received data.

In section 4.3, we study a WPCN composed of several nodes that rely on a HAP for their supply of energy. While the dominant focus of the literature has been on maximizing the throughput of this system in an AWGN channel, we optimize the performance of this system in a fading channel. In order to do so, we use reinforcement learning which can take a sequence of decisions to reach a higher long-term reward. In view of its ability to deal with continuous states and actions, we use the DDPG method to optimize this system. Thanks to its ability to deal with continuous states and actions, the DDPG algorithm can deal with the curse of dimensionality of this problem and solve it for a reasonable number of nodes. More specifically, simulation results confirm that the DDPG algorithm can easily learn to optimize a WPCN of five nodes and achieve a higher sum-rate compared with the traditional optimization schemes.

4.2 Learning to demodulate from few pilots via meta-learning

4.2.1 Overview

4.2.1.1 Motivation

For many standard channel models, such as additive Gaussian noise and fading channels with receive CSI, the design of optimal demodulators and decoders is well understood. Most communication links hence use pilot sequences to estimate CSI, which is then plugged into the optimal receiver with ideal receive CSI (see, e.g., [ODS+19]). This standard model-based approach is inapplicable if either of the following is true

1. An accurate channel model is unavailable;
2. The optimal receiver for the given transmission scheme and channel is of prohibitive complexity or unknown.

Examples of both scenarios are reviewed in [I20, S18a], and include new communication set-ups, such as molecular channels, which lack well-established models; and links with strong non-linear components, such as satellite channels with non-linear transceivers, whose optimal demodulators can be highly complex [I20, BRC98]. This observation has motivated a long line of work on the application of ML methods to the design of demodulators or decoders, from the 90s [I20] to many recent contributions, including [OH17, DSH+17, KAH+19] and references therein.

Demodulation and decoding can be interpreted as classification tasks, where the input is given by the received baseband signals and the output consists of the transmitted symbols for demodulation, and of the transmitted binary messages for decoding. Pilot symbols can hence be used as training data to carry out the supervised learning of a parametric model for the demodulator or decoder, such as SVMs or NNs. The performance of the trained “machine" as a demodulator or a decoder generally depends on how representative the training data are for the channel conditions encountered during test time and on the suitability of the parametric model in terms of trade-off between bias and variance [HTF09].

To the best of our knowledge, all of the prior works reviewed above assume that training is carried out using pilot signals from the same transmitter whose data is to be demodulated or decoded. This generally requires the transmission of long pilot sequences for training. In this section, we consider an IoT-like scenario, illustrated in Figure 4.2, in which devices sporadically transmit short packets with few pilot symbols. The number of pilots is generally insufficient to obtain an accurate estimate of the end-to-end channel, which generally includes the effects of fading and of the transmitter’s non-linearities [HDA17]. We propose to tackle this problem by using *meta-learning* [T98].

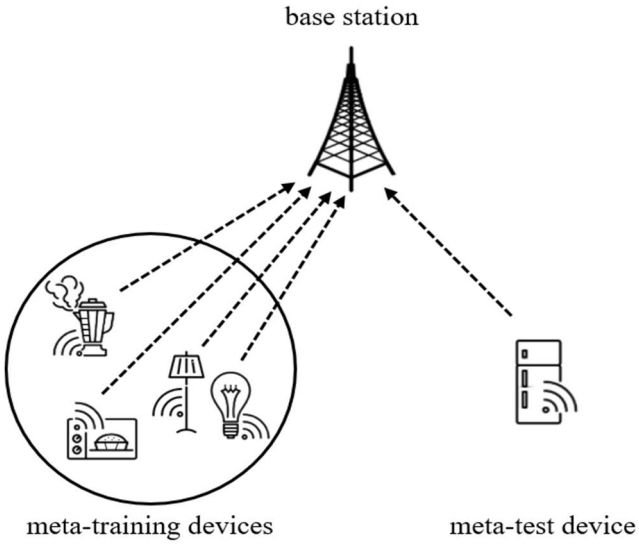


Figure 4.2 Illustration of few-pilot training for an IoT system via meta-learning

4.2.1.2 Meta-learning

Meta-learning, also sometimes referred to as “learning to learn", aims at leveraging training and test data from different, but related, tasks for the purpose of acquiring an inductive bias that is suitable for the entire class of tasks of interest [T98]. The inductive bias can be optimized by selecting either a model class, e.g., through a feature extractor, or a training algorithm, e.g., through an initialization of model parameters or the learning rate [GFL+18, SPK20]. An important application of meta-learning is the acquisition of a training procedure that allows a quick adaptation to a new, but related, task using few training examples, also known as *few-shot learning* [VBL+16]. For instance, one may have training and test labelled images for binary classifiers of different types of objects, such as cats vs dogs or birds vs bikes. These can be used as meta-training data to quickly learn a new binary classifier, say for handwritten binary digits, from few training examples.

Meta-learning has recently received renewed attention, particularly thanks to advances in the development of methods based on Stochastic Gradient Descent (SGD), including MAML [FAL17], REPTILE [NAS18], and fast CAVIA [ZSK+19]. Such techniques can be generally classified as either *offline*, in which case the meta-training data is fixed and given [FAL17, NAS18, ZSK+19]; or *online*, in which case all prior data from related tasks is treated as meta-training data in a streaming fashion [FRK+19].

4.2.1.3 Main contributions

As illustrated in Figure 4.3 and Figure 4.4, the key idea of this section is to use pilots from previous transmissions of other IoT devices as meta-training data in order to train a procedure that is able to quickly adapt a demodulator to new end-to-end channel conditions from few pilots.

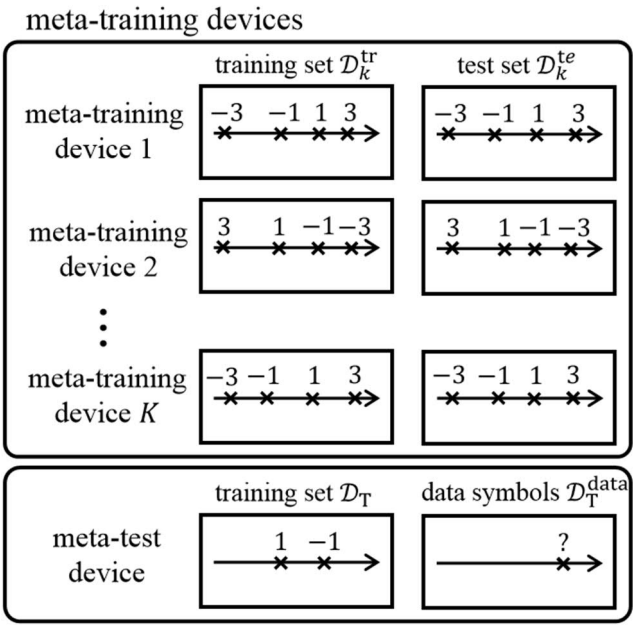


Figure 4.3 Offline meta-learning: Meta-training and meta-test data for 4-PAM transmission from set $S = \{-3, -1, 1, 3\}$.

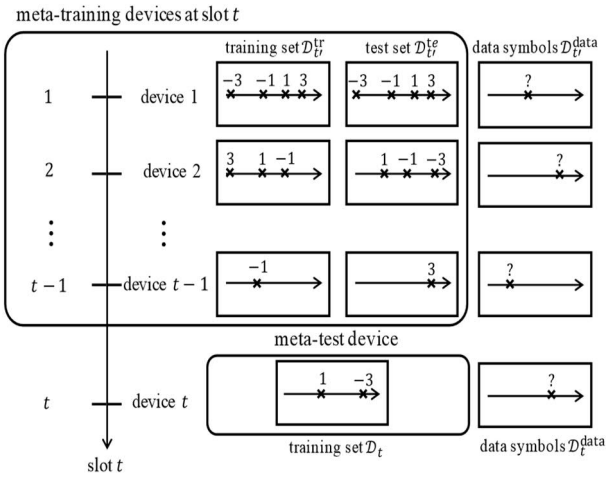


Figure 4.4 Online meta-learning: Meta-training and meta-test data for 4-PAM transmission from set $\boldsymbol{S} = \{-\mathbf{3}, -\mathbf{1}, \mathbf{1}, \mathbf{3}\}$.

Here, we focus on an offline algorithm, in which the set of previous transmissions is fixed. The main contributions are as follows

- We adapt to the problem at hand a number of state-of-the-art offline meta-learning solutions, namely MAML [FAL17], FOMAML [FAL17], REPTILE [NAS18], and CAVIA [ZSK+19]. Their relative merits and a unified interpretation in terms of the EM algorithm are discussed;
- We validate the advantage of meta-learning with extensive numerical results that provide comparisons with conventional model-based and learning-based communication schemes. A comparative study of the performance of various meta-learning solutions is also presented;

The results in this section have been partially presented in [PJS+19]. In particular, reference [PJS+19] derives an offline MAML based algorithm, and offers some preliminary numerical results. As compared to the preliminary conference version [PJS+19], this section presents additional analysis, including a general framework for meta-learning based on EM; novel algorithms, introducing a comprehensive evaluation of a larger number of meta-learning offline schemes; and more extensive discussions in terms of both algorithm definition and extra experiments.

4.2.1.4 Related works

In [JKA+19], which is concurrent to [PJS+19], the authors train a NN-based decoder that can adapt to the new channel condition with a minimal number of pilot symbols using meta-learning via FOMAML. In [MLL19], the authors train a NN-based channel estimator in OFDM system with meta-learning via FOMAML in order to obtain an effective CE given a small number of pilots. After the first submission of this work, several additional works have considered meta-learning for communication. Reference [SPK20] provides a review of meta-learning with applications to communication systems. In [YGZ+19], meta-learning is used for DL/UL channel conversion in Frequency-Division Duplex massive MIMO channels. References [PSK20a] and [PSK20b] consider meta-learning for end-to-end training of physical layer with and without a channel model, respectively. Finally, reference [GAH20] considers a related approach based on hypernetworks to aid NN-based MIMO detection.

The rest of the section is organized as follows. In subsection 4.2.2, we detail system model and offline meta-learning problem. In subsection 4.2.3, various offline meta-learning solutions are covered within a unified interpretation. Numerical results are presented in subsection 4.2.4.

4.2.2 Model and problem

4.2.2.1 System model

In this subsection, we consider the IoT system illustrated in Figure 4.2, which consists of a number of devices and a BS. For each device k, the complex symbol transmitted by the device and the corresponding received signal at the BS are denoted as $s_k \in S$ and y_k, respectively.

We also denote by S the set of all constellation symbols as determined by the modulation scheme. The end-to-end channel for a device k is defined as

$$y_k = h_k x_k + z_k, \tag{4-1}$$

where h_k is the complex channel gain from device k to the BS, which is constant over a transmission block according to the standard quasi-static fading model typically assumed for short-packet transmissions; $z_k \sim \mathcal{CN}(0, N_0)$ is additive white complex Gaussian noise; and

$$x_k \sim p_k(\cdot\,|s_k) \tag{4-2}$$

is the output of a generally random transformation defined by the conditional distribution $p_k(\cdot\,|s_k)$. This conditional distribution accounts for transmitter's non-idealities such as phase noise [CP02], I/Q imbalance [WF05], and amplifier's characteristics [HDA17] of the IoT device. Throughout this section for each device k, we assume pilots and data symbols to follow the same constellation S and to be subject to the transmitter's non-idealities defined by $p_k(\cdot\,|s_k)$. The average transmitted energy per symbol is constrained as $\mathbb{E}[|x_k|^2] \leq E_x$ for some positive value E_x for both pilot and data symbols. As an example for the transmitter's non-idealities (2), a common model for the I/Q imbalance assumes the following transformation [TM07]

$$x_k = (1+\epsilon_k)\cos\delta_k\,\mathcal{R}e\{s_k\} - (1+\epsilon_k)\sin\delta_k\,\mathcal{I}m\{s_k\} + j\big((1-\epsilon_k)\cos\delta_k\,\mathcal{I}m\{s_k\} - (1-\epsilon_k)\sin\delta_k\,\mathcal{R}e\{s_k\}\big), \quad (4\text{-}3)$$

where ϵ_k and δ_k represent the amplitude imbalance factor and phase imbalance factor, which are real constants or RVs. Note that we only explicitly model imperfections at the transmitter side. This is because, in practice, non-idealities on the receiver processing chain at the BS are expected to be much less significant than the mentioned non-idealities for IoT devices. Furthermore, receiver-side non-linearities at the BS can be also mitigated through offline designs prior to deployment.

Based on the reception of few pilots from a target device, we aim at determining a demodulator that correctly recovers the transmitted symbol s from the received signal y with high probability. The demodulator is defined by a conditional probability distribution $p(s|y,\phi)$, which depends on a trainable parameter vector ϕ.

4.2.2.2 Meta-learning problem

Following the nomenclature of meta-learning [FAL17], the target device is referred to as the *meta-test device*. To enable few-pilot learning, we assume here that the BS can use the signals received from the previous pilot transmissions of K other IoT devices, which are referred to as *meta-training devices* and their data as *meta-training data*. Specifically, as illustrated in Figure 4.3, the BS has *N* pairs of pilot s_k and received signal y_k for each meta-training device $k = 1, \dots, K$. The meta-training dataset is denoted as $D = \{D_k\}_{k=1,\dots,K}$, where $D_k = \{(s_k(n), y_k(n)) : n = 1, \dots, N\}$, and $(s_k(n), y_k(n))$ are the $n\text{-}th$ pilot-received signal pairs for the $k\text{-}th$ meta-training device. This scenario is referred to as offline meta-learning since the meta-training dataset D is fixed and given.

For the target, or the meta-test, device, the BS receives *P* pilot symbols. We collect the P pilots received from the target device in set $D_T = \{(s^{(n)}, y^{(n)}) : n = 1, \dots, P\}$. The demodulator can be trained using meta-training data D and the pilot symbols D_T from the meta-test device.

Training requires the selection of a parametric model $p(s|y,\phi)$ for the demodulator. The choice of the parametric model $p(s|y,\phi)$ should account for the standard trade-off between capacity of the model and overfitting [B06, S18b]. To fix the ideas, we will assume that the demodulator $p(s|y,\phi)$ is given by a multi-layer NN having L layers, with a softmax non-linearity in the final, $L\text{-}th$, layer. This can be written as

$$p(s|y,\varphi) = \frac{\exp\left(\left[f_{\{\varphi^{(L-1)}\}}\left(f_{\{\varphi^{(L-2)}\}}\left(\cdots f_{\{\varphi^{(1)}\}}(y)\right)\right)\right]_s\right)}{\sum_{s'\in S}\exp\left(\left[f_{\{\varphi^{(L-1)}\}}\left(f_{\{\varphi^{(L-2)}\}}\left(\cdots f_{\{\varphi^{(1)}\}}(y)\right)\right)\right]_{s'}\right)}, \quad (4\text{-}4)$$

where $f_{\varphi^{(l)}}(x) = \sigma\left(W^{(l)}x + b^{(l)}\right)$ represents the non-linear activation function of the $l\text{-}th$ layer with parameter $\phi(l) = \{W(l), b(l)\}$, with $W^{(l)}$ and $b^{(l)}$ being the weight matrix and bias vector of appropriate size, respectively; $[\cdot]_s$ stands for the element regarding s; and $\phi = \{\phi^{(l)}\}_{l=1,\dots,L-1}$ is the vector of parameters. The non-linear function $\sigma(\cdot)$ can be, e.g., a ReLU or a hyperbolic tangent function. The input y in (4-4) can be represented as a two-dimensional vector comprising real and imaginary parts of the received signal.

4.2.3 Meta-learning algorithms

In this subsection, we adapt state-of-the-art offline meta-learning algorithms to design the demodulator in (4-4) given meta-training and meta-test data. As discussed earlier, we view demodulation as a

classification task. To set the notation, for any set D_0 of pairs (s, y) of transmitted symbol s and received signal y, the standard cross-entropy loss function is defined as a function of the demodulator parameter vector ϕ as

$$f_{\varphi^{(l)}}(x) = \sigma\left(W^{(l)}x + b^{(l)}\right). \quad (4\text{-}5)$$

4.2.3.1 Joint training

As a benchmark, we start by considering a conventional approach that uses the meta-training data D and the training data D_T for the *joint training* of the model $p(s|y, \phi)$. Joint training pools together all the pilots received from the meta-training devices and the meta-test device, and carries out the optimization of the cumulative loss $L_{\{D \cup D_T\}}(\varphi)$ in (4-5) using SGD. Accordingly, the parameter vector ϕ is updated iteratively based on the rule

$$\varphi \leftarrow \varphi + \eta \nabla_{\varphi} \log p\left(s^{(n)}|y^{(n)}, \varphi\right), \quad (4\text{-}6)$$

by drawing one pair $\left(s^{(n)}, y^{(n)}\right)$ at random from the set $D \cup D_T$. In (4-6), the step size η is assumed to be fixed for simplicity of notation but it can in practice be adapted across the updates (see, e.g., [GBC16]). Furthermore, this rule can be generalized by summing the gradient in (6) over a mini-batch of pairs from the dataset $D \cup D_T$ at each iteration [GBC16].

4.2.3.2 A unified view of meta-learning

A useful way to introduce meta-learning in terms of the graphical model is illustrated in Figure 4.5 Accordingly, meta-learning assumes a demodulator $p(s|y, \varphi, \theta)$ that depends on a shared parameter θ common to all tasks, or users, and on a latent context variable φ, which is specific to each user. The specific parameterization $p(s|y, \varphi, \theta)$ and its relationship with (4-4) depend on the meta-learning scheme, and they will be discussed below. Note that, as illustrated in Figure 4.5, the context vector φ is assumed to be random, while θ is a shared (deterministic) parameter. Furthermore, from Figure 4.5, the shared variable θ can also affect the prior distribution of the context variable φ. In this framework, the key idea is that meta-training data D is used to estimate the shared parameters θ via the process of meta-learning, while the context variable φ is inferred from the meta-test data D_T.

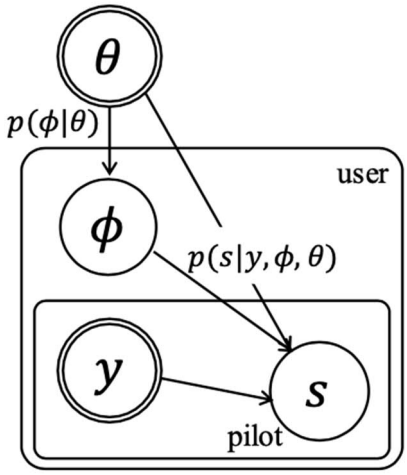


Figure 4.5 Graphical model assumed by meta-learning: The demodulator $\boldsymbol{p(s|y, \varphi, \theta)}$ depends on a user-specific, or context, RV $\boldsymbol{\varphi}$, as well as on a shared parameter $\boldsymbol{\theta}$, which may also affect the prior

distribution of the context variable $\boldsymbol{\varphi}$.

To elaborate, a principled way to train the model in Figure 4.5 would be to estimate parameter θ using the EM algorithm based on the meta-training data D. The EM algorithm is in fact the standard tool to tackle the problem of maximum likelihood estimation in the presence of latent variables, here the context variable φ (see, e.g., [B06, S18b, KF09]). EM maximizes the sum of marginal likelihoods

$$p(s|y,\theta) = \mathbb{E}_{\phi\sim p(\phi|\theta,\mathcal{D}_k)}[p(s|y,\phi,\theta)] \tag{4-7}$$

over the data pairs (s,y) from all data sets D_k in the meta-training data set D. In (4-7), the expectation is taken with respect to the posterior distribution $p(\varphi|\theta,D_k)$ of the context variable given the training data D_k of the k-th meta-training device. After EM training, one can consider the obtained parameter θ as fixed when inferring a data symbol s given a new observed signal y and the pilots D_T for the meta-test device. This last step would ideally yield the demodulator

$$p(s|y,\theta) = \mathbb{E}_{\phi\sim p(\phi|\theta,\mathcal{D}_T)}[p(s|y,\phi,\theta)], \tag{4-8}$$

where the average is taken over the posterior distribution $p(\varphi|\theta,D_T)$ of the context variable given the training data of the meta-test device.

The computation of the posteriors $p(\varphi|\theta,D_k)$ in (4-7) and $p(\varphi|\theta,D_T)$ in (4-8) is generally of infeasible complexity. Therefore, state-of-the-art meta-learning techniques approximate this principled solution by either employing point estimate of latent context variable φ [FAL17, NAS18, ZSK+19] or by a direct approximation of its posterior distribution [RB19, GBB+19, NDC19]. In this section, we focus on the more common point estimate based meta-learning techniques, which are reviewed next.

4.2.4 Results of meta-learning for Rayleigh fading with I/Q imbalance

In this subsection, we provide numerical results in order to bring insights into the advantages of meta-learning[1].

We consider a realistic scenario including Rayleigh fading channels $h_k \sim \mathcal{CN}(0,1)$ and model (4-3) to account for I/Q imbalance at the transmitters. We set $\epsilon_k = 0.15\epsilon'_k$ and $\delta_k = 15°\delta'_k$, where ϵ'_k~Beta(5,2) and δ'_k~Beta(5,2) are independent. Note that this implies that ϵ_k and δ_k are limited in the intervals [0,0.15] and [0,15°], respectively.

We assume 16-QAM for constellation S, and the sequence of pilot symbols in the meta-training dataset D and meta-test dataset D_T is fixed by cycling through the symbols in S, while the transmitted symbols in the test set for the metatest device are randomly selected from S. The number of meta-training devices is set as $K = 1000$; the number of pilot symbols per device is $N = 3200$, which are divided into N^{tr} training samples and $N^{\mathrm{te}} = N - N^{\mathrm{tr}}$ testing samples. The average SNR per complex symbol is given as E_x/N_0 = 20dB.

In Figure 4.6, the symbol error rate with respect to the number P of pilots for the meta-test device is illustrated when using an equal number of pilots for meta-training, i.e., $N^{\mathrm{tr}} = P$. As in Figure 4.4, we compare the performance of meta-training methods with conventional learning and joint training strategies, along with a conventional communication scheme based on MMSE CE with P pilots followed by a maximum likelihood demodulator. All the schemes with worse performance as compared to this

1 Code is available at https://github.com/kclip/ meta-demodulator.

conventional communication scheme are not shown. MAML, REPTILE, and CAVIA are seen to adapt quickly to the channel and I/Q imbalance of the target device, outperforming the conventional scheme based on MMSE CE, which is agnostic to the I/Q imbalance. MAML shows the best performance for sufficiently large P, while CAVIA is seen to outperform MAML when fewer pilots are available.

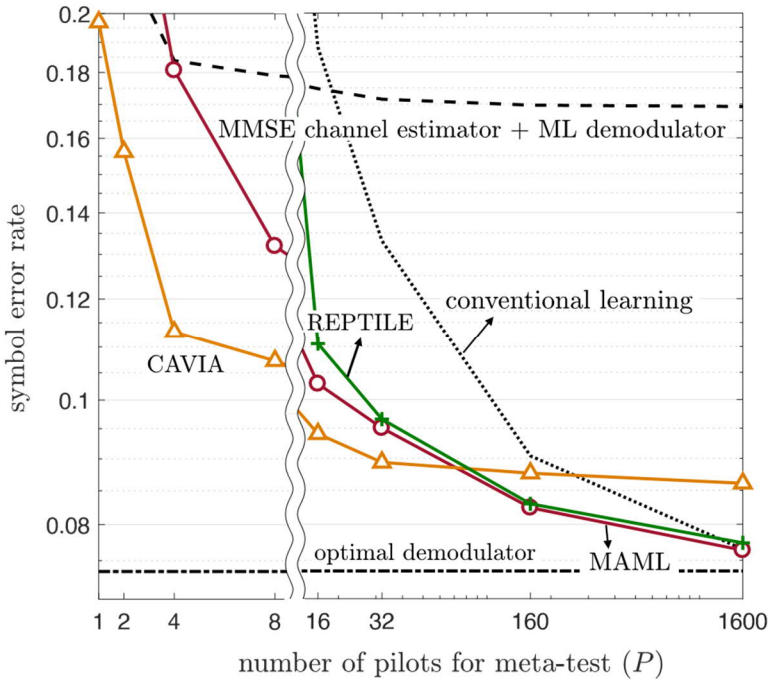


Figure 4.6 Symbol error rate with respect to the number $N^{\mathrm{tr}} = P$ of training pilots used for both meta-training and meta-testing for offline meta-learning with 16-QAM, Rayleigh fading, and I/Q imbalance with $K = 1000$ meta-training devices, $N^{\mathrm{tr}} + N^{\mathrm{te}} = 3200$ pilots for meta-training devices.

It is worth noting that, when there is a sufficient number P of pilots for the meta-test device, conventional learning can outperform meta-learning schemes. In this case, the inductive bias inferred by meta-training can hence cause a performance degradation [PSK20a].

In Figure 4.7, we consider the case where the number P of pilots for the meta-test device is different from the number N^{tr} used for meta-training, here set to $N^{\mathrm{tr}} = 4$. Despite this mismatch between meta-training and metatesting condition, MAML, CAVIA, and REPTILE are seen to outperform conventional communication when there is a sufficient number *P* of pilots for meta-test. In a manner similar to results in Figure 4.6, CAVIA shows the best performance with extremely few pilots, e.g., 4 pilots, while MAML is preferable for larger values of P. In fact, comparing Figure 4.6 and Figure 4.7 reveals that having $P > N^{tr}$ can even be advantageous for some meta-training schemes, such as CAVIA. This may be interpreted in terms of meta-overfitting, which refers to a degradation in meta-testing performance due to an excessive dependence of the meta-trained shared parameters on the meta-training data [AM17]. Using fewer pilots during meta-training can potentially reduce meta-overfitting, by making the shared parameters less dependent on meta-training data, and improve meta-testing performance.

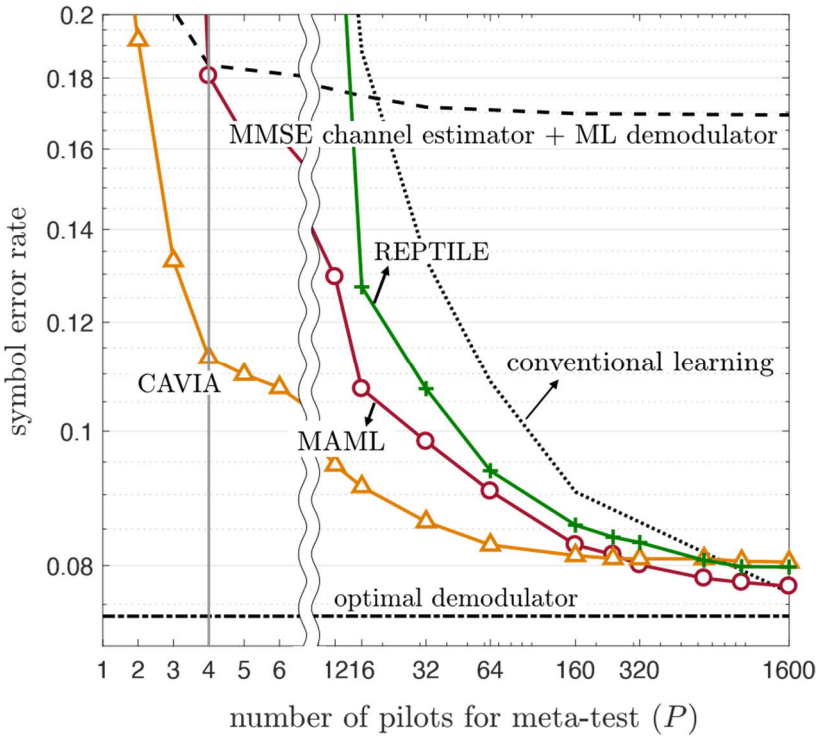


Figure 4.7 Symbol error rate with respect to the number P of pilots (used during meta-testing) for offline meta-learning with 16-QAM, Rayleigh fading, and I/Q imbalance for $K = 1000$ meta-training devices, N^{tr} = 4 (vertical line), N^{te} = 3196.

Finally, in Figure 4.8, the symbol error rate with respect to the number K of the meta-training devices is demonstrated. Joint training has a performance similar to conventional learning, hence being unable to transfer useful information from the K meta-training devices. In contrast, MAML and CAVIA show better performance when given data from more meta-training devices, up to a point where the gain saturates. This matches well with the intuition that there is only a limited amount of common information among different users that can be captured by meta-learning. Confirming the results in Figure 4.6 and Figure 4.7, MAML and CAVIA are seen to offer better performance than the conventional communication scheme with a sufficient number K of meta-training devices. Furthermore, CAVIA needs a larger value of K than MAML. This accounts again for CAVIA's architectural difference as compared to MAML: CAVIA needs to find a shared parameter vector θ for the demodulator $p(s|y, \varphi_{\mathrm{T}}, \theta)$ that is not adapted to the training symbols of the current device.

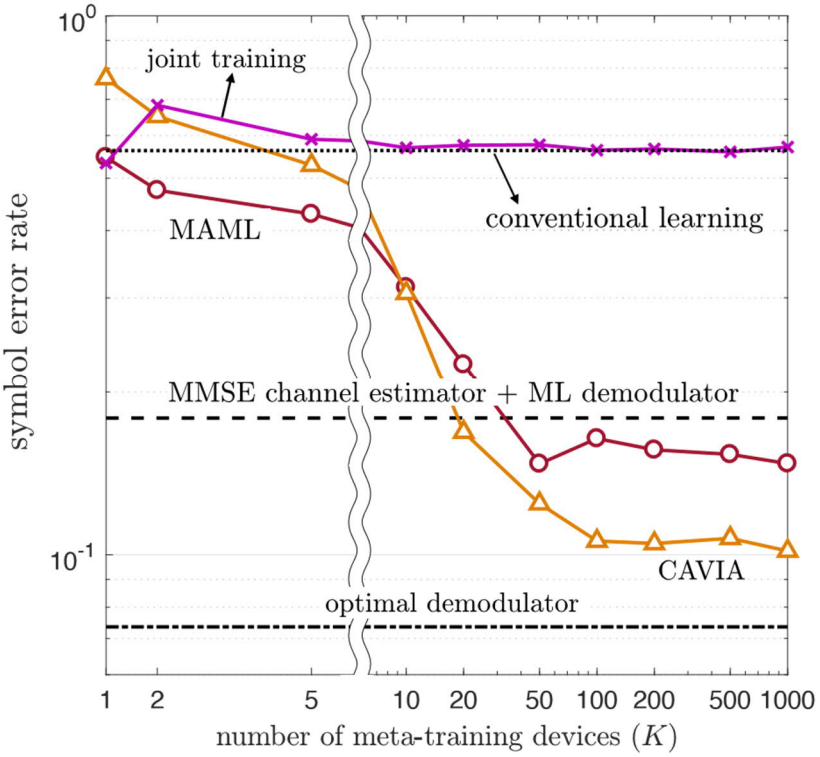


Figure 4.8 Symbol error rate with respect to number K of meta-training devices for offline meta-learning with 16-QAM, Rayleigh fading, and I/Q imbalance with N^{tr} = 4 and N^{te} = 3196 for meta-training devices, $P = 8$ pilots for meta-test devices.

4.3 Resource allocation in wireless-powered communication networks using deep deterministic policy gradient

4.3.1 Overview

Traditional mobile wireless devices rely on batteries for their supply of power. This has limited their application in areas where battery replacement or replenishment is difficult or even impossible. Lately, energy harvesting has become an alternative to sustain such wireless devices. These methods rely on natural resources such as wind or solar energy. However, these resources are uncontrollable and can be intermittent. An even more recent move toward a truly wireless network has been to harvest energy from far-field RF signals [JZ14]. The energy is transported using the RF energy from the AP to the nodes. The nodes, harvest and save the energy and send back information in return. This type of network is called the WPCN which has found application in many areas such as WSN, RFID systems, IoT networks, and perhaps more recently used in conjunction with UAVs. Examples of such applications are considered in [WYX19, PLE19, XXZ20] where UAV-mounted APs serve nodes which are located on the ground which are sometimes referred to as GTs.

WPCN resource allocation using conventional algorithms that rely on greedy optimization has been studied extensively in the literature, both originally [JZ14] as well as recently [ASP20, RNR19, ASP20]. Here, greedy refers to the HTT or the TWH protocols which maximize the throughput in every frame. The former, was first popularized in [JZ14] which proposed that the nodes first harvest the broadcast energy from the HAP, save the harvested energy, and then utilise all the harvested energy to transmit information in the UL. The latter protocol, i.e. the TWH protocol, allows the nodes to simultaneously harvest energy in the DL and utilise the harvested energy to transmit information in the UL. This approach, originally proposed in [ASP20], is only possible in FDD WPCN. Nevertheless, both these

techniques dictate that all the harvested energy in a single frame be consumed in the same frame, making them only optimal in AWGN channels. As was first argued in [BZ17], and later in [AP18], since these traditional methods make no assumptions about the variability of the channel and maximize the system throughput in every single frame, their approach is greedy and hence suboptimal.

Yet, higher expected long-term throughput may be gained by directly maximizing the long-term average throughput. This was considered in [BZ17, LGL19] where MDP was used to train a WPCN of two nodes. However, since the MDP cannot directly deal with continuous states and actions, to adapt this approach to the WPCN problem, the authors had to discretize the states and actions. This, in turn leads to the curse of dimensionality, making this solution exponentially more difficult to be applied to networks having more than two nodes. Another attempt to solve this problem using mathematical modelling of the ergodic throughput was proposed in [AP18]. The solution developed in this work, however, serves more as a theoretical bound or an upper limit of such a policy than a practical one.

It should be noted that, the key point in all such solutions is the emphasis on the long-term average throughput of the wireless networks in flat-fading channels, a notion which is properly called the ergodic throughput in communication systems. In this sense, reinforcement learning is relevant because of its ability to deal with a sequence of actions that leads to sacrifice of short-term reward for higher long-term reward. On the other hand, because the parameters in WPCN nodes are continuous, we use the DDPG algorithm [LHP+15] which can natively deal with such continuous problems without the need for discretization. The goal of the present work is to investigate such a problem. In short, in this section

1. The multi-node energy management problem in wirelessly powered communication problem is expressed as a reinforcement learning problem whose goal is to maximize the ergodic UL sum-rate. Our formulation makes the definitions of the states normalized which further helps the RL algorithm learn more efficiently.
2. We use the DDPG algorithm to calculate the optimal policy. In this algorithm, the actor net is in charge of learning the optimal policy, while the critic net prevents the curse of dimensionality by calculating the Q-value instead of using a Q-table.
3. We use simulation to confirm that the proposed policy can actually improve the conventional slot-oriented policy in a 5-node setup. The results show that, our algorithm yields higher expected sum-rate.

In subsection 4.3.2 we provide an overview of the DDPG. In subsection 4.3.3 we study the system model of the problem. After that, we formulate the WPCN resource allocation problem using the DDPG algorithm in subsection 4.3.4. We finally present the simulation results in subsection 4.3.5.

4.3.2 An overview of the deep deterministic policy gradient

Recently there has been growing interest in RL algorithms that can deal with continuous action and states. DDPG is one of such algorithms and is well suited for solving complex problems with many continuous action and states that would otherwise be nearly impossible to solve using other methodologies. DDPG [LHP+15] has two networks: The actor network which maps states to actions and generates a deterministic policy, and the critic network which maps states and actions to episode reward which simulates the Q-table.

These networks each have an online as well as a target network which have identical structures. We use Q for the online critic network and μ for the online actor network. Likewise, the target actor and critic networks are denoted by Q' and μ' respectively. Let θ represent the parameters in any of these networks. Along the same lines, we use θ^{μ} and θ^{Q} to denote the parameters of the online actor and critic networks respectively.

In DDPG, the Q-table equation is expressed as

$$Q^{\mu} = \mathbb{E}\left[r(\boldsymbol{s}[n], \boldsymbol{a}[n]) + \gamma\left[Q^{\mu}\left(\boldsymbol{s}[n], \mu(\boldsymbol{s}[n+1])\right)\right]\right], \quad (4\text{-}9)$$

where, n, as we will later elaborate, denotes the frame index, γ is the discount factor, and $r(\boldsymbol{s}[n], \boldsymbol{a}[n])$ is the reward of the state $\boldsymbol{s}[n]$ and action $\boldsymbol{a}[n]$. The loss of the critic network measures the difference between $Q(\boldsymbol{s}[n], \boldsymbol{a}[n]|\boldsymbol{\theta}^{Q})$ and $y[n]$; that is

$$L(\theta^{Q}) = \mathbb{E}[(Q(\boldsymbol{s}[n], \boldsymbol{a}[n]|\boldsymbol{\theta}^{Q}) - y[n])], \quad (4\text{-}10)$$

where $y[n]$ is defined as

$$[n] = r(\boldsymbol{s}[n], \boldsymbol{a}[n]) + \gamma Q(\boldsymbol{s}[n+1], \mu(\boldsymbol{s}[n+1])|\theta^{Q}). \quad (4\text{-}11)$$

In this structure, the actor network learns to perform the optimal action in every state while the critic network learns the long-term expected reward. As a result, the actor network can update the policy using the critic network as follows

$$\nabla_{\theta^{\mu}} J \approx \mathbb{E}\left[\nabla_{\boldsymbol{a}} Q(\boldsymbol{s}, \boldsymbol{a}|\boldsymbol{\theta}^{Q})|_{\boldsymbol{s}=\boldsymbol{s}[n], \boldsymbol{a}=\mu(\boldsymbol{s}[n])} \nabla_{\boldsymbol{\theta}^{\mu}} \mu(\boldsymbol{s}|\boldsymbol{\theta}^{\mu})|_{s=s[n]}\right]. \quad (4\text{-}12)$$

The training in DDPG is described as bellow [LHP+15]

1. Using $\mu(\boldsymbol{s}[n])$ from the actor net, the action is calculated by adding some noise $\boldsymbol{a}[n] = \mu(\boldsymbol{s}[n]) + \boldsymbol{m}[n]$, where $m[n]$ is the added noise.
2. Once applied to the environment, this action generates a reward $r[n]$ and a next state $\boldsymbol{s}[n+1]$.
3. This experience is saved in the replay buffer as the tuple $(\boldsymbol{s}[n], \boldsymbol{a}[n], r[n], \boldsymbol{s}[n+1])$.
4. A set of N randomly selected tuples are selected from the buffer as a mini-batch and applied to the actor and critic nets. The target actor net outputs action $\mu'(\boldsymbol{s}[n+1])$ from which the target critic net can calculate $y[n]$.
5. The optimizer can now update the online critic net.
6. The online actor net outputs the action $\boldsymbol{a} = \mu(\boldsymbol{s}[n])$ to the online critic net from which the action's gradient can be calculated as $\nabla_{a} Q(\boldsymbol{s}, \boldsymbol{a}|\theta^{Q})|_{\boldsymbol{s}=\boldsymbol{s}[n], \boldsymbol{a}=\mu(\boldsymbol{s}[n])}$.
7. The gradients of the online actor's gradient can be calculated as $\nabla_{\boldsymbol{\theta}^{\mu}} \mu(\boldsymbol{s}|\boldsymbol{\theta}^{\mu})|_{\boldsymbol{s}=\boldsymbol{s}[n]}$.
8. These two gradients can be used to update the online actor net.
9. The target actor and critic nets are softly updated using the following relations.

$$\boldsymbol{\theta}^{Q'} \leftarrow \tau\boldsymbol{\theta}^{Q} + (1-\tau)\boldsymbol{\theta}^{Q'}$$
$$\boldsymbol{\theta}^{\mu'} \leftarrow \tau\boldsymbol{\theta}^{\mu} + (1-\tau)\boldsymbol{\theta}^{\mu'},$$

were τ is a small constant.

4.3.3 System model

We consider a WPCN shown in Figure 4.9 consisting of one HAP, which is a DBS in this case and transfers power wirelessly in the DL to K nodes. These nodes harvest the transmitted power in the DL and transmit information in the UL using the harvested energy. All nodes and the HAP have single omnidirectional antennas. In what follows, we first describe the frame structure, then the energy, power and batteries, after that, the UL and DL communication channels, and finally the UL and DL phases.

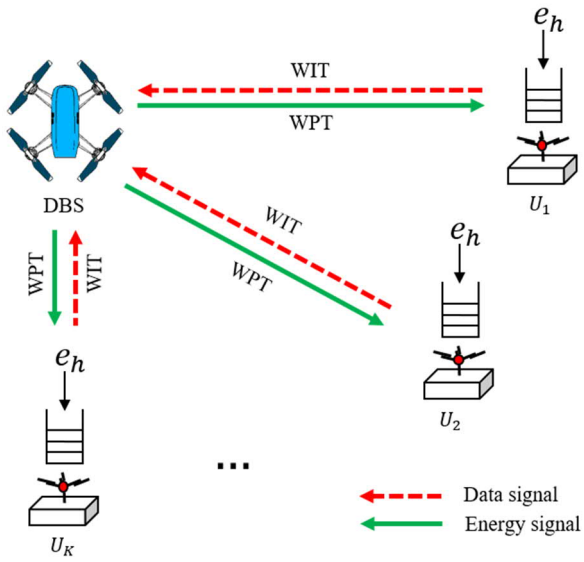


Figure 4.9 Schematic of the multimode WPCN.

4.3.3.1 Frame structure

The communication occurs in frames with indices denoted by n. As shown in Figure 4.10, each frame is of length T seconds and is divided into two phases. The WPT phase takes $\tau_0[n]T$ seconds and the WIT phase takes $(1-\tau_0[n])T$ seconds. The nodes use the WPT phase to harvest energy and the WIT phase to transmit their data to the DBS in a TDMA fashion. More specifically, the $k\text{-}th$ user transmits information with UL transmit power of $p_k[n]$ during $\tau_k[n]T$. Needless to say, $\sum_{k=0}^{K}\tau_k[n] = 1$, so that the whole frame is used.

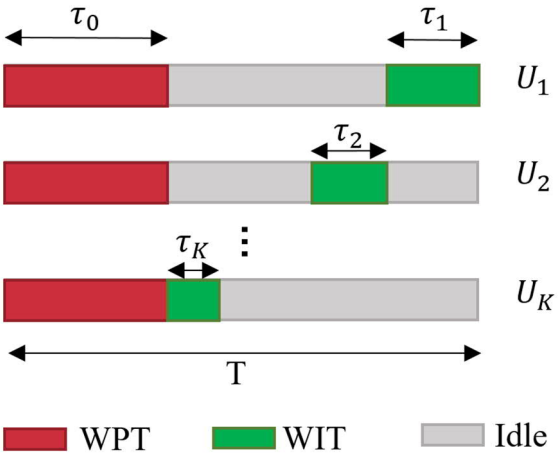


Figure 4.10 Frame structure of the multi-node WPCN using TDMA.

4.3.3.2 Energy, power, and batteries

Each node relies solely on wirelessly transmitted power from the HAP and is equipped with a built-in battery of capacity $\hat{b}_k$ that saves the harvested energy during the WPT phase. For simplicity, in this section, we assume all node batteries have the same capacity, and hence $\hat{b}_k = \hat{b}$.

At each frame, the battery receives $E_k^{\Sigma}[n]$ joules of energy and spends $E_k^{\Delta}[n]$. We presume the WPT phase is the first phase and the WIT phase is the second in each frame. As a result, the battery charge evolves according to the following equation

$$b_k[n] \leftarrow \max\{0, \min\{\hat{b}, b_k[n-1] + E_k^{+}[n]\}\}. \quad (4\text{-}13)$$

That is, the charge cannot go beyond the maximum value of $\hat{b}$ or become negative. In addition, the DBS and node antenna transmit power values should be less than or equal to the maximum allowed transmit power of $P_{\max}$ and $p_{\max}$ respectively. Since higher HAP transmit power always results in higher received power at the noes, there will be no disadvantage in HAP transmitting with the maximum allowed power. Hence, we assume the HAP's transmit power is $P_{\max}$.

4.3.3.3 Uplink and downlink channels

We assume the Rayleigh block fading model applies to both the DL and UL channels and that the reciprocity applies. At the $n\text{-}th$ frame, the channel power gain between the HAP and the $k\text{-}th$ node is described by $g_k[n]$, which is exponentially distributed as follows

$$g_k[n] \sim \text{Exp}(\mu_k), \quad (4\text{-}14)$$

in which μ_k is the large-scale fading coefficient that is modelled by $\mu_k = c_0 d_0^{\delta} d_k^{-\delta}$, where c_0 is a constant representing signal power attenuation at a reference distance of d_0, d_k is the distance of the $k\text{-}th$ node from the HAP, and $\delta > 0$ is the pathloss exponent.

4.3.3.4 Uplink phase

We assume that there is always enough data in the data queues of devices. Using the saved energy in the previous time frames, the $k\text{-}th$ node transmits $R_k[n]$ bits of information in the UL

$$R_k[n] = TB\tau_k[n] \log_2\left(1 + \frac{g_k[n]p_k[n]}{\sigma_0^2}\right), \quad (4\text{-}15)$$

where B is the channel bandwidth and σ^2 is the variance of the channel noise.

At each frame, node k uses $E_k^{\Delta}[n]$ joules of energy in the battery for UL information transmission, where

$$E_k^{\Delta}[n] = p_k[n]\tau_k[n]. \quad (4\text{-}16)$$

Depending on the state of the system, the UL transmission powers $p_k[n]$, $1 \le k \le K$ and the UL time allocations $\tau_k[n]$, $0 \le k \le K$ can change in every frame and are the optimization variables of the problem.

4.3.3.5 Downlink phase

The harvested energy at every frame for $k\text{-}th$ node is

$$E_k^{\Sigma}[n] = \eta P_0 \tau_0[n] g_k[n], \quad (4\text{-}17)$$

where η is the efficiency of the energy harvester, and p_0 is the transmit power of the HAP.

4.3.4 Resource allocation using the deep deterministic policy gradient method

4.3.4.1 Problem formulation

The optimization variables are $p_k[n], 1 \leq k \leq K$ and $\tau_i[n], 0 \leq k \leq K$. On the other hand, the states in this problem are the battery levels $b_k[n], 1 \leq k \leq K$ and channel power gains $g_k[n], 1 \leq k \leq K$ for every frame n. However, we are not going to use these variables directly. More specifically, since we will be using the atanh(·) function as the last layer of the actor network, it is much easier to transform the actor variables $p_k[n], 1 \leq k \leq K$ and $\tau_i[n], 0 \leq k \leq K$ to $\tilde{p}_k[n], 1 \leq k \leq K$ and $\tilde{\tau}_i[n], 0 \leq k \leq K$ which are confined to the $[-1, +1]$ interval. This helps us avoid having to hard-limit the variables which is known to hinder the training of DDPG. In addition, it is known that variables are easier to infer from once they are normalized. Thus, we will also normalize all state variables to the same interval of $[-1, +1]$. Finally, note that our goal is to maximize the long-term expected throughput in the UL. In other words,

$$\lim_{n\to\infty} \sum_{k=1}^{K} \frac{1}{N} \sum_{n=1}^{N} R_k[n]. \tag{4-18}$$

In the following subsections, we define the states, actions and reward.

4.3.4.2 States

As previously alluded to, the state variables in this problem are the battery and channel states of node k at frame n, or $b_k[n]$ and $g_k[n]$. Since the battery state has a clear maximum of $\hat{b}$, using a simple linear transformation

$$\tilde{b}_k[n] = 2b_k[n]/\hat{b} - 1, \tag{4-19}$$

we can easily feed this parameter into the DDPG algorithm. However, the channel gain $g_k[n]$ is unbounded. In order to make it easier for the algorithm to use this parameter, we use the following transformations

$$g[n] = -\mu_k \log\left(\frac{(\tilde{g}[n] + 1)}{2}\right), \tag{4-20}$$

or

$$\tilde{g}_k[n] = 2\exp(-g_k[n]/\mu_k) - 1. \tag{4-21}$$

In this way, $\tilde{g}[n]$ bounded between -1 and 1 leads to $g[n]$ between 0 and $+\infty$. Furthermore, a uniform distribution for $\tilde{g}[n]$ leads to exponential distribution of $g[n]$.

Thus, the state is $2K$-tuple represented as

$$\boldsymbol{s}[n] = \left(\tilde{\boldsymbol{g}}[n], \tilde{\boldsymbol{b}}[n]\right). \tag{4-22}$$

4.3.4.3 Actions

At each frame, there are two variables to control for each node. These variables are $\tau_k[n], 1 \leq k \leq K$ and $p_k[n],\ 1 \leq k \leq K$ as well as $\tau_0[n]$ which does not correspond to any node.

Since the node UL powers should satisfy $0 \leq p_k \leq p_{\max}$, we can simply apply a linear transformation to place them in the $[-1, +1]$ range. However, because the distances of the nodes to the HAP can change the UL transmit power needed to achieve a certain SNR logarithmically, we use another transformation for the UL power. The required transformation should map $[-1, +1]$ to the $[0, p_{\max}]$ range and at the same time make it easier for the algorithm to find the optimal value.

Assuming that the node distances to the HAP vary between $d_{\min}$ and $d_{\max}$ we can transform the power using the following semi-logarithmic transformation

$$\tilde{p}_k = 2 \log\left(1 + s \frac{p_k}{p_{\max}}\right) / \log(1 + s) - 1, \tag{4-23}$$

where s is a scaling parameter which we set at $s = (d_{\max}/d_{\min})^{\delta}$. In some sense s is a scaling parameter which determines how logarithmic or linear this transformation is. This is so because this equation is approximately linear bellow and around $p_{\max}/s$ and approximately logarithmic above this value. Using this transformation, s can be expressed in terms of $\tilde{p}_k$ as

$$p_k[n] = \frac{p_{\max}}{s}\left((1 + s)^{\left(\frac{\tilde{p}_k[n]+1}{2}\right)} - 1\right). \tag{4-24}$$

The same argument applies to $\tau_k[n], 0 \leq k \leq K$. As a result, we can apply the same transformations that we used for $p_k[n], 1 \leq k \leq K$ to $\tau_k[n], 0 \leq k \leq K$

$$\bar{\tau}_k[n] = \left((1 + s)^{\left(\frac{\bar{\tau}_k[n]+1}{2}\right)} - 1\right) / s. \tag{4-25}$$

Note that we have defined $\bar{\tau}_k[n]$, not $\tau_k[n]$. The reason is that $\tau_k[n], 0 \leq k \leq K$ are additionally constrained by $\sum_{k=0}^{K} \tau_k[n] = 1$. Hence, we use $\bar{\tau}_k[n], 0 \leq k \leq K$ to define $\tau_k[n], 0 \leq k \leq K$ as follows

$$\tau_k[n] = \frac{\bar{\tau}_k[n] + 1}{2 \times \sum_{l=0}^{K} \bar{\tau}_l[n]}. \tag{4-26}$$

The advantage is that, using $\bar{\tau}_k[n], 1 \leq k \leq K$, we don't have to implement extra constraints.

In summary, the action is a $2K + 1$-tuple represented as

$$\boldsymbol{a}[n] = (\tilde{\boldsymbol{\tau}}[n], \tilde{\boldsymbol{p}}[n]).$$

4.3.4.4 Reward

We use the Bellman equation to define the long-term average sum-rate

$$Q^{\mu}(\boldsymbol{s}[n], \boldsymbol{a}[n]) = \mathbb{E}[R_k[n] + \gamma Q^{\mu}(\boldsymbol{s}[n], \boldsymbol{a}[n])], \tag{4-27}$$

where ν is the slot-oriented policy [JZ14]. This has the added benefit that it is much easier to see if the trained agent is outperforming the traditional scheme during training.

4.3.5 Simulation results

The training is divided into intervals called episodes. Each episode starts at a random initial state (battery value and channel gain) and proceeds for 200 consecutive time frames. At the end of each episode, the state is reset to another random initial state, which, sets the battery to different initial values. We train the agent for 1000 episodes for each node configuration.

In the following, we first enumerate the WPCN parameters and then describe the DDPG setup. After that, we present the simulation results in two scenarios.

4.3.5.1 WPCN setup

We use a network having five nodes and one HAP. The fading distribution is assumed to be Rayleigh. The following parameters are assumed: P_0 = 30dBm , $P_{\max}$ = 10dBm . $B = 100\text{KHz}$, $T = 1\text{ms}$, $\hat{b}$ = −20dBm, $\delta = 3$, σ^2 = −120dBm.

4.3.5.2 DDPG Setup

The network structure shown in Figure 4.11 is considered for actor and critic networks. Since the output of the actor network needs to be in the range $[-1, +1]$, we use an atanh$(\cdot)$ for the final layer of this network. For the other cases we use the ReLU layer. We set the learning rates of the actor and critic nets to 2×10^{-5} and 2×10^{-4} respectively. The agent discount factor γ is set to 0.99 while the noise variance is set to 0.2 and the noise variance decay rate is set to 1.7×10^{-5}. Finally, we choose a mini batch size of 128 and a buffer length of $episodes \times simlength$ so that the buffer never overflows.

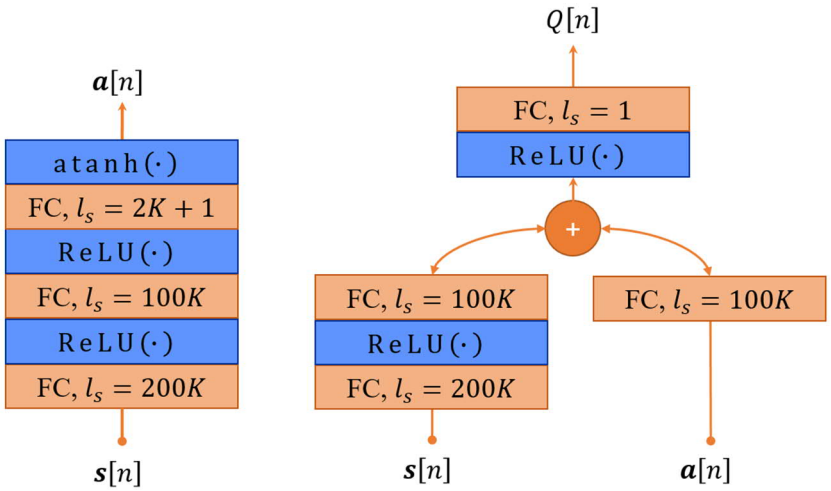


Figure 4.11 The network structure of the actor and critic. FC stands for Fully Connected and l_s is the size of FC layers.

4.3.5.3 Performance comparison

We test all the saved agents for the following scenarios and choose the one with the highest minimum relative throughput.

In the first scenario, we keep the distance of node 1 fixed and linearly increase the distance of the other nodes from 1m to 5m. In other words, we define the distances of the nodes to the HAP according to d_k = 1m + $(k-1)(x - 1\text{m})/4$. The sum-rate of the nodes in this scenario is plotted in Figure 4.12. As can be seen, as x increases, the sum-rate in both methods decreases which is anticipated since the distances of nodes 2 through 5 increase with the increase in x. However, relative to the slot-oriented

method, the sum-rate is higher in the DDPG method. The increase in the sum-rate is from 1.5% at $x = 1\text{m}$ to 4.5% at $x = 4\text{m}$. The lower increase at $x = 1\text{m}$ is due to the fact that at this distance the nodes are the most competitive for the channel resources. As x increases, the nodes distances become more and more different leading to less competition and hence higher throughput over the slot-oriented method.

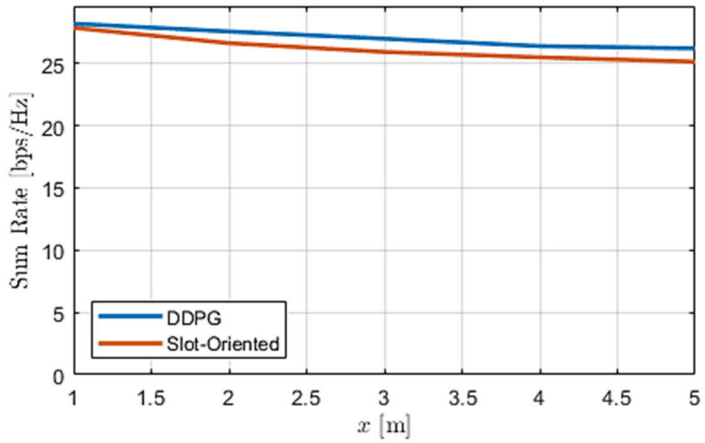


Figure 4.12 The sum-rate in the first scenario in which d_k = 1m + $(k-1)(x-1\text{m})/4$.

In the second scenario, we keep the distance of node 5 fixed and linearly increase the distance of the other nodes from 1m to 5m. In other words, we define the distance of the nodes according to d_k = 5m + $(k-5)(x-5\text{m})/4$. The sum-rate of the nodes in this scenario is plotted in Figure 4.13. As can be seen, compared to the first scenario, the sum-rate decreases faster in both methods which is to be expected because at the same x in this scenario, the nodes distances are equal to or greater than those in the first scenario. The increase in the sum-rate is from 4% at $x = 1\text{m}$ to 14% at $x = 4\text{m}$. Note that since at the same x the node distances to the HAP are equal to or greater than those in the first scenario.

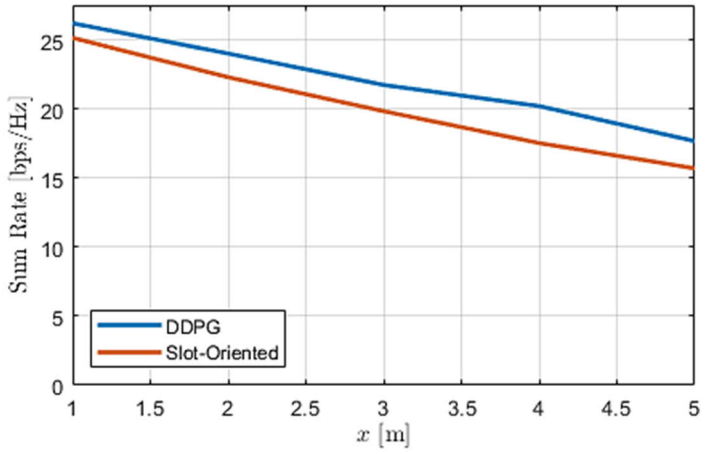


Figure 4.13 The sum-rate in the second scenario in which d_k = 5m + $(k-5)(x-5\text{m})/4$.

4.4 Summary

This chapter considered the wireless networking aspects of AI-assisted technologies in wireless

communication. We studied two problems in this realm, namely learning to demodulate from few pilots via meta-learning, and resource allocation in WPCN using DDPG.

In section 4.2, we proposed the use of online meta-learning for IoT scenarios by adapting state-of-the-art meta-learning schemes, namely MAML, FOMAML, REPTILE, and CAVIA, in a unified framework. Extensive numerical results validated the advantage of meta-learning as compared to conventional ML schemes. Moreover, comparisons among the mentioned meta-learning schemes reveal that MAML and CAVIA are preferable, with each scheme outperforming the other in different regimes in terms of amount of available meta-training data.

In section 4.3, we optimized a multi-node WPCN using the DDPG. Instead of the conventional schemes which maximize the sum-rate of such a scheme in every frame and is hence only optimal in an AWGN channel, the proposed scheme optimizes the long-term sum-rate of the system and, hence, yields a higher sum-rate in a flat-fading channel. Simulation results showed that the DDPG algorithm, in combination with the HTT scheme is able to learn to optimize a WPCN of five nodes and outperform the sum-rate achieved in the conventional schemes.

5 AI on edge

5.1 Overview

In AI on edge, AI offers edge computing approaches, algorithms and methodologies [DZW+20]. AI on edge can further be categorized into model training and model inference. Nevertheless, in this document, we only consider model training, where the purpose is to seek new frameworks for model training on the edge. Figure 5.1 is a schematic showing some of the problems discussed and some of the technologies exhibited in chis chapter.

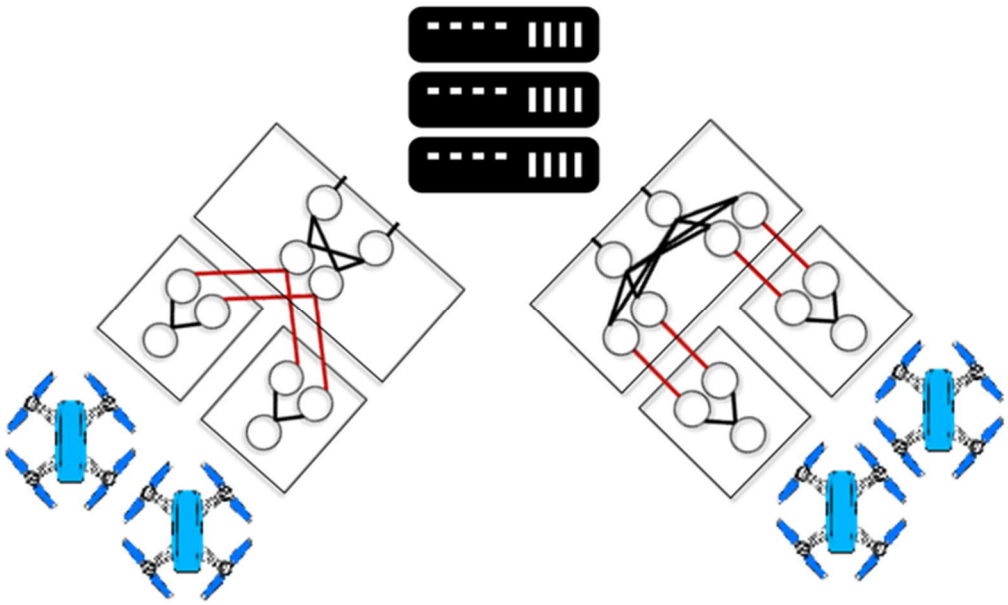


Figure 5.1 Schematic of the AI on edge.

As can be seen in Figure 5.1, using the technologies developed in this chapter, UAVs can train various ML and deep learning algorithms efficiently over the dynamic wireless communication environment that, given the huge amount of data that the image capturing sensors on the drone gather, would have otherwise been impossible.

In section 5.2, we analyse three popular distributed model training methods, namely FL, SL, and SFL, and compare their performances in terms of convergence time, processing delay and communication overhead. A simulation is run on image recognition tasks, using the classical public image dataset, MNIST, which consists of handwritten digit images. The simulation results show that SL and FL with five local training show negligible accuracy degradation relative to central learning while SFL shows relatively lower accuracy. On the other hand, SL shows significantly high processing time delay and communication overhead, which decreases with increasing the number of clients. On the other hand, these two performance indices stay the same in FL.

Section 5.3 is a study on a communication-efficient on-device ML framework suitable for operation in cellular networks with asymmetric UL-DL capacity. More specifically, we design a distributed learning technique, coined Mix2FLD, which considers both test accuracy and latency in a non-symmetric

environment. Numerical evaluation shows that Mix2FLD outperforms FL in terms of both test accuracy and latency on cellular networks with asymmetric UL-DL capacity. In addition, Mix2FLD's test accuracy promptly converges in both iid and non-iid data sets. This convergence is in proportion to the number of participating devices.

Section 5.4 studies various types of SL as well as FL and federated distillation and compares their performances. We divide SL based on whether the training method is performed in serial or in parallel, as well as whether the upper layer is concatenated or not. This results in three types of SL: sequential SL, SL architecture 1 (parallel & concatenated SL), and SL architecture 2 (parallel & non-concatenated SL). Simulation results show that both SL architecture 1 and FL have the best performance in terms of test accuracy. However, FL takes a long time to converge and a device memory size issue may occur at the inference step for SL architecture 1. Considering all these points, SL architecture 2, which utilizes averaging in the pooling layer to reduce the dimension of the upper layer, is the best architecture.

Section 5.5 is a study on the trade-off of accuracy and latency as the mini-batch size changes in SL, as well as the optimal mini-batch size that maximizes objective function which reflects both accuracy and latency. The optimization problem shows that by changing a single parameter in the objective function, we can adjust the test accuracy and latency. When the test accuracy in the objective function is the largest, the test accuracy decreases and the optimal mini-batch size both decrease. In contrast, when we increase the test accuracy, the test accuracy and the optimal mini-batch size increase.

Finally, to improve the outage-sensitive feature of SL, we supplement additional UL-DL phases to the existing SL method in section 5.6. We evaluate the test accuracy and latency by allocating a given resource in various ways between the newly introduced UL-DL and the existing SL UL-DL. Numerical simulations show that, using this method, the test accuracy increases by up to 19.4%, while the latency increases by up to 6.35 times.

5.2 Comparative analysis of distributed model training

Distributed model training is a paradigm in which a model is trained on multiple microprocessors, also known as worker nodes for the purpose of speeding up training, and/or enhancing data privacy and security. In this section, we assess three common model training methods, namely FL, SL, and SFL.

5.2.1 Federated learning

To extract information from the generated big data, ML models, especially deep learning, have become important to train the big data and obtain the trained models which then can be directly used to predict, classify or recognise new input. While the big data is always generated by different distributed users, the general idea is to let them send their local data to a central server to perform model training. This method, however, not only incurs a large communication delay, but also reveals users' private information. With the development of integrated circuit, local devices become more powerful and can support computing engines such as the CPU, GPU and DSP (e.g., energy efficient Qualcomm Hexagon Vector extensions on Snapdragon 835 [QualcoMM19]) for solving diverse ML problems.

As a technique to overcome these challenges, FL first uses the computing abilities of distributed users to train their local models with their local dataset. A central parameter server then collects their local models and perform model averaging to get the global model, which is sent back to each user. Finally, each user can update their local models with this global model [MMR+17]. With FL, users don't need to send their data to the central server so that the communication cost is reduced, and their privacy is protected. By only sharing model parameters, FL benefits wireless communication networks, like cellular networks, UAV networks, and IoT networks, which in return reduces communication cost and saves bandwidth, as well as achieves distributed learning.

However, due to the dynamic environment and limited resources, it is challenging for wireless communication links to provide reliable model parameters for FL, and as a result, many research works have been investigating this problem through user scheduling and resource allocation [AGK+20, KXN+20, CYS+19], enabling ultra-reliable, low-latency and highly efficient communication links [SMS+19, PTB+20], and designing incentive mechanism for users to contribute training [ZLQ+20]. Reference [AGK+20], for example, proposed a novel device scheduling policy and resource allocation techniques to choose the participant users in each FL iteration while allocating limited wireless resources considering channel conditions and the significance of the local model updates at the devices. A FL-based joint transmit power and resource allocation framework was proposed in [SMS+19], and it was proved to have as good accuracy as the centralized solution but with up to 79% reduction in the amount of exchanged data. Reference [CYS+19] provided a novel performance optimization scheme by considering a fundamental connection between the performance of FL algorithms and the underlying wireless networks. References [KXN+20, NY19] designed client selection scheme to make FL operate more reliably over mobile networks. Reference [KXN+20] focused on selecting the users with reliable dataset, while resource conditions of users were considered while selecting them to be aggregate by the server [NY19]. Moreover, [ZLQ+20] proposed a DRL-based incentive mechanism and determined the optimal pricing strategy for the parameter server and the optimal training strategies for edge nodes.

Figure 5.2 illustrates the framework of federated averaging method proposed in [MMR+17] where with aim of solving an ML problem, each user u_i is tries to train a local model with its own fraction of dataset to minimize the loss function $l_i(\omega)$, where ω represents the model parameters.

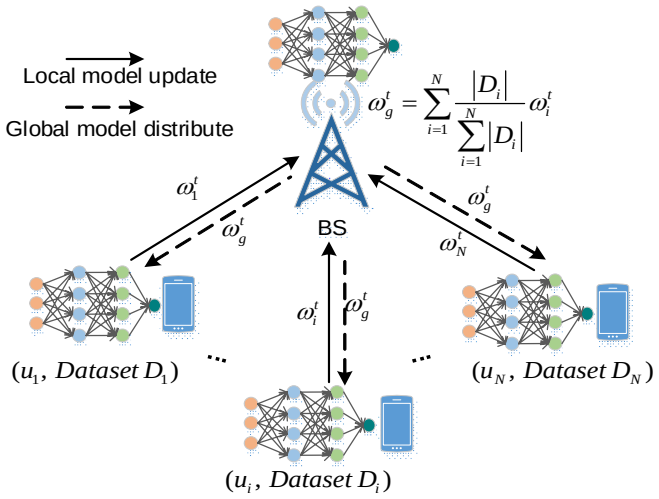


Figure 5.2 Illustration of FedAvg method [MMR+17].

Assuming the data that needs to be trained is distributed over N users, and that $|D_i|$ is the fraction of the whole dataset owned by user u_i, the ML optimization problem is formulated as

$$\min_{\omega \in R} \sum_{i=1}^{N} \frac{|D_i|}{\sum_{i=1}^{N}|D_i|} l_i(\omega) \tag{5-1}$$

To solve this problem, the federated averaging [MMR+17] algorithm was proposed with each user u_i locally training its local dataset once and taking one step of gradient descent such that the local model parameters ω_i are updated. Then, the BS server takes a weighted average of all the local models and generates the global model as

$$\omega_g^t = \sum_i^N \frac{|D_i|}{\sum_{i=1}^N |D_i|} \omega_i^t \quad (5\text{-}2)$$

After that, the server distributes the global model back to each user, and finally each local model is updated.

5.2.2 Split learning

SL was first proposed in [GR17] to train DNNs over multiple data sources while mitigating the need to share raw labelled data directly. Let us define a DNN as a function F, which has a chain structure including a sequence of layers $\{L_0, L_1, \dots, L_N\}$, with the input data D. The output of this function is $F(D)$ which is computed by sequentially going through the layers as

$$F(D) = L_N\left(L_{N-1} \dots \left(L_0(D)\right)\right) \quad (5\text{-}3)$$

After obtaining the output, loss function $l(\text{output}, \text{label})$ is used for computing the gradient of the final layer. Then the gradients are back-propagated over each layer to generate gradients of the previous layer and to update the current layer. Let $\{L_1^T, L_2^T, \dots, L_N^T\}$ denote the backpropagation over each layer and $F^T(\nabla l)$ denote the backpropagation over the entire NN. Similar to forward propagation, the backpropagation process over the layers is

$$F^T(\nabla l) = L_1^T\left(L_2^T \dots \left(L_N^T(\nabla l)\right)\right) \quad (5\text{-}4)$$

Figure 5.3 illustrates the SL framework with multiple data entities in [GR18], where each user holds a fraction of the dataset while they contribute to training the DNN $F(D)$. User u_1 is initialized with the first few layers of $F(D)$ as $F_a(D_1) = \{L_0, L_1, \dots, L_n\}$ with its weights indicated by $\omega_{1,1}$, and the BS is initialized with the rest of the layers of $F(D)$ as $F_b(a) = \{L_{n+1}, L_{n+2}, \dots, L_N\}$ with their weights as $\omega_{i,2}$. The training process begins with u_1, which trains its local $F_a(D_1)$ with its local dataset, and then sends the output of $F_a(D_1)$, a_1^t, to the BS for training $F_b(a_1)$. Afterwards, the BS calculates the gradients of the layer L_n, g_1^t, and send them back to u_1. Then, u_1 backpropagates the gradients it received. Before user u_2 starts training, it updates its local weights $\omega_{2,1}^t$ with last trained user u_1's weights $\omega_{1,1}^t$. This repeats until the training of the last user is finished. Then one global epoch is finished.

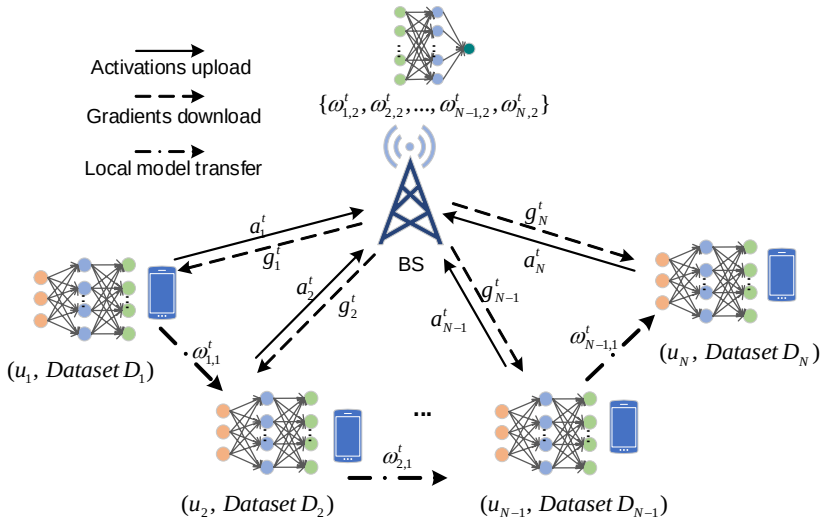


Figure 5.3 Illustration of SL over multiple users [GR18].

Several configurations of SL were illustrated in [VGS+18] to meet different practical settings. Comparisons with other distributed learning algorithms, including large batch SGD and FL demonstrated that SL is dramatically resource efficient and is scalable to large-scale settings. Reference [SVG+19] compared the communication efficiencies of both popular distributed ML approaches, SL and FL. By deploying them in various practical scenarios, [SVG+19] shows that increasing the number of clients or model size favours SL over FL, while FL is more communication efficient with increasing the number of data samples and keeping the number of clients and model size small. With real-world IoT network settings, [GKA+20] confirmed the same findings in [SVG+19] by evaluating splitNN and FL in terms of communication overhead, time and power consumption. Moreover, a HetSLAgg was proposed in [KPB+20], where the BS-side NN segment fuses RF signals and uploads image features without collecting raw images. It was, furthermore, demonstrated that this approach could reduce the prediction error by 44% and achieve over 20% gains in terms of communication and energy cost reduction. Reference [JK20] also proposed a parallel SL method, in which mini-batch sizes are selected and trained at each node and then trained as a full batch at the server.

5.2.3 SplitFed learning

Through the discussions of two recently popular distributed ML approaches, that is, FL and SL, FL and SL show contrasting strengths and weaknesses. In an effort to eliminate the drawbacks of the two approaches, [TCC20] proposed a novel approach, called SFL, which amalgamates the two approaches. Figure 5.4 illustrates the SFL framework of $splitfedv1$ in [TCC20]. SFL combines the strength of FL with parallel training among distributed clients, and the strength of SL with splitting the network into client-side and server-side during training.

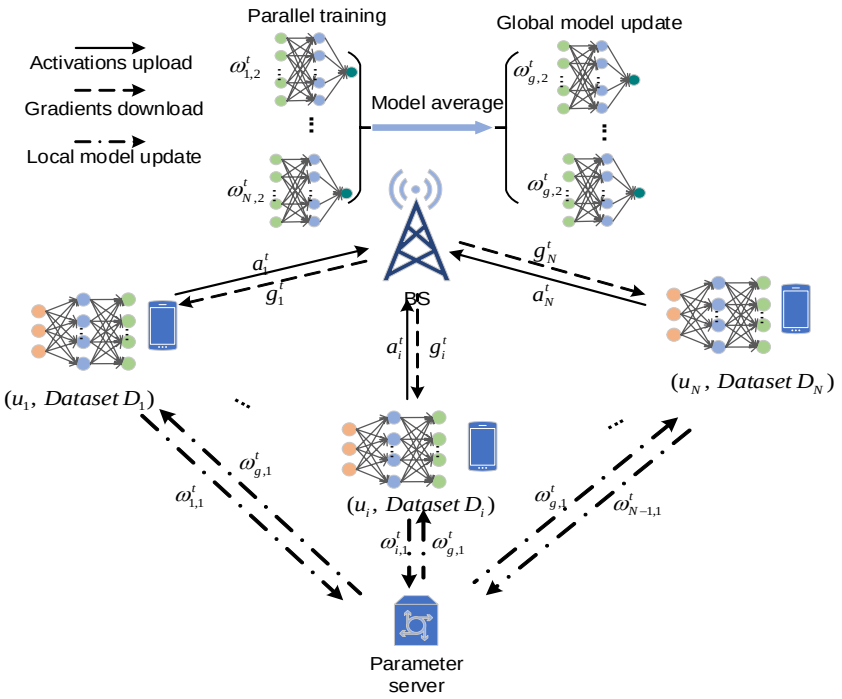


Figure 5.4 Illustration of SFL framework [TCC20].

In this approach, users $\{u_1, \dots, u_i, \dots, u_N\}$ train their local client-side models $F_a(D_1), \dots, F_a(D_N)$ with their local datasets $\{D_1, \dots, D_N\}$ in paralel. Then, they pass the output of local models, $\{a_1^t, \dots, a_N^t\}$, to the BS server, which is assumed to have high computation capacity, and can process the forward propagation and backpropagation on server-side models $F_b(a_1), \dots, F_b(a_N)$ in parallel. Afterwards, it calculates the gradients of the final layer, backpropagates and updates the server-side models, and then sends the gradients of the cut layer $\{g_1^t, \dots, g_N^t\}$ to respective clients for their backpropagation and client-side local models update. Then, the server updates its global model weights $\omega_{g,2}^t$ by performing a weighted average of the parallelly trained models weights $\{\omega_{1,2}^t, \dots, \omega_{N,2}^t\}$, and the clients send their local model weights $\{\omega_{1,1}^t, \dots, \omega_{N,1}^t\}$ to a parameter server, which conducts federated averaging of client-side local update and sends the generated global model weight $\omega_{g,1}^t$ to all the participant clients. This process, defined as one global epoch, repeats until convergence is achieved.

5.2.4 Comparative performance analysis of federated learning, split learning and SplitFed learning

The simulations are run on a laptop with NVIDIA RTX 2070 GPU and Intel i7-10750H CPU, where the server program is run on the GPU, while the clients' program is run on the CPU. During the experiments, we investigate the performance by observing the processing time (including training time and transmission time) with respect to global epochs, and the amount of data communication by each client. In our setup, we assume that all participants update the model in each global epoch.

The classical public image dataset, MNIST dataset, which consists of handwritten digit images of 0 to 9 (i.e., 10 classes), is considered to conduct our experiment. Each image has 28 × 28 pixels with a pixel value ranging from 0 to 255. It has a total of 60000 training and 10000 test samples. The ML model architecture Net is considered using CNN with two convolution layers and two fully connected layers. It uses two 5 × 5 sized kernels in its layers. For all experiments in SL and Splifed, the network layers are split at the second layer (after 2D MaxPool layer).

By conducting experiments with MINST dataset, the comparison results on test accuracy, processing time delay and communication overhead of these algorithms are illustrated as follows.

5.2.4.1 Convergence analysis

LL where each client has its own dataset and trains its own model locally, and CL where the server receives the whole dataset from all the clients, are considered as the benchmarks. As shown in Figure 5.5, all the experiments were conducted with 10 clients and one server, and all the learning algorithms have converged within 50 global epochs. As can be seen in Figure 5.5, CL has the highest test accuracy, while the LL has the lowest. This is because LL has limited dataset used for training the learning model. Additionally, SL has the closest test accuracy in CL, FL with 5 local training before global model average has almost the same accuracy as SL, while FL with one local training is slightly worse. Finally SFL shows similar accuracy performance as FL(1).

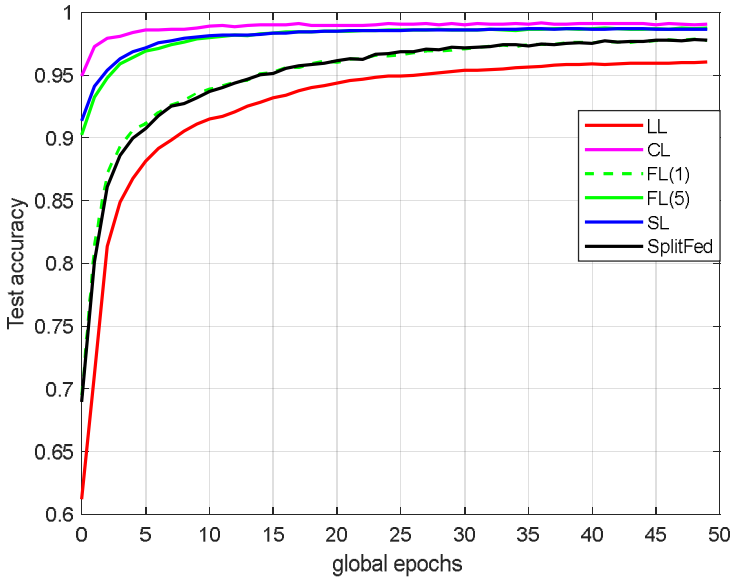


Figure 5.5 Test convergence of different learning with ten clients.

5.2.4.2 Processing time

To show the time efficiency of different learning algorithms, we analyse the processing time taken for one global epoch (the time cost to train the whole dataset) shown in Figure 5.6. Here, the processing time mainly contains training time and transmission time. CL contains the training time of the server and the transmission time of all the clients by transmitting their local dataset which is performed in parallel. In LL, the clients train their own dataset locally. This can be done in parallel as well, which means it only takes local training time. FL contains training time on both the server and clients, as well as transmission time which depends on the model size and the communication conditions of the worst client. So, the processing time per global epoch stays the same with increasing the number of clients since clients carry out the training in parallel. For SL, the total processing time for one global epoch depends on the product of the number of clients and the time to process one global epoch by the clients and the server, while in SFL, all the clients train their local models simultaneously and the server can be a supercomputer that can train the edge models from all the clients in parallel. Therefore, the processing time of SFL is much shorter. Moreover, in SL and SFL, the transmission time depends on the number of data samples that each client has as well as the communication conditions. Hence, SFL is trained faster with increasing the number of clients.

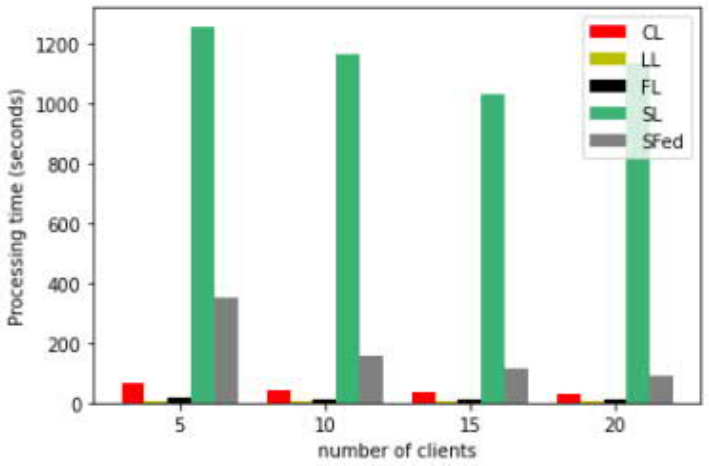


Figure 5.6 Processing time delay of different learning algorithms.

5.2.4.3 Communication overhead

The communication overhead including (UL and DL overhead) indicates the operability of a distributed learning approach in a resource constrained environment. The comparisons of communication overhead are performed under the amount of communication overhead per client in one global epoch shown in Figure 5.7. CL has the communication overhead by each client uploading their local dataset, while LL has no communication overhead where each client trains their dataset locally. FL contains local clients' model parameters UL and global model parameters distribute, which is independent of the dataset size of the client. For SL and SFL, we observed the same amount of communication overhead and it's reducing with the increasing client number. This is because in SL and SFL, each client needs to upload the activations and download the gradients of the cutlayer, which depend on the dataset size of each client.

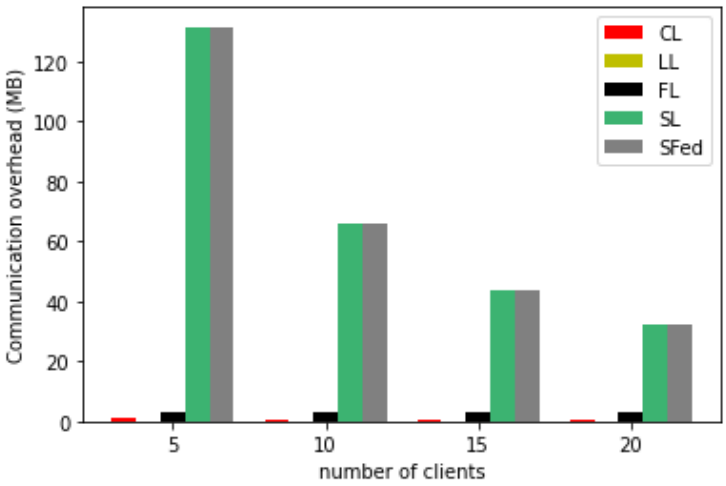


Figure 5.7 Communication overhead of different learning algorithms.

5.3 Mix2FLD: downlink federated learning after uplink federated distillation

with two-way mixup

When distributed ML operates on a cellular network consisting of several handheld devices and a server, the asymmetry of UL and DL capacity due to the low transmit power of the handheld devices poses challenges to learning. FL has a high test accuracy, but because of exchanging model weight, outages occurring frequently especially in the UL transmission result in large latency. FD can solve the outage problem, but its test accuracy is very low. Thus, our goal in this section, is to design a distributed learning technique that considers both test accuracy and latency.

Assuming that distributed learning is taking place in a communication system consisting of servers and handheld devices, there occurs an imbalance in UL and DL capacity due to the transmission power gap between devices and servers.

FL proposed in [YLC+19] is the most widely used privacy-preserving technique among distributed learning techniques in which the devices and the server exchange model weights. Yet, in the case we just discussed the UL capacity is limited and hence when the device sends the model weights to the server, a communication bottleneck is highly probable.

In the case of FD proposed in [JOK+18], in order to solve the communication problem that occurs when such distributed learning runs on an actual network, output distribution with a very small communication payload size is exchanged, but the test accuracy is relatively inferior.

In the case of FD proposed in [JOK+18], in order to solve the communication problem that occurs when such distributed learning runs on an actual network, output distribution with a very small communication payload size is exchanged, but the test accuracy is relatively inferior.

To ensure test accuracy while resolving UL-DL asymmetric capacity, in our proposed Mix2FLD, we utilize output distribution at UL as in FD and model weight at DL as in FL. At this time, the central server shares some samples of each device for output-to-weight conversion. To solve the sample privacy leakage problem that occurs, using the mixup algorithm proposed in [ZCD+18], each device uploads the sample passed through the mixup and the mixing ratio information used during the mixup to the server, and the server generates an inversely-mixed sample with the same label as that of the raw sample of the device through inverse mixup depending on the mixing ratio. Based on the generated inversely-mixed sample and the uploaded output distribution, the server side creates a global model through knowledge distillation proposed in [HVD14]. The overall process of Mix2FLD is illustrated and elaborated in Figure 5.8 and Algorithm 5.1 respectively.

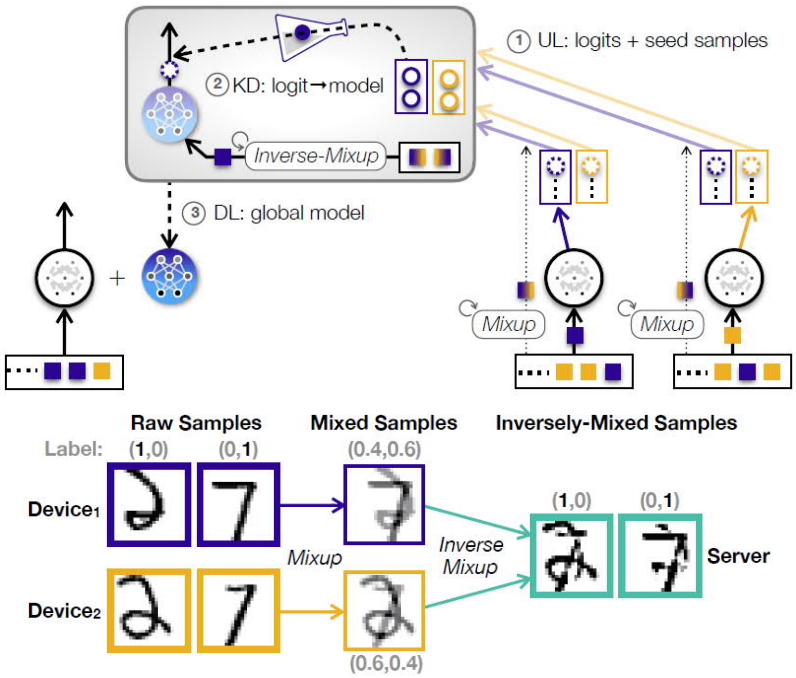


Figure 5.8 Operation of Mix2FLD, Mixup and Inverse-Mixup.

Algorithm 5.1 Operation of Mix2FLD.

Require: $\{\mathbf{S}_d\}$ with $d \in \mathcal{D}$, $\lambda \in (0,1)$
while $|\mathbf{G}^{p}_{\text{out},n} - \mathbf{G}^{p-1}_{\text{out},n}|/|\mathbf{G}^{p-1}_{\text{out},n}| \geq \varepsilon$ **do**
Device $d \in \mathcal{D}$: ▷ *Output upload*
  if $p=1$ **generates** $\{\hat{\mathbf{s}}_d^{[i,j]}\}$ via (6) **end if** ▷ *Mixup*
  updates $\mathbf{w}_d^{(k)}$ in (1) and $\bar{\mathbf{F}}^{p}_{d,n}$ in (2) for K iterations
  unicasts $\{\bar{\mathbf{F}}^{p}_{d,n}\}$ (with $\{\hat{\mathbf{s}}_d^{[i,j]}\}$ **if** $p=1$) to the server
Server: ▷ *Output-to-model conversion*
  if $p=1$ **generates** $\{\tilde{\mathbf{s}}_{d,d',n}^{[i,j][i',j']}\}$ via (7) **end if** ▷ *Inverse-Mixup*
  computes $\mathbf{G}^{p}_{\text{out},n}$
  updates $\mathbf{w}_s^{(k)}$ via (5) for K_s iterations
  broadcasts $\mathbf{G}^{p}_{\text{mod}} = \mathbf{w}_s^{(K_s)}$ to all devices
$p \leftarrow p+1$
Device $d \in \mathcal{D}$ **substitutes** $\mathbf{w}_d^{(0)}$ with $\mathbf{G}^{p}_{\text{mod}}$ ▷ *Model download*
end while

Figure 5.9 shows the performance comparison with distributed ML algorithms such as FL and FD, including Mix2FLD, in asymmetric and symmetric UL-DL capacities, respectively.

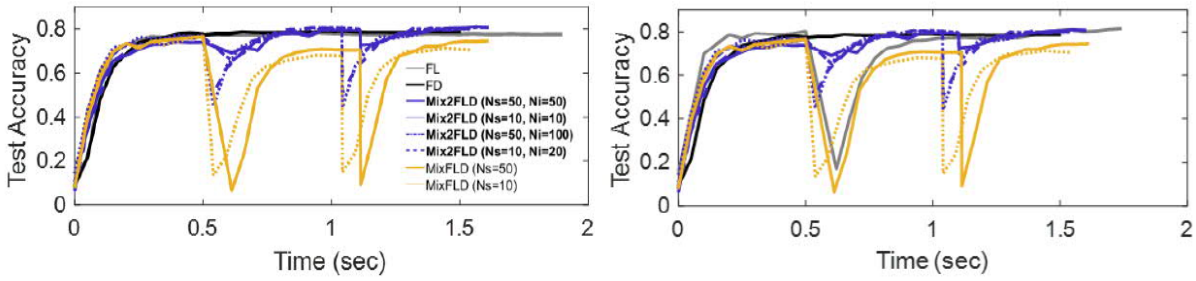


Figure 5.9 Learning curves of distributed ML under asymmetric & symmetric channels.

On the other hand, in a symmetrical environment, the transmission power of devices is 40dBm [3GPP16-36942]. This value is the same as that at the server, which can be considered an ideal channel environment. In this environment, FL, which has the largest amount of information exchanged through UL or DL, has the highest performance in terms of test accuracy. However, in an asymmetric environment, where the transmission power of devices is 23dBm [3GPP16-36942], two cases may occur: when FL is employed, frequent outages occur in the UL, and when FD is used, communication is successful, but accuracy is not high. On the other hand, the proposed Mix2FLD, in addition to having the highest performance in such an asymmetric environment, achieves considerably high performance in a symmetric environment.

Figure 5.10 shows the average and variance of the test accuracy of Mix2FLD as the number of devices is changed in iid and non-iid data set environments, respectively. In both iid and non-iid environments, as the number of devices increases, the average of the test accuracy gradually increases, while the variance decreases. In particular, it can be seen that the trend is more clearly obvious in iid case.

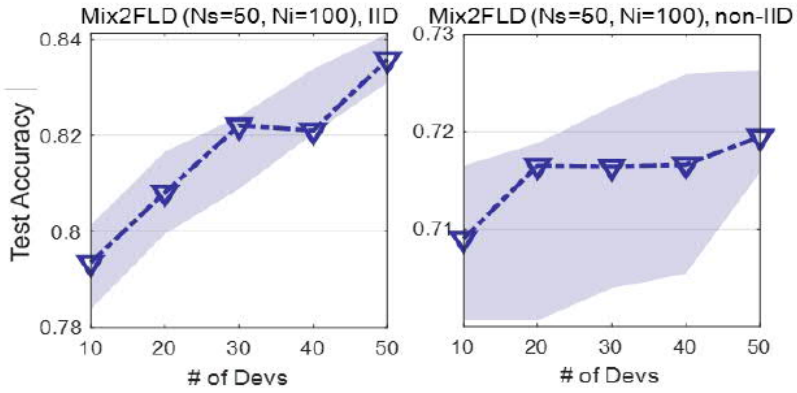


Figure 5.10 Test accuracy of Mix2FLD with respect to the number of devices.

5.4 Wireless split learning

SL, first proposed in [GR17] is implemented in a sequential manner which prevents processing multiple inputs simultaneously. By utilizing parallel computing in SL, performance can be improved in terms of test accuracy or overhead incurred in forward and back propagation. With the standalone structure shown in Figure 5.11 as the baseline, we propose two types of SL, namely SL architecture 1 and SL architecture 2, both shown in Figure 5.11. A common idea for training in SL is to synchronize the data labels of data samples across devices. SL architecture 1 and SL architecture 2 can be classified by the input dimension of the upper layer stored in the server. A model whose input dimension linearly increases with the number of devices is SL architecture 1, and a model with a fixed input dimension is

SL architecture 2. In particular, in the case of SL architecture 2, in order not to change the input dimension of the server even when the number of devices change, techniques such as averaging or random scheduling can be used.

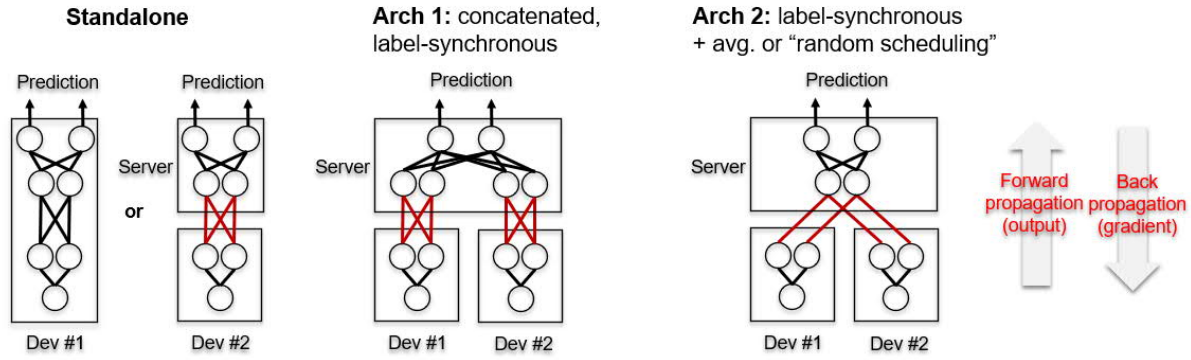


Figure 5.11 Structures of standalone, SL architecture 1, and SL architecture 2.

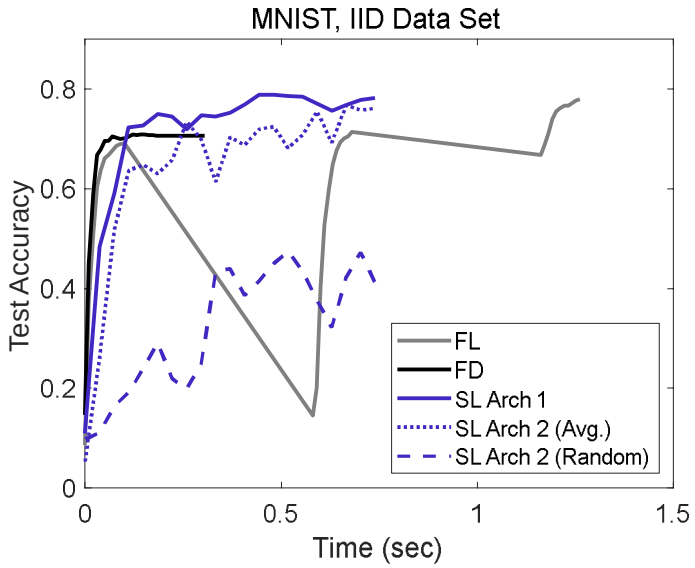


Figure 5.12 Learning curves of various distributed ML including various types of SL.

Figure 5.12 compares the performance of the proposed SL architecture 1, and SL architecture 2 using averaging and random scheduling, as well as, FL and FD. In terms of test accuracy, both FL and SL architecture 1 have high accuracy. Nevertheless, considering the latency, the performance of SL architecture 1 is the highest.

In addition, after learning is completed, considering that all global models are downloaded and used in the inference stage, in the case of SL architecture 1, the input dimension of the upper layer increases in proportion to the number of devices. Thus, it is practically limited by the memory size or computation power. Moreover, when the environment with many outages is considered, the time it takes to reach convergence in SL architecture 1, which has more weight to learn, may be longer. Considering these communication and computing-related constraints, while the test accuracy of SL Architecture 2 is slightly inferior to that of Architecture 1, SL Architecture 2 is extremely efficient in terms of latency.

5.5 Accuracy-latency trade-off for mini-batch size

In SL, based on the intermediate layer, the devices have the partition of the lower layer and the parameter server has the upper layer. In this structure, for model update, each device transmits the output of the cut layer (forward-propagation), and the server transmits the gradient information (back-propagation) through DL. Compared to other distributed learning techniques, SL communicates a smaller packet size more often, and this leads to capacity loss due to the packet sizes being small.

In addition, since forward propagation and back propagation in SL are entirely dependent on communication, an increase in payload size (which is proportional to batch size) results in a gain in test accuracy but tends to increase the latency. Especially, when the MSGD technique is applied, a trade-off between test accuracy and latency occurs which depends on the mini-batch size. Therefore, we examine the above trends as the batch size changes in SL, and find out what the optimized batch size is.

When applying the MSGD to SL, each device transmits the output value to the parameter server after passing the total number of samples of mini-batch size in parallel through the local model, and the server calculates the gradient through the average loss and broadcasts it to the device. This procedure is periodically repeated until the local model converges.

Suppose that the number of samples reflected during local update is fixed, that is, (mini batch size) × (number of communication rounds) is fixed. When mini-batch size increases by N times, the UL payload size also increases by N times. However, since the UL transmission period decreases by $1/N$ times, the UL traffic per unit time remains constant. In case of DL, it has the same payload size regardless of the mini-batch size, but the transmission period changes in inverse proportion to mini-batch size. Thus, as mini-batch size increases, the communication payload size per unit time decreases.

Next, since SL exchanges a single intermediate layer's instantaneous activation and gradient whose corresponding packet length can be very short, when the packet length is very small, the Shannon's capacity formula is no longer accurate. Therefore, the following capacity model [YVS10] is used

$$R = C - \sqrt{\frac{V}{N}}\,Q^{-1}(\varepsilon) + \frac{1}{2N}\log_2 N$$

Here, as mini-batch size increases, the UL payload size N decreases which leads to an increase in data rate.

Also, the number of iterations required for convergence N_{Update} is in inverse proportion to the payload size N as proposed in [PKK+19], which is as follows

$$N_{\mathrm{Update}} = N_{\infty} + \frac{\alpha}{N}$$

Finally, in terms of accuracy, when comparing gradient updates multiple times with a small mini-batch size and relatively fewer gradient updates with a large mini-batch size, the former performance is higher. This means test accuracy and latency trade-off for mini-batch size.

Figure 5.13 shows the test accuracy and latency in SL while mini-batch size changes from 2^3 to 2^{11}. It can be seen that when the mini-batch size increases, both the test accuracy and latency decrease.

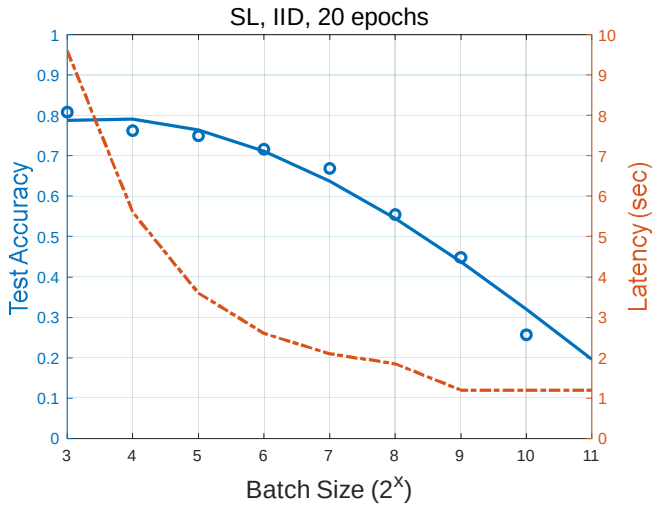


Figure 5.13 Test accuracy and latency with respect to batch size in SL.

To find the optimal mini-batch size that maximizes the objective function, we define the objective function to be $S_b - \lambda T_b$, where S_b and T_b are the test accuracy and test latency respectively.

Figure 5.14 shows the objective functions with respect to different λ and batch sizes. In Figure 5.14, as λ increases, the optimal batch point shifts to the right. This means that the larger λ is, the greater the proportion of latency in the objective function is, and accordingly, it changes in the direction of further minimizing the latency, that is, in the direction of increasing the mini-batch size.

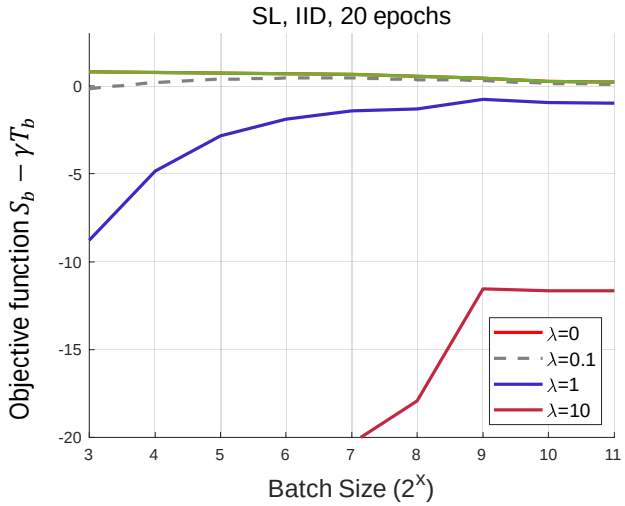


Figure 5.14 Optimal batch size with respect to different λ

Given the defined objective function, when λ is 0 (considering test accuracy only) and 0.1, the optimal mini-batch size is 2^7, and when λ is 1 and 10, the optimal mini-batch size is 2^9.

5.6 UL resource allocation

We introduced SL in subsection 5.2.2 and section 5.4 as a technique to train DNNs over multiple data sources while alleviating the need to share raw labelled data directly. However, some of the limitations

of SL are

1. When outage occurs in either UL or DL, the weight is not updated. In other words, it is sensitive to outage.
2. Where the non-iidness of samples of devices is too severe, the SL training based on label-synchronous may not proceed properly.

To compensate for the performance degradation resulting from these limitations, we introduce additional UL-DL communication that exchanges local gradient information (from devices that have successfully received gradient information) and global gradient information (obtained from server by averaging local gradient information) after UL-DL communication that exchanges gradients and outputs from the intermediate layer of the existing SL. We use UL_1 to denote the UL transmission phase that sends the output of the intermediate layer and UL_2 to represent the UL transmission phase by devices which have successfully received UL_1.

In this structure, both UL communications (devices to server) can be communication bottlenecks. Therefore, under the assumption that the outage probability in the DL is 0, we investigate the result of resource allocation of UL_1 and UL_2. Here, a trade-off occurs for resource allocation of UL_1 and UL_2.

First, if more resources are allocated to UL_1, the number of stragglers who fail to upload local output information decreases. Instead, when receiving local gradient information from receivers, the probability of occurrence of outage is high, and as a result, the probability of transmission failure or low quality of the averaged gradient information to be given to stragglers is high.

Conversely, when the allocated resource blocks in UL_2 increase, a lot of stragglers occur in UL_1. Instead, in UL_2, receivers can upload more local gradient information, and stragglers can be compensated through higher quality global gradient information.

In order to check the trade-off of UL_1 and UL_2 resource allocation and the optimal resource allocation method, we have a total of 10 devices in a TDMA environment. No device has any samples corresponding to 2 labels out of the total 10 labels.

Table 5.1. Test accuracy, latency, and straggler for various resource allocation methods. There are a total of 5500 slots of length 1ms.

Number of slots in UL_1	300	400	500	600	700
Number of slots in UL_2	5200	5100	5000	4900	4800
Number of stragglers	10	8	4	1	0
Number of successfully uploaded local gradients	0	2	6	1	0
Number of stragglers who received global gradients	0	8	4	1	0
Test Accuracy S_b	0.128	0.269	**0.728**	0.525	**0.534**
Latency T_b (Averaged per 1 cycle)	0.732	1.4705	**3.9622**	2.7196	**0.6239**

In Table 5.1, the total number of time slots allocated to UL_1 and UL_2 is 5500, and the duration of each time slot is 1ms.The number of slots allocated to UL_1 varies from 300 to 700, and accordingly, the number of slots allocated to UL_2 varies from 5200 to 4800. When 300 slots are allocated to UL_1, the maximum number of stragglers is 10, and when 700 slots are allocated, the minimum number of stragglers is 0. Except for when the number of stragglers is 10 or 0, when the receivers send local gradients, the most sent case is when the number of slots allocated to UL_2 is 5000, and the minimum is when the number of slots allocated to UL_2 is 4900. This is because when the number of slots allocated

to UL_2 was 4900, there was only one straggler in UL_1, and 9 receivers attempted to upload local gradients, causing many collisions.

Note that when there are 400 or 500 slots allocated to UL_1, 10 devices have local or global gradients. However, when looking at the actual accuracy, the latter case shows much higher performance as 0.728. It can be concluded that when the number of stragglers is too large, the quality of the global gradient is too low, so that compensation does not work properly.

Finally, in the case of UL_1 having 500 slots and UL_2 having 5000 slots, the performance is the highest in terms of test accuracy while compromising latency. Compared to the case where 700 and 4800 slots are allocated to UL_1 and UL_2, respectively, that is, the conventional SL that does not utilize UL_2, the additional UL-DL increases the test accuracy up to 19.4%, while latency increased up to 6.35 times. Since this result is measured when additional UL-DL is applied for every UL-DL, it can be considered as the worst case in terms of latency. However, this can be solved by optimizing the frequency of applying the additional UL-DL. In the case of original SL, which actually operates without UL_2 with 700 UL_1 slots, high non-iidness is not overcome.

5.7 Summary

This chapter studied, model training in AI on edge. The ideas, methodologies and approaches proposed in this chapter can help ML algorithms learn the vast amount of information gathered while performing different fire-fighting manoeuvres.

In section 5.2, different distributed model training schemes for ML, that is FL, SL, SFL were introduced. The basic learning mechanisms of these methods were illustrated and compared in terms of test accuracy, processing time delay and communication overhead. The simulation results showed that SL and FL with five local training have closer test accuracy to central learning while SFL shows relatively lower accuracy. SL showed significantly high processing time delay and communication overhead, which decreases with increasing the number of clients. On the other hand, because of their dependencies on model size, these two performance indices stay the same in FL.

In section 5.3, we designed a distributed learning procedure that achieves an optimal trade-off between accuracy and latency in a non-symmetric environment. The proposed approach, called Mix2FLD, outperforms FL in terms of both test accuracy and latency on cellular networks.

In section 5.4, a study on the different types of SL as well as FL and federated distillation and their performances was conducted. We categorized SL based on its architecture and its operation into three categories: sequential SL, SL architecture 1, and SL architecture 2. Based on the simulation results, SL architecture 2 is the best architecture.

In section 5.5, we examined the effect of changing the mini-batch size in SL. By defining an objective function and using it in an optimization problem to increase accuracy and decrease latency, we observed a trade-off between test accuracy and latency. In particular, it was shown that when we increase the test accuracy, the optimal mini-batch size and the test accuracy both decrease and conversely when we increase the test accuracy, the optimal mini-batch size and the test accuracy both increase.

Lastly, in an effort to combat the adverse effects of outage on SL, we added supplementary UL-DL phases to the SL method in section 5.6. We evaluate the test accuracy and latency by allocating a given resource amount in various ways between the newly introduced UL-DL and the existing SL UL-DL. According to the simulation results, when more resources are allocated to the existing UL-DL, the number of stragglers become small, but the test accuracy remains low because the severe non-iidness cannot be overcome. When more resources are allocated to the additional UL-DL, the quality of the

global gradient increases, but the test accuracy is low because there are many stragglers in the existing UL. That is, a trade-off for test accuracy occurs as the quality of the global gradient and the number of UL stragglers change depending on which UL-DL is allocated. As a result, compared to existing SL, the test accuracy increases by up to 19.4%, while the latency increases by up to 6.35 times.

6 Conclusions and outlooks

The main objective of this deliverable is to provide a final status report about the work that has been conducted as part of WP4 AI assisted communications.

In chapter 2, novel drone localization and coordination technologies were shown. First, to improve the performance of DAE, we proposed a new denoising framework, named nlDAE which can maximize the efficiency of the ML approach for wireless communications where noise is typically easier to regenerate than original data owing to its stochastic characteristics. Second, we proposed a Q-learning-based low complexity beam tracking algorithm for mmWave MIMO systems which can track the directional of signal with high resolution and requires only a few beam searches with low overhead. Third, to solve the autonomous BS management in a distributed manner, ML-based approaches were used in distributed systems settings. The proposed scheme offers a promising solution to find optimal trajectories in the operating area and network coverage control of DBSes that can cover as many users as possible.

In chapter 3, we used both a centralized we well as a decentralized approach for offloading computation. First, we introduced FlexSensing, a centralized task offloading decision making algorithm and then we presented a decentralize task offloading decision making. Finally, we introduced an IoV network for distributed caching in video streaming services provided via D2D links. Using DRL, we jointly optimized video delivery decisions by maximizing the average video quality under the constraints on the playback delays and the data rate guarantees.

In chapter 4, topologies of edge using AI were illustrated. First, we proposed resource allocation in WPCNs using the DDPG. It was shown that the DDPG is able to deal with the complexity problem of a network of five nodes and an HAP. It can even outperform the traditional slot-oriented schemes in terms of long-term expected throughput. Second, meta-learning approaches were proposed to tackle the insufficient pilot problem for IoT scenarios.

In chapter 5, we analysed three popular distributed model training methods in terms of convergence time, processing delay and communication overhead. Then, we designed a distributed learning method that achieves an optimal trade-off between accuracy and latency in an asymmetric environment. After that, we conducted a study on the different types of SL as well as FL and federated distillation and compared their performances. Then, we investigated the trade-off between accuracy and latency as the mini-batch size changes in SL. By crafting an objective function which takes into the account these two criteria, we were able to find the optimal mini-batch size that achieves the optimal trade-off between the two. Finally, we added supplementary UL-DL communication phases to the standard SL method which lead to an increase in test accuracy by up to 19.4% and an increase in latency by up to 6.35 times.

References

[3GPP16-36942] 3GPP TR 36.942, “Radio Frequency (RF) system scenarios”, January 2016.

[A09] N. Andrei, “Accelerated conjugate gradient algorithm with finite difference Hessian/vector product approximation for unconstrained optimization,” Journal of Computational and Applied Mathematics, vol. 230, no. 2, pp. 570–582, Aug. 2009.

[AES18] A. Antoniou, H. Edwards, and A. Storkey, “How to train your MAML,” arXiv preprint arXiv:1810.09502, 2018.

[AGK+20] M.M. Amiri, D. Gündüz, S.R. Kulkarni, "Convergence of Update Aware Device Scheduling for Federated Learning at the Wireless Edge." arXiv, 2020

[AHN+15] M. A. Alsheikh, D. T. Hoang, D. Niyato, H. P. Tan, and S. Lin, "Markov decision processes with applications in wireless sensor networks: A survey," IEEE Communications Surveys & Tutorials, 17(3), pp. 1239-1267, April 2015

[AL+17] K. Asadi and M. L. Littman, “An alternative softmax operator for reinforcement learning,” in International Conference on Machine Learning, PMLR, pp. 243-252, 2017

[AM17] R. Amit and R. Meir, “Meta-learning by adjusting priors based on extended PAC-Bayes theory,” arXiv preprint arXiv:1711.01244, 2017.

[AOC19] A. Azari, M. Ozger, and C. Cavdar, “Risk-aware resource allocation for URLLC: Challenges and strategies with machine learning,” IEEE Communications Magazine, vol. 57, no. 3, pp. 42–48, 2019.

[AP18] A. Ahmadian and H. Park, "Maximizing Ergodic Throughput in Wireless Powered Communication Networks," 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 1-6, doi: 10.1109/GLOCOM.2018.8647192.

[ASP20] A. Ahmadian, W. Shin and H. Park, "Max-min Throughput Optimization in FDD Multi-Antenna Wirelessly-Powered IoT Networks," IEEE Internet of Things Journal, pp. 1-1, 2020, doi: 10.1109/JIOT.2020.3033227.

[B06] C. M. Bishop, Pattern recognition and machine learning. Springer, 2006.

[B57] R. Bellman, Dynamic Programming. Princeton, NJ, USA: Princeton Univ. Press, 1957.

[BRC98] S. Bouchired, D. Roviras, and F. Castanié, “Equalisation of satellite mobile channels with neural network techniques,” Space Communications, vol. 15, no. 4, pp. 209–220, 1998/1999.

[BYA+13] Y. Bengio, L. Yao, G. Alain, and P. Vincent, “Generalized denoising auto-encoders as generative models,” Advances in neural information processing systems, pp. 899–907, 2013.

[BZ17] A. Biason and M. Zorzi, "Battery-Powered Devices in WPCNs," IEEE Transactions on Communications, vol. 65, no. 1, pp. 216-229, Jan. 2017, doi: 10.1109/TCOMM.2016.2621109.

[CCR+20] H. Chiang, K. Chen, W. Rave, M. K. Marandi, and G. Fettweis, “Multi-UAV mmWave beam tracking using Q-learning and interference mitigation,” in Proc. IEEE Int. Conf. Commun. (ICC Workshops), Jun. 2020, pp. 1–7.

[CCS+20] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, “Artificial neural networks-based machine learning for wireless networks: A tutorial,” IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3039–3071, 2019.

[CL+06] N. Cesa-Bianchi and G. Lugosi, “Prediction, learning, and games,” Cambridge university press, 2006.

[CP02] E. Costa and S. Pupolin, “M-QAM-OFDM system performance in the presence of a nonlinear amplifier and phase noise,” IEEE Trans. Commun., vol. 50, no. 3, pp. 462–472, Mar. 2002.

[CRT20] U. Challita, H. Ryden, and H. Tullberg, “When machine learning meets wireless cellular networks: Deployment, challenges, and applications,” IEEE Communications Magazine, vol. 58, no. 6, pp. 12–18, 2020.

[CYG17] J. Chen, U. Yatnalli, and D. Gesbert, “Learning radio maps for UAV-aided wireless networks: A segmented regression approach,” Pro-ceedings of IEEE International Conference on

Communications (ICC). IEEE, 2017, pp. 1–6.

[CYS+19] M. Chen, Z. Yang, W. Saad, "Performance optimization of federated learning over wireless networks." In 2019 IEEE Global Communications Conference (GLOBECOM), pp. 1-6. IEEE, 2019.

[CZL+18] F. Cheng, S. Zhang, Z. Li, Y. Chen, N. Zhao, F. R. Yu, and V. C. Leung,“ UAV trajectory optimization for data offloading at the edge of multiplecells,” IEEE Transactions on Vehicular Technology, vol. 67, no. 7, pp.6732–6736, 2018.

[DSH+17] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, “Deep learning based communication over the air,” IEEE Journal of Selected Topics in Signal Processing, vol. 12, no. 1, pp. 132–143, 2017.

[DZW+20] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar and A. Y. Zomaya, "Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence," IEEE Internet of Things Journal, doi: 10.1109/JIOT.2020.2984887

[FAL17] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic metalearning for fast adaptation of deep networks,” arXiv preprint arXiv:1703.03400, 2017.

[FRK+19] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, “Online meta-learning,” arXiv preprint arXiv:1902.08438, 2019.

[G91] D. Goldberg, “What every computer scientist should know about floating-point arithmetic,” ACM Computing Surveys (CSUR), vol. 23, no. 1, pp. 5–48, 1991.

[GAH20] M. Goutay, F. A. Aoudia, and J. Hoydis, “Deep HyperNetwork-Based MIMO Detection,” arXiv preprint arXiv:2002.02750, 2020.

[GBB+19] J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. Turner, “Meta-learning probabilistic inference for prediction,” in Proc. International Conference on Learning Representations (ICLR), New Orleans, United States, May 2019.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[GDW+15] Z. Gao, L. Dai, Z. Wang and S. Chen, “Spatially common sparsity based adaptive channel estimation and feedback for FDD massive MIMO,” IEEE Trans. Signal Processing, vol. 63, no. 23, pp. 6169-6183, Dec., 2015.

[GFL+18] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, “Recasting gradient-based meta-learning as hierarchical bayes,” arXiv preprint arXiv:1801.08930, 2018.

[GKA+20] Y. Gao, M. Kim, S. Abuadbba, "End-to-End Evaluation of Federated Learning and Split Learning for Internet of Things." arXiv preprint arXiv:2003.13376 (2020).

[GR17] O. Gupta and R. Raskar, “Secure Training of Multi-Party Deep Neural Network.” U.S. Patent Application No. 15/630,944, 2017.

[GR18] O. Gupta, R. Raskar, "Distributed learning of deep neural network over multiple agents." Journal of Network and Computer Applications 116 (2018): 1-8.

[GSS14] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” arXiv preprint arXiv:1412.6572, 2014.

[H57] J. Hannan, “Approximation to bayes risk in repeated play,” Contributions to the Theory of Games, vol. 3, pp. 97–139, 1957.

[HCS+20] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, “Meta-reinforcement learning for trajectory design in wireless UAV networks,”arXiv preprintarXiv:2005.12394, 2020.

[HDA17] A. G. Helmy, M. Di Renzo, and N. Al-Dhahir, “On the robustness of spatial modulation to I/Q imbalance,” IEEE Commun. Letters, vol. 21, no. 7, pp. 1485–1488, July 2017.

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman, The Elements Of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed. New York: Springer, 2009.

[HVD14] G. E. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” in Proc. of NIPS Deep Learning Wksp., (Montr`eal, Canada), Dec. 2014.

[I20] M. Ibnkahla, “Applications of neural networks to digital communications–a survey,” Signal

Processing, vol. 80, no. 7, pp. 1185–1215, July 2000.

[JDS+19] N. Jiang, Y. Deng, O. Simeone, and A. Nallanathan, “Online supervised learning for traffic load prediction in framed-ALOHA networks,” IEEE Commun. Letters, vol. 23, no. 10, pp. 1778– 1782, Oct. 2019.

[JK20] J. Jeon, J. Kim, "Privacy-Sensitive Parallel Split Learning." In 2020 International Conference on Information Networking (ICOIN), pp. 7-9. IEEE, 2020.

[JKA+19] Y. Jiang, H. Kim, H. Asnani, and S. Kannan, “Mind: Model independent neural decoder,” in Proc. IEEE Signal Processing Advances in Wireless Commun. (SPAWC), Cannes, France, July 2019.

[JMC+17] S. Jayaprakasam, X. Ma, J. W. Choi, and S. Kim, “Robust beam-tracking for mmWave mobile communications,” IEEE Commun. Lett., vol. 21, no. 12, pp. 2654–2657, Dec. 2017.

[JOK+18] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, “Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data,” presented at Conference on NeurIPS MCPCD Wksp., 2018.

[JZ14] H. Ju and R. Zhang, "Throughput Maximization in Wireless Powered Communication Networks," IEEE Transactions on Wireless Communications, vol. 13, no. 1, pp. 418-428, January 2014, doi: 10.1109/TWC.2013.112513.130760.

[K19] P. Kairouz et al., “Advances and open problems in federated learning,” arXiv preprint arXiv:1912.04977, 2019.

[KAH+19] M. Khani, M. Alizadeh, J. Hoydis, and P. Fleming, “Adaptive neural signal detection for massive MIMO,” arXiv preprint arXiv:1906.04610, 2019.

[KB14] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[KF09] D. Koller and N. Friedman, Probabilistic Graphical Models: Principles and Techniques. MIT Press, 2009.

[KHK+19] S. Kim, H. Han, N. Kim, and H. Park, “Robust beam tracking algorithm for mmWave MIMO systems in mobile environments,” in Proc. IEEE 90th Veh. Technol. Conf. (VTC-Fall), Sep. 2019, pp. 1–5.

[KP20] G. Kwon and H. Park, “Limited feedback hybrid beamforming for multimode transmission in wideband millimeter wave channel,” IEEE Trans. Wireless Commun., vol. 19, no. 6, pp. 4008–4022, Jun. 2020.

[KPB+20] Y. Koda, J. Park, M. Bennis, "Distributed heteromodal split learning for vision aided mmWave received power prediction." arXiv preprint arXiv:2007.08208 (2020).

[KS17] N. S. Keskar and R. Socher, “Improving generalization performance by switching from ADAM to SGD,” arXiv preprint arXiv:1712.07628, 2017.

[KSP19] R. Kassab, O. Simeone, and P. Popovski, “Information-centric grant-free access for IoT fog networks: Edge vs cloud detection and learning,” arXiv preprint arXiv:1907.05182, 2019.

[KXN+20] J. Kang, Z. Xiong, D. Niyato, "Reliable federated learning for mobile networks." IEEE Wireless Communications 27, no. 2 (2020): 72-80.

[KYY16] E. Kalantari, H. Yanikomeroglu, and A. Yongacoglu, “On the number and 3D placement of drone base stations in wireless cellular networks,” Proceedings of IEEE Vehicular Technology Conference (VTC). IEEE, 2016, pp. 1–6.

[LBO+12] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, “Efficient backprop,” in Neural Networks: Tricks of the Trade. pp. 9–48, Springer, 2012.

[LCT+18] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, “Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach,” IEEE Journal on Selected Areas Communications, vol. 36, no. 9, pp. 2059–2070, 2018.

[LGL19] Q. Li, J. Gao, H. Liang, L. Zhao and X. Tang, "Optimal Power Allocation for Wireless Sensor Powered by Dedicated RF Energy Source," IEEE Transactions on Vehicular Technology, vol. 68, no. 3, pp. 2791-2801, March 2019, doi: 10.1109/TVT.2019.2892770.

[LHP+15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” arXiv preprint arXiv:1509.02971, 2015.

[LLX17] L. Liang, G. Y. Li and W. Xu, “Resource Allocation for D2D-Enabled Vehicular Communications,” IEEE Transactions on Communications, vol. 65, no. 7, pp. 3186-3197, July 2017

[LLX17] L. Liang, G. Y. Li and W. Xu, “Resource Allocation for D2D-Enabled Vehicular Communications,” IEEE Transactions on Communications, vol. 65, no. 7, pp. 3186-3197, July 2017.

[LQC+19] Y. Liu, Z. Qin, Y. Cai, Y. Gao, G. Y. Li, and A. Nallanathan, “UAV communications based on non-orthogonal multiple access,” IEEE Wireless Communications, vol. 26, no. 1, pp. 52–57, 2019.

[LYC+20] L. Lu, Z. Yang, M. Chen, Z. Zanget al., “Machine learning for predictive deployment of UAVs with multiple access,”arXiv preprintarXiv:2003.02631, 2020.

[MLL19] H. Mao, H. Lu, Y. Lu, and D. Zhu, “RoemNet: Robust meta learning based channel estimation in OFDM systems,” in Proc IEEE Int. Conf. Commun. (ICC), Shanghai, China, May 2019.

[MMR+17] B. McMahan, E. Moore, D. Ramage, "Communication-efficient learning of deep networks from decentralized data." In Artificial Intelligence and Statistics, pp. 1273-1282. PMLR, 2017.

[MSB+16] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, “Optimal transport theory for power-efficient deployment of unmanned aerial vehicles,” Proceedings of IEEE International Conference on Communications(ICC). IEEE, 2016, pp. 1–6.

[NAS18] A. Nichol, J. Achiam, and J. Schulman, “On first-order metalearning algorithms,” arXiv preprint arXiv:1803.02999, 2018.

[NDC19] C. Nguyen, T.-T. Do, and G. Carneiro, “Uncertainty in modelagnostic meta-learning using variational inference,” arXiv preprint arXiv:1907.11864, 2019.

[NS13] M. J. Neely and S. Supittayapornpong, “Dynamic Markov decision policies for delay constrained wireless scheduling,” IEEE Transactions on Automatic Control, 58(8):1948–1961, Aug. 2013.

[NY19] T. Nishio, R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge." In ICC 2019-2019 IEEE International Conference on Communications (ICC), pp. 1-7. IEEE, 2019.

[OAE16] T. Oliveira, A. P. Aguiar, and P. Encarnação, “Moving path following for unmanned aerial vehicles with applications to single and multiple target tracking problems,” IEEE Transactions on Robotics, vol. 32, no. 5, pp.1062–1078, 2016.

[ODS+19] J. Östman, G. Durisi, E. G. Ström, M. C. Coşkun, and G. Liva, “Short packets over block-memoryless fading channels: Pilotassisted or noncoherent transmission?” IEEE Trans. Commun., vol. 67, no. 2, pp. 1521–1536, Feb. 2019.

[OH17] T. O’Shea and J. Hoydis, “An introduction to deep learning for the physical layer,” IEEE Trans. Cognitive Commun. and Netw., vol. 3, no. 4, pp. 563–575, Dec. 2017.

[OKB+17] M. Orsag, C. M. Korpela, S. Bogdan, and P. Y. Oh, “Dexterous aerial robots - mobile manipulation using unmanned aerial systems,” IEEE Transactions on Robotics, vol. 33, no. 6, pp. 1453–1466, 2017.

[P94] M. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming. Hoboken, NJ, USA: Wiley, 1994.

[PGZ+16] L. D. P. Pugliese, F. Guerriero, D. Zorbas, and T. Razafindralambo,“Modelling the mobile target covering problem using flying drones,” Optimization Letters, vol. 10, no. 5, pp. 1021–1052, 2016.

[PJS+19] S. Park, H. Jang, O. Simeone, and J. Kang, “Learning how to demodulate from few pilots via meta-learning,” Proc. IEEE Signal Processing Advances in Wireless Commun. (SPAWC), Cannes, France, July 2019.

[PKK+19] M. P. Perrone, H. Khan, C. Kim, A. Kyrillidis, J. Quinn, and V. Salapura, "Optimal Mini-Batch Size Selection for Fast Gradient Descent," arXiv preprint arXiv:1911.06459, 2019.

[PLE19] J. Park, H. Lee, S. Eom and I. Lee, "UAV-Aided Wireless Powered Communication Networks: Trajectory Optimization and Resource Allocation for Minimum Throughput Maximization," IEEE

Access, vol. 7, pp. 134978-134991, 2019, doi: 10.1109/ACCESS.2019.2941278.

[PSK20a] S. Park, O. Simeone, and J. Kang, “Meta-learning to communicate: Fast end-to-end training for fading channels,” in Proc. IEEE 45th Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP), Barcelona, Spain, May 2020.

[PSK20b] S. Park, O. Simeone, and J. Kang, “End-to-End Fast Training of Communication Links Without a Channel Model via Online Meta-Learning,” arXiv preprint arXiv:2003.01479, 2020.

[PTB+20] S.R. Pandey, N.H. Tran, M. Bennis, "A crowdsourcing framework for on-device federated learning." IEEE Transactions on Wireless Communications 19, no. 5 (2020): 3241-3256.

[QHC+19] C. Qiu, Y. Hu, Y. Chen and B. Zeng, "Deep Deterministic Policy Gradient (DDPG)-Based Energy Harvesting Wireless Communications," IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8577-8588, Oct. 2019, doi: 10.1109/JIOT.2019.2921159.

[QualcoMM19] We are Making On-Device AI Ubiquitous. Accessed: Dec. 2019. [Online]. Available: https://www.qualcomm.com/news/onq/2017/08/16/we-are-making-device-ai-ubiquitous

[RB19] S. Ravi and A. Beatson, “Amortized bayesian meta-learning,” in Proc. International Conference on Learning Representations (ICLR), New Orleans, United States, May 2019.

[RL15] X. Rao and V. K. N. Lau, “Compressive sensing with prior support quality information and application to massive MIMO channel estimation with temporal correlation,” IEEE Trans. Signal Processing, vol. 63, no. 18, pp. 4914-4924, Sept.15, 2015.

[RLL+15] Y. Ren, F. Liu, Z. Liu, C. Wang, and Y. Ji, “Power Control in D2D-based Vehicular Communication Networks,” IEEE Transactions on Vehicular Technology, vol. 64, no. 12, pp. 5547–5562, Dec. 2015.

[RNR19] O. Rezaei, M. M. Naghsh, Z. Rezaei and R. Zhang, "Throughput Optimization for Wireless Powered Interference Channels," IEEE Transactions on Wireless Communications, vol. 18, no. 5, pp. 2464-2476, May 2019, doi: 10.1109/TWC.2019.2901491.

[RSP+14] W. Roh, J. Seol, J. Park, B. Lee, J. Lee, Y. Kim, J. Cho, K. Cheun, and F. Aryanfar, “Millimeter-wave beamforming as an enabling technology for 5G cellular communications: Theoretical feasibility and prototype results,” IEEE Commun. Mag., vol. 52, no. 2, pp. 106–113, Feb. 2014.

[S12] S. Shalev-Shwartz, “Online learning and online convex optimization,” Foundations and Trends in Machine Learning, vol. 4, no. 2, pp. 107–194, 2012.

[S18a] O. Simeone, “A very brief introduction to machine learning with applications to communication systems,” IEEE Trans. Cognitive Commun. and Netw., vol. 4, no. 4, pp. 648–664, Nov. 2018.

[S18b] O. Simeone, “A brief introduction to machine learning for engineers,” Foundations and Trends in Signal Processing, vol. 12, no. 3-4, pp. 200–431, 2018.

[SB98] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. Cambridge, MA, USA: MIT Press, 1998.

[SKL19] M. Shin, J. Kim, and M. Levorato, “Auction-based charging scheduling with deep learning framework for multi-drone networks,” IEEE Trans-actions on Vehicular Technology, vol. 68, no. 5, pp. 4235–4248, 2019.

[SMS+19] S. Samarakoon, M. Bennis, W. Saad, "Distributed federated learning for ultra-reliable low-latency vehicular communications." IEEE Transactions on Communications 68, no. 2 (2019): 1146-1159.

[SPK20] O. Simeone, S. Park, and J. Kang, “From Learning to MetaLearning: Reduced Training Overhead and Complexity for Communication Systems,” in Proc. 6G Wireless Summit, Lapland, Finland, Mar. 2020.

[SPZ+19] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, “Application of machine learning in wireless networks: Key techniques and open issues,” IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3072–3108, 2019.

[SS04] O. Simeone and U. Spagnolini, “Adaptive pilot pattern for OFDM systems,” in Proc. IEEE Int. Conf. Commun. (ICC), pp. 978–982, Paris, France, June 2004.

[SSF+16] S. Sukhbaatar, A. Szlam, and R. Fergus., “Learning multiagent communication with backpropagation”, [Online]. Available: https://arXiv:1605.07736.

[SVG+19] A. Singh, P. Vepakomma, O. Gupta, "Detailed comparison of communication efficiency of split learning and federated learning." arXiv preprint arXiv:1909.09145 (2019).

[T98] S. Thrun, “Lifelong learning algorithms,” in Learning to Learn. Springer, 1998, pp. 181–209.

[TCC20] C. Thapa, M.A.P. Chamikara, S. Camtepe, "SplitFed: When Federated Learning Meets Split Learning." arXiv preprint arXiv:2004.12088 (2020).

[TM07] D. Tandur and M. Moonen, “Joint adaptive compensation of transmitter and receiver IQ imbalance under carrier frequency offset in OFDM-based systems,” IEEE Trans. Signal Processing, vol. 55, no. 11, pp. 5246–5252, Nov. 2007.

[UNM+19] H. Ullah, N. G. Nair, A. Moore, C. Nugent, P. Muschamp, and M. Cuevas, “5G communication: An overview of vehicle-to-everything, drones, and healthcare use-cases,” IEEE Access, vol. 7, pp. 37251–37268, 2019.

[VBL+16] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, “Matching networks for one shot learning,” in Proc. Advances in Neural Information Processing Systems (NIPS), pp. 3630–3638, Barcelona, Spain, Dec. 2016.

[VGD+15] H. Van Hasselt, A. Guez, D. Silver, “Deep reinforcement learning with double q-learning”, arXiv preprint arXiv:1509.06461, 2015.

[VGS+18] P. Vepakomma, O. Gupta, T. Swedish, "Split learning for health: Distributed deep learning without sharing raw patient data." arXiv preprint arXiv:1812.00564 (2018).

[VVH16] V. Va, H. Vikalo, and R. W. Heath, Jr., “Beam tracking for mobile millimeter wave communication systems,” in Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP), Dec. 2016, pp. 743–747.

[WCY+19] Y. Wang, M. Chen, Z. Yang, T. Luo, and W. Saad, “Deep learning for optimal deployment of UAVs with visible light communications,”arXivpreprint arXiv:1912.00752, 2019.

[WF05] M. Windisch and G. Fettweis, “On the performance of standardindependent I/Q imbalance compensation in OFDM directconversion receivers,” in Proc. IEEE European Signal Processing Conference, pp. 1–5, Antalya, Turkey, Sep. 2005.

[WLA+] Y.-P. E. Wang, et al., “A primer on 3GPP narrowband Internet of Things,” IEEE Commun. Mag., vol. 55, no. 3, pp. 117–123, Mar. 2017.

[WYX19] F. Wu, D. Yang, L. Xiao and L. Cuthbert, "Energy Consumption and Completion Time Tradeoff in Rotary-Wing UAV Enabled WPCN," IEEE Access, vol. 7, pp. 79617-79635, 2019, doi: 10.1109/ACCESS.2019.2922651.

[XXZ20] L. Xie, J. Xu and Y. Zeng, "Common Throughput Maximization for UAV-Enabled Interference Channel With Wireless Powered Communications," IEEE Transactions on Communications, vol. 68, no. 5, pp. 3197-3212, May 2020, doi: 10.1109/TCOMM.2020.2971488.

[XZ+17] Y. Xiao and C. Zhu. "Vehicular fog computing: Vision and challenges," IEEE International Conference on Pervasive Computing and Communications Workshops, 2017.

[YGZ+19] Y. Yang, F. Gao, Z. Zhong, B. Ai, and A. Alkhateeb, “Deep Transfer Learning Based Downlink Channel Prediction for FDD Massive MIMO Systems,” arXiv preprint arXiv:1912.12265, 2019.

[YLC+19] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated Machine Learning: Concept and Applications,” ACM TIST, vol. 10, Feb. 2019.

[YVS10] P. Yury, P. H. Vincent, and V. Sergio, “Channel coding rate in the finite blocklength regime,” IEEE Transactions on Information Theory, vol. 56(5), pp. 2307-2359, 2010.

[YZ19] L. Yang and W. Zhang, “Beam tracking and optimization for UAV communications,” IEEE Trans. Wireless Commun., vol. 18, no. 11, pp. 5367–5379, Nov. 2019.

[ZCD+18] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “Mixup: Beyond Empirical Risk Minimization,” in Proc. of 6th ICLR, 2018.

[ZCH17] D. Zhu, J. Choi, and R. W. Heath, Jr., “Auxiliary beam pair enabled AoD and AoA estimation in closed-loop large-scale millimeter-wave MIMO systems,” IEEE Trans. Wireless Commun., vol.

16, no. 7, pp. 4770–4785, Jul. 2017.

[ZCXJ+20] C. Zhu, Y.H. Chiang, Y. Xiao, and Y. Ji. "FlexSensing: A QoI and Latency Aware Task Allocation Scheme for Vehicle-based Visual Crowdsourcing via Deep Q-Network." IEEE Internet of Things Journal, 2020.

[ZLQ+20] Y. Zhan, P. Li, Z. Qu, "A learning-based incentive mechanism for federated learning." IEEE Internet of Things Journal (2020).

[ZPR+16] D. Zorbas, L. D. P. Pugliese, T. Razafindralambo, and F. Guerriero, “Optimal drone placement and cost-efficient target coverage,”Journalof Network and Computer Applications, vol. 75, pp. 16–31, 2016.

[ZPXY+18] C. Zhu, G. Pastor, Y. Xiao and A. Ylajaaski, “Vehicular fog computing for video crowdsourcing: Applications, feasibility, and challenges,” IEEE Communications Magazine, 56(10), 58-63, 2018.

[ZPYLY+18] C. Zhu, G. Pastor, Y. Xiao, Y. Li, and A. Ylae-Jaeaeski. "Fog following me: Latency and quality balanced task allocation in vehicular fog computing," IEEE International Conference on Sensing, Communication, and Networking (SECON), pp. 1-9, 2018.

[ZSK+19] L. Zintgraf, K. Shiarli, V. Kurin, K. Hofmann, and S. Whiteson, “Fast context adaptation via meta-learning,” in Proc. International Conference on Machine Learning (ICML), pp. 7693–7702, Long Beach, California, June 2019.