Distributed Artificial Intelligence-driven open and programmable architecture for 6G networks

D3.2 Distributed AI framework for CrowdSourcing AI – Initial

Document Summary Information

Project Identifier	HORIZON-JU-SNS-2022. Project 101095363	**Acronym**	ADROIT6G
Full Title	Distributed Artificial Intelligence-driven open and programmable architecture for 6G networks		
Start Date	01.01.2023	**Duration**	36 months
Project URL	https://adroit6g.eu/		
Deliverable	Distributed AI framework for CrowdSourcing AI – Initial		
Work Package	WP3: Distributed AI Framework for CrowdSourcing AI		
Contractual due date	30/04/2024	**Actual submission date**	01/07/2024
Type	R	**Dissemination Level**	Public
Lead Beneficiary	CNIT		
Responsible Author	Carla Fabiana Chiasserini (CNIT), Antonella Molinaro (CNIT)		

Co-funded by the European Union

6GSNS

'ADROIT6G project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grand Agreement No 101095363. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.'

Contributions from	Vasos Vassiliou (CYENS), Christophoros Christophorou (CYENS), Iacovos Ioannou (CYENS), Eleni Keliri (eBOS), Christos Verikoukis (ISI/ATH), Maria Trigka (ISI/ATH), John Vardakas (IQU), Marius Iordache (ORO), Erin Seder (NXW), Giacomo Bernini (NXW), Giulio Franzese (EURE), Pietro Michiardi (EURE), Carla Fabiana Chiasserini (CNIT), Antonella Molinaro (CNIT), Chaouki Ben Issaid (UOULU), Nadia Adem (UOULU)
Peer reviewer(s)	Giacomo Bernini (NXW), John Vardakas (IQU), Vasos Vassiliou (CYENS), Christos Verikoukis (ISI/ATH), Eleni Keliri (eBOS)

Revision history (including peer reviewing & quality control)

Version	Issue Date	% Complete	Changes	Contributor(s)
V1.0	22.01.2024	5	Initial Table of Contents	Carla Fabiana Chiasserini, Antonella Molinaro (CNIT)
V1.1	29.02.2024	50	Intermediate Deliverable version	All partners
V1.2	08.03.2024	75	Partner contributions to first draft available	All partners
V1.3	18.03.2024	90	Merging of partner contributions and 1st review	All partners
V1.4	31.03.2024	90	Integration of modifications after 1st review	All partners
V1.5	15.04.2024	95	Quality Review	Philippos Philippou (eBOS)
V1.6	22.04.2024	95	New Draft for approval available (2nd review)	NXW, IQU
V1.7	18.05.2024	98	QM Check, submission preparation	All partners
V1.8	20.05.2024	99	Final Draft for approval available (3rd review)	Christos Verikoukis (ISI/ATH)
V1.9	25.06.2024	99	QM Check, submission preparation	All partners
V2.0	01.07.2024	100	Submitted Version – End deliverable creation process	John Vardakas (IQU)

© ADROIT6G, 2023

Disclaimer

The content of the publication herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.

While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the ADROIT6G consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the ADROIT6G Consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the ADROIT6G Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

© ADROIT6G Consortium, 2023-2025. This Deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.

Table of Contents

1 Executive Summary 12

2 Introduction 14
2.1 Mapping ADROIT6G Outputs 14
2.2 Deliverable Overview and Report Structure 18

3 CrowdSourcing AI Concept and Functional Architecture 20
3.1 AI/ML Framework Architecture 20
3.2 Interaction with the AI-driven Management & Orchestration and Control frameworks 25
3.3 Preliminary Implementation of the AI/ML Framework 27

4 CrowdSourcing Intelligence: The AI/ML Orchestrator 31
4.1 Enabling Robust Cooperative Training 31
4.1.1 ADROIT6G specific enhancements over state of the art 32
4.1.2 Functional architecture 32
4.1.3 Preliminary results 35
4.2 Cooperative Transfer Learning 35
4.2.1 ADROIT6G specific enhancements over state of the art 36
4.2.2 Functional architecture 38
4.2.3 Preliminary results 40
4.3 FL Designing Fair and Distributionally Robust Distributed Algorithms 42
4.3.1 ADROIT6G specific enhancements over state of the art 42
4.3.2 Functional architecture 43
4.3.3 Preliminary results 45
4.4 Meta-learning for Model Selection and Crowdsourced Model Sharing 46
4.4.1 ADROIT6G specific enhancements over state of the art 46
4.4.2 Functional architecture 47
4.4.3 Preliminary results 49
4.5 Energy-efficient Federated Learning Algorithms 50
4.5.1 ADROIT6G specific enhancements over state of the art 50
4.5.2 Functional architecture 51
4.5.3 Preliminary results 53
4.6 Decentralized Collaborative ML Techniques 53
4.6.1 ADROIT6G specific enhancements over state of the art 54
4.6.2 Functional architecture 55
4.6.3 Preliminary results 56
4.7 Edge-mobile Device Cooperative Inference 57
4.7.1 ADROIT6G specific enhancements over state of the art 57
4.7.2 Functional architecture 58
4.7.3 Preliminary results 59
4.8 Scheduling and Offloading Techniques Across Multiple Domains 61
4.8.1 ADROIT6G specific enhancements over state of the art 61
4.8.2 Functional architecture 62

© ADROIT6G, 2023

5 Data Generation Pipeline 65
5.1 Multimodal Generative Modelling and Data Generation 65
5.1.1 ADROIT6G specific enhancements over state of the art 65
5.1.2 Functional architecture 66
5.1.3 Preliminary results 70
5.2 Integration Algorithms to Speed Up Model and Data Generation 72
5.2.1 ADROIT6G specific enhancements over state of the art 72
5.3 Multi-view Extension 73
5.3.1 ADROIT6G specific enhancements over state of the art 73
5.3.2 Functional architecture 73

6 Service Orchestration and Management 75
6.1 Robust Algorithms for Joint Resource Management satisfying Performance and Energy Constraints 75
6.1.1 ADROIT6G specific enhancements over state of the art 75
6.1.2 Functional Architecture 76
6.2 Traffic and Energy-Efficient BDIx agent Configurations 78
6.2.1 ADROIT6G specific enhancements over state of the art 79
6.2.2 Functional architecture 81
6.3 Traffic predictor 83
6.3.1 ADROIT6G specific enhancements over state of the art 84
6.3.2 Functional architecture 84
6.3.3 Preliminary results 86
6.4 Optimal D2D Topologies Formation 87
6.4.1 ADROIT6G specific enhancements over state of the art 88
6.4.2 Functional architecture 88
6.5 Predictors for Real-time VNF Migration, Placement and Scale-out 89
6.5.1 ADROIT6G specific enhancements over state of the art 90
6.5.2 Functional architecture 91
6.6 Migration of Containerized AI/ML-based Applications 95
6.6.1 ADROIT6G specific enhancements over state of the art 95
6.6.2 Functional architecture 96

7 Conclusions 98

8 References 99

© ADROIT6G, 2023

List of Figures

Figure 3.1 Architecture of the AI/ML framework realizing the concept of CrowdSourcing AI and interaction with the Management and Orchestration framework. 20

Figure 3.2 AI/ML training and inference functions 21

Figure 3.3 AI/ML framework architecture 21

Figure 3.4 Workflow for the end-to-end (e2e) service provisioning with AI-based zero-touch closed loops 27

Figure 3.5 Initial implementation of the AI/ML framework 28

Figure 3.6 Swagger UI for a model serving instance 30

Figure 4.1: Operational pipeline in the case of supervised ML. 31

Figure 4.2: The main steps of the DepL solution strategy. 33

Figure 4.3: Training cost vs guaranteed loss. DepL performance is compared to the optimum and the best-exp strategy. 35

Figure 4.4: Illustration of the dual-step transfer learning (DSTL)-based prediction model in a cellular network with clusters based on the similarity in traffic distribution among gNodeBs 37

Figure 4.5: Localisation Module 39

Figure 4.6: Performance metrics representation for sms, call, and internet traffic when the base model of cluster 1 is applied on the aggregated dataset of cluster 2 and 3. 40

Figure 4.7: Training and the validation loss of the transferred model vs epochs. 41

Figure 4.8: Performance metrics representation for sms, call, and internet traffic when the base model of cluster 3 is applied on the aggregated dataset of Cell 5577 41

Figure 4.9: Illustration of the CA-AFL algorithm functional architecture. 44

Figure 4.10: Performance of CA-AFL compared to baselines in terms of the total energy. 45

Figure 4.11: Integration of MERGE in the functional AI/ML framework. 48

Figure 4.12: Performance of the proposed MERGE meta-RL framework for CrowdSourcing AI. 49

Figure 4.13: Illustration of the Fed-Sophia algorithm functional architecture. 52

Figure 4.14: Performance of Fed-Sophia against other baselines for MNIST and FMNIST datasets using a CNN model 53

Figure 4.15: Illustration of the DIN algorithm functional architecture. 56

Figure 4.16: Performance of Q-DNGAM against other baselines for the a1a dataset. 57

Figure 4.17: Reference scenario. 58

Figure 4.18: Application-level message exchange for inference execution at an edge server (left) and at a mobile device (right) 60

Figure 4.19: Total delay for the two possible model deployments, full and pruned models (pruning factor 70%), and variable edge server load, for DNN model: (a) ResNet-152, (b) DenseNet201, (c) MobileNet. 60

Figure 4.20: Online Computation Offloading Architecture [72] 63

© ADROIT6G, 2023

Figure 5.1: Overview of the MLD method: list of unimodal autoencoders and joint diffusion model 67
Figure 5.2: Implementation details about the Score Network architecture. 68
Figure 5.3: Interaction across different blocks for the Data Generation Pipeline 69
Figure 5.4: implementation scheme of the multi-view extension of the MLD framework. 73
Figure 6.1. Binary tree representation of the resource orchestration system model in Low-part. 77
Figure 6.2: BDIx architecture. 79
Figure 6.3: A D2D Communication Network Example Using BDIx Agents. 81
Figure 6.4: Interaction of all ADROIT components with BDIx agents 83
Figure 6.5: System model for Traffic prediction of 6G multivariate heterogeneous mobile network. 85
Figure 6.6: Prediction of (a) Network management traffic (b) Live audio/video traffic (c) HTTP and live streaming and video traffic (d) Fun applications and online gaming traffic (e) Downloading and system update traffic (f) Unidentified traffic. 87
Figure 6.7. Double DQN agent. 92
Figure 6.8. Algorithm for training Dueling DDQN 93
Figure 6.9: Migration protocol under MOSE. 97

List of Tables

Table 2.1: Adherence to ADROIT6G's GA Deliverable & Tasks Descriptions 14

Table 2.2 Links to other Project's Outputs/ Work 19

Table 4.1: Error metric for the dual-step transfer learning-based prediction model for cell 5577 41

Table 4.2: Computation/communication energy costs and corresponding carbon footprints for MNIST with CNN for target accuracy of 75% 53

Table 4.3: Computation/communication energy costs and corresponding carbon footprints for the Phishing dataset for a target optimality gap of 10^{-5} 57

Table 5.1: Performance of MLD against competitors on the MNIST-SVHN dataset 71

Table 5.2: Performance of MLD against competitors on the MHD dataset 71

Table 5.3: Performance of MLD against competitors on the CelebA-HQ dataset 71

© ADROIT6G, 2023

Glossary of terms and abbreviations

Abbreviation / Term	Description
3GPP	Third Generation Partnership Program
ADMM	Alternating Direction Method of Multipliers
AI	Artificial Intelligence
ANFIS	Adaptive Neuro-Fuzzy Inference System
BDIx	Beliefs-Desires-Intention eXtended
BS	Base Station
BSoTA	Beyond State of The Art
CA-AFL	Channel-Aware Agnostic Federated Learning
CNN	Convolutional Neural Network
CN+OLS	Conditional Neural Network with Optimistic Linear Support
CPE	Customer-Premise Equipment
C-RAN	Cloud-Radio Access Network
D2D	Device-to-Device
DAI	Distributed Artificial Intelligence
DDDQN	Dueling Double Deep Q-learning Network
DDQN	Double Deep Q-learning Network
DepL	Dependable Learning
DL	Deep Learning
DNN	Deep Neural Network
DPI	Deep Packet Inspection
DQN	Deep Q-learning Network
DRL	Deep Reinforcement Learning
DU	Distributed Unit
e2e	end-to-end
EE	Energy Efficiency
FID	Frechet Inception Distance
FL	Federated Learning
GA	Grant Agreement

© ADROIT6G, 2023

Gnb	Next Generation Node B (5G Node B)
ICO	Intelligent Computational Offloading
IID	Independent and Identically Distributed
IoT	Internet of Things
KPI	Key Performance Indicator
M&O	Management and Orchestration
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MCDA	Multi-Criteria Decision Analysis
MDP	Markov Decision Process
MEC	Mobile-Edge Computing
MERGE	MEta Reinforcement learning framework for tunable RL agents at the edge
MIMO	Multiple Input Multiple Output
MLD	Multi-modal Latent Diffusion
MINLP	Mixed Integer Non-Linear Programming
ML	Machine Learning
MLaaS	ML as a service
MOFQI	Multi-Objective Fitted Q-Iteration
MORL	Multi-Objective Reinforcement Learning
MOSE	Migration Orchestration framework for microservices at the Edge
MSE	Mean Squared Error
MVLD	Multi-view Latent Diffusion
NFV	Network Functions Virtualization
NN	Neural Network
NOP	No-Operation
O-RAN	Open Radio Access Network
PMF	Probability Mass Function
PS	Parameter Server
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network

© ADROIT6G, 2023

RL	Reinforcement Learning
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RNN - BLSTM	Recurrent Neural Network - Bidirectional Long Short Term Memory
RU	Radio Unit
SDE	Stochastic Differential Equation
SLA	Service Level Agreement
SNR	Signal-to-Noise Ratio
SoTA	Sate of The Art
TL	Transfer Learning
TO	Technical Objective
UAV	Unmanned Aerial Vehicle
UE	User Equipment
UI	User Interface
UE-VBS	User Equipment – Virtual Base Station
Umi	Urban Micro
URLLC	Ultra-Reliable Low-Latency Communications
VAE	Variational Autoencoder
VM	Virtual Machine
VNF	Virtual Network Function

© ADROIT6G, 2023

1 Executive Summary

ADROIT6G proposes an Artificial Intelligence (AI)/Machine Learning (ML) framework that interprets the concept of ML-as-a-Service (MLaaS) in an innovative manner, offering mechanisms for the management and distributed provisioning of AI/ML pipelines efficiently and effectively. It does so by enabling the novel concept of CrowdSourcing AI to serve an AI/ML service request made by the AI-driven Management and Orchestration (M&O) framework. The CrowdSourcing AI concept allows for collaboratively training, updating/fine-tuning, sharing, and usage of AI/ML models among the network nodes, thus enabling and exploiting a continuous cycle for learning, validation, refinement, and transfer among different domains and the nodes therein. Remarkably, through model compression techniques such as pruning, the CrowdSourcing AI framework can also create ML models that adapt to the computing and energy resource availability of different domains, thus making AI/ML-based solutions scalable and suitable for training and inference execution at heterogeneous devices.

This deliverable reports the activity performed within WP3 towards the design of the CrowdSourcing AI framework (see *Technical Objective (TO) 3: Architect a distributed and secure CrowdSourcing AI*) that realizes the above innovative concept. The document first presents the framework architecture, the role of the entities therein, and how they interact to implement an efficient AI/ML pipeline. This is a refined design with respect to what reported in D2.2 and will serve as input to D2.3, which will present the final system architecture. The document also specifies the synergy between the CrowdSourcing AI framework and the AI-driven M&O framework. One of the essential entities in the CrowdSourcing AI framework is the AI/ML Orchestrator where the framework intelligence resides. The deliverable thus introduces the algorithmic solutions that have been developed within the project to implement such intelligence. They include mechanisms for distributed training processes that meet the services requirements on the AI/ML models training while using the system resources efficiently and, thus, making the use of AI/ML model sustainable. Such algorithms leverage on various emerging approaches, including sequential, transfer, federated, and decentralized learning. Further, they address the aspect of ML model selection through, e.g., metalearning-based solutions. Notably, the proposed mechanisms do not address only ML model training but also the execution of inference tasks through ML models. Specifically, we propose schemes that exploit the cooperation, through task offloading, between end-users and edge servers for AI/ML-based services requiring low latency or privacy preservation. Subsequently, the document delves into the management of data – a crucial aspect in data driven approaches and an essential part of the AI/ML pipeline management. In this context, the deliverable tackles the issue of using multimodal data, whose transfer may cause both high bandwidth consumption and information loss or degradation. To cope with such issues, we have developed novel techniques for generating synthetic data at the processing entities whose distribution resembles that of real-world data, without requiring the continuous streaming of data samples in the network system. Finally, the document presents a set of network services that take advantage of the AI/ML pipeline implemented through the CrowdSourcing AI framework. Such network services exploit AI/ML trained models for (i) the dynamic and energy-efficient management of network and computing resources, (ii) the implementation of collective intelligence of Belief-Desire-Intent Extended (BDIx) agents, informed by precise localization information, to improve network management and create device-to-device network topologies, (iii) traffic prediction, and (iv) the migration of network functions.

The activity performed within WP3 and reported in this deliverable has revealed the high potential of the developed CrowdSourcing AI framework, showing that it effectively contributes to the achievement of:

- *TO3 KPI 1: x10 improvement of energy efficiency (under light load)*, through energy-efficient training and resource management (Sections 4.1, 4.5, 4.6, 4.8, 6.1, and 6.2);
- *TO3 KPI 2: 70% more types of devices contributing to learning and inference processes*, through distributed learning and inference (Section 4);
- *TO3 KPI 3: 50% reduction in data transferred for decision making,* through transferred learning and synthetic data generation (Sections 4.2, 4.4, 4.7, 5);
- *TO4 KPI 1: >50% reduction in network bandwidth occupation,* through efficient distributed learning (Sections 4.3, 6.2, 6.3, 6.4, 6.5, 6.6);

© ADROIT6G, 2023

- *TO4 KPI 2: >30% increased robustness to node faults*, through synthetic data generation (Section 5).

2 Introduction

2.1 Mapping ADROIT6G Outputs

This section presents the ADROIT6G 's Grant Agreement (GA) commitments, as extracted from the formal deliverable and task description, in respect to their outputs and work to be performed.

Table 2.1: Adherence to ADROIT6G's GA Deliverable & Tasks Descriptions

ADROIT6G GA Component Title	ADROIT6G GA Component Outline	Respective Document Chapter(s)	Justification
DELIVERABLE			
D3.2 Distributed AI framework for CrowdSourcing AI – Initial	First release of software and AI algorithmic components of ADROIT6G implementing energy efficient automated AI orchestration, multimodal generative models for scalable monitoring and secure data distribution, CrowdSourcing AI and AI algorithms for joint resource management.	Section 3 Section 4 Section 5 Section 6	This deliverable reports the activity performed within WP3 towards the design of the CrowdSourcing AI framework ("TO3: Architect a distributed and secure CrowdSourcing AI"). It specifies the functional architecture of the framework and the algorithmic solutions to be executed for an efficient and collaborative AI/ML pipeline. It then presents a set of relevant AI/ML-based network services.
TASKS			
Task 3.1 Energy-efficient & automated AI orchestration	This task is responsible for the development, interfacing, and configuration of the automated AI orchestrator according to the latest ETSI OSM releases and ensuring its integration with the other ADROIT6G architectural elements and underlying testbed infrastructure. Novel data-driven algorithms that will create predictors for VNF placement in real time will be developed, reaching an optimal trade-off between performance, deployment complexity and cost. More specifically, Deep Reinforcement Learning will be leveraged to predict the correct placement of the services at MEC nodes, as well as decide their live migration and scale-out when needed, with energy optimization being one of the main objectives. Due to the large number of	Section 6	Task 3.1's objectives are aligned with the contributions described in Section 6. More specifically, in Section 6.4, CYENS focuses on optimal D2D topologies formation through the integration of BDIx Agents and the application of state-of-the-art Minimum-Weight Spanning Tree ML Algorithms, utilizing the capabilities of Graph Neural Networks. ISI/ATH and IQU, in Section 6.5, exploit Advanced Deep Q-Learning Network architectures to

	parameters involved in the envisioned 6G MANO environment, it is not possible to capture all possible state transitions and apply standard MDP methods. Instead, as a first step, dynamic Q-learning will be applied, that can be proven to converge to optimal configurations, without a priori knowledge. NXW will contribute to this task by implementing the dynamic provisioning and configuration of AI/ML pipelines. CNIT will develop the algorithms for the sustainability-aware orchestration of distributed AI. ATH and IQU will develop the predictors for real time VNF placement, migration and scale-out. CYENS will develop energy optimization and orchestration algorithms under a DAI framework. (Output: D3.2, D3.3).		enhance scaling and migration of any given service function in the ADROIT6G infrastructure. Finally, CNIT, in Section 6.6, deals with the management of containerized AI/ML functions, tackling the problem of migrating such functions whenever AI/ML-based network services are instrumental to provide continuous time-critical services to mobile end users.
Task 3.2 Multimodal generative modelling & data generation	The focus of this task is on the analysis and implementation of multimodal generative models. The first key objective is to define a sensible modelling approach to multimodality, which involves the study of the broad framework of energy-based models. EURE will focus on score-based diffusion models, and their extension to multiple input modalities. This task calls for the definition of novel stochastic differential equations describing the information diffusion process that "destroys" multiple inputs into a simple, easy to sample noise distribution, on the one hand, and the derivation of an inverse diffusion process that begins with a noise sample and lands into the original multimodal distribution of input space. A second key objective relates to the design of efficient numerical integration algorithms to speed-up both model training and sample generation, which are two known bottlenecks of score-based diffusion models in general. A third important objective of this task is to enable cross (or conditional) modality generation: given an available input modality the goal is that of defining a training and inference procedure that allows the generation of missing modalities. The ability for the generative models developed in this project to perform conditional generation allows their integration within the ADROIT6G orchestration framework, which will optimize	Section 5	The contributions of Task 3.2 are elaborated in Section 5, and, as foresee, they focus on the development and integration of multimodal generative models within the ADROIT6G framework. Specifically, in Section 5.1, EURE delves into the deployment of diffusion-based generative models, addressing the intricate challenges of multimodal data synthesis and reconstruction through advanced diffusion processes (MLD, Multimodal Latent Diffusion). These models are crucial for maintaining the integrity and usability in multimodal scenarios, especially under conditions where data modalities may be incomplete or corrupted during transmission. Section 5.2 contains advanced numerical integrations schemes. A key aspect of this contribution is the operational shift towards

	system resource utilization by inducing missing modalities in the interest of performance and parsimony. A by-product of this task is the study of modality importance, which amounts in the definition and computation of a ranking of modalities for their contributions in reconstruction tasks, a signal that is going to inform the ADROIT6G orchestration framework optimization procedures. The output of this task will be software components implementing the generative models, model training procedures, and the integration of the model within the ADROIT6G architecture. ORO will contribute to the design and analysis of the multimodal model and studies for the resource's optimization utilization and integration within ADROIT6G orchestration framework. IQU will contribute to the development of numerical integration algorithms to speed up model and sample generation. EURE will contribute to the development and analysis of diffusion-based generative models that accept multiple input modalities, novel integrators for improving training and inference, and cross/conditional multimodal generation. (Output: D3.2, D3.3).		using latent space for generative models. This shift allows data to be encapsulated in a more condensed, abstract format, simplifying the dynamics involved and reducing the computational complexity. Finally, Section 5.3 presents an extension of MLD to the multi-view scenario to effectively address challenges in managing multi-view data from diverse sources. MVLD reconstructs missing views in multiple sources streams, enhancing data analysis and optimizing network efficiency.
Task 3.3 CrowdSourcing AI in multi-stakeholder scenarios	A first goal of this task is to develop the overarching control framework that implements the concept of CrowdSourcing AI in the distributed ecosystem. It will define the internal architecture of the control framework, as well as the information and the AI model flow therein. Furthermore, it will specify the interactions between the control framework and the ecosystem stakeholders for efficient discovery, selection, and update of AI models. Such stakeholders include various, heterogeneous domains as well as external users. The framework architecture will have to be not only highly modular and scalable, but also flexible to accommodate any existing and yet-to-be AI/ML paradigm. Importantly, it will have to guarantee a light interaction between the different entities through open interfaces and integrate a data pipeline that is both pervasive and tailored to the AI/ML models to be used. A second goal of the task is to develop a meta-learning	Section 4	The main contribution of Task 3.3 refers to the initial development of the algorithmic control framework implementing the intelligence needed in the AI/ML Orchestrator to run an AI/ML pipeline for efficient model selection, model training/updating, and inference execution. Sections 4.1, 4.2, and 4.3 describe cooperative learning mechanisms, which (resp.) leverage on sequential, transfer, and federated learning. Section 4.4 presents a meta-learning mechanism for ML model selection and sharing. Section 4.7 reports an early study on efficient inference

	approach that optimally selects and configures the AI paradigm, thus enabling a sustainable decision-making process and fitting the domain where this will have to take place. The meta-learning solution must choose the AI model and paradigm that best suit the purpose of the decision process and the related operation conditions, given the task to be performed, the existing suitable models, the available data, as well as the characteristics of the domain and of the nodes therein. The envisioned approach must be, again, scalable, and flexible, and able to perform swiftly and efficiently. The output of this task will be the software implementation of (i) the internal architecture and information/AI model flow for the CrowdSourcing AI control framework; (ii) the open APIs between the control framework and the relevant stakeholders in the ecosystem scenario; (iii) the definition and software implementation of the meta-learning approach for a task and context-aware selection of the best AI strategy to use. CNIT will lead this task and contribute to the development of the control layer and of the interactions among the ecosystem entities implementing the concept of CrowdSourcing AI. NXW will implement the interfaces with the M&O and control frameworks for discovery, selection, and update of AI models. OULU will propose a framework based on matching theory to match data and algorithms providers. This will be carried out in a decentralized manner according to constraints in terms of applications requirements. (Output: D3.2, D3.3).		task execution by exploiting cooperation between edge and user devices.
Task 3.4 AI Algorithms for joint resource management	This task will propose and develop distributed, communication and energy efficient model training and inference algorithms under resource and application constraints. We will explore a gamut of techniques including federated learning and federated distillation for both star (parameter server) and random topologies (D2D communication and other meshed topologies). This task will also investigate how to allocate resources to help train faster and more robust models. In terms of sustainability	Section 4 Section 6	Section 4.5 aligns with the task's goal of developing distributed, communication and energy-efficient model training algorithms. It proposes enhancements to FL to improve the energy efficiency in the star (parameter server) topology. Section 4.6 explores techniques like FL for various random

	to reduce the energy bill of model training we will propose novel model quantization algorithms that strike a balance between model accuracy and energy savings (accuracy per watt). This will be done by leveraging recent advances in model pruning and quantization. To ensure robustness, we will leverage Bayesian learning to provide uncertainty quantification for mission-critical applications. Finally, this task will investigate how communication theoretic principles (diversity and redundancy) can help speed up model training while ensuring privacy. OULU leads this task and will contribute to the development of AI algorithms for resource management in terms of bandwidth and compute power. In addition, OULU will characterize the energy bill for various model training algorithms and inference considering the application requirements, including the trade-off between accuracy, energy, and reliability. ATH will contribute to the development of robust algorithms for Joint Resource Management, satisfying performance and energy constraints. CYENS will contribute to the development of REST services on BDIx agents (DAI Framework) for coordinating the agents in terms of Joint Resource Management, satisfying performance and energy constraints. EBOS will contribute to the development of algorithms for scheduling and offloading actions across the edge and cloud domains, guaranteeing tight service delay constraints and minimum energy consumption. (Output: D3.2, D3.3).		topologies (D2D communication and other meshed topologies), as called for in the task description. Regarding the task's sustainability goal to balance model accuracy and energy savings, Sections 4.5 and 4.6 examine the trade-off between model performance and energy efficiency in FL by studying the energy bill for various model training algorithms. Section 4.8 explores techniques for efficiently scheduling and offloading AI/ML workloads across different domains or nodes in the network. Section 6.1 focuses on designing robust algorithms for joint resource management that aim to satisfy both performance and energy constraints. Section 6.2 ties into the task's objective of developing REST services on BDIx agents (DAI Framework) for coordinating the agents in terms of joint resource management, satisfying performance and energy constraints.

2.2 Deliverable Overview and Report Structure

The rest of the deliverable is organized as follows:

- Section 3 presents the functional architecture of the CrowdSourcing AI framework, the role of the entities therein, and how they interact to implement an efficient AI/ML pipeline, besides highlighting the synergy between the CrowdSourcing AI framework and the AI-driven M&O framework.
- Section 4 introduces the algorithmic solutions to be implemented at the AI/ML Orchestrator for distributed learning and inference processes that satisfy the services requirements on the AI/ML models training and execution while using the system resources efficiently.
- Section 5 introduces data management mechanisms with particular emphasis on the generation of multimodal synthetic data, so as to avoid information loss or data quality degradation, and to reduce the amount of data that the system has to collect for training.

© ADROIT6G, 2023

- Finally, Section 6 presents a set of relevant network services that take advantage of the AI/ML pipeline implemented through the CrowdSourcing AI framework.

Finally, Table 2.2 provides the link of this deliverable with other deliverables of the project.

Table 2.2 Links to other Project's Outputs/ Work

WP #	Task #	Deliverable No related	Content
2	2.1, 2.3	D2.2	CrowdSourcing AI framework (architecture of the AI/ML)
2	2.1, 2.3	D2.3	Refined CrowdSourcing AI framework (architecture of the AI/ML)
3	3.1, 3.2, 3.3, 3.4	D3.3	Refinement and performance assessment of the CrowdSourcing AI framework
4	4.2, 4.3	D4.1	AI-driven Management and Orchestration and Control frameworks incorporating closed-loop control functions into network services
5	5.2	D5.3	Initial testing of implemented CrowdSourcing AI-related solutions and further improvement

© ADROIT6G, 2023

3 CrowdSourcing AI Concept and Functional Architecture

CrowdSourcing AI in ADROIT6G aims to be an innovative vision of the Machine Learning as a Service (MLaaS) approach in distributed systems. The concept enables collaborative training of ML models and the execution of inference tasks through such models, using distributed techniques and enabling a continuous cycle for learning creation, validation, refinement, and transfer of ML models across domains. Crowdsourcing also enables saving in computational power and memory associated with ML training where a domain requiring an ML model can retrieve an already-trained model and further refine it using its own data. In addition, the concept uses meta-learning techniques to support discovery and selection of the best available ML model for the execution of a given task and for the training of a model with the aim at establishing the best solution for the amount of energy consumed by ML model training/updating/inference and the quality of the decisions made. The orchestration of ML model training and inference takes advantage of the number and multiple types of computing entities (devices, network nodes) and the amount of (computing/networking) resources that can be dedicated by such entities to the training and inference tasks.

The AI/ML framework enables the cooperative, energy-efficient approach to AI offered through Crowdsourcing. The framework covers deployment of AI/ML services across multiple domains, allows an efficient sharing of ML models, and supports the execution of the methods and functions for Crowdsourcing Intelligence in ML model training and inference described in Sections 4 and 5. Below, we provide an overview of the functional architecture of the framework that is conceived to implement the concept of CrowdSourcing AI, in alignment with what has been defined in WP2 and deliverable D2.2 (and shown in Figure 3.1), specifying (i) the role of the different entities composing the framework as well as their interaction, and (ii) the synergy between the CrowdSourcing AI and the AI-driven Management and Orchestration (M&O) frameworks.

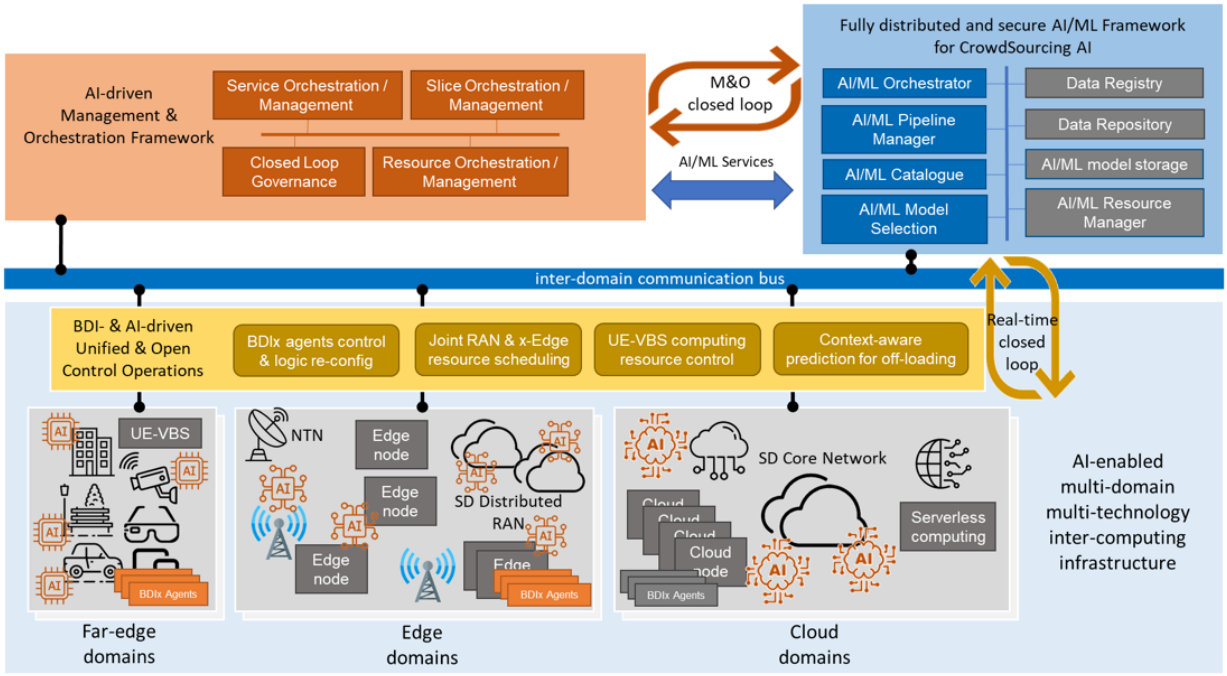


Figure 3.1 Architecture of the AI/ML framework realizing the concept of CrowdSourcing AI and interaction with the Management and Orchestration framework.

3.1 AI/ML Framework Architecture

Starting from the high-level specification provided in WP2 and deliverable D2.2, as part of the WP3 activities the design of the AI/ML Framework has been further elaborated to fully support Crowdsourcing functionalities essential to creating a distributed ecosystem for the lifecycle of AI/ML services. This includes supporting a variety

of techniques described in detail later in the deliverable, including cooperative and collaborative training of models, transfer and federated learning, intelligent model selection, and intelligent placement and management of AI/ML services in distributed execution environments. The framework deploys and manages AI/ML training and inference functions (Figure 3.2) via agents on virtualized infrastructures spanning across multiple domains from terrestrial and non-terrestrial far edge and cloud. The main target in ADROIT6G is the integration of cloud-native infrastructures (e.g., based on k8s), as well as extreme edge (e.g., with Belief-Desire-Intent Extended (BDIx), User Equipment – Virtual Base Station (UE-VBS) and satellite computing where applicable). This evolved version of the AI/ML framework functional architecture serves as input for the final ADROIT6G framework specification and design in WP2.

The AI/ML framework architecture is broken-down into 8 functional blocks as depicted in Figure 3.3, containing a mix of both innovative (blue blocks) and State of The Art (SoTA) (grey blocks) elements.

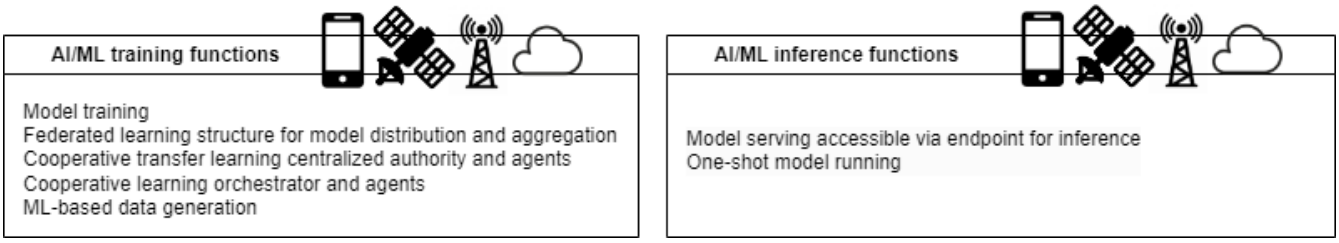


Figure 3.2 AI/ML training and inference functions

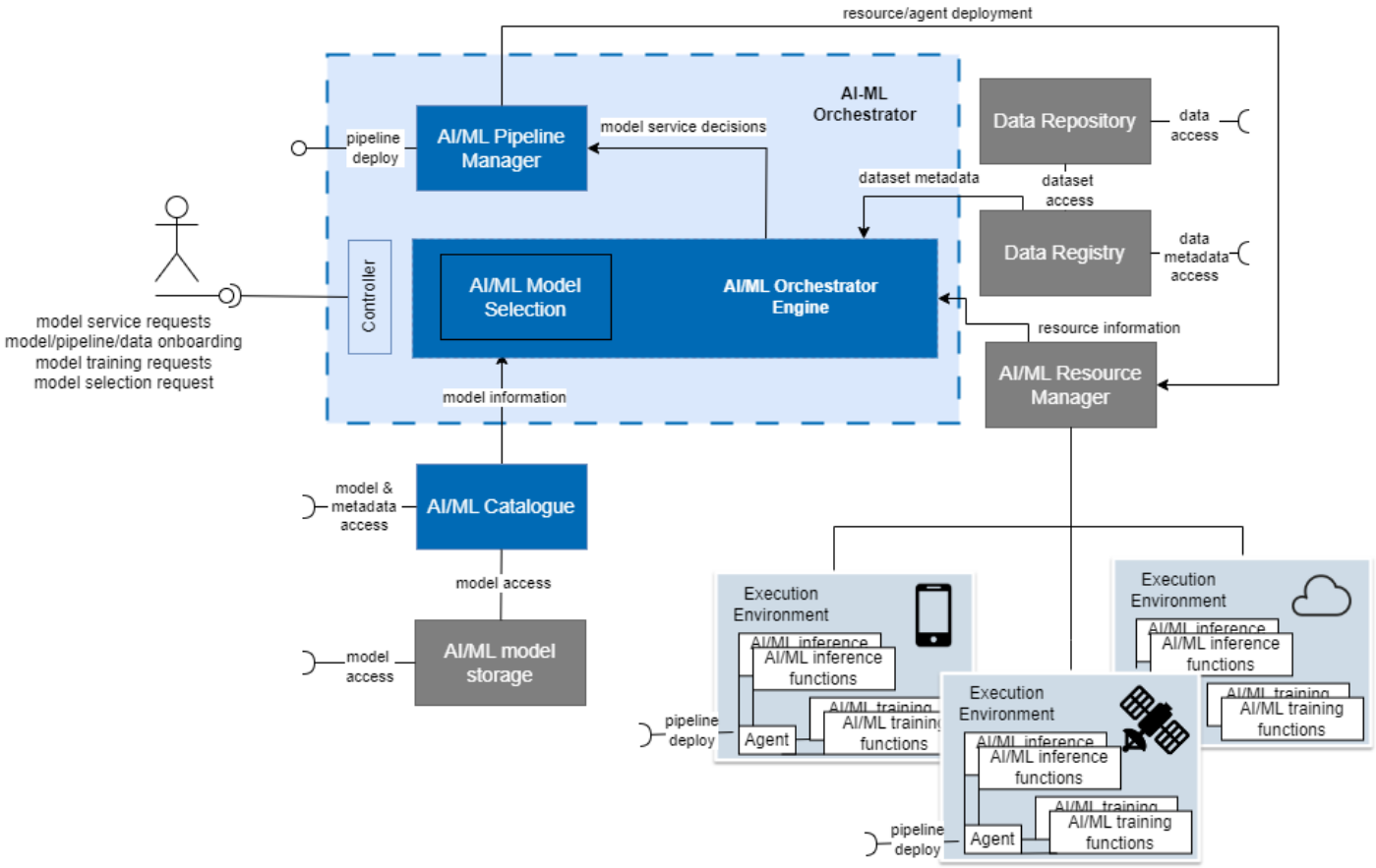


Figure 3.3 AI/ML framework architecture

AI/ML inference functions are automatically deployed by the AI/ML framework to realize model serving services. This includes the automated deployment of model serving instances, where a model which is selected either by an external user or the AI/ML framework based on supplied criteria, is deployed and reachable via an endpoint (e.g. REST-based) for continuous run-time inference requests. It also includes model inference services where a model is run on a specified dataset and an inference result is returned to the user. AI/ML training functions are associated with the training/retraining of a model, which can be via isolated, cooperative, collaborative, or federated methods, executed as single entities for isolated training, or interconnected over multi-domain in the case of federated, cooperative and collaborated learning strategies. They are also related to pipelines for ML-based data generation. The choice of deployment location of AI/ML inference and training functions can be specified at the time of AI service request or can be autonomously chosen based on resource availability and/or policy information on the AI service. Model monitoring capabilities can be incorporated into inference function services as AI agents, where the performance of the model is tracked, and actions can be taken for further rounds of optimization and training as new data becomes available or as the model's performance varies over time.

The following gives an overview of the functionalities and interactions of each block of the AI/ML framework.

AI/ML Orchestrator

The AI/ML Orchestrator serves as the central component within the AI/ML Framework for CrowdSourcing AI, it includes the AI/ML Pipeline Manager for the execution of training and inference functions, the AI/ML Orchestration Engine for deciding a service orchestration approach including the Model Selection for the choice and optimization of a model for an AI/ML service, as represented in Figure 3.3. The Orchestrator executes requests including for model service, model/pipeline/data onboarding, model section, and model training from users.

The AI/ML Orchestrator offers the following services through its northbound interface:

- upload models into the framework for training and distribution;
- onboard training pipelines to be executed by the framework;
- upload datasets to be available in the training and sharing of data across domains;
- request for model services including model selection for a given task, model serving for continuous inference and model execution for single/batch inference results;
- request a model training or re-training using an onboarded a pipeline.

AI/ML Orchestrator Engine

The AI/ML Orchestrator Engine is tasked with determining the orchestration approach for each requested AI/ML task. It is supported in its decision-making processes by the other functional blocks of the AI/ML Framework, either by supplying input data or executing its directives. If data need to be synthetically generated for model training, the Orchestration Engine performs the selection of the nodes and resources for executing data generation.

The main functionalities of an AI/ML Orchestrator Engine have been determined in part by the HEXA-X project [1], including:

- Optimized model selection;
- Selection of the node(s), resources, and dataset(s) to be used for model training, or for a model that needs to be updated;
- Selection of the nodes and resources for executing an inference.

The AI/ML Orchestrator Engine incorporates innovative technologies aimed at delivering services across far-edge, edge, and cloud domains. It consumes the APIs of various blocks within the AI/ML Framework including:

- The AI/ML Pipeline Manager, to which it triggers the on-boarding, management and execution processes for pipelines;
- The AI/ML Resource Manager, from which it receives the information on the network, computational, and energy resource capability of the system nodes that can contribute to the AI/ML tasks;
- Data Registry, from which it receives information on available datasets;
- AI/ML Catalogue, from which it receives information on available models.

By leveraging information from these blocks, the AI/ML Orchestrator Engine aims to streamline resource management and enhance training and inference services delivery efficiency. This approach enables seamless integration and utilization of resources across different domains, fostering improved performance and scalability. Overall, the AI/ML Orchestrator Engine's development underscores a strategic focus on enabling and facilitating collaborative AI, leveraging technology to optimize resource allocation and service deployment in diverse computing environments.

AI/ML Model Selection

The AI/ML Model Selection is contained as a subcomponent in the AI/ML Orchestrator Engine. It contains the algorithms and methods of model selection based on service and resource requirements (e.g., full-size or pruned models). This can include the optimization of a model from parameters (set in a configuration and implementation file) which are used as input for techniques (i.e., Energy-Efficient Model Selection, Multi-Criteria Decision Analysis (MCDA) for Model Selection) for the selection of specific models using their metadata such as accuracy, latency, energy consumption, and model robustness. Moreover, after the correct model selection, the selection of the model's hyperparameters is executed using AUTOML, which is the automatic model selection of the most updated model according to the highest accuracy.

The AI/ML Model Selection also contains the methods for the BDIx agent and UE-VBS component within a network or system architecture, enabling dynamic selection of the most appropriate AI/ML models based on specific criteria or configurations. This advanced feature allows for optimal performance, efficiency, and accuracy in operations requiring intelligence and adaptability. The selection process involves evaluating various models based on predefined criteria such as accuracy, latency, computational efficiency, and the specific requirements of the task at hand. This ensures that the model chosen is best suited for the real-time analytics or complex data analysis tasks it will perform. Configurable parameters within the model selection configurations can include performance thresholds, resource usage limits, and environmental context, such as network conditions or device capabilities. These parameters allow the system to tailor model selection to the current operational context and objectives.

The AI/ML Model Selection includes the following capabilities:

- Real-time Model Adaptation: Developing mechanisms that allow for the real-time adjustment of models in response to changing data characteristics or operational requirements, ensuring optimal performance across different UCs;
- Energy-Efficient Model Selection: Introducing criteria for selecting models not only based on their accuracy and computational efficiency but also their energy consumption, particularly important for mobile and edge computing applications;
- MCDA for Model Selection: Implementing MCDA techniques to evaluate models based on multiple criteria simultaneously, such as accuracy, latency, energy consumption, and model robustness, offering a holistic approach to model selection.

The block interacts with the other functional blocks in the architecture, including:

- AI/ML Orchestrator Engine, from which it receives requests of model selection for a give configuration;
- AI/ML Catalogue, to which it requests and receives information on available models.

© ADROIT6G, 2023

AI/ML Pipeline Manager

A main function of the AI/ML Pipeline Manager is to manage the deployment of training and runtime AI/ML functions defined through pipelines, using a service-based approach. These pipelines are onboarded to the Pipeline Manager and can be executed on demand, or via scheduled deployments through exposed interfaces.

Baseline state of the art solutions include pipeline and workflow orchestrators such as Prefect [2], Argo [3], Apache Airflow [4], and Kubeflow [5]. Leveraging on the available solutions, the functionality of the ADROIT6G AI/ML Pipeline Manager provides innovations in two different directions. First, the AI/ML Pipeline manager supports deployment over distributed domains throughout multiple computing technologies domains (e.g., infrastructure as a service – IaaS, container as a service – CaaS) to be able to assign and execute pipeline deployments via agents on the available physical/virtual infrastructures. The choice of infrastructure, location, and time is assigned by the AI/ML Orchestrator, or, when applicable by an external user of the framework (e.g., in the case the AI/ML framework is used in standalone mode for AI/ML services and models management) or based on policies configured within the AI/ML Orchestrator. Similarly, the AI-driven Management and Orchestration framework, as a consumer of the AI/ML framework services (e.g., the AI-Driven Management and Orchestration Framework) or based on policy information on the AI service. This flexibility allows for optimization of deployed AI/ML services and models based on a variety of metrics (e.g., availability of computing resources, performances of the models, proximity to data, etc.). The second innovation is the seamless interaction and integration with the Model catalogue and AI/ML Orchestrator to realize a comprehensive and automated AI/ML framework, and maintaining and exposing metadata of the deployed trainings and runtime ML model inference services. In this way, the associated models and metadata onboarded into the AI/ML Framework remain fully traceable for the entire AI/ML service and models lifecycle, therefore enabling the implementation of the ADROIT6G crowdsourcing concept.

The interaction of the AI/ML Pipeline Manager with the other blocks of the framework is presented in Figure 3.3. This includes the AI/ML Orchestrator Engine for determining model service orchestration decisions, the AI/ML Resource Manager for allocation of resources for Agents and functions located in the execution environments, and the Agents in execution environments for pipeline deployment. The AI/ML Pipeline Manager offers the following services through its northbound interface:

- the onboarding of training and execution pipelines;
- request of deploying or schedule a deployment of a pipeline;
- management of a pipeline (e.g., deletion, update);
- retrieval of information on a deployed pipeline;
- puling of pipeline deployment jobs by agents located in execution environments.

The AI/ML Pipeline Manager compiles an onboarded pipeline into a deployment object which is to run a model training or serving on a supported infrastructure. The available resources for pipeline deployment, dynamic in nature (e.g., computing resources availability), are accessed through interaction with the AI/ML Resource Manager. The request of deploying a pipeline is received from the AI/ML Orchestrator through an exposed interface for the triggering of the execution and management of pipelines. The triggering can pass run-time parameters to the deployed pipeline such as tuning parameters for the model and data locations. Information about a deployed pipeline is available through the Service Instance Catalogue, including the overall timestamped workflow, metrics on the process and training, and user-defined metadata defined in the onboarded pipeline.

AI/ML Catalogue

The ADROIT6G AI/ML Catalogue is the access point for the tracking of model training and published model versions. Metadata stored in the catalogue contains beyond state-of-the-art (BSoTA) information on models including training runs, data used, training results, model signatures, model specific execution environments, suitability for a specific AI/ML task, location of saved model versions, as well as version-controlled model publication.

© ADROIT6G, 2023

The AI/ML Catalogue leverages on the state-of-the-art solution MLFlow (https://mlflow.org/), designed for managing ML experiments, expanding the functionality to track complete ML solutions throughout the service lifecycle, expanding on definitions held in model metadata.

The required model metadata for the management of network level AI/ML service lifecycles is an active topic in the Standard Development Organization community, including recent developments in 3GPP [6] and [7]. These works set out to define several aspects related to sharing and selecting ML models including the information necessary to be able to select an ML model based on a task, model interoperability information (e.g., file format, execution environment), performance metrics for model assessment, model signature for inference, validity period for time sensitive model information, etc. This work is leveraged in ADROIT6G to create a dictionary for metadata information as well as the structure in storage, to create a complete enhanced model management entity.

The exposed interfaces of this block allow:

- onboarding of models and metadata;
- retrieval of models. metadata, training and version history;
- registry of models for deployment (version control);
- deletion or update of stored models.

In addition, the AI/ML Model Catalogue includes a web-based user interface (UI) for browsing registered and shared models, with comparison of trainings, execution environment requirements, data information and location of model files.

The four other blocks which form the AI/ML Framework implement SoTA solutions to provide the needed services and functions of the AI/ML Orchestrator:

- **AI/ML Resource Manager**
  The AI/ML Resource Manager is responsible for assigning and reserving computational and communication resources necessary for ML algorithm execution. It implements resource allocation decisions set forth by the AI/ML Orchestrator.
- **Data Registry**
  Within ADROIT6G, a cutting-edge data registry module is being be explored, responsible for managing metadata, schemas, and data definitions across the network infrastructure. This module is intricately involved in organizing, validating, and standardizing data structures and semantics within the network environment. Moreover, it plays a crucial role in guaranteeing data consistency, quality, and interoperability within network applications, thereby fostering streamlined data management, governance, and innovation across the network ecosystem.
- **Data Repository**
  ADROIT6G aims to integrate a SoTA data repository module, designed to securely store and efficiently organize data within a structured environment. This module not only ensures robust data security but also enables seamless data access, retrieval, and sharing processes.
- **Model Storage**
  The ADROIT6G Model Storage consists of a database used for storing the ML models that have been onboarded in the system, either through training pipelines or directly as already trained, or yet to be trained models. The associated model metadata are stored in the AI/ML Model Catalogue described above. The AI/ML model storage uses SoTA object storage tools including minIO (https://min.io/).

3.2 Interaction with the AI-driven Management & Orchestration and Control frameworks

The CrowdSourcing AI system introduces a comprehensive AI/ML framework designed to support both the AI-driven Management and Orchestration and Control frameworks in their decision-making processes, as illustrated

in Figure 3.1. This interaction operates bidirectionally, with the AI/ML framework facilitating energy-efficient provisioning of AI agents, initial training of AI models, and their discovery and selection, leveraging meta-learning principles. Conversely, the AI-driven Management and Orchestration and Control frameworks actively participate in the continual validation and improvement of AI models, fostering a collaborative and secure learning environment conducive to enhancing the performance of AI models. This reciprocal exchange of information and resources between the AI/ML framework and the AI-driven Management and Orchestration and Control frameworks establishes a symbiotic relationship aimed at achieving optimal decision-making and operational efficiency within the CrowdSourcing AI system.

As detailed in D4.1 ("AI-driven Management & Orchestration and Control Frameworks"), the AI-driven Management and Orchestration and Control frameworks incorporate closed-loop control functions into network services for their dynamic management. These closed-loops are AI-driven and require the complete management of AI/ML models during their lifecycle, including model selection and definition, training, evaluation, provisioning, monitoring, and modification. To do this, the AI-driven Management and Orchestration requests model services from the AI/ML framework of the Crowdsourcing AI. A workflow highlighting an interaction between the AI-driven Management and Orchestration and the AI/ML Framework for the end-to-end (e2e) service provisioning with AI-based zero-touch closed is shown in Figure 3.4.

When a request is received by the AI-driven Management and Orchestration that contains a service with an AI-equipped zero-touch closed loop, the first step is to establish which model should be attached to the closed-loop for optimal performance based on service requirements. A request is made to the AI/ML Orchestrator which calls the AI/ML Orchestrator Engine via the controller for appropriate model selection. This process uses the supplied model parameters for selection and could include optimization of an existing model in the AI/ML Model Catalogue in terms of pruning or hyperparameter adjustments. In this case, the AI/ML Orchestrator Engine requests a training to the AI/ML Pipeline Manager with the appropriate model, pruning factor, hyperparameters, data, and target execution environment. The AI/ML Pipeline Manager then configures and deploys a training pipeline to an agent deployed in the execution environment which creates an AI/ML training function to run the training. After the training, the AI/ML training function registers and pushes the model to the AI/ML Catalogue and sends a response to the AI/ML Pipeline Manager of the completed training. The result is then passed back to the AI/ML Orchestration Engine who relays the selected model back to the AI-driven Management and Orchestration via the AI/ML Orchestrator controller.

If the requirements of the closed-loop include a model to be served for inference requests, the AI-driven Management and Orchestration requests the provisioning of an AI/ML service to the AI/ML Orchestrator for runtime model inference. This request is then passed to the AI/ML Pipeline Manager who provisions and configures a model serving deployment pipeline as well as any monitoring function pipelines requested for monitoring the state and quality of the model's performance. The AI/ML Pipeline Manager then deploys these pipelines to an Agent deployed in the execution environment which creates AI/ML runtimes function to run the model and monitoring with exposure provided via API. The result including the metadata and endpoints of the model service is then returned to the AI-driven Management and Orchestration.

The AI-driven Management and Orchestration then configures the closed-loop with the model endpoint and completes the instantiation of the closed-loop. Other operations not represented in this workflow that involve interactions between the AI-driven Management and the AI/ML framework include model training, validation, migration of ML model services, as well as data management.

© ADROIT6G, 2023

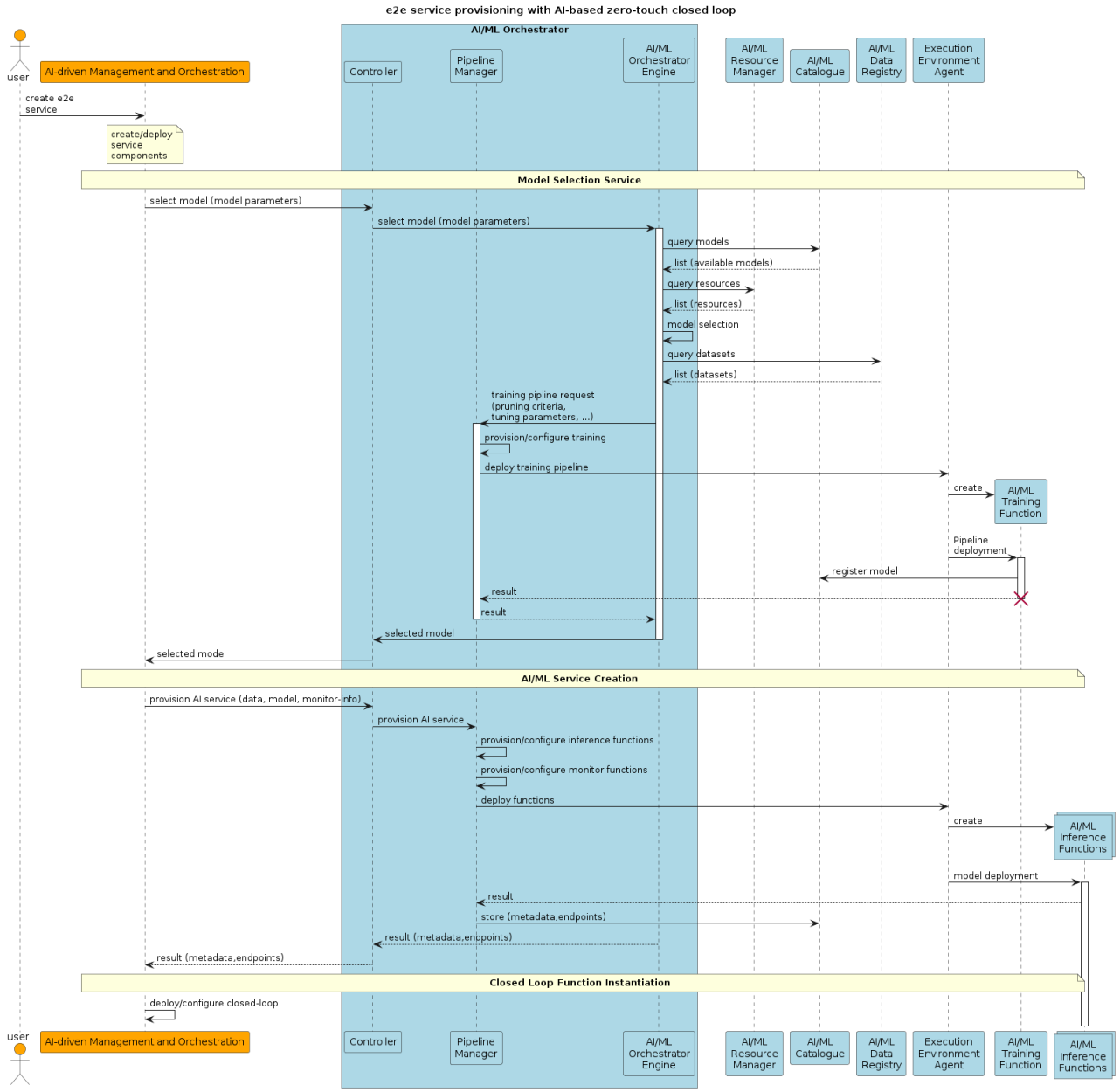


Figure 3.4 Workflow for the end-to-end (e2e) service provisioning with AI-based zero-touch closed loops

3.3 Preliminary Implementation of the AI/ML Framework

A preliminary implementation of several of the functionalities of the AI/ML framework, mostly pertaining to pipeline management and ML models tracking through the ML model catalogue has been done and tested using ML model training, retraining, and serving pipelines (Figure 3.5). The goal of this first implementation is to validate the AI/ML Pipeline Manager functionalities and related innovation in providing ML services across multiple cloud-native domains (based on Kubernetes execution environments) to execute a training pipeline, a re-training pipeline, and a model serving deployment in an automated way. The first version of the AI/ML Pipeline Manager has been realized through the Prefect tool, which has been used to handle the deployment of python-based pipelines spanning several domains. Prefect is a descriptor-based workflow orchestrator, allowing the definition of required technologies for a pipeline, runtime input parameters to the pipeline, and dynamic location

of a training algorithm and data. For this first implementation, the descriptors for the pipelines used were loaded locally into Prefect, with the incorporation of an API-driven approach to be adopted in the future. Prefect uses workpools to assign pipeline workflows to execution environments (e.g., domains). Workpools are associated to a specific technology (ex. Kubernetes, docker) and are polled by one or more agents which are located in the execution environment as depicted in Figure 3.3. For the organization of model trainings, metadata, and metrics, MLFlow was implemented as part of the Model Catalogue. MLFlow was further integrated with MinIO to store the models and data.

The configuration of the implementation involved one virtual machine (VM) hosting initial components of the AI/ML Pipeline, AI/ML Model Storage, and AI/ML Catalogue, and two separate Kubernetes execution environments (K3S and minikube) running on two VMs, depicted in Figure 3.5.

In this scheme, the AI/ML Pipeline Manager exposes an interface for the manual onboarding of training and deployment descriptors, which contain the information on the pipeline, input data, model, infrastructure, and location for deployments. The AI/ML Pipeline Manager assigns the deployment to a workpool which is polled by agents located at the 2 execution environments.

The service executions and deployments used on this framework implementation involved three pipelines for the training, retraining, and deployment of an ML Model. In the first two pipeline deployments the focus is on training an exemplary ML model. The model used was a linear regression with 11 input parameters describing the network state, 1 quality of experience output value, and 2 adjustable hyper parameters (alpha, and l1 ratio). The first pipeline contains a model definition, an initial dataset, and an algorithm for training. The training process is recorded by the ML Catalogue via a python library, with permissions to store the information provided securely by the Pipeline Manager at the time of pipeline deployment.

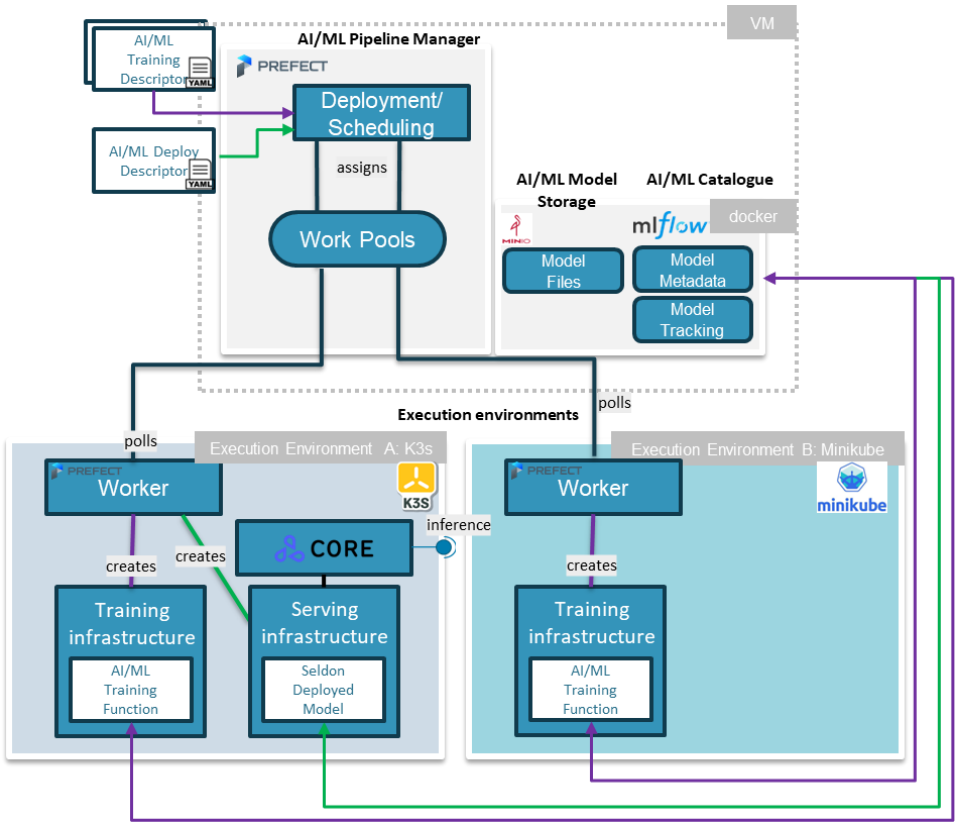


Figure 3.5 Initial implementation of the AI/ML framework

The sequence of deployments, shown as purple arrows in Figure 3.5, was as follows:

- A service descriptor for a model definition and training is manually onboarded into the AI/ML Pipeline Manager;
- A request for the deployment of the onboarded service is made to the AI/ML Pipeline Manager, defining model tuning parameters via a web-based UI;
- The AI/ML Pipeline Manager assigns the deployment to a workpool, and an agent in the K3S execution environment pulls the assignment and creates a pod to execute the training pipeline;
- The trained model and metadata (performance metrics, tuning parameters, and required environment for model serving) are sent to the AI/ML Catalogue which stores the model in the AI/ML Model Storage;
- A second service descriptor is onboarded for a retraining of a model in the AI/ML Catalogue with new data, and a request is made for a deployment through the web-based UI of the AI/ML Pipeline Manager;
- The same deployment process as before is done, but this time using the minikube execution environment to execute the pipeline;
- The target model is pulled from the AI/ML Catalogue, trained and evaluated on a new set of data defined through the pipeline's runtime parameters set on the UI of the AI/ML Pipeline Manager;
- The newly trained model and metadata are uploaded to the AI/ML Catalogue as a new version of the original model that was pulled for retraining;
- Comparison of the performance of the newly trained model and the previous version are shown on a web-based UI of the AI/ML Catalogue.

At this point, the focus is shifted to providing a model serving instance for the model that was trained in the steps above. A model serving instance exposes an API that can be used to receive an inference from a model stored in the Model Catalogue. This is accomplished with the use of Seldon core, which converts ML models (Tensorflow, Pytorch, H2o, etc.) into production REST/GRPC microservices. The model serving pipeline contains all the information that is required to load the model into Seldon, including the required conda environment which is provided in the Model Catalogue. The sequence of deployments, shown as a green arrow in Figure 3.5, was as follows:

- A service descriptor is onboarded in the AI/ML Pipeline Manager for a model serving instance, and a request is made for a deployment, defining the desired model through the web-based UI of the AI/ML Pipeline Manager
- The AI/ML Pipeline Manager assigns the deployment to a workpool, and an agent in the K3S execution environment pulls the assignment and creates a pod to execute the model serving pipeline
- The pipeline pulls the defined model and metadata necessary for running the model from the AI/ML Catalogue and creates a new pod under an instance of Seldon core that serves the model for inference requests via REST API
- Inference from the model is requested with new data via the external API Swagger (Figure 3.6), the request contains values for the 11 network input parameters of the model (Requests per second, Data IO, Average response time rate, Peak response time, Hardware utilization, Thread count, Disk usage, Load, Network bandwidth, Uptime, HTTP server error rate) and a result is returned.

The external API exposed by Seldon core also allows for feedback on a model's inference (e.g., a correct labeling). Future developments in the implementation of the framework for Crowdsourcing AI will expand to include more functionalities of the framework, including those of the AI/ML Orchestrator for interaction with the AI-driven Management and Orchestration Framework for automated deployment, and supporting a selection of the crowdsourcing intelligence solutions described in Section 4 for cooperative and collaborative training of models, transfer and federated learning, as well as Model Selection for use in BDIx agent and UE-VBS deployments at the far edge.

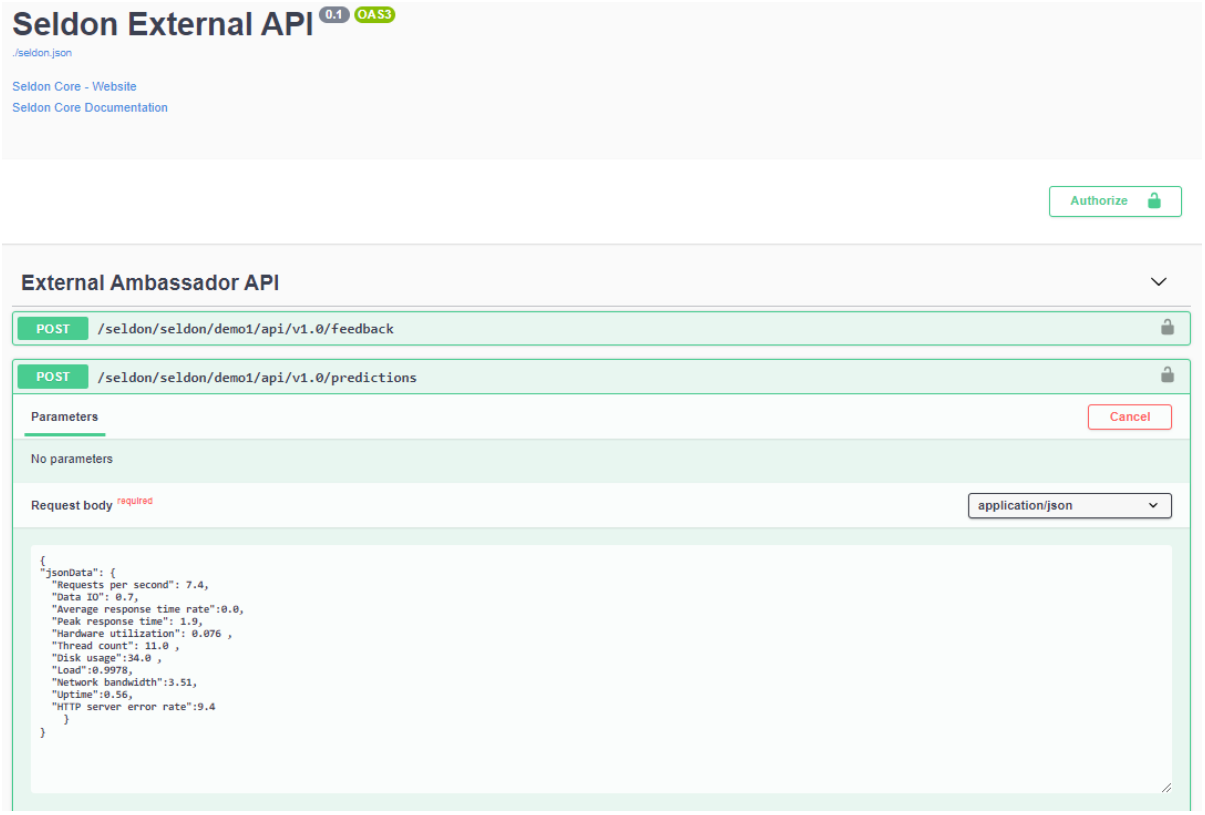


Figure 3.6 Swagger UI for a model serving instance

4 CrowdSourcing Intelligence: The AI/ML Orchestrator

This section focuses on the AI/ML Orchestrator Engine -- an essential component of the AI/ML Orchestrator and of the entire CrowdSourcing AI framework introduced above. The AI/ML Orchestrator Engine executes algorithmic solutions, whose output is then used for effectively and efficiently deploying and managing the AI/ML operational pipeline. As exemplified in Figure 4.1 for a supervised ML approach, such a pipeline takes as input data (streams of batches), extracts the features from each data sample, uses such features to either train a model, or as input to the inference task if the model has been already trained. Among the different stages, the main two on which we focused for the development of the CrowdSourcing AI concept are: ML model training and inference.

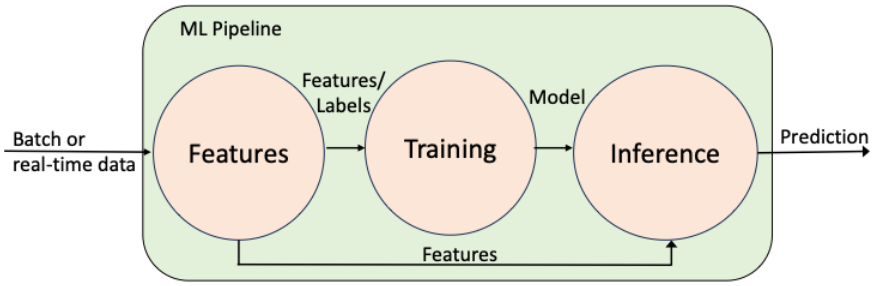


Figure 4.1: Operational pipeline in the case of supervised ML.

Specifically, the algorithms included in this section aim to implement the intelligence needed in the AI/ML Orchestrator Engine for collaborative training and inference, so that the AI/ML pipeline (i) is sustainable, i.e., efficiently utilizes the resources offered by the existing network nodes and minimizes energy consumption, and (ii) ensures that training and inference performance requirements are satisfied.

The rest of the section is organized as follows:

- Section 4.1, Section 4.2, and Section 4.3 focus on *distributed learning mechanisms* that contribute to realizing the concept of CrowdSourcing AI. Specifically, by leveraging a cooperative approach, they provide ML models that are properly trained, while making the use of ML sustainable and meeting the resource limitations of the system nodes in terms of communication, computing, and data resources. The presented methods differ in the distributed learning approach that they leverage on, including *sequential learning,* –(Section 4.1), *transfer learning* (Section 4.2), and *federated learning* (Section 4.3);
- Section 4.4 presents a *meta-learning* approach for *ML model selection and reuse*, in a distributed network scenario;
- Section 4.5 specifically addresses the *energy efficiency of federated learning*, proposing an implementation thereof that minimizes communication resource consumption while guaranteeing a satisfactory model convergence time;
- Section 4.6 introduces a *decentralized learning* scheme, which is particularly useful when the system cannot elect, or opts for not electing, a central coordinator;
- Section 4.7 and Section 4.8 present solutions that aim at performing *inference tasks* efficiently by exploiting the cooperation between edge servers and end devices.

4.1 Enabling Robust Cooperative Training

ML models, and Deep Neural Networks (DNNs) in particular, have become essential components of several services and a great deal of attention has been devoted to making them more and more capable. This however has implied an increasing complexity and training cost. This issue has been tackled through two complementary approaches: distributed training and model compression. The former exploits the resources (e.g., computational) and data available at different nodes, thus better distributing the training burden. The

latter includes a wealth of different techniques (e.g., model pruning and knowledge distillation) allowing for the transfer of knowledge from a model to a (usually, simpler) one.

4.1.1 ADROIT6G specific enhancements over state of the art

The existing work on the distributed training and compression of ML models [8],[9],[10],[11] has consistently overlooked the distribution of the achieved learning quality, focusing instead on its average value. This has led to a poor dependability of the resulting ML models, whose performance may be much worse than expected. Within the project, we have filled this gap by envisioning DepL (for dependable learning), a framework to be implemented at the AI/ML Orchestrator for dependable learning, able to make high-quality, efficient decisions on:

- the data to leverage for learning,
- the models to use and when to switch among them, and,
- the clusters of nodes, and the resources thereof, to exploit.

For concreteness, we consider as possible available models a full DNN and its compressed versions. Note, however, that although we focus on model compression as the source of alternative models, the problem we pose and the solution we envision extend to arbitrary models.

The main enhancements of DepL to the state of the art are thus as follows:

- Through a comprehensive system model, capturing all relevant aspects of distributed ML training, it selects the datasets to be used, the model versions to train, and the system nodes that should contribute to the training process, so that the target learning quality distribution is achieved at the minimum cost;
- It exhibits polynomial complexity;
- As highlighted by preliminary results, it closely matches the optimal (prohibitively complex) optimal solution and consistently outperforms by over 27% compared to the considered benchmark.

4.1.2 Functional architecture

We consider a distributed learning scenario where nodes in different parts of the network (e.g., far edge, and edge) cooperatively train a DNN model aided by a learning orchestrator. In this setting, we develop our DNN training scheme called DepL, allowing the AI/ML Orchestrator to make joint decisions about: (a) the datasets to exploit for learning; (b) the models to train for the ML task at hand; (c) the node clusters to use at different stages of training. Unlike previous work, DepL accounts for the mutual interaction between such decisions, and the effect that each of them has on the expected learning quality as well as its distribution. The AI/ML Orchestrator runs DepL within the AI/ML Orchestrator Engine, performing the following steps as per the entities' interaction illustrated in Figure 3.3:

1. Receive from the AI/ML Resource Manager the information on the available nodes and their capability;
2. Access the available datasets from the Data Repository and their meta data in the Data Registry;
3. Access the available versions of an ML model suitable for the task at hand, which are stored in the AI/ML model storage, and their meta data in the AI/ML Catalogue;
4. Return to the AI/ML Pipeline Manager the generated selection of dataset, ML models, and nodes.

The AI/ML Pipeline Manager will then implement the AI/ML pipeline as per the AI/ML Orchestrator's instructions by instructing the AI/ML Resource Manager, which, in its turn, interacts with the selected nodes and provides them with the indication of the selected datasets and model versions.

To ensure dependable training, DepL seeks to keep a target quantile ω of the loss distribution below a maximum threshold value that is deemed to be acceptable for the application at hand. Second, the complexity of the decisions to make, which, along with the sheer amount of existing options to explore,

makes finding the optimal solution impractical for realistic-sized problem instances. To address the first issue, DepL makes three main decisions:

1. selecting the datasets to use for learning;
2. choosing the model to use at each epoch, switching models as needed;
3. identifying the cluster of learning nodes to call for at each epoch, and the resources they should commit.

Then, to tackle the second issue, DepL efficiently makes orchestration decisions by decoupling them, while iterating as needed so as to account for their mutual influence. To this end, it is critical to establish the order in which decisions are made. DepL properly accounts for the decisions' mutual influence by making decisions that strongly influence each other sequentially. We observe that the mutual influence is strongest between (i) dataset and model selection, since different models may call for different data, and (ii) model selection and resource allocation, as insufficient resources may cripple the model performance. Additionally, DepL seeks to make first decisions with the deepest impact on the overall training performance. To do so, we remark that:

- Dataset selection is related to the model selection and affects the resources needed at the nodes, as more data invariably means more processing at the nodes;
- Model selection is less critical, yet very relevant as it impacts both the learning progress (hence, how many epochs we need) and the required computational resources (hence, how long each epoch will take);
- The impact of the resources and nodes to use is limited to the epoch duration and cost.

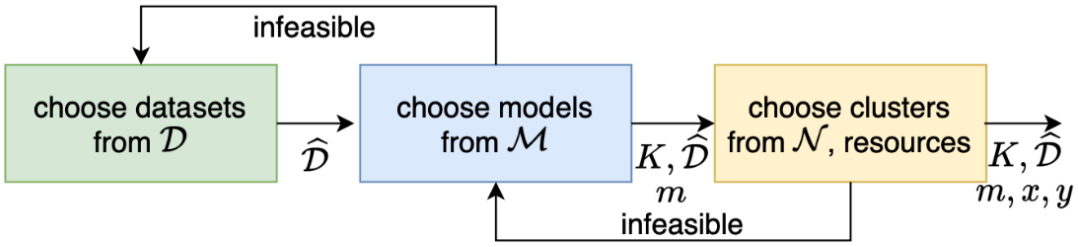


Figure 4.2: The main steps of the DepL solution strategy.

Based on the above, DepL takes the approach depicted in Figure 4.2, making dataset selection decisions in the outermost loop, then model selection, and finally resource allocation. Importantly, a further reason to make decisions in the order highlighted in the figure has to do with how difficult it is to estimate their consequences. Indeed, choosing the data sources to use for learning requires sophisticated statistics and feedback loops. The impact of model selection on the overall learning is somehow better understood, though with significant uncertainty, and finding the right resources to run a given model can be mapped to many well-studied problems. By making harder-to-estimate decisions less often, we can reduce the impact of estimation errors on the overall solution quality. Furthermore, in view of the above observations, we make each decision by considering the least restrictive option for the subsequent ones. As an example, while choosing the data, we assume that the most complex (i.e., full) version of the model is chosen in the subsequent step, and then that the cluster with the largest amount of available resources is used and they are fully allocated to the model training. Indeed, since adding more data is the most consequence- fraught decision to make, we want to avoid doing that, unless it is utterly necessary; similarly, a simpler model is used only if the available computational and networking resources are insufficient to make a more complex model work.

Dataset selection. Adding new datasets to improve learning quality may have undesirable consequences: (i) adding a new dataset implies that more training should be performed (hence, more resources are needed at the nodes using the dataset); (ii) more datasets typically imply more network resources to consume for

their collection and, hence, a larger network. Accordingly, DepL begins with no selected datasets and adds them as needed. Importantly, at this stage, it takes the learning time as proxy of the overall training cost, as it does not directly tackle resource allocation. We remark that adding a new dataset has two contrasting effects on the training epochs. On the one hand, more learning is performed at each epoch – hence, fewer epochs are needed to reach the target learning quality. On the other hand, more data is collected and processed at each epoch – hence, each epoch lasts longer. Our key observation is that these two effects do not have the same evolution: (i) Adding more data decreases the number of necessary epochs according to a logarithm law [12]; (ii) Processing and time to transmit model updates, i.e., an epoch duration, grow instead linearly with the quantity of data. It follows that, intuitively, if adding a new dataset does not help (specifically, it does not make convergence faster), adding another one will also not help. More formally, the total training time is submodular. Since DepL takes the learning time as proxy of the cost, it just needs to minimize the training time as a function of the selected datasets, which is an unbounded submodular minimization problem [13]. Such problems can be solved to optimality in polynomial time via several different algorithms [14] that achieve strongly polynomial runtime, only depending upon the problem size. This means that DepL solves to optimality the first step of the solution strategy, although not the NP- hard problem formulation that jointly and optimally selects datasets, models, and nodes.

Model selection. Given the datasets identified at the previous stage, model selection at each epoch is still a combinatorial problem. To address such complexity, we build an expanded graph where:

- vertices represent the possible states of the system; each vertex is associated with a 4-uple composed of the model under use in the state represented by the vertex, the epoch index, the elapsed learning time; the ω-quantile of the current loss;
- edges represent the possible decisions and their effects; the weight of each edge represents the cost associated with a model switch. Edges are also created between vertices corresponding to the same model, i.e., indicating no model switch;
- the best decision is given by the path (i.e., a sequence of decisions) that connects the current system state with a feasible one and has the lowest weight (i.e., cost). Note that such decisions may not be optimal solutions to the original problem, as not all states are represented by vertices in the graph. However, by increasing the resolution with which we represent the system states, we can reduce this issue and get arbitrarily close to the optimum, at the cost if increasing the size of the graph and the solution complexity.

We underline that at this stage, DepL assumes that the most capable node cluster, and all the resources therein, can be used. Furthermore, if no model can be selected that meets the target learning time and quality, DepL goes back to the previous stage: the current dataset choice is blacklisted and the datasets selection is repeated, still considering the most complex model. Optimized model selection will then be attempted again under the new data selection.

Node selection. Once the datasets and models are chosen, the only remaining decisions to make concern: (i) the clusters to use for learning, and (ii) which resources each cluster has to commit. Importantly, neither decision affects how many epochs are needed for convergence, nor the distribution of the resulting loss value; on the other hand, clusters and resource selection have a very strong influence on the learning time (i.e., the duration of each epoch) and the resulting cost. We make the key observation that the resulting problem – including its decisions and their effects – can be mapped, one-to-one, to the classic problem of Virtual Network Function (VNF) placement, also referred to as service embedding. Specifically: (i) clusters are mapped to servers, which can run VNFs; (ii) models are mapped to the VNF themselves; (iii) model complexity is mapped to VNF requirements. As a result, we can use any VNF placement strategy to solve this stage of our problem. VNF placement is a very well-studied problem, with several efficient and effective schemes existing for it. In particular, we take [15] as a reference and obtain the same $O(1/\varepsilon)$ complexity.

Again, in the case where no cluster or resource allocation can be found that can make the model be trained within the target time, DepL goes back to the previous decision stage, i.e., model switching, again blacklisting

the model choices that have proven insufficient and considering that any cluster and resource can be used. The overall process depicted in Figure 4.2 may therefore be iterated till a feasible solution is found, or the existing options have been exhausted.

4.1.3 Preliminary results

Since the dataset selection is provably optimal, we focus on model and node selection and benchmark DepL against two alternatives:

- Optimal decisions, obtained by brute force (labelled as optimal in plots);
- A state-of-the-art solution, aiming to optimize the expected loss, inspired to [10] (labelled as best-exp in plots). The best-exp scheme makes decisions of the same nature as our DepL; however, it accounts for the expected loss instead of any quantile thereof. Thus, comparing against best-exp is a valuable way to gauge the impact of accounting for the distribution of the loss besides its expected value.

As possible model versions, we consider either three versions of the AlexNet DNN pruned, respectively, with factors F1=0.5, F2=0.75, and F3=0.9, or three versions of the MobileNet network, pruned with the same factors. The network infrastructure includes two types of servers (hence, clusters nodes): A-class nodes, with capabilities set to 100 and processing cost 1, and B-class nodes, with capability 50 and processing cost 1.2. For simplicity, we assume that clusters contain one node only.

Figure 4.3 shows the cost incurred for different target values of the loss 99th percentile. Clearly, the tighter the loss requirements, the more the necessary training, hence, the higher cost. More interestingly, DepL outperforms best-exp and virtually matches the optimum. The difference is especially large when the requirements are *less* strict, hence, there is more margin for learning strategies that optimize cost over sheer performance.

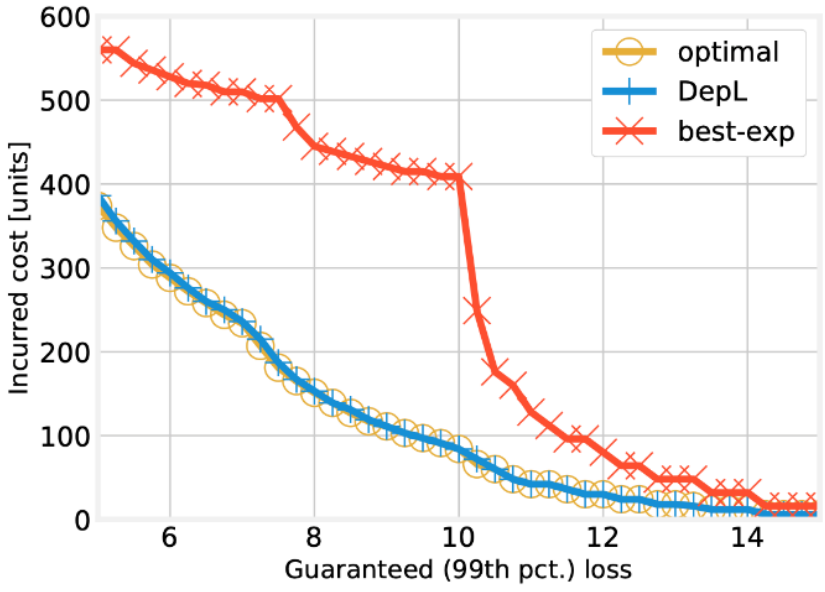


Figure 4.3: Training cost vs guaranteed loss. DepL performance is compared to the optimum and the best-exp strategy.

4.2 Cooperative Transfer Learning

Cooperative Transfer Learning is an advanced ML strategy that facilitates knowledge sharing among various models or systems to boost learning efficiency and performance, especially beneficial in contexts with limited data, computational resources, or diverse environments. This approach differs from traditional transfer learning

© ADROIT6G, 2023

by involving multiple entities in a collaborative learning effort [16]. A key aspect of Cooperative Transfer Learning is the role of a centralized authority in examining and optimizing hyperparameters across all entities involved, to ensure peak efficiency and effectiveness. This centralized authority plays a critical role in fine-tuning hyperparameters and managing computational loads across all entities involved in the collaborative learning, ensuring an optimized and balanced learning ecosystem. Below, a brief explanation of how Cooperative Transfer Learning operates, is provided:

- Entities like models, systems, or network base stations are grouped by their similarities in data, tasks, or objectives. This crucial step helps pinpoint which entities could gain most from mutual knowledge exchange, enhancing their learning processes;
- An initial model is trained with aggregated data from a selected group or entity, forming a knowledge base that encapsulates key patterns or features relevant to the collective tasks. This phase is overseen by a centralized authority that optimizes the model by selecting suitable architectures, algorithms, and hyperparameters;
- Once the base model is trained, its parameters are shared across different entities or clusters, setting the stage for further learning. Entities adapt these parameters to their specific data, applying transfer learning principles to efficiently address new but related challenges with reduced data and computational needs;
- Then, entities customize the shared model with their data, tweaking its parameters to reflect their unique data traits. The centralized authority aids in making this process effective by offering optimization strategies and guidelines.

This learning approach offers numerous advantages, such as enhanced learning efficiency, lower computational and data demands, and improved model performance in varied settings. It is particularly useful where data is geographically dispersed, making data centralization impractical due to privacy, security, or logistical constraints. Through the strategic collaboration and efficient learning process distribution, Cooperative Transfer Learning paves the way for more effective and scalable ML solutions, enabling the creation of robust and adaptive models, adept at tackling complex real-world problems.

The Cooperative Transfer Learning approach, as detailed in the context of its application to cellular traffic modelling and beyond, brings several specific enhancements over traditional state-of-the-art ML and transfer learning methodologies. These enhancements are particularly relevant in the context of distributed learning environments, such as those found in telecommunications networks, but also have broader implications across various domains where data is decentralized, and computational resources are at a premium.

4.2.1 ADROIT6G specific enhancements over state of the art

Next-generation intelligent mobile networks rely heavily on deep learning-enabled cellular traffic modelling and prediction. It can assist in proactive resource management operations in large-scale data-driven cellular networks. Conventional methods often utilise a widely recognised forecasting model for single-variable time series prediction and necessitate a substantial volume of training data. We introduce a dual-step transfer learning-based prediction model for predicting multivariate spatiotemporal cellular traffic. This approach categorises the base stations into several clusters according to the correlation of traffic. The base model is trained on the combined dataset of a single cluster using a hybrid of Recurrent Neural Network and Bidirectional Long-Short Term Memory (RNN-BLSTM). Parameter-based transfer learning involves providing the base model to other clusters as a starting point. The base model then undergoes the process of fine-tuning on an intra-cluster aggregated dataset. Subsequently, the intra-cluster prediction model is distributed to all base stations within the cluster, necessitating only minimal training and parametric modifications. We perform comprehensive tests on an actual cellular traffic dataset, and the findings demonstrate that the suggested dual-step transfer learning-based prediction model can attain high prediction accuracy while decreasing the demands for data, time, and computing [16] (as shown in Figure 4.4).

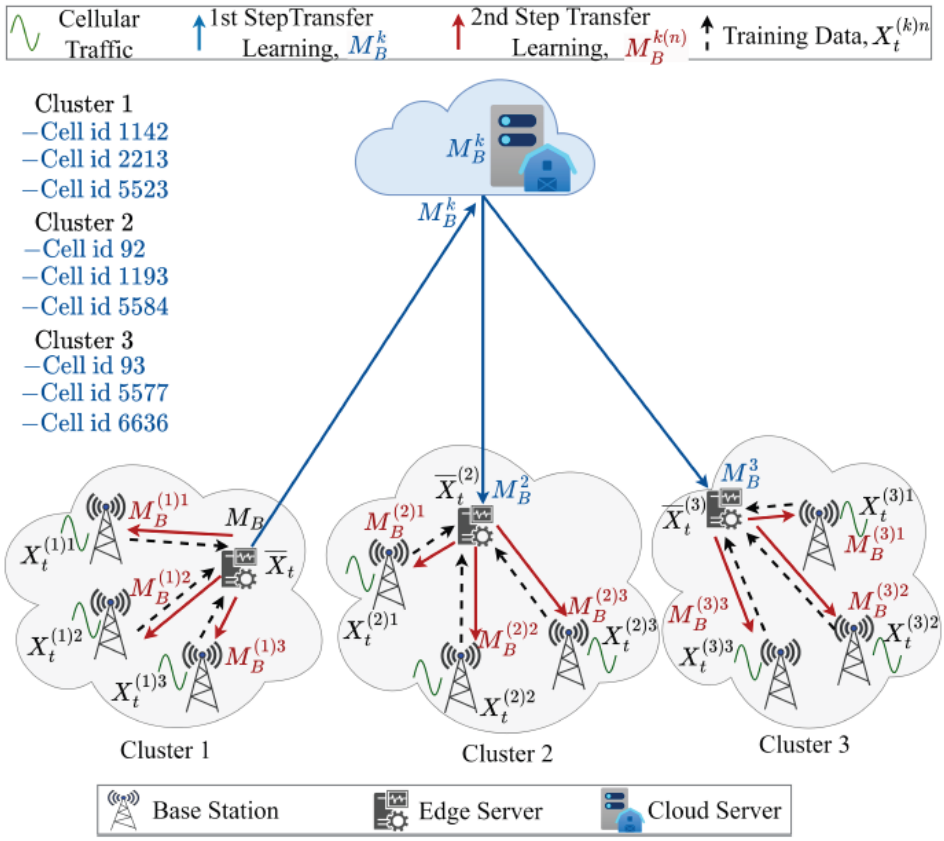


Figure 4.4: Illustration of the dual-step transfer learning (DSTL)-based prediction model in a cellular network with clusters based on the similarity in traffic distribution among gNodeBs

Here are the key enhancements that Cooperative Transfer Learning offers over the state of the art:

- Efficient Utilization of Distributed Data: Unlike traditional approaches that may require centralizing data for training, Cooperative Transfer Learning efficiently leverages data where it resides. By training models locally at the source of data (e.g., base stations or devices) and only sharing model parameters or insights, this approach respects privacy and reduces the need for data transmission, addressing both privacy concerns and bandwidth limitations.
- Reduced Computational and Data Requirements: By sharing a base model across similar entities and allowing for local fine-tuning, Cooperative Transfer Learning significantly reduces the need for each entity to train a complex model from scratch. This results in lower computational and data requirements, making advanced ML models accessible to entities with limited resources.
- Rapid Adaptation to New Environments: The framework enables models to be quickly adapted to new environments or scenarios by reusing pre-trained models. This is particularly beneficial in dynamic environments where data distributions change over time, as it allows for continual learning and adaptation without the need for extensive retraining.
- Improved Model Performance through Specialization: The fine-tuning phase allows each local model to specialize based on its unique data, leading to improved performance compared to a one-size-fits-all model. This specialization is crucial in applications like cellular traffic prediction, where traffic patterns can vary significantly from one location to another.
- Centralized Optimization of Hyperparameters: The centralized authority's role in examining and optimizing hyperparameters ensures that models are not only tailored to their specific environments but also optimized for performance and efficiency. This centralized oversight helps in identifying the best strategies for model training and deployment across diverse scenarios.

- Scalability and Flexibility: Cooperative Transfer Learning is inherently scalable, as it can accommodate new entities or data sources by simply incorporating them into the existing framework. This flexibility makes it suitable for growing networks and evolving datasets, where new nodes or data types can be integrated with minimal disruption.
- Enhanced Learning through Knowledge Sharing: The approach facilitates a form of collective intelligence by enabling entities to benefit from each other's learning. This knowledge sharing can lead to more robust models that are better equipped to handle a wide range of scenarios, thanks to the diverse data and experiences they encapsulate.

The suggested approach, which combines Cooperative Transfer Learning with a Federated Learning framework, represents a substantial advancement in achieving the goal of implementing CrowdSourcing AI. This approach leverages the combined knowledge and computing power of multiple devices or nodes, such as mobile phones, IoT devices, or base stations, to collectively improve AI models. An essential benefit of this strategy is in its influence on hyperparameter optimisation, which is typically a limiting factor in conventional methods due to its time and computational complexity. Specifically, by dividing the learning process over numerous nodes, each with its own dataset, the system may simultaneously explore different hyperparameter settings, significantly reducing the time required to identify the best configurations. Simultaneous investigation is enhanced by the variety of data sources and localised computation, enabling a wider and more effective coverage of the search space without overwhelming any individual computer resource. Integrating power reserve attributes into this system tackles an additional crucial obstacle that is effectively controlling computational workload to avoid excessive power consumption on participating devices, particularly relevant for edge devices or mobile phones. The Cooperative Transfer Learning approach employs techniques such as adaptive learning, in which nodes modify their involvement based on their power state, and efficient communication, reducing the energy consumption during data transmission. By implementing selective participation, devices with greater power reserves are assigned more demanding computational tasks, thus preventing the depletion of resources on any individual device.

The proposed Cooperative Transfer Learning and Federated Learning framework seamlessly incorporates these strategies to expedite the process of hyperparameter optimisation. Additionally, it guarantees the sustainability of device participation in the learning network without compromising their functionality. This method facilitates the development of CrowdSourcing AI that is not only more effective and practical, but also capable of generating AI models that are precise, resilient, and safeguard privacy. This strategy represents a significant advancement in the collaborative creation of AI systems by leveraging a large and diverse dataset while ensuring the privacy and constraints of contributors.

4.2.2 Functional architecture

With reference to the AI/ML architecture shown in Figure 3.3, the Cooperative Transfer Learning module will be included in the AI/ML Orchestrator. More specifically it will be included, as a method module of evaluating and improving the AI/ML models, in the AI/ML Model Selection module, and used for AI/ML optimization. The AI/ML Orchestrator serves as the central hub for managing the lifecycle of AI/ML models, including deployment, monitoring, and updating models across distributed nodes. It orchestrates the overall process of model training, hyperparameter optimization, and model deployment in a federated learning environment. However, integrating the proposed Cooperative Transfer Learning within a Federated Learning framework necessitates a clear understanding of the input/output dynamics and the interaction with other modules that are under the AI/ML Orchestrator architectural component (i.e., Localization Module). This integration aims to streamline the process of hyperparameter optimization across distributed nodes, leveraging localized data while ensuring optimal computational efficiency and resource utilization. Below we describe how Cooperative Transfer Learning module operates within the AI/ML framework, along with its specified inputs and outputs.

The Cooperative Transfer Learning module will interact with the Localization Module (Figure 4.5) that is located the AI/ML Model Storage component of the AI/ML Framework. The Localization Module is responsible for identifying the geographical distribution of nodes (e.g., mobile devices, edge devices) and their data

characteristics. It provides crucial information for clustering nodes based on data similarity or geographical proximity, which is essential for efficient Cooperative Transfer Learning. Basically, it holds the trees structures of the D2D communications networks of each gNB under the ADROIT6G.

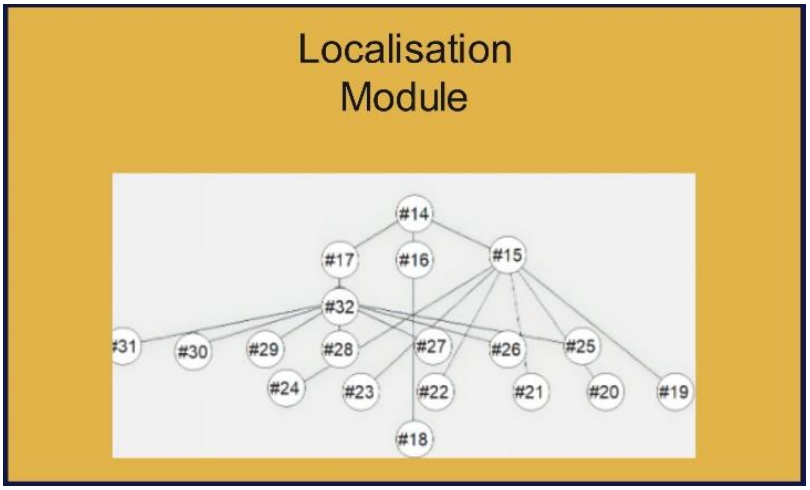


Figure 4.5: Localisation Module

Inputs to the Cooperative Transfer Learning module:

- Model Type: Information about the specific type of model that requires hyperparameter optimization. This helps in determining the kind of data and computational capabilities needed for the task. Examples of the hyperparameter optimization are the following:
  - **Grid Search** (Optimizing an SVM by evaluating combinations)
  - **Random Search** (Tuning a random forest model by selecting random values)
  - **Bayesian Optimization** (Finding optimal **hyperparameters** for a gradient boosting model)
  - **Gradient-Based Optimization** (Using gradient **methods** to tune neural network hyperparameters like learning rate)
  - **Evolutionary Algorithms** (Applying genetic algorithms to choose hyperparameters for deep learning models)
  - **Simulated Annealing** (Sequentially exploring hyperparameters of a decision tree by cooling down choices based on a temperature parameter)
- Model Location: Details regarding where the model is currently deployed or where the initial model training should occur. This can include specific nodes or clusters that have been identified as optimal starting points based on data availability and relevance.
- Parameters for Hyperparameter Optimization: A set of parameters that define the scope and constraints of the hyperparameter optimization process, including the range of hyperparameters to explore, the criteria for success (e.g., accuracy, efficiency), and any computational limitations of the nodes.

Outputs from the Cooperative Transfer Learning module:

- Best Hyperparameters Identified: After the optimization process, the component returns the optimal set of hyperparameters for the given model and task. This output includes the specific values for each hyperparameter that resulted in the best performance during the testing phase.

Process Flow:

- Initialization: The AI/ML Orchestrator initiates the process by sending a request to the Localization module, specifying the model type, the desired location for model deployment or optimization, and the parameters for hyperparameter optimization.

© ADROIT6G, 2023

- Hyperparameter Optimization: Utilizing the inputs, the Localization module identifies the best hyperparameters by coordinating with distributed nodes. This involves parallel processing and evaluation of different hyperparameter settings across nodes, leveraging their localized data for training and testing.
- Feedback Loop: The results of the hyperparameter optimization process, including the identified set of best hyperparameters, are sent back to the AI/ML Orchestrator. The Orchestrator then uses this information to update the model configurations and redeploys the optimized model across the relevant nodes.
- Deployment and Monitoring: With the optimized model deployed, the AI/ML Orchestrator continues to monitor performance, potentially initiating further rounds of optimization as new data becomes available or as the model's performance varies over time.

This integration into the functional architecture ensures that the Cooperative Transfer Learning and Federated Learning framework can dynamically and efficiently adapt to the evolving data landscapes and computational environments of distributed nodes. It maximizes the potential for collaborative AI development, leveraging crowdsourced data and computational resources while maintaining privacy and efficiency.

4.2.3 Preliminary results

This part provides the preliminary results regarding the cooperative transfer learning using a dual-step strategy [16]. A point of significance in this work is using a dual-step transfer learning strategy. Specifically, the parameter-based TL technique is investigated to predict multivariate spatio-temporal cellular traffic.

First Step Transfer Learning: In the first step, the pretrained model of cluster 1 is transferred to the other clusters via the cloud server. When the pre-trained base model is implemented on the aggregated data of clusters 2 and 3 without any fine-tuning, although the losses are higher, it can be observed from Figure 4.6 that the TL-based prediction model is capable of predicting the target cluster traffic with minor adjustments, for example, the RMSE of internet traffic for clusters 2 and 3 is 5.04 and 4.75, while it is 2.1 for cluster 1. Figure 4.7 shows the training and the validation loss when the transferred model is trained on the aggregated data of clusters 2 and 3, respectively. After training, clusters 2 and 3 will update the model parameters at the edge server. Figure 4.8 shows the prediction losses for cluster 3 after minor adjustments. For example, the MAE, MAPE, MSE, and RMSE for the internet traffic in cluster 3 are improved by 219%, 244%, 562%, and 225%, respectively.

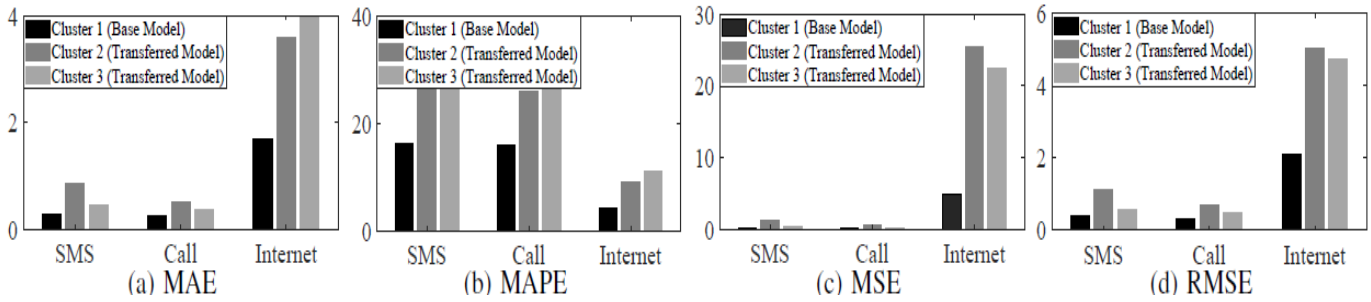


Figure 4.6: Performance metrics representation for sms, call, and internet traffic when the base model of cluster 1 is applied on the aggregated dataset of cluster 2 and 3.

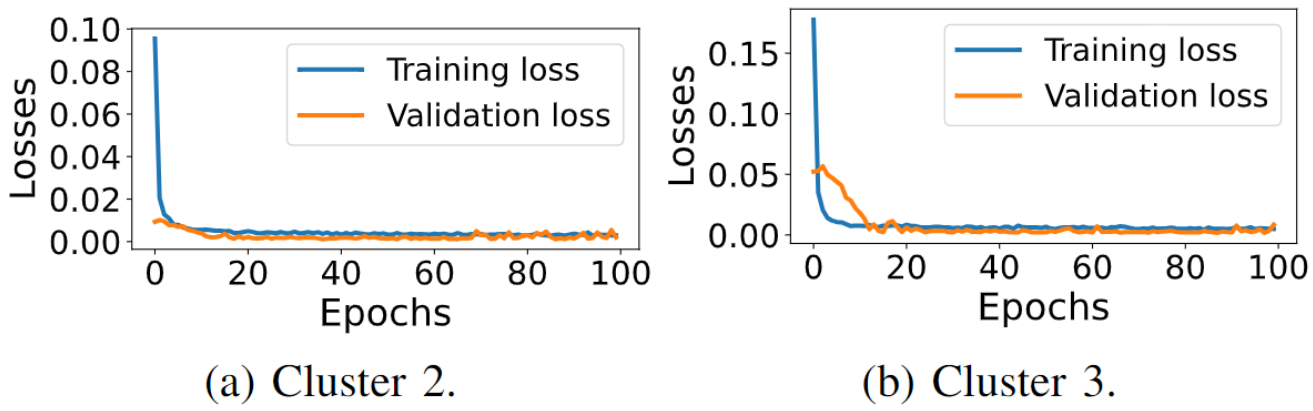


Figure 4.7: Training and the validation loss of the transferred model vs epochs.

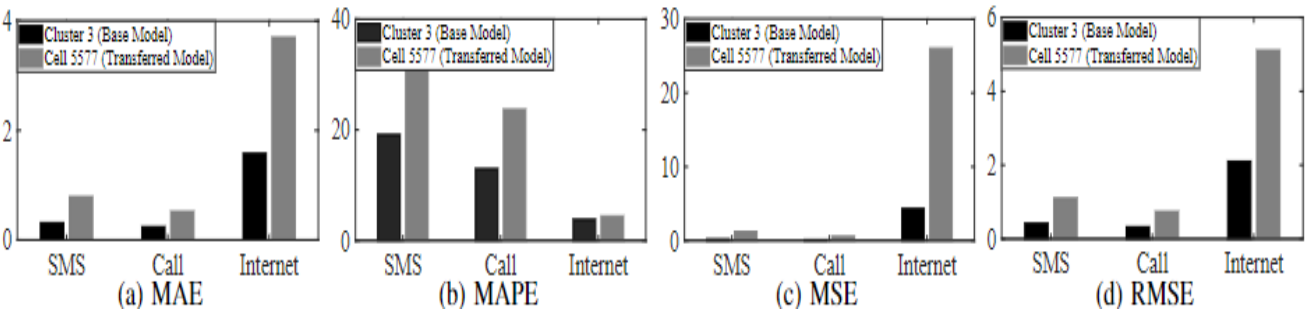


Figure 4.8: Performance metrics representation for sms, call, and internet traffic when the base model of cluster 3 is applied on the aggregated dataset of Cell 5577

Second-Step Transfer Learning: The intra-cluster base model is transferred to the BSs inside each cluster via the edge server in the second step. For simulation purposes, we are taking one BS from cluster 3 with cell ID 5577. When the intra-cluster base model is applied to the BS 5577 dataset, due to the similarity in the traffic patterns, the MAPE value for internet traffic is just 4.4%. Finally, the Intra-cluster base model is trained on the traffic pattern of the BS. After minor adjustments, it can be observed in Table 4.1 that the prediction accuracy improves significantly. MAE, MAPE, MSE, and RSME for the internet traffic are 1.58, 3.8, 4.32, and 2.1, respectively. It can be seen that the MAPE value reduces by 12%, from 4.4 to 3.8 for internet traffic. Thus, by applying a dual-step TL-based prediction model, the data, time, and computational resources can be reduced for the model training and prediction without degrading the prediction performance.

Table 4.1: Error metric for the dual-step transfer learning-based prediction model for cell 5577

Cluster 3	Traffic Type	MAE	MAPE	MSE	RMSE
Cell id 5577 (intra-cluster model training on BS dataset)	SMS	0.313	19.12	0.18	0.42
	Call	0.24	12.98	0.1	0.324
	Internet	1.58	3.8	4.32	2.1

In summary, Cooperative Transfer Learning offers a strategic advantage in scenarios where data is distributed, privacy or bandwidth constraints limit traditional centralized training, or where computational resources are unevenly distributed. Its enhancements over the state of the art lie in its efficient, scalable, and flexible approach to leveraging distributed data and computational resources, enabling rapid adaptation and improved performance across a wide array of applications.

4.3 FL Designing Fair and Distributionally Robust Distributed Algorithms

As the world becomes increasingly digitized, edge devices ranging from smartphones to Internet of Things (IoT) gadgets are generating massive volumes of private data, presenting significant challenges in data storage, capacity, and processing. The Federated Learning (FL) paradigm [17] has traditionally facilitated collaborative model learning without centralizing data, thus preserving privacy to a degree, and reducing the transmission overhead. FL enables devices to collaboratively learn a shared global model without the need to transfer large datasets to a central repository, thus facilitating model learning through the exchange of model parameters or gradients and providing some privacy preservation. However, standard FL presents unique challenges due to the diversity of edge devices and the unrealistic assumption of identical data distribution across these devices, leading to biases and fairness issues. Furthermore, it leads to skewed model performance and challenges in scalability and energy consumption.

4.3.1 ADROIT6G specific enhancements over state of the art

To counteract the challenges encountered by standard FL, a robust min-max formulation was introduced in [18], [19], optimizing model performance across the worst-case local data distributions through a communication-efficient iterative descent-ascent algorithm that selects subsets of clients for each communication round. Nonetheless, the integration of FL into wireless edge-centric systems faces major challenges like scalability, latency, and energy consumption. AirComp [20] has been identified as a promising solution for its efficiency and the ability to leverage the superposition property of wireless channels, addressing the exponential growth of IoT devices. Dynamic client scheduling, considering channel conditions and gradient norms for client selection [21], aims to improve energy efficiency but faces challenges in predictability and dependency on tuning parameters. Amongst these challenges, there is a growing need for enhancing edge device performance through energy-efficient and robust distributed learning methods.

By leveraging the components of the ADROIT6G architecture, we propose a framework that facilitates running an efficient, scalable, and adaptable FL process, ensuring optimal performance against non-independent and identically distributed (non-IID) data distributions. Our proposed solution, coined ***channel-aware agnostic federated learning (CA-AFL)*** [22], contributes to the realization of CrowdSourcing AI by optimizing the distributed learning process across wireless edge devices. By leveraging the collective learning power of edge devices while minimizing energy consumption and addressing data heterogeneity, CA-AFL advances the deployment of AI models that are both performant and sustainable. This integration of energy efficiency and robust learning into the FL framework exemplifies how distributed AI can be achieved without compromising on critical operational parameters, pushing the boundaries of what is possible with CrowdSourcing AI in next-generation networks.

The major contributions CA-AFL establishes over the state-of-the-art baselines are:

- Innovative client selection**:** Unlike existing solutions that focus singularly on robustness or energy efficiency, CA-AFL merges these aspects through a combined approach. It introduces a unique client selection mechanism that combines deterministic energy efficiency with a probabilistic robustness sampling approach.
- Theoretical and practical advances**:** We address the dual challenge of energy efficiency and robustness to data heterogeneity, showing that existing methods are merely special cases of this framework. Moreover, we demonstrate, through simulations, the algorithm's superior energy savings and robustness compared to existing baselines, highlighting its practical effectiveness.
- Configurable energy-conservation tuning**:** We present a flexible approach to balancing energy efficiency and distributional robustness through a configurable parameter. This adaptability allows for seamless transitions

between energy-focused and robustness-focused FL, offering tailored solutions to varying network conditions and requirements.

Energy-efficient and fair client selection. The concept of enhancing energy efficiency while ensuring fair distribution in the client selection process for FL involves complex challenges. This is especially evident in the implementation of communication-efficient descent-ascent algorithms, which tackle this problem by probabilistically selecting clients to upload their models. A significant concern is that clients with suboptimal communication channels might be chosen, leading to high energy consumption for model uploads. To address this, we developed a novel client selection strategy that employs two key metrics: (i) energy consumption and (ii) probability distribution, converting the straightforward energy-based selection into a bias-adjustable probability mass function (PMF).

Probability sampling as a product of experts. In our approach, we use probability sampling to reconcile two potentially conflicting insights. By combining the probabilities from two PMFs and adjusting the combined result, we can more accurately weigh the importance of different clients. This method ensures clients who score highly in both PMFs have more chance of being selected, while those excelling in only one are given a lower selection probability. Those ranking low in both are given the least priority. This approach is inspired by the product of experts' method in ML, which combines several models to better understand complex situations. Each PMF in our scenario acts as an "expert," with one focusing on the robust distribution aspect and the other on saving energy. The balance between these experts is controlled by a tuning parameter, which adjusts their influence based on the scenario, allowing for a blend of insights from both aspects.

To sum up, the CA-AFL algorithm runs both descent and ascent steps to manage client selection and model updates. In the descent step, a subset of clients is chosen based on predefined PMFs. These selected clients then proceed to update and upload their models with a new data batch. After this, the Parameter Server (PS) compiles these updates into an aggregated model and distributes it to all clients. Conversely, the ascent step involves uniformly selecting a subset of clients to refine the probability simplex vector's values using another data batch. This update involves only a scalar value, ensuring minimal communication overhead and facilitating efficient exchange with the PS over a control channel.

4.3.2 Functional architecture

The CA-AFL algorithm's integration into the ADROIT6G framework enhances its distributed AI capabilities, allowing it to function smoothly across varied and dynamic network environments. This setup features a distributed learning approach where edge nodes work together under a learning orchestrator to train an ML model. The subsection provides a comprehensive overview of the algorithm's implementation stages and how it interacts within the ADROIT6G infrastructure, highlighting the synergy between distributed AI and the framework's architecture for efficient learning across networks, as seen in Figure 4.9.

- **AI/ML Orchestrator.** The AI/ML Orchestrator, serving as the PS, is crucial for the integration of the CA-AFL algorithm within an FL framework, focusing on energy efficiency and handling non-IID data distributions. It optimizes the balance between energy conservation and robust data handling, notably through adjustments like the energy-conservation factor C. Client selection is strategically managed based on energy profiles and channel conditions, leveraging insights from the AI/ML Resource Orchestrator regarding available network and energy resources. This approach ensures the FL process remains energy efficient. The Orchestrator's responsibilities encompass the entire model development lifecycle, from the selection of the initial model to monitoring its evolution and performance, facilitated by the AI/ML Repository and Catalogue. This enables the deployment of effective models grounded in empirical performance data. Furthermore, the Orchestrator manages the CA-AFL algorithm's comprehensive deployment, including setup, client selection, and model updates, while overseeing

performance monitoring through AI/ML Monitor Agents. This ensures models remain adaptable to evolving data distributions, underscoring the Orchestrator's central role in enhancing FL efficiency.

- **AI/ML Pipeline Manager.** Serving as the communication hub, the AI/ML Pipeline Manager ensures seamless interaction between the AI/ML Orchestrator and the system nodes. It facilitates the transmission of updated global models to the clients and the collection of local model updates and loss computations, which are crucial for the ascent step. This manager plays a pivotal role in implementing the algorithm's orchestration decisions, and handling data flows that include client metrics necessary for computing the PMF and scheduling decisions.
- **AI/ML Resource Orchestrator.** This component assesses and abstracts the network, computational, and energy resources of the system nodes. Its insights allow the AI/ML Orchestrator to make informed decisions on client selection, focusing on optimizing the FL process for energy efficiency. The Resource Orchestrator's data is vital for calculating the energy-conservative PMF, which guides the selection of nodes based on their channel quality and energy usage, aligning with the CA-AFL's emphasis on energy-aware FL.

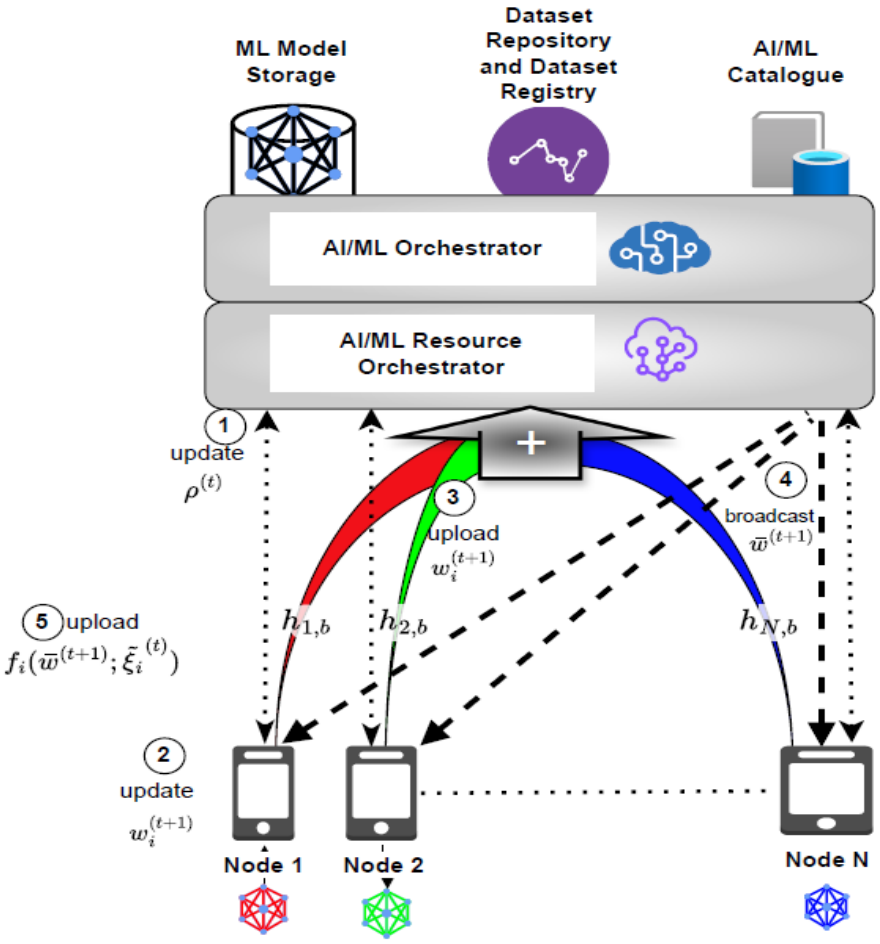


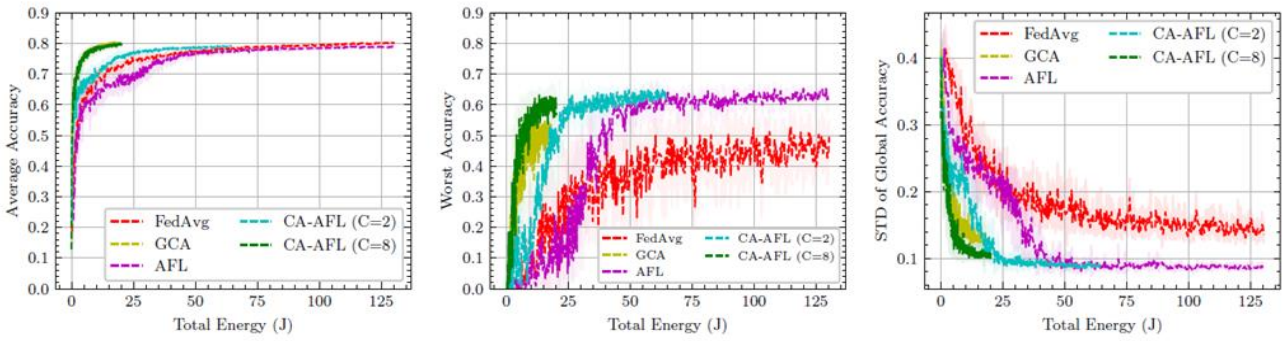
Figure 4.9: Illustration of the CA-AFL algorithm functional architecture.

- **Dataset Repository and Dataset Registry.** Although the CA-AFL algorithm operates under the premise that data remains decentralized on client nodes, the Dataset Repository and Registry provide essential support for initial model training and validation. They offer a structured inventory of available datasets and their characteristics, aiding in the selection of relevant data for preliminary testing, simulation of FL scenarios, and validation of the global model's efficacy across diverse data distributions.

- **AI/ML Catalogue.** These components store detailed records of the CA-AFL learning process and its evaluation metrics, for different models, at different training epochs, including the initial global model and its iterations. The repository and catalogue are instrumental in tracking the performance, updates, and metadata associated with each model type, facilitating the future selection and deployment of the most effective models based on empirical performance data.
- **ML Model Storage.** Model Storage securely houses the sequence of global models generated during the CA-AFL process. This repository enables efficient access to and deployment of models for ongoing training, inference, and performance analysis, supporting the iterative nature of federated learning by providing a historical archive of model evolution.
- **System Nodes (Clients).** The nodes, encompassing edge devices and servers, are the workforce of the CA-AFL algorithm, performing computations on local data to update model parameters. These nodes embody the distributed computing paradigm, with their participation in the learning process directly influenced by the AI/ML Orchestrator's scheduling decisions, based on energy efficiency and data robustness considerations.

4.3.3 Preliminary results

This section showcases the performance analysis of our CA-AFL algorithm, demonstrating its effectiveness through experiments on the Fashion MNIST dataset and a logistic regression model. Our experimental setup involved 100 devices, with 40 participating in each communication round. We compared CA-AFL's performance against three baseline algorithms: (i) FedAvg [17], (ii) AFL [18], and (iii) GCA [21].Figure 4.10 Figure 4.10 (a) shows that while the proposed CA-AFL achieves similar average global accuracy, around 80%, to the baselines, it requires as much total energy as CGA for C=8. Furthermore, we note that FedAvg and AFL are the most energy-intensive, as they are not channel-aware. Figure 4.10 (b) demonstrates that CA-AFL leads to the same worst test accuracy as the robust benchmark, AFL, but with substantially lower energy demands. In fact, CA-AFL achieves the worst test accuracy of 0.6 with half of the total energy compared to AFL. At C=8, our proposed algorithm matches GCA's energy efficiency while achieving the best attainable worst client accuracy for GCA and FedAvg with much less energy consumption. Finally, Figure 4.10 (c) examines the standard deviation (STD) of the global accuracy, measuring how the performance varies across the different devices. At C=8, CA-AFL surpasses GCA in terms of STD while consuming the same amount of energy, even with GCA operating at its optimal hyperparameter configuration. Furthermore, the proposed CA-AFL matches the performance of the benchmark AFL while consuming only one-third of the energy.



(a) Average test accuracy (b) Worst test accuracy (c) STD

Figure 4.10: Performance of CA-AFL compared to baselines in terms of the total energy.

4.4 Meta-learning for Model Selection and Crowdsourced Model Sharing

As the complexity of ML models increases and the need for involving in the AI/ML pipeline heterogeneous nodes becomes critical, it is crucial to develop mechanisms for optimal learning in distributed networks. In a nutshell, it is essential *to learn how to learn* so that the quality of trained models satisfies the learning requirements while the training process meets the characteristics of computationally constrained environments. To this end, it exploits a *meta-learning, CrowdSourcing approach*. In this section we present a *meta-learning, CrowdSourcing approach,* which leverages the knowledge (i.e., the datasets) owned by the different network nodes by aggregating the models trained locally at such nodes. Furthermore, the proposed approach tailors the aggregated model to the availability of computing resources at the node that must execute the model by compressing (e.g., pruning) the model accordingly. In doing so, our solution, called MEta Reinforcement learning framework for tunable RL agents at the edGE (MERGE), realizes the concept of CrowdSourcing AI and efficiently creates ML models that match both the limitations and the performance requirements at the radio interface.

To develop the above concept, MERGE takes as reference scenario a Radio Access Network (RAN) where access nodes need to execute a *radio model* to configure their radio interface. Notably, this scenario also includes the case of a gNB functional split architecture, as promoted by the relevant open RAN (O-RAN) initiative, where the Distributed Unit (DU) executes the radio model and provides the resulting radio policy to the Radio Unit (RU) [23].

4.4.1 ADROIT6G specific enhancements over state of the art

It is worth remarking that considering large-scale connectivity with diverse channel fading characteristics and service requirements, the application of standard optimization frameworks for solving the problem of radio resource allocation is often intractable. Learning-based resource allocation algorithms [24], instead, have the potential to effectively cope with the system complexity for providing automated network control. However, also in the case of learning-based approaches, several hurdles exist in developing scalable, efficient solutions. The study in [25] has shown that conventional centralized approaches may scale poorly, while distributed radio resource allocation can be much more effective. It is worth observing that the latter can also be a better match for the architecture of a virtual RAN consisting of several radio nodes (e.g., DUs). On the other hand, a radio node has typically limited computing and storage resources; moreover, implementing learning-based resource allocation algorithms at a radio node would imply that the learning agent is exposed only to local network dynamics, thereby limiting its generalization capability and thus impacting the quality of resource allocation decisions. To address the above issues, it is imperative to envision solutions for the learning of distributed agents that suit both the availability of compute resources at the radio nodes and the need for model generalization.

MERGE builds on the intuition that, as noted above, a learning approach for radio resource allocation cannot be handled at the radio node alone. Instead, the radio node can host and execute a *pre-trained, reduced-size (or pruned) version of a learning agent* that matches its computational constraints and has been properly created by the AI/ML Orchestrator. Thanks to its architectural role and access to a larger storage capability, the AI/ML Orchestrator can indeed collect knowledge from all the radio nodes through the AI/ML Resource Manager and use it to improve the capability of radio agents to deal with unseen scenarios (e.g., sudden increase in network traffic, or SINR variations). Furthermore, to guarantee a timely delivery of up-to-date radio agents, the AI/ML Orchestrator can leverage pruned versions of a model [26] and create model versions of different size, hence complexity level, and deliver an agent of the right size upon a radio node's request.

To summarize, the proposed solution allows for the realization of the CrowdSourcing AI concept, leading to the following enhancements over the state of the art:

- It defines a meta-learning framework for the cooperative training of ML models;
- It introduces a *Model Maker* logic (to run at the AI/ML Orchestrator), which comprises designing a *Deep Reinforcement Learning (DRL) agent* to select a subset of ML models based on the availability of

computing resources at the requesting radio node, and an *Aggregator*, to combine selected crowdsourced ML models for creating one that is of suitable size and provides high-quality decisions;
- Finally, through preliminary results in the context of RL for radio interface configuration, it demonstrates the potential of the CrowdSourcing AI approach against two benchmarks and shows that the decision-making quality of MERGE matches the ideal benchmark with 25% less computational requirements, while, for a given computational resource, its decision-making performance is about 20% better than that of a single compressed model.

We remark that the existing meta-learning approaches [27], [28], [29] deal with the edge compute constraints to some extent, however, none of them addresses radio resource allocation in distributed vRANs. Also, MERGE overcomes the lack of generalization of ML pruning methods [30], [31], while creating agents suitable for resource-constrained environments.

4.4.2 Functional architecture

With reference to the AI/ML architecture shown in Figure 3.3, the MERGE solution is implemented and run at the AI/ML Orchestrator and, more specifically, within the AI/ML Model Selection module. Below we describe how MERGE operates within the AI/ML framework, while a pictorial representation is given in Figure 4.11:

1. The Controller receives a request for a trained model to be used for inference at the radio node and relays such request to the AI/ML Orchestrator Engine;
2. The request is processed by the AI/ML Model Selection module, which accesses the metadata of the ML models, and the ML models themselves, that have been previously trained by radio nodes and that are stored into, respectively, the AI/ML Catalogue and the AI/ML model storage. Note that the information of the radio nodes' capabilities is provided to the AI/ML Orchestrator (hence the AI/ML Model Selection module) by the AI/ML Resource Manager;
3. The AI/ML Orchestrator provides the AI/ML Pipeline Manager with the new model that meets the radio node's capabilities and that the AI/ML Pipeline Manager delivers to the AI/ML Resource Manager, along with instructions on the resource to be allocated at the radio node.
4. The AI/ML Resource Manager deploys the model and allocates the resources to the radio node, as per the received instructions.
5. The AI/ML Resource Manager receives periodically the updated RL radio model and the related performance from the radio node and sends them to the AI/ML Orchestrator, which in its turn updates the AI/ML Catalogue (and thus the AI/ML model storage) and uses the information on performance to keep learning.

The assessment over time of the quality of the output of the local interference task must be performed by the AI/ML Monitor agent running withing each radio node.

We underline that the ML model to be executed at the radio node can be of any type, while the characterizing core of the proposed solution is represented by the Model Maker. To implement the Model Maker, MERGE leverages reinforcement learning wherein the statistical properties of crowdsourced models from the radio nodes and the requesting node's computational constraints are used as the input, and the Key Performance Indicators (KPIs) satisfaction is accounted in a carefully designed reward function.

Here we remark that while the state-of-the art distributed learning techniques, such as federated learning, require all the nodes to periodically and synchronously share their models with a coordinator, MERGE takes an asynchronous approach. Although it requires that radio nodes share their updated radio models with the AI/ML Orchestrator, the sharing is sporadic and depends on the change in the training environment of the radio models at the respective radio nodes. Further, different radio nodes may not necessarily share their ML models with the AI/ML Orchestrator simultaneously.

The Model Maker is a Deep Q-learning Network (DQN)-based model, which implements meta learning for improving the training of the radio models. Its objective is to build pre-trained radio agents that meet the constraints imposed by the computing-constrained edge while providing a high-quality radio interface configuration when used for inference at the radio nodes. The elements of the Model Maker are as follows.

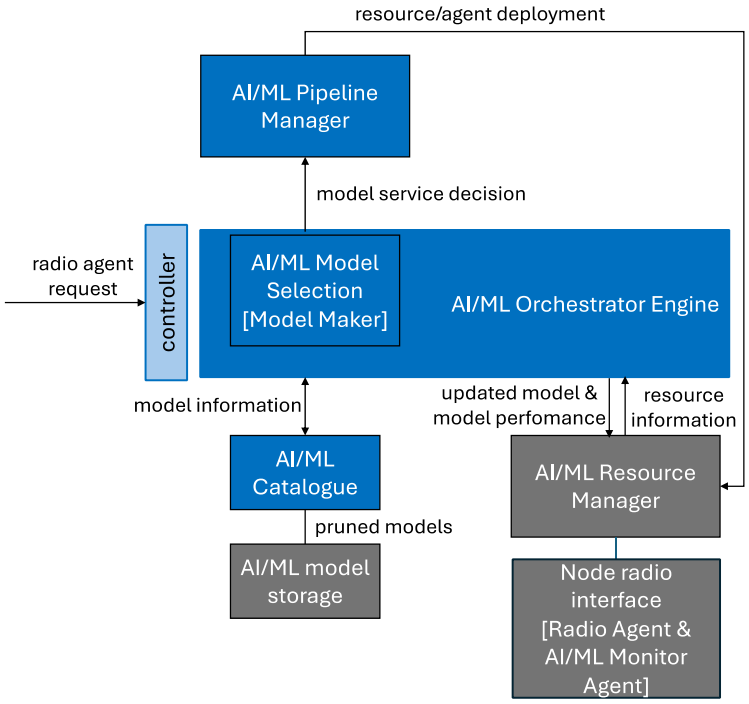


Figure 4.11: Integration of MERGE in the functional AI/ML framework.

Context space. The selection of pruned versions of the radio model is primarily governed by two factors: (i) the availability of computing resources at the radio node and (ii) the learning quality-related statistics of the pruned radio models stored in the AI/ML model storage. The context is composed of the following elements:

- for each radio model in the AI/ML Catalogue, the size of input layer, output layer, and hidden layers, number of hidden layers, mean, variance, and sparsity of the weight matrix;
- the available computational resource at the node that needs to receive a radio model.

Action space. The design objectives of the model maker are to identify how many and which (possibly pruned) radio models to select for creating an aggregated inference model. To jointly meet these objectives, we formulate the model maker's actions as vectors, each denoting a probability distribution over the choice of models to be selected. The non-zero elements of this vector will indicate the selected radio models to be aggregated to create the new model to be delivered to the radio node.

Reward. We assign two KPIs to the model maker: (i) the amount of computing resources consumed by the inference model, and (ii) the reward value of the radio model at its convergence, which represents the satisfaction of the radio KPIs for the served UEs. Then the total reward of the model maker is then given by summing the above two components.

DQN learning and pruned models' selection. Q-learning is a model-free RL technique in which the agent learns to choose actions for a given state of the environment by maximizing its long-term reward. For the model maker, we adopt the definition of long-term reward given as the discounted expected return [32]. In practice, since the optimal Q-values follow the Bellman's recursive equation, they are iteratively learned over successive decision windows by minimizing the difference between the optimal Q-value and its estimate. In a DQN, both such values are approximated using two separate deep neural networks, called the target and the prediction network. The target network output behaves as the ground truth for the prediction network, while the output of the latter governs the choice of selected actions. Thus, the DQN is trained by minimizing the squared error loss function given by [32]. The Q-value estimation is followed by an ε-greedy action selection policy. Notice that the cardinality of the action space quickly scales with an increase in the number of the radio models available in the AI/ML model storage, in turn increasing the complexity of the DQN implementation. To this end, we limit the size of the output layer of the prediction network to the number of available radio models and augment it with a

softmax layer. Thus, the prediction network output is a probability distribution vector with the generic element indicating the preference of choosing the corresponding pruned radio model. Subsequently, we evaluate Hellinger's distance between the probability distribution predicted by the DQN and the possible actions and, with probability 1-ε, select the greedy action that minimizes such a distance. Over successive decision windows, ε decays by a factor of 10^{-4} to allow for more exploitation rather than exploration of the environment. Models corresponding to non-zero elements in the above probability vector are the desired subset of selected pruned radio models.

Inference model. Since the selected pruned radio models may be heterogeneous owing to their different pruning factors, and hence different sizes, we follow the hierarchical strategy in [33] to aggregate them into a radio model to be used for inference at the requesting radio node. Once the aggregated model is ready, it is output to the AI/ML Pipeline Manager, and it is delivered through the AI/ML Resource Manager to the radio node for which it was created. The performance of the model is observed at the radio node through the AI/ML Monitor agent in terms of reward of the radio model at convergence, which is then sent back to the model maker for its learning.

4.4.3 Preliminary results

Consider 7 pruned versions of the full model to be locally stored, out of which 2 are pruned by a factor of 0.5, and the rest by 0.1, 0.3, 0.6, 0.8, and 0.9 each. The pruned models are further trained using different context and action scenarios to emulate statistical network diversity. Such pre-trained pruned radio models are then selectively combined using the meta learning policy of the model maker. We focus on a node requesting a radio model and consider that it is serving 2 UEs. To enable the pre-training and learning of the radio agents, we used the datasets reporting the CPU constraints and the corresponding KPIs for various contexts and actions, available at https://github.com/corrado113/VERA.

We show the performance of MERGE in terms of KPIs of the model maker, i.e., the radio agent reward and the computing resource consumption; the latter is expressed as the ratio of the consumed CPU/GPU ϕ_c to the available one ϕ_a. Figure 4.12 highlights that, barring the initial time period when the model maker is still learning, a high value of reward at convergence is maintained at the radio agent, i.e., the inference model is making high-quality radio resource allocation decisions at the radio node. Further, Figure 4.12 shows that, as the model maker learns the meta learning policy, the ratio of computational resources consumed by the model with respect to the available computational resources at the radio node converges close to 1. In other words, the proposed MERGE framework can ensure sufficiently high-quality decisions while satisfying the computational constraints of the radio node.

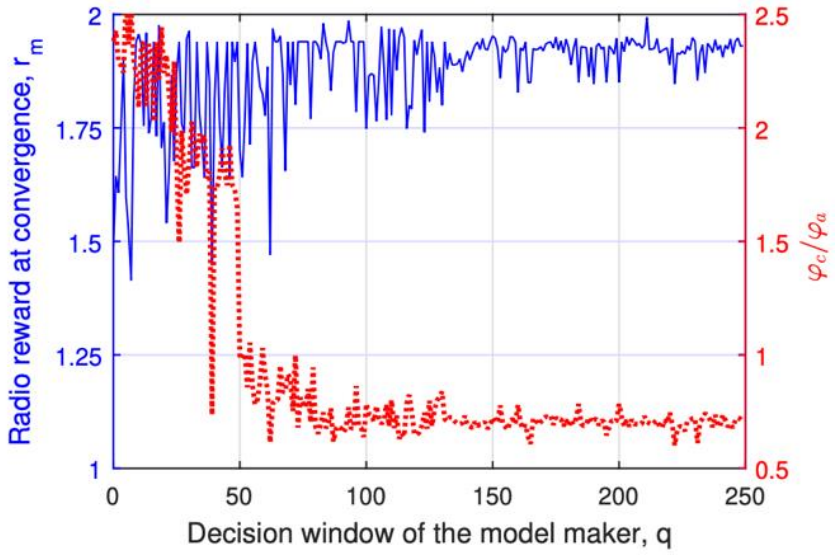


Figure 4.12: Performance of the proposed MERGE meta-RL framework for CrowdSourcing AI.

4.5 Energy-efficient Federated Learning Algorithms

AI/ML technologies are the way to provide effective resource orchestration to improve performance. However, unless these technologies are well managed, such improvement may come at the cost of scarfing resourcing utilization and sustainability across networks. This section delves into a methodology aimed at enhancing the management of AI/ML algorithms. Through discussions and elucidations supported by several preliminary results, this section provides an overview of our advancements and contributions in AI/ML resource orchestration, showcasing their alignment with ADROIT6G's innovative architecture. The section, more specifically, presents solutions to enhance state-of-art distributed AI/ML techniques in energy consumption and communications efficiency that are proven to be scalable, privacy preserving, and adaptable to heterogeneous settings.

4.5.1 ADROIT6G specific enhancements over state of the art

The evolution of FL presents both challenges and opportunities for enhancing communication, computational, and energy efficiency within distributed networks. Accelerating the convergence of FL algorithms can significantly save energy and enhance communication efficiency by reducing the number of necessary communication rounds between the coordinator and edge devices. This approach streamlines the learning process, allowing for quicker achievement of model accuracy with less energy consumption and bandwidth usage. First-order FL methods [17], [34], [35] perform a uniform step size across all dimensions of the optimization problem. By doing this, these methods ignore the curvature information of the loss landscape. Hence, they often exhibit a slower convergence, which means that more communication rounds are needed between the coordinator and the edge devices, resulting in more energy consumption. Unlike first-order methods that rely solely on gradient information, second-order methods [36], [37] rely on the Hessian's information to accelerate the convergence process. However, the adoption of second-order methods introduces additional computational and communication burdens due to the complexities involved in computing and sharing the Hessian's information. While recent second-order FL methods employ various techniques for Hessian approximation [37], [38] and compression [39] to make second-order optimization more feasible in FL settings, challenges remain, including the impracticality of computing full Hessians for large models and the potential for convergence to suboptimal points due to inaccurate curvature information.

Within the ADROIT6G project, we have developed Fed-Sophia [40], a scalable, communication and energy-efficient algorithm, that addresses the challenges faced by second-order FL methods by leveraging curvature information of the loss function to guide and expedite the convergence process through a lightweight estimation of the Hessian's diagonal, combined with a weighted moving average of the gradient and a clipping operation to determine the descent direction. Specifically, Fed-Sophia outperforms existing methods by requiring fewer communication rounds to achieve convergence and by offering a more computation, communication, and energy-efficient solution for FL, particularly in large model scenarios such as deep learning, and in heterogeneous settings.

The key improvements Fed-Sophia introduces over the current state of the art are outlined as follows:

- **Communication Efficiency:** By adopting a lightweight Hessian estimation and sharing only model parameters, Fed-Sophia significantly reduces the communication overhead compared to both first and second-order FL methods. This efficiency is paramount in distributed networks where bandwidth and data transmission costs are critical constraints.
- **Energy Efficiency:** Numerical evaluations underscore Fed-Sophia's superior energy efficiency. The algorithm's design minimizes both computational and communication energy demands, aligning with the ADROIT6G project's sustainability goals.

© ADROIT6G, 2023

- **Robustness to Data Heterogeneity:** The algorithm is designed to operate effectively across heterogeneous data distributions among devices, a common scenario in real-world applications. Its robustness enhances model performance and reliability, addressing the drift in local updates and variance reduction challenges.

Hessian estimation. Fed-Sophia explores the concept of using the Hessian matrix as a pre-conditioner in guiding the FL optimization problem. The main idea is that the Hessian matrix, by capturing the curvature of the objective function, can significantly improve the convergence properties of gradient-based optimization methods. This leads to faster convergence and potentially overcomes some of the challenges associated with FL, such as data heterogeneity and communication and energy constraints. However, in scenarios involving extremely large DNNs, calculating and sharing the full Hessian matrix is unfeasible. Consequently, this limitation hinders the applicability of second-order FL methods. A possible approach is to approximate the Hessian by focusing solely on its diagonal components using the Gauss-Newton-Bartlett estimator. These diagonal entries are significant as they encapsulate the curvature information pertinent to each dimension in the optimization landscape.

Adaptive step size. By focusing on adaptive mechanisms to enhance convergence and learning efficiency, Fed-Sophia introduces an innovative approach to adjusting the step size. The adaptive step size is obtained by dividing the gradient's moving average by the estimated diagonal Hessian. The intuition behind this is that the optimal step size should inversely correlate with the Hessian values in each dimension. Instead of employing a uniform update step for all dimensions of the loss function, the strategy involves fine-tuning this step to accommodate dimensions that are either sharp or flat appropriately. For instance, the step size should be smaller in sharper dimensions compared to flatter dimensions. By dividing by the estimated Hessian, Fed-Sophia enables a mechanism for adaptive step size, which, consequently, allocates varying step sizes based on the loss curvature's form. In fact, adjusting the learning steps based on the local curvature information provided by the Hessian allows for more informed updates to the model parameters.

Clipping operation. A clipping mechanism has been implemented in Fed-Sophia to mitigate two common issues that stem from the optimization formulation in FL, which are:

- A risk of the optimization process converging to a saddle point in cases where a non-convex loss function is used, e.g., when dealing with NNs**.**
- The Hessian matrix's values could incorrectly influence the convergence, especially when the loss function's curvature experiences abrupt changes.

This technique helps protect the optimization process from incorporating misleading data into the diagonal of the Hessian matrix and ensures the convergence to the optimal solution.

In summary, Fed-Sophia incorporates mechanisms to minimize energy consumption during both the computation and communication phases as follows:

- Devices compute and share the Hessian estimation periodically every number of communication rounds and not every time they communicate with the PS. By doing so, this reduces the frequency of energy-intensive operations.
- The adaptive step sizes, based on the lightweight Hessian estimation, further enhance computational and communication efficiency by dynamically adjusting to the curvature of the loss function, thereby reducing unnecessary computation.

4.5.2 Functional architecture

This subsection details the Fed-Sophia's implementation phases and interaction with the ADROIT6G infrastructure. We consider a distributed learning scenario in which edge nodes cooperatively train an ML model, e.g., the weights and biases of an NN, with the help of a learning orchestrator. Fed-Sophia, with its lightweight

computation and communication complexity, makes a perfect fit for the framework. Figure 4.13 illustrates the Fed-Sophia algorithm functional architecture.

- **AI/ML Orchestrator** is an essential part of the Fed-Sophia Algorithm. The AI/ML Orchestrator aggregates the edge node's trained models by averaging them to better combine individual local models into a unified global one. Acting as a PS, the AI/ML orchestrator is also responsible for distributing the averaged model back into the edge nodes, facilitating the convergence of the model while adapting it to their local data.
- **AI/ML Pipeline Manager** has the responsibility of broadcasting of the global model from the AI/ML Orchestrator to the edge nodes. It enhances this process and ensures a robust transmission of the global model. For the convergence of the learning problem to happen, the AI/ML Pipeline handles the process of combining local models' results from each client, making it an essential part of monitoring the convergence of the algorithm.
- **ML Resource Orchestrator.** For the orchestration of the model transmission to happen, the resource orchestrator is responsible for providing insightful information about the status of the clients participating in each communication round. This information could be further processed by the AI/ML orchestrator to mitigate any clients that have low to negative contributions that affect the performance of the global model.
- **Dataset Repository and Dataset Registry.** The Dataset Repository and Registry serves as a storage for any relevant public dataset that can be used to further test the performance and the robustness of the algorithm of the global model, that resulted out of the aggregation process, to handle different distributions.
- **AI/ML Catalogue** stores such metrics as the average test loss, and test accuracy based on different models through the convergence of the learning process, to enable further analysis of the different metrics used to monitor the learning process.
- **The ML Model Storage** enables and enhances the deployment, rolling back, and versioning of different global models. It also serves as an aid for the convergence analysis and mentoring of the algorithm behavior throughout the learning process.
- **System Nodes (Clients)** are the places where the actual training of the models happens. It performs local computation of the Fed-Sophia steps, enabling the local model to better represent the edge node's local data. Every edge node computes the estimated moving average of the gradient and Hessian, enabling the convergence process to be enhanced by the local loss functions' curvature information.

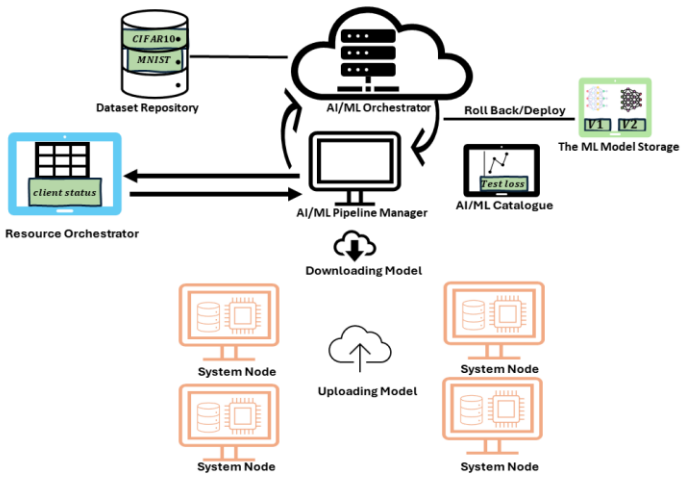


Figure 4.13: Illustration of the Fed-Sophia algorithm functional architecture.

4.5.3 Preliminary results

We evaluate the performance of Fed-Sophia against two state-of-the-art baselines: (i) FedAvg [17], which is a first-order method relying solely on gradient information, and (ii) DONE [38], an approximate Newton-type method. In this experiment, we consider 32 edge devices connected to a PS. We used two datasets, MNIST and Fashion MNIST, with a Convolutional Neural Network (CNN) employed as the classification model. The results, as depicted in Figure 4.14, demonstrate that Fed-Sophia surpasses both FedAvg and DONE by requiring fewer communication rounds. Additionally, it achieves superior test accuracy compared to the baselines. Table 4.2 demonstrates that, when aiming for a test accuracy of 75% on the MNIST dataset using a CNN model, Fed-Sophia significantly outperforms both DONE and FedAvg in minimizing computation energy consumption. In fact, Fed-Sophia's computational energy requirement is nearly half that of FedAvg and it utilizes up to 4000 times less computational power compared to DONE. Furthermore, Fed-Sophia's communication energy is substantially lower than the other baselines, requiring only about 20% of the energy needed by its nearest competitor, FedAvg. This significant reduction in both computational and communication energy solidifies Fed-Sophia's position as a highly efficient method, demonstrating its minimal overall energy footprint in comparison with the other two baselines.

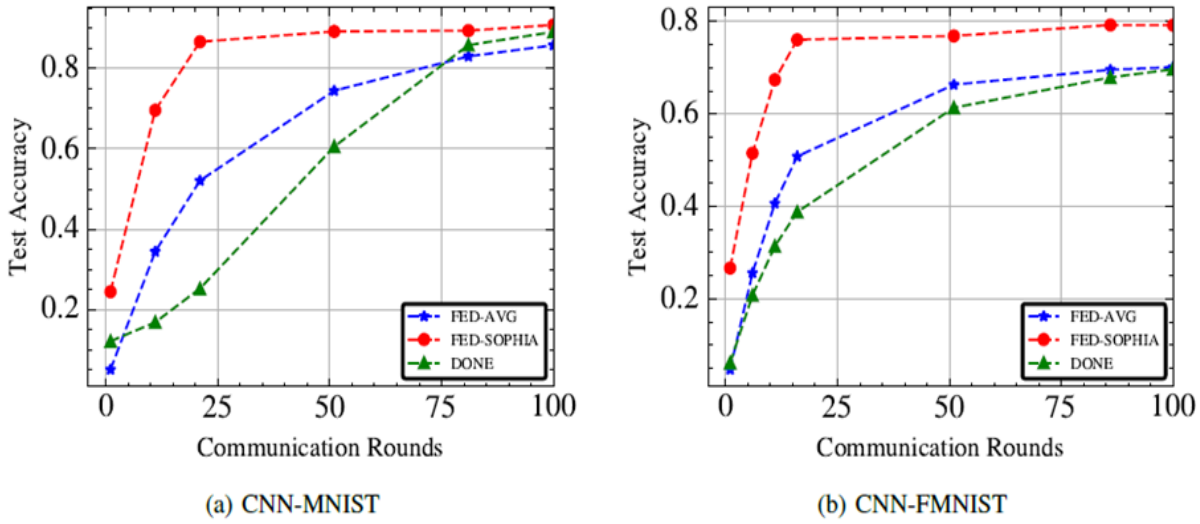


Figure 4.14: Performance of Fed-Sophia against other baselines for MNIST and FMNIST datasets using a CNN model

Table 4.2: Computation/communication energy costs and corresponding carbon footprints for MNIST with CNN for target accuracy of 75%

Algorithm	Computation Energy [MJ]	Communication Energy [MJ]	Total Footprint [kg-CO2-eq]
DONE	$6170E-4$	153.2	$1097E+3$
FedAvg	$2.2E-4$	139.2	$0.123E+3$
Fed-Sophia	$1.4E-4$	27.8	$0.004E+3$

4.6 Decentralized Collaborative ML Techniques

The surge in data collection driven by technological advancement has fuelled the use of ML algorithms for complex tasks. However, traditional approaches often require centralized dataset storage, which raises privacy concerns for sensitive data and can be computationally expensive. Furthermore, centralized solutions may not be possible due to the size of dataset as it incurs high communication costs when transmitting it to a central server, as in the case of large-size datasets. Decentralized implementation of FL tackles these challenges by

enabling collaborative model training and eliminates the need for a central server, a potential point of failure. In decentralized FL, individual devices like smartphones train local copies of the model using their own data. This ensures privacy by keeping sensitive data on the device. Clients then exchange model updates, which capture the learning from their data, directly with each other. This makes it increasingly prominent in healthcare, and the IoT, paving the way for a new era of privacy-preserving and better resource usage by enabling edge computing.

4.6.1 ADROIT6G specific enhancements over state of the art

The proliferation of connected devices and the exponential growth of data have necessitated advancements in decentralized learning methods to address the challenges of privacy, communication efficiency, and energy constraints in AI/ML environments. A communication and energy-efficient approach to decentralized learning is paramount to addressing the challenges associated with data distribution and computational resources across the network. Conventional methods in FL primarily utilize first-order gradient-based techniques that suffer from slow convergence and demand extensive communication, leading to high latency and costs. To counter these limitations, techniques such as quantization [41], sparsification [42], and gradient reuse [43], were introduced to enhance communication efficiency. Alternatively, second-order FL was recently introduced to accelerate the convergence speed, hence decreasing the total number of required communication rounds [44]. However, as pointed out in Section 5.1, sharing the Hessian in second-order FL comes with a huge communication and computational burden. In this section, we introduce two novel approaches, Q-DNGAM [45] and DIN [46], to decentralized learning that emphasize communication and energy efficiency, a critical factor in the distributed AI/ML landscape, especially for bandwidth-limited systems. Both proposed algorithms rely on second-order Newton-type workflow to significantly reduce communication and energy overhead by efficiently utilizing second-order information without the need for transmitting Hessian information between devices. Moreover, we bypass the need for transmitting the Hessian matrix at each iteration using an approximation for the inverse Hessian-gradient product, thereby preserving privacy and reducing communication and energy costs. To further enhance communication and energy savings, Q-DNGAM leverages sparse and quantized updates instead of full Hessian matrices.

Our approach introduces the following innovations compared to state-of-the-art methods:

- **Communication and Energy Efficiency:** By formulating the Newton direction learning problem as a sum of separable functions subject to a consensus constraint, our approach ensures that each device shares only a model-sized vector, compared to second-order methods that require sharing the Hessian matrix, which drastically reduces the communication and energy cost per iteration. Hence, we attain the fast convergence property of second-order methods and low communication cost per iteration of first-order methods.
- **Privacy Preservation:** In our approach, devices share only an approximation for the inverse Hessian-gradient product, which enhances privacy by avoiding the explicit sharing of gradient and Hessian information.
- **Stochastic Quantization:** To further diminish the communication load, we implement stochastic quantization. This technique allows for the transmission of significantly compressed model updates without compromising the learning process's integrity or convergence speed.

Our approach consists of reformulating the original Newton's step to solve the FL problem into a bilevel optimization problem: (i) an inner problem that consists of minimizing a quadratic problem to get the descent direction, and (ii) an outer problem that updates the ML model according to this direction.

Approximating the descent direction (inner problem). In the inner problem, we aim to learn the Hessian inverse-gradient product without sharing individual gradients and Hessians, improving privacy and communication/energy efficiency. The Hessian inverse-gradient product is a solution to a quadratic problem that

is approximated using a single ADMM (Alternating Direction Method of Multipliers) step at each iteration, replacing the need for transmitting Hessian information.

Newton step (outer problem). Using the approximated Hessian inverse-gradient product from the inner problem, devices perform one Newton's descent step update. This step ensures the fast convergence characteristic of Newton-type methods while maintaining the benefits of communication and energy efficiency since only vector-values quantities are shared among devices.

Stochastic quantization. To further enhance communication efficiency, Q-DNGAM incorporates stochastic quantization, a technique that quantizes the transmitted variable, thereby reducing the size of the data needed to be sent over the network. This step significantly lowers communication and energy costs by compressing the amount of information exchanged without compromising the algorithm's performance or convergence properties. Stochastic quantization is particularly beneficial in scenarios where bandwidth is limited, or the cost of communication is a critical concern.

4.6.2 Functional architecture

DIN framework can seamlessly integrate with the ADROIT6G architecture to enhance its distributed AI capabilities, whether in centralized or decentralized settings. Though the DIN algorithm runs in a decentralized fashion, we limit the discussion in this section to the special case of the star topology, i.e., in the presence of a PS. DIN's integration offers increased flexibility and improves learning speed by utilizing second-order information while reducing energy consumption. The proposed setup establishes a distributed learning framework wherein edge devices collaboratively learn a model without the need to share data. In the following section, we elaborate on how our framework can be integrated into the ADROIT6G architecture and mapped to its functional units, as shown in Figure 4.15.

- **AI/ML Orchestrator.** The AI/ML Orchestrator plays a crucial role in integrating the DIN algorithm into an FL framework. Acting as a coordinator or PS during training, the orchestrator is responsible for several tasks. It selects the model to be trained, initiates the training process, and triggers nodes to share their local updates. These updates are then aggregated and sent back to the nodes. Throughout this process, the orchestrator monitors performance using the AI/ML pipeline manager and catalogue.
- **AI/ML Pipeline Manager.** The AI/ML Pipeline Manager facilitates smooth communication between the AI/ML Orchestrator and system nodes. It serves as the intermediary for sharing model updates, enabling the transfer of update directions between nodes, and aggregating the local updates used by the orchestrator.
- **Dataset Repository and Dataset Registry.** While the DIN algorithm assumes the existence of local data storage on client nodes and no sharing of any of the data, the Dataset Repository and Dataset Registry enable initial model training and validation on the PS side. They provide an organized collection of available datasets and their attributes, assisting in selecting suitable datasets for PS training and testing.
- **AI/ML Catalogue.** The AI/ML catalogue serves as a repository and catalogue, storing details of the training and evaluation metrics of multiple training rounds. Those records are then used in tracking the performance and updates associated with each model, thereby assisting in the future model selection and deployment based on set performance targets.
- **ML Model Storage.** This component stores the sequence of global models and aggregates update directions generated throughout the training process of DIN. This repository simplifies the deployment of models and enables continuous training and inference.
- **System Nodes (Clients).** The nodes, which consist of edge devices and the PS, form the core of the DIN framework. They perform computations on local data to determine the update direction, using both first- and second-order information. Supported by the orchestrator and AI/ML pipeline manager, the edge

nodes transmit their local updates to the PS, which aggregates them. Subsequently, they receive these aggregates to complete one communication round and update the local model.

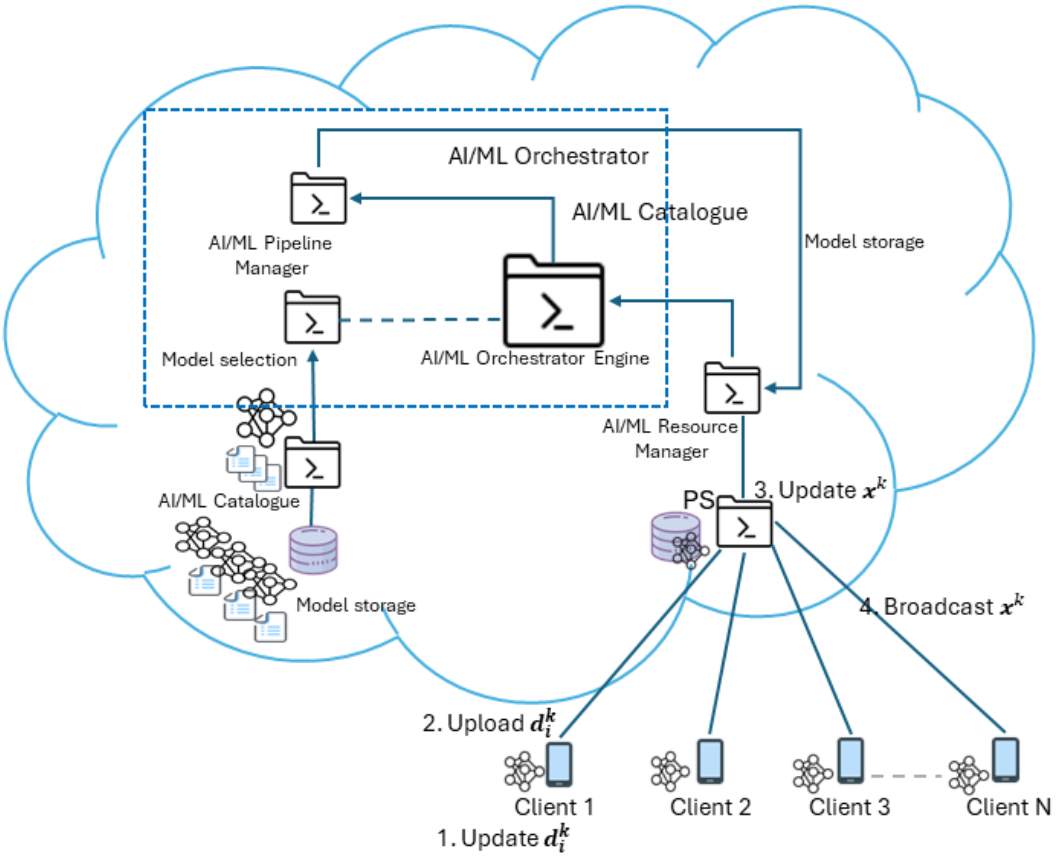


Figure 4.15: Illustration of the DIN algorithm functional architecture.

4.6.3 Preliminary results

Figure 4.16 plots the loss of Q-DNGAM and 5 other baselines as a function of the transmission energy per node and under various signal-to-noise ratio (SNR) conditions. Figure 4.16 (left) highlights Q-DNGAM's energy efficiency compared to the other baselines. Notably, Q-DNGAM necessitates less than half the transmission energy per node compared to the first-order quantized baseline Q-GADMM [47]. Figure 4.16 (right) shows that Q-DNGAM successfully reaches the desired loss level at an SNR of 8 dB, outperforming the other baselines that fail to meet the target even at an SNR of 20 dB. The superior performance of Q-DNGAM can be attributed to its efficient use of a smaller number of transmitted bits per iteration and its rapid convergence rate. Additionally, it is observed that increasing the SNR value boosts the performance of the baseline algorithms, except for DSGD and Newton tracking [48]. These exceptions are due to their need for a greater number of iterations to achieve convergence, which in turn requires more communication resources. In Table 4.3, the communication and computation energies and the carbon footprint of four algorithms to achieve a 10^{-5} optimality gap on the phishing dataset are presented. It is noted that DGD exhibits the greatest total energy use and carbon footprint. Despite DGD's lower computational demands, its extensive need for communication rounds to reach the desired optimality gap results in significant communication energy expenses. Conversely, Network Newton [49] incurs the most computational energy costs, as it executes two matrix inversions per communication round. Meanwhile, DIN stands out for its efficiency in both computational and communication energy, attributed to its rapid convergence and the execution of a single matrix inversion per communication round.

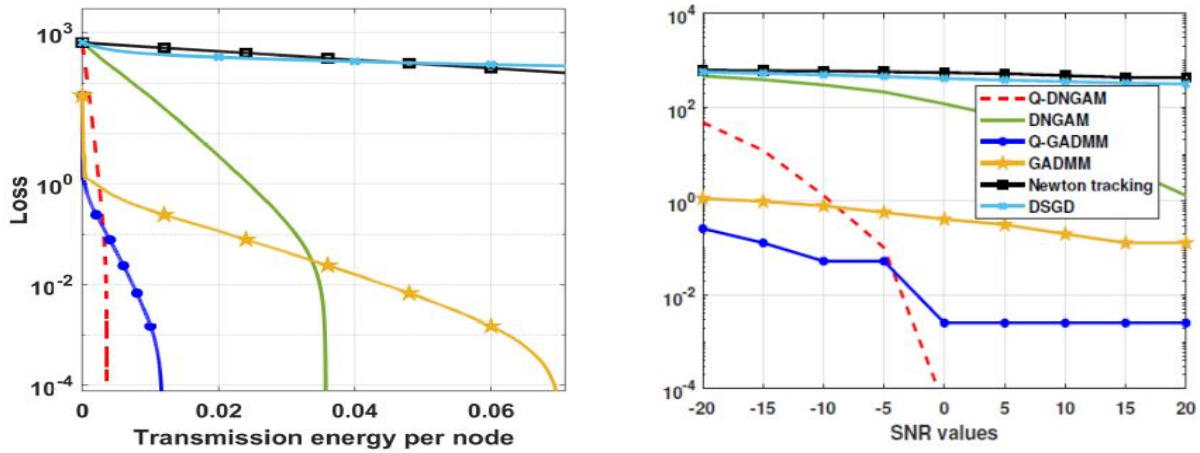


Figure 4.16: Performance of Q-DNGAM against other baselines for the a1a dataset.

Table 4.3: Computation/communication energy costs and corresponding carbon footprints for the Phishing dataset for a target optimality gap of 10^{-5}

Algorithm	Comp. Energy [J]	Comm. Energy [J]	Total Footprint [g-CO2-eq]
DGD	$5.97E-2$	25.54	$4.55E-4$
DIN	$1.32E-2$	4.60	$0.82E-4$
Newton Tracking	$2.69E-2$	7.88	$1.41E-4$
Network Newton	$7.96E-2$	10.21	$1.83E-4$

4.7 Edge-mobile Device Cooperative Inference

Nowadays, ML is becoming pervasive and crucial to the provisioning of intelligent services in several application domains, from smart manufacturing to autonomous driving, from digital health to smart agriculture [50]. Many end-devices generate big volumes of data that need to be processed to get valuable knowledge. Traditionally, such data are processed and stored in cloud centers, where powerful machines have the necessary resources to run ML algorithms and take smart decisions. With the increased capabilities of end-devices and the growing demand for low-latency applications and data privacy, we are witnessing a paradigm shift where data processing and decisions are taken at the edge or even locally on end-devices. Executing ML models in resource-constrained edge environments needs to account for potential accuracy loss due to model compression, and for latency and energy consumption.

4.7.1 ADROIT6G specific enhancements over state of the art

This section reflects preliminary results of a research activity that aims at investigating solutions for performing inference tasks efficiently by exploiting the CrowdSourcing AI concept in resource-constrained edge scenarios, dominated by the presence of mobile and embedded devices. Inference tasks need trained ML models to make decisions and predictions. Performing inference, although less demanding than model training, can still require large resource footprint, especially considering that many intelligent services rely on the execution of compute-intensive DNN models, characterized by a large number of parameters that necessitate substantial memory space and frequent memory access.

Model compression techniques like pruning, which eliminate e.g., the low-magnitude, model parameters and reduce model size and complexity, are well known solutions to alleviate the burden on resource-constrained devices and reduce energy consumption. However, pruning may result in degraded model output quality and reduced accuracy. Related works [51],[52],[53],[54] evaluated the effectiveness of pruning algorithms for neural networks, considering such factors as computational complexity and accuracy. The state-of-the-art also

addressed DNN model compression for diverse devices, optimal training location decisions, and proposed methods to process complex applications on board of resource-constrained devices, improving overall resource efficiency.

Executing DNN models at resource-constrained edge environments requires careful consideration to balance model size, latency, energy efficiency, and accuracy. Unlike prior studies, in this project we examine the benefits and costs associated with executing inference tasks on mobile devices, considering energy consumption, latency, and accuracy. We delve into the comparison between executing inference tasks at the mobile devices versus offloading them to an edge server [55] by considering:

- impact of model pruning on inference accuracy, latency, and computational cost,
- computational footprint of inference processes,
- energy consumption and latency costs incurred by data transfer between the network edge and mobile devices.

Our primary enhancements to the state-of-the-art are, therefore, as follows:

- characterizing the performance of pruning and inference execution at edge servers and mobile devices through a series of extensive experiments, focusing on execution time and accuracy across various DNN models and pruning factors,
- identifying conditions under which executing inference at the mobile device using a pruned DNN model proves to be more advantageous compared to conventional inference offloading approaches. This analysis considers factors such as edge load, different DNN models, and pruning factors, evaluating inference delay, data exchange volume, and energy consumption throughout the process.

4.7.2 Functional architecture

We consider the cellular edge domain illustrated in Figure 4.17, where mobile devices, under a Base Station coverage, need to execute an inference task, e.g., image recognition.

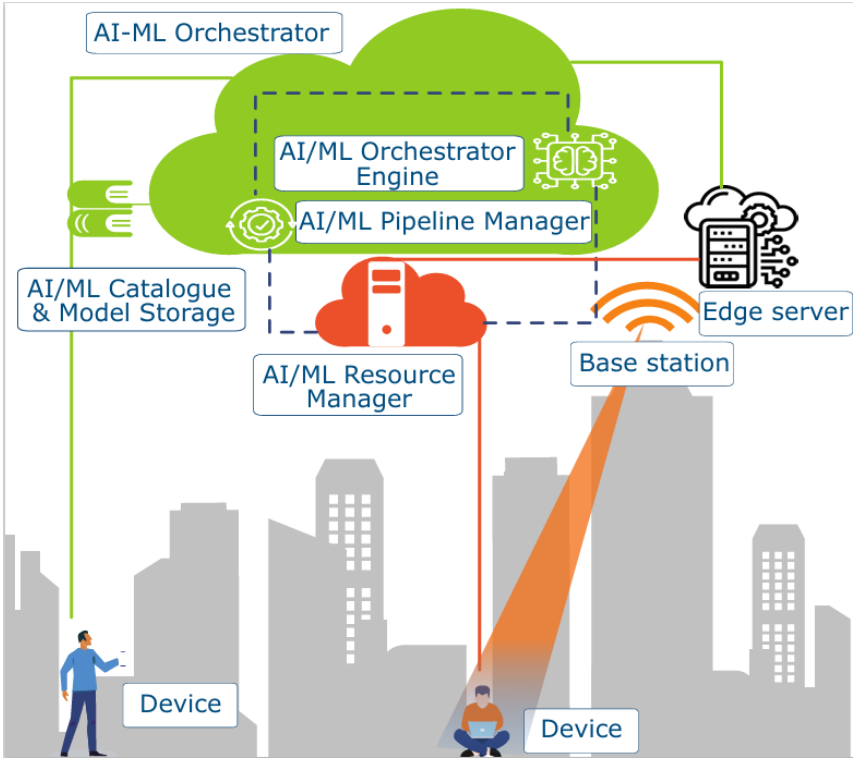


Figure 4.17: Reference scenario.

With reference to the CrowdSourcing AI/ML architecture shown in Figure 3.3, the *AI/ML Orchestrator Engine* is in charge of making decisions regarding whether to execute inference locally at the device or to offload it to the edge. By means of its *AI/ML Model Selection* component, it also identifies the most suitable trained model instance, pruned or not, as described in the *AI/ML Catalogue* and stored in the *AI/ML model storage.* The selected model has to meet the capabilities and needs of the requesting device. For this purpose, the AI/ML Orchestrator Engine (i) receives from the *ML Resource Manager* information on the network, computational, and energy resource capabilities of the nodes, i.e., the end device and the edge server, and (ii) provides the *AI/ML Pipeline Manager* with the selected model and with instructions to deliver to the AI/ML Resource Manager about the node where the AI/ML tasks should be pruned (if this operation is deemed necessary) and executed as well as the resources to be used.

We consider the *application-level* message exchange in Figure 4.18 in the two cases where the AI/ML Orchestrator Engine decides that the inference is to be executed at the edge server (left image) or at the mobile device (right image).

- *Inference at the edge.* The inference request together with the data input, represented by a sample of data, is transmitted from the device to the edge server where the model is deployed. The edge server then executes inference and sends the resulting output back to the mobile device. If the device requests further inference tasks of the same type, the new sample data is transmitted to the edge server that executes inference by leveraging the available ML model and sends results back to the device.
- *Inference at the device.* The device sends a model request to perform a given task. The edge server performs the necessary operations as per the instructions received by the AI/ML Resource Manager (i.e., model pruning) and then sends the model to the device, where inference is executed locally. Upon receiving the DNN model, the device locally stores it and uses it for all subsequent required inference tasks of the same type. Thus, in this case, successive task requests only require local inference execution by leveraging on the same (pruned) model and no further interaction with the edge server is needed.

The aim of this initial study is to derive experimental results on the inference costs in terms of latency and energy consumption of the two above ML model deployments. These results will be leveraged for future work in order to design a decision algorithm to be implemented in the AI/ML Orchestrator Engine and Model Selection components regarding whether it is more convenient to execute inference at the device or at the edge, and using which model version (i.e., full-size or pruned model).

4.7.3 Preliminary results

Latency on the radio channel is computed by simulating the sub-6 GHz channel between the end device and the Base Station by using the 3GPP Urban Micro (UMi) street canyon modeling in a MATLAB environment, with a 5G New Radio Base Station operating at 3.5 GHz. For inference and pruning times computation, we conducted real experiments. A virtual machine (VM), an Azure Standard D4ds v5 instance with 4 virtual CPUs, has been configured as an edge server, running on an Intel(R) Xeon(R) Platinum 8370 C physical processor with 16 GB of RAM. The VM works at full CPU power. We simulate various edge load conditions with up to 100 active Docker containers active in the VM for inference tasks. The mobile device has one physical core, equipped with an Intel Core i7-9750H processor, 4 GB of RAM, and 30 GB of disk space.

We focused on image recognition inference tasks, considering different DNN models (ResNet-152, DenseNet201, and MobileNet) and different pruning factors, with the ImageNet-1000 dataset used for model training. The input image size is 300 kB and the output size is 2 kB. The technique of magnitude weight pruning, which removes neural connections associated with the least relevant weights, was applied using Tensorflow Model Garden, Tensorflow Model Optimization, and Keras Surgeon.

Our results showed that *executing inference at the end-device may lead to 60% faster decision-making compared to execution at a highly loaded edge*.

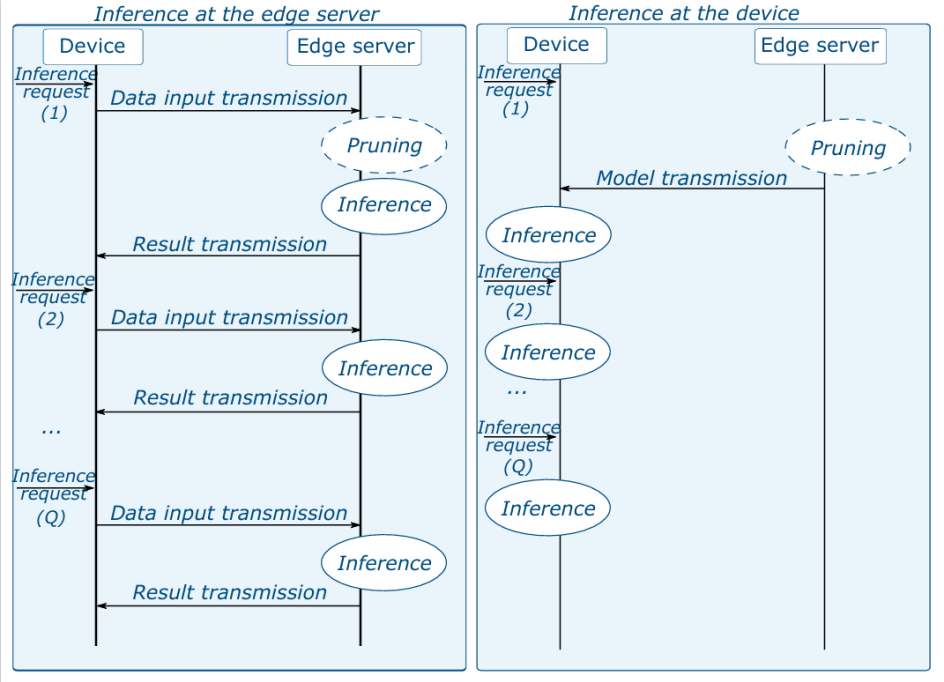


Figure 4.18: Application-level message exchange for inference execution at an edge server (left) and at a mobile device (right).

Figure 4.19 shows the average time for obtaining an inference result at the end device once the request has been issued, for the cases of inference executed at the edge (curves labelled as *Edge*), and at the device (curves labelled as *Device*), and when inference is executed through (i) the full model, (ii) a pre-pruned model stored at the edge server, or (iii) a model pruned at edge on-demand, i.e., upon the edge receives a request from a device. The higher the edge load and model complexity, the longer the inference time at the edge. Notably, for all models, performing inference at the device (blue curves) is faster than running models at the edge (red curves) beyond a given edge load, with such load value getting smaller as a lighter model is considered.

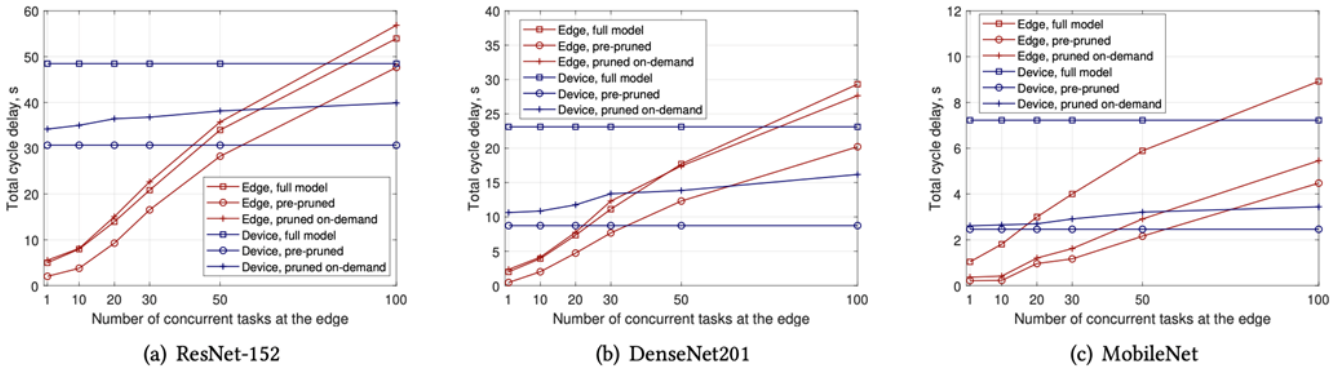


Figure 4.19: Total delay for the two possible model deployments, full and pruned models (pruning factor 70%), and variable edge server load, for DNN model: (a) ResNet-152, (b) DenseNet201, (c) MobileNet.

Another notable result is that *executing a pruned model provides the device with time and energy savings up to 40% and 53%, respectively, w.r.t. the unpruned model*. More precisely, the energy gain is in the order of 1.5, 2.7, and 5.4 for ResNet-152, DenseNet201, and MobileNet, respectively (curves not shown here but available in [55]). The number of inference executions (Q) has clearly an impact on the energy drain. Considering common battery capacity values for mobile and embedded devices (3.7 Wh for embedded devices, 1.1 Wh for smartphones, and 18.5 Wh for PCs), MobileNet Q may be higher than 100, before the device runs out of battery.

© ADROIT6G, 2023

As a final comment, regarding model accuracy, we have run experiments to assess the impact of the pruning factor on the accuracy metric and demonstrated that the accuracy of the considered DNN models remains stable for pruning factors less than or equal to 70%, thus indicating no need for fine tuning. The results shown refer to a pruning factor equal to 70%. Further work is planned in which the effect of fine tuning will be included for higher values of pruning factors when the model accuracy degrades visibly. Fine-tuning involves additional training of the pruned model, allowing the remaining weights to adapt to information previously overlooked during pruning. Importantly, fine-tuning may require the use of private data, which raises issues in terms of privacy preservation. We expect that, despite the benefits of increased accuracy, the cost of fine-tuning must be assessed carefully to wisely distribute fine-tuning and inference operations between edge server and end-devices.

4.8 Scheduling and Offloading Techniques Across Multiple Domains

Intelligent Computational Offloading (ICO) is an effective method to improve the computational and energy performance of Mobile-Edge Computing (MEC) networks. By deploying MEC servers at the edge of radio access networks, such as cellular base stations, the edge devices can offload resource intensive computational tasks to nearby Edge Servers to reduce energy and latency costs. Compared to the naive scheme of offloading all tasks to the edge, ICO can dynamically assign tasks to be computed either locally (on the edge devices) or at the Edge Server. The ICO has shown significant performance improvements under varying network conditions, such as under fluctuating energy levels, task dependent input-output delicacies and edge caching availability, among others [56], [57], [58]. Typically, this process involves solving a Mixed Integer Non-Linear Programming (MINLP) problem that jointly takes the binary offloading and resource allocation decisions. Due to the computational complexity, especially in large networks, many studies have focused on developing algorithms with reduced complexity, such as local-search based heuristics, decomposition-oriented search and convex relaxations of the binary variables [59], [60], etc. These algorithms result in computational latency, as they require many numerical iterations to produce a satisfactory solution. Furthermore, the MINLP must be frequently resolved when system parameters, such as wireless link quality, change. Therefore, implementing conventional optimization algorithms in a highly dynamic MEC environment can be suboptimal, making it necessary to find more efficient solutions.

4.8.1 ADROIT6G specific enhancements over state of the art

The ICO uses the data driven solutions, such as DRL provide a promising alternative solution that uses DNNs to learn the optimal mapping from the "state" (system parameters) to the "action" (resource allocation and offloading decisions) to maximize the "reward" (energy efficiency and latency) by learning from the environment modelled in the context of ICO. This approach eliminates the need for the complicated computation of MINLP and is particularly advantageous for online implementation. The existing DRL based methods are either policy based, or value based. Commonly used value based DRL methods include DQN [61], [62], [63] and its variants such as Double DQN (DDQN). However, the DQN-based methods are suboptimal when there are exponential number of offloading decisions to be taken for the edge devices. The policy-based approach such as deep deterministic policy gradient methods [64], [65] and actor-critic DRL [66], [67], [68] can provide an alternative solution that can generate an optimal mapping policy from input states to the output actions using the DNN. [68] and [64] use two learning modules to allocate the resources in a sequential manner. In [65], the edge devices make the discrete offloading decision without considering the power requirements. In [68], a smaller number of binary offloading decisions are generated that are fed into a model-based critic module. This module then selects the best decision by analytically solving the optimal resource allocation problem. The DRL-based methods have the limitation of performing under long-term performance constraints e.g., average power requirement and queue stability. These long-term performance constraints can be achieved by applying Lyapunov optimization to design an online offloading strategy as studied in [69], [70], [71]. The Lyapunov optimization converts the multistage stochastic problem to a frame wise deterministic subproblem. However, it still needs to solve a hard MINLP problem at each substage of the subproblem for jointly optimizing the resource allocation and offloading decisions. In this work, we investigate a multi-user MEC network, which involves computation task data being

randomly and sequentially received by the edge devices. Our objective is to create an ICO algorithm that can make decisions for each time frame, without relying on knowledge of the future. We aim to optimize the network's data processing capacity while maintaining the long-term stability of the data queues and adhering to average power limitations. To address this challenge, we propose an approach, which combines the advantages of Lyapunov optimization and DRL. This method enables real-time, optimal decisions in response to rapid changes in channel fading and fluctuating task arrivals, ensuring the long-term stability of the system.

Overall, the proposed solution contributes to the realization of CrowdSourcing AI by providing an efficient and scalable framework for offloading computation tasks to far edge devices in MEC systems, thereby leveraging the collective intelligence and computing capabilities of a crowd of distributed devices for AI applications. In particular:

- Energy Efficient Resource Allocation: The optimization algorithms developed in the proposed solution help in allocating resources under power constraints in MEC systems. This is crucial for CrowdSourcing AI as it involves distributing computational tasks among various edge devices according to their energy levels.
- Adaptability and Scalability: Lyapunov-guided deep reinforcement learning allows the system to adapt to changing conditions and scale effectively as the number of edge devices or computational tasks varies. This adaptability and scalability are important characteristics of CrowdSourcing AI systems, which often involve many distributed computing nodes and dynamic workloads.
- Optimization of Wireless Resource Allocation: The solution considers the optimization of wireless resource allocation, which is critical for CrowdSourcing AI applications operating over wireless networks. By efficiently managing wireless resources, the proposed solution can improve the overall performance and reliability of AI tasks executed in a crowdsourced manner.
- Real-Time Decision Making: The use of deep reinforcement learning enables real-time decision-making in offloading computation tasks to edge devices. This is essential for CrowdSourcing AI applications that require timely processing of data and responses to user requests.

4.8.2 Functional architecture

This subsection provides a comprehensive overview of the algorithm's implementation stages and how it interacts within the ADROIT6G infrastructure, highlighting the synergy between distributed AI and the framework's architecture. With reference to the AI/ML architecture shown in Figure 3.3, the ICO solution is implemented and run at the *AI/ML Orchestrator*. The AI/ML Orchestrator is responsible for deciding whether to execute the task locally at the device or offload it to the edge. It also identifies the capabilities and needs of the requesting device. For this purpose, the *AI/ML Orchestrator* receives from the *AI/ML Resource Orchestrator* information on the network, computational and energy resource capabilities of the end device and edge server. The *AI/ML Resource Orchestrator* insights allow the *AI/ML Orchestrator* to make informed decisions on offloading decisions, focusing on optimizing the energy efficiency and computational resources.

Figure 4.20 represents the architecture of the Online Computation Offloading. It provides an online binary offloading strategy to perform computation tasks with *non-partitionable datasets*. The proposed offloading solutions are then fed to a model-based optimization module, which optimizes the allocation of communication and computation resources and provides an accurate estimate of the reward value for each potential offloading decision. The different modules of the Online Computation Offloading are explained below:

- Input of DNN: This block represents the input to the DNN, which includes the current channel gains (h_t), the data queue length (Q_t), and the energy queue length (E_t), These inputs encapsulate the current state of the system.
- Actor Module:
  - DNN (μ_{th}): The deep learning model that outputs the action to be taken based on the current state of the system. The DNN uses weights (*theta*), that are trained to make decisions.
  - NOP Action Quantization: This is a block that quantizes the No-Operation (NOP) action, which is a choice the system can make to not offload computation in a certain time step.

- Critic Module:
  - DNN (Q_{phi}): Another deep learning model that estimates the quality, or (Q_{value}) of the current action taken by the actor module based on the system state. This model uses weights (*phi*) during training.
  - Compute optimal resource allocation (*P**): Based on the input state and the action taken, this block computes the optimal resource allocation.
- Training samples: Samples used to train the DNN models of the actor and critic modules. These samples are generated from the system's experience by *interacting* with the environment.
- Replay Memory: A database of past experiences, where each experience is a tuple of the state, action, reward, and next state. It is used for training the DNNs in the actor and critic modules to improve decision-making.
- Q Policy Update Module: Using the replay memory, this module updates the policy, which determines how the actor selects actions. It's about refining the policy to improve performance based on the outcomes of past actions.
- Execute the joint action: After determining the joint action (x_t,y_t) this block represents the execution of the chosen action in the environment, which affects the data processing and energy consumption of the system.
- Update Queues: Update the state of the queues (Q_{t+1}, y_{t+1}) after the action is executed, so the new lengths of queues are determined.

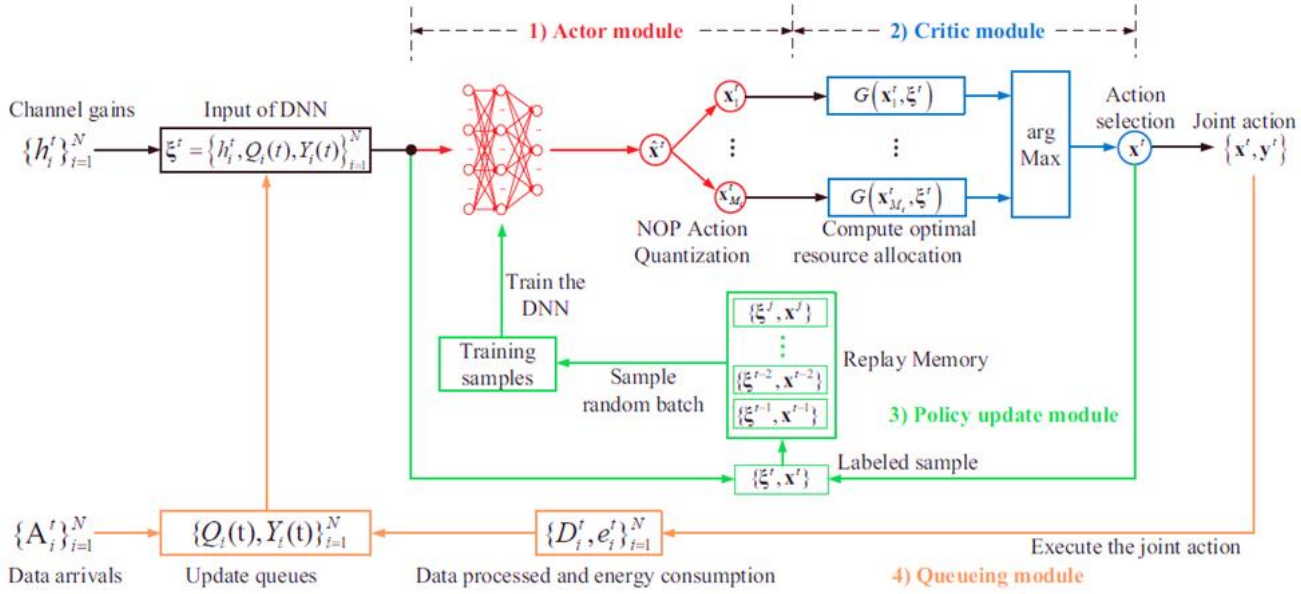


Figure 4.20: Online Computation Offloading Architecture [72]

Necessary Inputs/Outputs:

Inputs:

- Channel gains (h_t): This represents the condition of the channel at time which impacts the communication rate for offloading data.
- Data queue lengths (Q_t): This denotes the amount of data queued at each user device that is waiting to be processed.
- Energy queue lengths (E_t): This indicates the available energy for each user device at time t, which will affect the ability of the device to perform computation and communication tasks.

These inputs (ht, Qt, Et): represent the state of the system at time t and are provided to the Input of DNN block within the actor module. The DNN uses this information to output an initial set of offloading decisions, which may then be further processed (such as through the NOP Action Quantization block) to yield the final offloading actions for the system.

Outputs:

The outputs for the diagram are as follows:

- Action quantization (x_t, y_t): This is the immediate output from the actor module after the NOP action quantization, providing the decision on whether the computation should be offloaded and the quantized level of computation that should be offloaded at time t.
- Computational action (x_{*t}): This represents the optimal action chosen by the critic module after evaluating the action quantization values; it essentially determines the final offloading decision.
- Joint action: The combination of the computational action (x_{*t}) from the critic module and the quantization (x_t, y_t) from the actor module. This denotes the definitive actions executed by the system, encompassing both computation offloading and resource allocation decisions.
- Updated queue states (Q_{t+1}, $E_{t+1)}$: Following the execution of the joint actions, the queues are updated to reflect the new states of the data and energy queues after processing and energy consumption have occurred.

5 Data Generation Pipeline

A vital part of the AI/ML pipeline is data management, as data are essential for both ML model training and inference tasks execution. This Section specifically focuses on the Data Generation Pipeline, whose goal is to ensure that any AI/ML model involved in the ADROIT6G architecture, or any end-user application can continue operating with the required quality in the cases where transmitted data is lost or received with an unsatisfactory level of corruption.

Different, a.k.a. multimodal, data sources producing, e.g., images, text, audio, and sensor data, by their very nature, offer a richer, more comprehensive datasets for AI/ML applications, providing nuanced insights that would not be possible with unimodal data. However, while collecting and transferring data samples from multimodal data sources to the entities that use them for training or inference is of critical importance, this richness comes with its own set of challenges. Firstly, the transfer of such may take a high toll on the system resources in terms of bandwidth and energy consumption. Secondly, there is a potential loss or corruption of modalities during data transmission. This issue is not just a technical inconvenience; it fundamentally compromises the integrity and efficacy of AI/ML models, rendering them less reliable or even unusable in critical applications.

Recent approaches have therefore emerged to generate synthetic data at the processing entities with a distribution that resembles that of real-world data. Such data samples can then be used for ML model training and inference tasks, instead of real-world data. Importantly, the ability to compensate for lost modalities through synthetic data generation ensures that AI/ML models within the CrowdSourcing AI framework can maintain high levels of accuracy and reliability, even in less-than-ideal transmission conditions. This capability is especially crucial in applications where data integrity directly influences safety, operational efficiency, or critical decision-making processes. Moreover, by addressing the issue of data loss proactively, we enhance the resilience of the CrowdSourcing AI framework, making the system more robust to the inherent unpredictability of real-world data transmission environments. This section thus focuses on such innovative aspects, highlighting solutions for generation of synthetic multimodal data that can effectively and efficiently address the above challenges.

The rest of the section is organized as follows:

- Section 5.1 presents an innovative technique for multimodal data generation using a diffusion model, which aims at both providing high-quality synthetic data and maintaining coherence across different data modalities;
- Section 5.2 investigates accelerated numerical integration schemes for stochastic differential equations so as to improve the efficiency of the models used for data generation and enable their use in resource-constrained scenarios;
- Section 5.3 extends the developed techniques for multimodal data generation to multi-view settings.

5.1 Multimodal Generative Modelling and Data Generation

ADROIT6G places a significant emphasis on the generation of synthetic data to address the critical issue of modalities that may be lost or corrupted during transmission and to reduce the bandwidth cost of data transfer from wireless devices/nodes to processing entities. This strategic focus is rooted in the understanding that in real-world applications the completeness and integrity of multimodal data are paramount. The loss of a single modality can severely impact the performance of AI/ML models, leading to inaccurate analyses, flawed decision-making, or outright system failures.

5.1.1 ADROIT6G specific enhancements over state of the art

ADROIT6G's approach to tackling the mentioned challenge involves the deployment of an advanced data generation framework that leverages the latest in generative modelling techniques [73]. This framework is

specifically designed to reconstruct missing or corrupted modalities, thus preserving the functionality and reliability of AI/ML models in the face of data transmission challenges.

The methodology applied to multimodal generative modelling introduces a revolutionary approach by leveraging diffusion models to navigate the prevalent coherence-quality trade-off observed in existing ML methods. Traditional strategies often struggle to strike a balance between generating high-quality data and maintaining coherence across different data modalities. The employment of multi-modal diffusion models allows this approach to bypass the limitations faced by previous models, enabling the generation of data that is unparalleled in quality and exhibits remarkable coherence across modalities.

Diffusion models play a key role in this innovative process, synthesizing data that accurately captures the intricate relationships between various modalities. This method surpasses its competitors by producing data that is unmatched in quality and coherence. This method contributes significantly to the robustness of systems reliant on multimodal sensor data collection, which may otherwise be prone to unreliability. This enhancement ensures that learning models can more effectively interpret and utilize data from diverse sources, improving both their accuracy and their adaptability in the considered use cases. Additionally, thanks to the proposed technique, it is now possible to enhance the capabilities of other learning algorithms by supplying them with larger datasets.

The process works according to the following three steps:

1. **Detection of Lost Modalities:** The system continuously monitors the integrity of incoming multimodal data, quickly identifying instances where certain modalities are missing or have been compromised during transmission.
2. **Cross-generation with Generative Models:** Upon detecting a loss or corruption, the system activates its generative models, which are pre-trained on extensive datasets encompassing a wide range of scenarios and modalities. These models are adept at generating synthetic data that closely mimics the original characteristics of the lost modalities.
3. **Integration of Synthetic Data:** The generated synthetic data is then seamlessly integrated into the data, replacing the missing or corrupted modalities. This ensures that the data remains complete and coherent, allowing AI/ML models or any end-user application to proceed with their analyses as if no data loss had occurred.

Currently, a specific multimodal transmission scheme does not exist capable of reconstructing in a semantically meaningful way the missing modalities in case of transmission loss. The proposed Multi-modal Latent Diffusion (MLD) framework allows instead this, guaranteeing excellent results in terms of both coherency and generation quality.

The MLD framework introduces a new approach to generative modelling across multiple data modalities, addressing and overcoming the longstanding challenge of the coherence-quality trade-off faced by existing multi-modal Variational Autoencoders (VAEs). MLD distinguishes itself through its ability to generate high-quality, coherent data across different modalities, as demonstrated through rigorous experimental evaluation on various real-life datasets.

Traditional multi-modal generative models have struggled to balance the generation of high-quality data with maintaining coherence across modalities. This balance is critical in applications where data from different sources or formats need to be synthesized or augmented in a consistent manner. The MLD approach revolutionizes this space by employing a set of independently trained unimodal deterministic autoencoders that encode data into a common latent space. This space is then utilized by a masked diffusion model to enable generative modelling. This innovative strategy sidesteps the pitfalls of latent variable collapse and information loss that commonly plague multi-modal VAEs, enabling MLD to maintain high fidelity and coherence in generated data.

5.1.2 Functional architecture

The proposed solution introduces a new cross-modal generative model, which integrates seamlessly into the

functional architecture aimed at guaranteeing operationality of the network even in case of missing or corrupted data. The MLD framework introduces an innovative approach to address the coherence-quality trade-off in multi-modal generative modelling [73]. Leveraging independently trained uni-modal deterministic autoencoders combined with a novel diffusion process in latent space, MLD achieves unparalleled performance in generating coherent and high-quality data across multiple modalities. The main building blocks are presented in Figure 5.1 and the architecture leverages the following key innovations:

- **Integration of Uni-modal Autoencoders:** These components independently process each data modality, guaranteeing a precise and comprehensive representation. This ensures that the unique characteristics of each modality are accurately captured and reflected in the generated synthetic data, as depicted in Figure 5.1.
- **Masked Diffusion Process:** A sophisticated enhancement to the generative process, allowing for the creation of synthetic data that maintains high coherence and quality across modalities. These processes are further described by Equations (6) and (7), enhancing the generation process.

$$\mathrm{d}R_t = m(A_1) \odot [\alpha(t) R_t \mathrm{d}t + g(t)\mathrm{d}W_t], \quad q(r,0) = q_\psi(r^{A_1} \mid z^{A_2})\delta(r^{A_2} - z^{A_2}).$$

$$\mathrm{d}R_t = m(A_1) \odot \left[\left(-\alpha(T-t)R_t + g^2(T-t)\nabla \log\left(q(R_t, T-t \mid z^{A_2})\right)\right)\mathrm{d}t + g(T-t)\mathrm{d}W_t\right]$$

- **Multi-time Training Technique:** Significantly improves the generative capabilities of the system. The Score network architecture, illustrated in Figure 5.2, is adept at handling both clean and corrupted modalities, providing accurate noise estimation for each.

To fully leverage the potential of the MLD framework within the AI/ML architecture, it is crucial to understand the specific inputs required to drive the system and the outputs it produces. This understanding ensures seamless integration and operational efficiency across the network, particularly in scenarios dealing with incomplete or corrupted data sets.

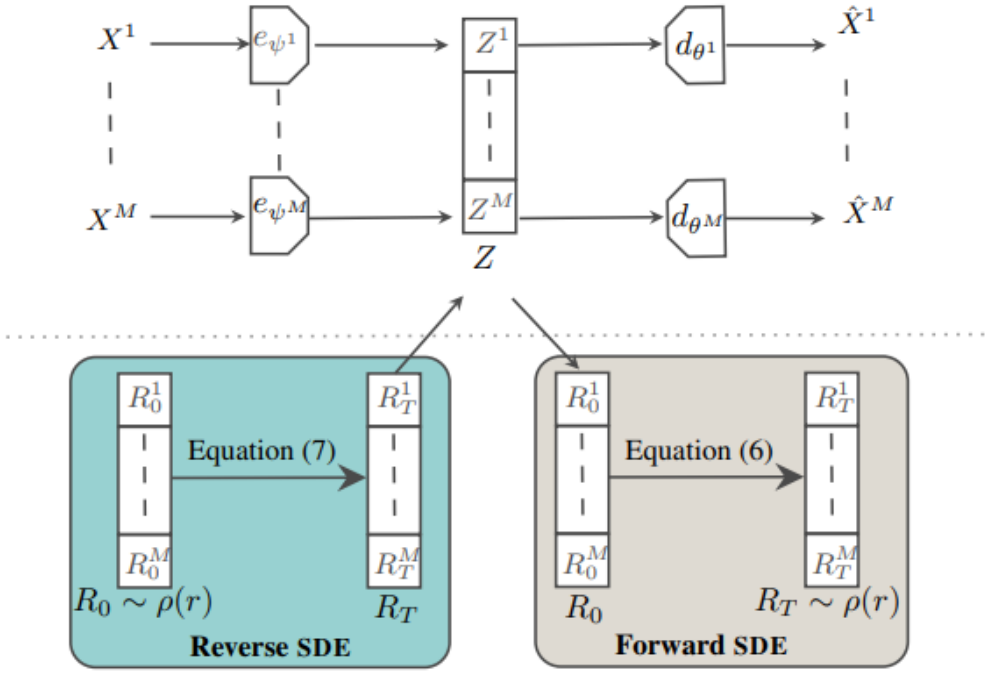


Figure 5.1: Overview of the MLD method: list of unimodal autoencoders and joint diffusion model

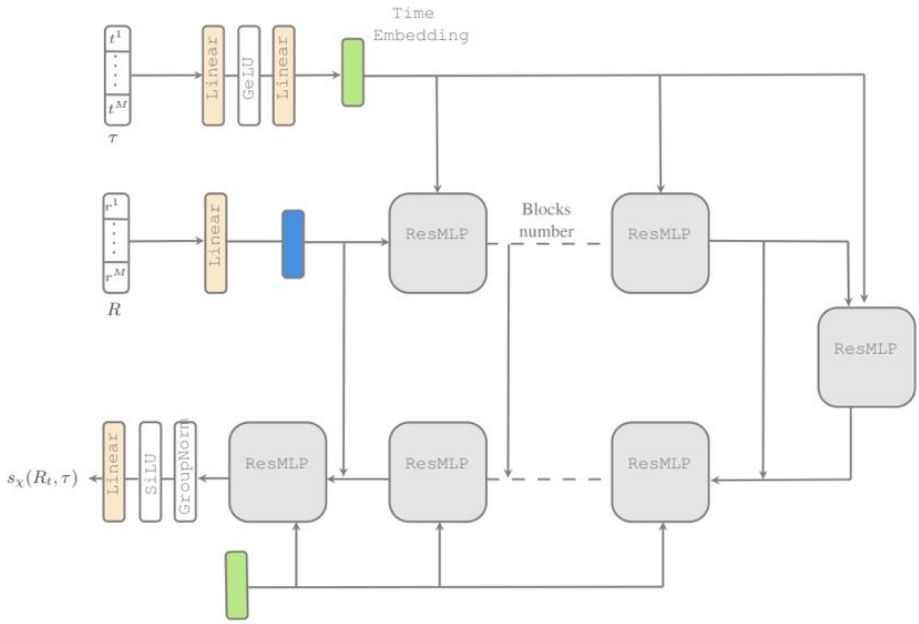


Figure 5.2: Implementation details about the Score Network architecture.

Required input for the MLD framework.

- **List of Uni-modal Autoencoder Models:** The foundation of the MLD framework is built upon a suite of uni-modal autoencoders, each tailored to a specific data modality. These models are meticulously designed to capture the intrinsic characteristics of individual modalities, such as text, images, sounds, or sensor data, with high fidelity. Obtaining these models from a centralized model repository ensures that they are consistently updated and optimized based on the latest research and performance feedback. This repository acts not just as a storage unit but as a dynamic library that evolves with the project's needs.
- **Joint Diffusion Model:** Central to the MLD's innovative approach is the joint diffusion model, a sophisticated algorithm that orchestrates the blending of multiple modalities in the latent space. This model, also sourced from the model repository, employs advanced diffusion techniques to gradually introduce and control noise within the data, thereby enabling the generation of new, synthetic instances that maintain coherence across modalities. The model's parameters and structure are the result of extensive research and experimentation, designed to maximize generative quality and inter-modal coherence.
- **List of Available Modalities at the Transmitter and Receiver:** The effectiveness of the MLD framework is partially contingent on the availability and variety of modalities it can process. A comprehensive list of currently accessible modalities is essential for determining the scope and direction of data generation processes. This list not only informs the system of the types of data at its disposal but also allows for the strategic planning of data enhancement and synthesis to fill in gaps or augment the dataset as necessary.

Provided outputs by MLD framework.

- **Concatenated, Compressed Latent Code at the Transmitting Edge:** One of the key outputs of the MLD framework is a concatenated and compressed latent code, representing a dense, information-rich encoding of all available modalities. This code is the culmination of the framework's processing, encapsulating the essence of the multimodal dataset in a form that is both efficient for transmission and ready for further manipulation or generation tasks. The creation of this latent code is a testament to the framework's ability to distil complex, multimodal information into a unified representation, facilitating easier and more effective data handling and transmission.

- **Regenerated Joint Latent Code and Corresponding Reconstructed Missing Modalities at the Receiving Edge:** Upon reaching the receiving edge, the MLD framework delivers two critical pieces of output. Firstly, the joint latent code, regenerated from the transmitted condensed code, serves as the foundation for reconstructing the original multimodal dataset or generating new instances thereof. Secondly, the framework provides the reconstructed versions of any missing modalities, seamlessly filling in gaps in the dataset with synthetically generated data that closely mirrors the quality and coherence of the original. This dual-output mechanism ensures that the receiving end has access to a complete and coherent set of modalities, ready for analysis, learning, or other application-specific processes.

Through the careful design and implementation of its input and output protocols, the MLD framework offers a robust and flexible solution for managing and enhancing multimodal data within complex AI/ML architectures. By addressing the unique challenges posed by multimodal datasets, particularly in terms of maintaining coherence and quality across diverse types of data, the MLD framework stands as a significant advancement in the field of generative modelling.

Integration with the architecture. In the ADROIT6G architecture framework, the MLD model introduces a substantial enhancement in multimodal semantic communications, primarily engaging at a strategic layer closely tied to end-user applications. This integration of MLD showcases the model's seamless fit into ADROIT6G's ecosystem, spotlighting its crucial role in attaining semantic goals through targeted communications.

MLD's design — encompassing encoding, latent diffusion, and decoding processes for multimodal data — aligns well with the architecture's upper layers, where it directly contributes to achieving semantic communication objectives. This alignment ensures MLD's contributions are deeply integrated with the logic and data specific to user applications, a departure from models that function at more foundational network layers. By situating MLD in this context, it leverages semantic goals, such as achieving coherence at the class level, to directly influence communication based on defined semantic aims.

The process of integrating MLD also highlights the critical role of developing application-tailored components, including encoders, decoders, and the generative model itself, all implemented in PyTorch. Tailoring these components to specific data sets and pre-training them before deployment ensures their effectiveness across various scenarios, reflecting the model's adaptability.

As described previously, the MLD process involves three fundamental steps: 1) Detection of Lost Modalities, 2) Cross-generation with Generative Models, 3) Integration of Synthetic Data. The process is generic and applies to a variety of different applications, each with its own specific models. We proceed to describe the interaction with the key elements of the ADROIT6G architecture (reported schematically in Figure 5.3).

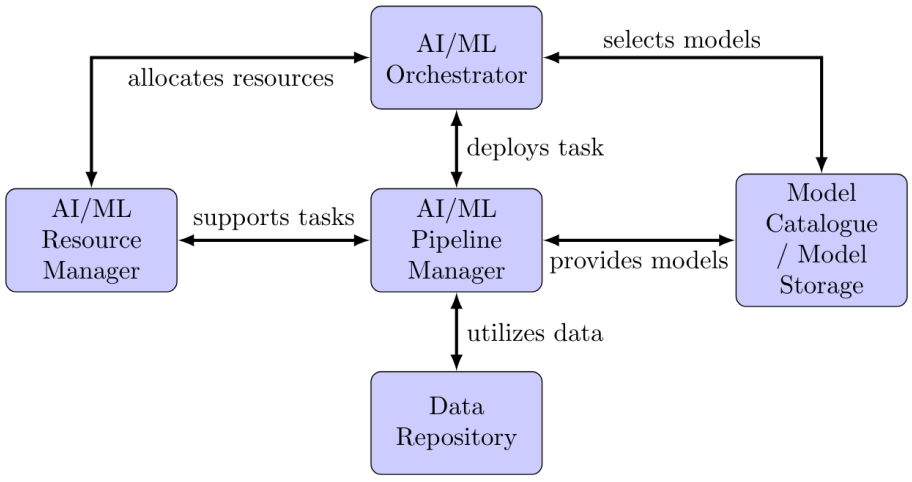


Figure 5.3: Interaction across different blocks for the Data Generation Pipeline

- **AI/ML Orchestrator**: takes the decision to cross-generate missing modalities whenever a modality loss is detected. In this case, it needs to understand which modality needs regeneration. Then, it decides from the list of available models 1) which joint diffusion model should be adopted for the cross generation of the missing portion of the latent code corresponding to the missing modality, and 2) which decoders should be selected to map back to the original data space the cross generated latent code. Moreover, the Orchestrator decides which nodes and resources should be used for performing the cross-generation task.
- **AI/ML Pipeline Manager**: it is instructed by the Orchestrator to deploy the cross-generation task of interest.
- **AI/ML Resource Manager:** It collects system node information for the actual available computational resources for the cross-generation task.
- **Model Catalogue/Model storage**: contain, respectively, the list of available models and their actual storage. In this specific scenario, the models listed in the catalogue (and stored) are the i) uni-modal encoders /decoders, and ii) the joint diffusion models.
- **Data Repository**: contains multi-modal datasets of interest, which can be used to fine-tune existing multi-modal models or train new ones.

In the scope of WP5 all these components will be deployed within the ORO's laboratory facility implementing a private cellular 5network. In this regard, ORO is working on the infrastructure preparation needed for the integration of the multimodal model application developed by EURE into the laboratory testbed, updated with 6G features and components from ADROIT that will be used for autonomously scaling of network resources depending on the load. This way, three surveillance cameras were placed in the laboratory and connected to the private network through Customer-Premises Equipment (CPEs) specifically configured for this task. Initial footage from the three cameras was recorded by ORO and shared with the partners from EURE for the training and calibration activities associated with the multimodal model application development.

Furthermore, ORO prepared the virtualized infrastructure that will host EURE's AI/ML application by integrating in its OpenStack cluster two new servers fitted with Nvidia Tesla T4 GPUs. These hardware resources will be accessed by EURE through a VM deployed on top of the OpenStack cluster, through a secured access procedure, via an IPSEC VPN tunnel, for which the connectivity parameters and security rules were already defined by ORO. On top of the given VM, EURE are going to deploy their inference algorithm, which will be based on the resulting pre-trained ML model and will consume the live video feeds coming from the surveillance cameras. The laboratory piloting phase of the solution is expected to start in the second half of 2024 followed by the analysis of its benefits in terms of resource's optimization.

5.1.3 Preliminary results

The efficacy of MLD is empirically validated across several benchmarks, where it consistently outperforms existing state-of-the-art models in both joint and conditional generation tasks. In particular, the experiments showcase MLD's superior performance in scenarios involving loosely correlated modalities, a traditionally challenging area for multi-modal generative models.

The results are supported by quantitative metrics, such as Frechet Inception Distance (FID) and coherence percentages, alongside qualitative assessments through visual inspections of generated samples. As an example, Table 5.1, Table 5.2, and Table 5.3 present a comprehensive comparison of MLD's performance against other models, highlighting its superior generative quality and coherence across multiple datasets, including MNIST-SVHN and MHD, among others.

In the context of the ADROIT6G architecture, models trained on standard ML datasets such are crucial for demonstrating how these technologies can be adapted to 'real-world' scenarios specific to the ADROIT framework. These models, initially developed and validated on diverse and representative datasets, are well-suited for integration into the ADROIT6G's modular architecture (see in particular Figure 5.3), which is designed

to handle various data modalities like images from surveillance cameras, audio recordings, and semantic text descriptions.

The flexibility of these pre-trained models allows them to be fine-tuned and optimized for specific applications within the ADROIT6G ecosystem, such as enhancing image recognition accuracy in video surveillance or improving the relevance of audio processing in urban environments. This approach not only accelerates the deployment of advanced ML solutions across different nodes of the network but also ensures that the solutions are robust, scalable, and directly aligned with the low-latency data processing needs of the ADROIT6G architecture.

Table 5.1: Performance of MLD against competitors on the MNIST-SVHN dataset

Models	Coherence (%↑)			Quality (↓)			
	Joint	M → S	S → M	Joint(M)	Joint(S)	M → S	S → M
MVAE	38.19	48.21	28.57	13.34	68.9	68.0	13.66
MMVAE	37.82	11.72	67.55	25.89	146.82	393.33	53.37
MOPOE	39.93	12.27	68.82	20.11	129.2	373.73	43.34
NEXUS	40.0	16.68	70.67	13.84	98.13	281.28	53.41
MVTCAE	48.78	81.97	49.78	12.98	**52.92**	69.48	13.55
MMVAE+	17.64	13.23	29.69	26.60	121.77	240.90	35.11
MMVAE+(K=10)	41.59	55.3	56.41	19.05	67.13	75.9	18.16
MLD (ours)	**85.22**	**83.79**	**79.13**	**3.93**	56.36	**57.2**	**3.67**

Table 5.2: Performance of MLD against competitors on the MHD dataset

Models	Joint	I (Image)			T (Trajectory)			S (Sound)		
		T	S	T,S	I	S	I,S	I	T	I,T
MVAE	37.77	11.68	26.46	28.4	95.55	26.66	96.58	58.87	10.76	58.16
MMVAE	34.78	**99.7**	69.69	84.74	99.3	85.46	92.39	49.95	50.14	50.17
MOPOE	48.84	99.64	68.67	99.69	99.28	87.42	99.35	50.73	51.5	56.97
NEXUS	26.56	94.58	83.1	95.27	88.51	76.82	93.27	70.06	75.84	89.48
MVTCAE	42.28	99.54	72.05	99.63	99.22	72.03	99.39	92.58	93.07	94.78
MMVAE+	41.67	98.05	84.16	91.88±	97.47	81.16	89.31	64.34	65.42	64.88
MMVAE+(k=10)	42.60	99.44	**89.75**	94.7	99.44	**89.58**	95.01	87.15	87.99	87.57
MLD (ours)	**98.34**	99.45	88.91	**99.88**	**99.58**	88.92	**99.91**	**97.63**	**97.7**	**98.01**

Table 5.3: Performance of MLD against competitors on the CelebA-HQ dataset

Models	Attributes		Image				Mask	
	Img + Mask F1	Img F1	Att + Mask FID	Mask FID	Att FID	Joint FID	Img+Att F1	Img F1
Wesego & Rooshenas (2023)								
SBM-RAE	0.62	0.6	84.9	86.4	85.6	84.2	0.83	0.82
SBM-RAE-C	0.66	0.64	83.6	82.8	83.1	84.2	0.83	0.82
SBM-VAE	0.62	0.58	81.6	81.9	78.7	79.1	0.83	0.83
SBM-VAE-C	0.69	0.66	82.4	81.7	76.3	79.1	0.84	0.84
MOPOE	0.68	**0.71**	114.9	101.1	186.8	164.8	0.85	**0.92**
MVTCAE	0.71	0.69	94	84.2	87.2	162.2	**0.89**	0.89
MMVAE+	0.64	0.61	133	97.3	153	103.7	0.82	0.89
Supervised classifier		0.79					0.94	
MLD (ours)	**0.72**	0.69	**52.75**	**51.73**	**53.09**	**54.27**	0.87	0.87

5.2 Integration Algorithms to Speed Up Model and Data Generation

In the context of the ADROIT6G project, the exploration into accelerated numerical integration schemes for Stochastic Differential Equations (SDEs) constitutes an important component of advancing efficiency in generative modeling. This extensive investigation, underpinned by a rigorous examination of contemporary advancements in the domain, plays a crucial role in enhancing the comprehension and utilization of these mathematical constructs within AI/ML frameworks. The assimilation of these schemes significantly augments the project's capability to accurately simulate complex systems and phenomena, an essential factor for the crafting of advanced generative models.

Advanced Numerical Integration Schemes. SDEs are integral to the mathematical modeling of systems' evolution under the influence of stochastic processes. Within the domain of generative models, especially those predicated on diffusion processes, SDEs articulate the temporal evolution of data distributions under stochastic influences. The challenge resides in the precise and efficient simulation of these equations, necessitating the adoption of sophisticated numerical integration schemes. The investigation encompasses an array of state-of-the-art numerical methods devised to address the challenges inherent in the integration of SDEs. Classical schemes for the numerical integration of SDEs are based either on strong or weak integration schemes [74]. In data generation, strong schemes should be preferred to ensure faithfulness and quality of the generated data. The AI community has investigated several schemes tailored specifically for the simulation of backward diffusion processes, see e.g., [75],[76]. Others investigated the possibility of directly reducing the amount of time spent in the backward diffusion by truncating the diffusion to smaller times [77],[78]. Notice that the different approaches can be combined.

5.2.1 ADROIT6G specific enhancements over state of the art

A significant inflection point in this inquiry is the acknowledgment of the efficacious nature of operations within latent space [79], [73]. Generative models, as employed in the ADROIT6G project [73][74], predominantly operate in a latent space wherein data is encapsulated in a more condensed, abstract format. This paradigm shift bears profound implications for the application of numerical integration schemes. Operations in latent space effectively simplify the dynamics to be modeled, reducing computational complexity and, by extension, mitigating the requirement for highly intricate integration schemes. This realization is in harmony with the principles delineated in the referenced materials, where the emphasis on latent space dynamics engenders a more streamlined simulation process, leveraging the reduced dimensionality and the intrinsic structure of latent representations.

The transition towards operations in latent space heralds several key implications. Primarily, it denotes a shift towards a more resource-efficient modeling strategy, imperative for the scalability of AI/ML solutions within the ADROIT6G initiative. This strategic pivot is not merely about achieving operational efficiency but also about harnessing the full potential of latent space to enhance the fidelity and utility of generative models. The inherent efficiency of operating within this condensed data realm allows for the streamlining of processes that would otherwise require significantly more computational resources. This efficiency is crucial in environments where processing power and memory are at a premium, making the ADROIT6G project's approach highly advantageous.

Furthermore, this shift accentuates the criticality of selecting numerical integration schemes that are congruent with the operational characteristics of the generative models in development. The design and deployment of these models, in conjunction with the joint effort of other tasks, highlights the pivotal role of collaboration in optimizing latent space operations. The concerted effort to integrate models across different domains without the necessity for more complex or computationally demanding integrators underscores a key project insight: efficiency is paramount. This result showcases how strategic cooperation between modules and a focused approach to model design within latent spaces can substantially boost efficiency without the incessant search for more efficient integrators. This alignment with the project's dual objective: efficient transmission of latent codes and ensuring the lean operation of AI/ML solutions.

© ADROIT6G, 2023

5.3 Multi-view Extension

As networks and systems increasingly rely on multimodal data from diverse sources, it becomes imperative to address scenarios where these sources provide different perspectives of the same physical field, each contributing unique and complementary information. Such scenarios are prevalent in complex environments where the integration of multiple viewpoints can significantly enhance the comprehensiveness and utility of the data analysis. However, these systems often encounter issues such as incomplete data due to sensor failures or physical obstructions, which can severely impact the accuracy and reliability of subsequent analyses.

This section introduces the Multi-view Latent Diffusion (MVLD) model, an innovative extension of the previously established MLD framework.

5.3.1 ADROIT6G specific enhancements over state of the art

A relevant scenario is represented by a network system where different data sources can have a different perspective of a physical field and, thus, they can capture data of the same field but featuring different aspects, hence carrying different information. The previous MLD approach has therefore been extended to address such a relevant scenario.

For concreteness, we focus on an application specific use case addressing the reconstruction of missing views of video streams collected by multiple cameras. Our work builds on the work introduced in [80], where a MVLD model is proposed.

MVLD introduces a pioneering approach to leverage multi-view observations, offering a comprehensive understanding of real-world phenomena by enabling the conditional generation of any missing view at inference time. This methodology significantly enhances the practical deployment of multi-view data systems, addressing the common issue of incomplete data due to sensor failures or obstructions. Similarly to MLD, MVLD's methodology integrates a deterministic autoencoder with a score-based diffusion model. This combination effectively imputes missing views, enriching data analysis and decision-making processes. The framework's potential extends to communication systems, proposing a novel application for efficient multi-view images transmission, thus optimizing network load and image quality. The implementation scheme is depicted in Figure 5.4.

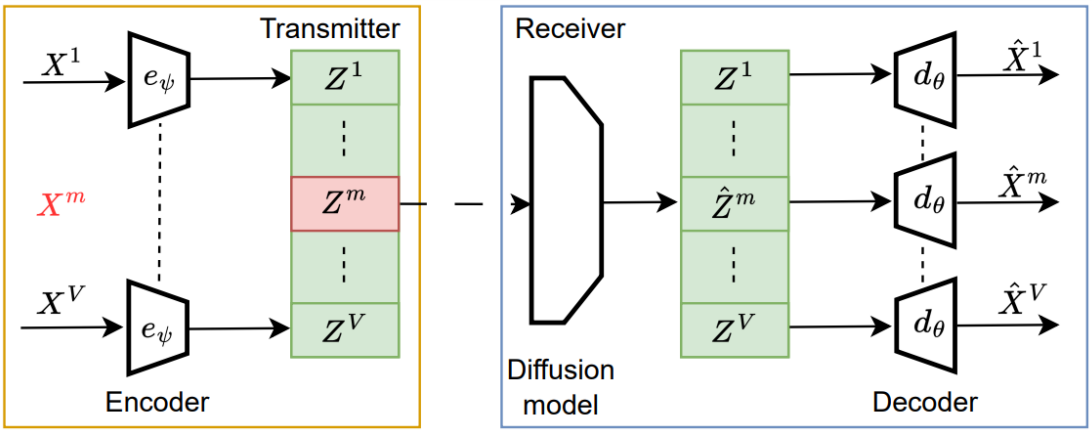


Figure 5.4: implementation scheme of the multi-view extension of the MLD framework.

5.3.2 Functional architecture

The proposed solution introduces a new cross-view generative model, leveraging the results of [73],[80]. The main building blocks are presented in Figure 5.4 and the architecture leverages the following key innovations:

- **Integration of a single Uni-modal Autoencoder:** a single autoencoder which can process input views from arbitrary perspectives. This amortization with a single ML model allows efficiency in training and in terms of model storage.
- **Masked Diffusion Process:** A joint diffusion model which is trained to recover noised version of the latent views given an arbitrary number of reference views.

In particular, MVLD has the following set of inputs and outputs:

- **List of arbitrary views autoencoder models:** These models are obtained from the centralized model repository and can be tailored to the specific scenario of interest (e.g., urban vs rural scenario).
- **Joint Diffusion Model**: This object allows to recover noiseless version of the corrupted modalities.
- **List of Available views at the Transmitter and Receiver sides:** A comprehensive list of currently accessible views at the transmitter (some views maybe missing due to sensor fault) and at the receiver (some views maybe lost during transmission).
- **Concatenated (transmitted) joint latent code:** it is the result of the concatenation of the different latent codes corresponding to the available views at the transmitting side.
- **Reconstructed latent code:** it is the output of the cross-generative diffusion model at the receiver side, which reconstruct any number of missing modalities.
- **Reconstructed views:** it is the decoding output of the single view decoders for the single portion of the reconstructed latent codes.

Integration with the Architecture. Envisioning MVLD in a communication context, it offers an innovative solution for transmitting multi-view data over wireless channels. Whenever some of the views are lost during transmission, MVLD can reconstruct missing ones at the receiver, conserving channel capacity. Current efforts are focused on extending the MVLD model's innovative methodology to the domain of video streams, particularly those collected from multiple cameras. This initiative aims to harness MVLD's capabilities for conditional generation and imputation of missing views in dynamic, real-world phenomena captured on video. The key challenge lies in optimizing the architecture of the deterministic autoencoder and the score-based diffusion model to ensure scalability and efficiency. This optimization is crucial for the methodology's application in real-time video analysis and transmission, promising significant advancements in multimedia content generation, analysis, and semantic communication strategies.

We proceed to describe how MVLD interacts with the key components of the ADROIT6G architecture (the schematic interaction is similar to the one reported in Figure 5.3):

- **AI/ML Orchestrator:** it serves the purpose of triggering missing views cross-generation tasks. Upon decision of which AI model to be used, it informs AI/ML Pipeline Manager to deploy the cross-generation task of interest;
- **AI/ML Resource Manager:** its role is to maintain an updated view of the available computational resources which can be used for the encoding/decoding of the single views and the cross-generation tasks;
- **Model Catalogue/Model storage:** contain respectively the list of available models and their model weights storage. In this scenario, the models listed in the catalogue are the i) single view autoencoders and ii) the joint diffusion models.

To effectively integrate the Multi-view Latent Diffusion (MVLD) model within the ADROIT6G architecture, the next step will involve deploying the model on real static images. This preliminary application will serve as a critical testbed to refine the model's capabilities in handling real-world data. By starting with static images, the project can methodically assess and enhance the model's performance in reconstructing and processing multimodal data. This step is essential to ensure that the MVLD framework is robust, efficient, and fully capable of supporting the sophisticated demands of the ADROIT6G architecture in subsequent phases.

© ADROIT6G, 2023

6 Service Orchestration and Management

AI/ML approaches are at the forefront of transforming telecommunications through enhanced network services, offering automation and intelligent decision-making capabilities, significantly improving network operations, service delivery, and user experience. This section thus presents a set of relevant AI/ML-based network services that can benefit of the AI/ML pipeline implemented through the CrowdSourcing AI framework, and solutions that can make the implementation and use of AI/ML models for network services more efficient as well as more robust to harsh network conditions due to radio channel variability and users' mobility. Relevant examples of AI/ML-based network services include: (i) Traffic prediction, where algorithms analyze historical and real-time data to forecast network traffic patterns, facilitating proactive network resource allocation and minimizing congestion; (ii) RAN configuration, where AI/ML enables dynamic adjustment of network parameters, optimizing performance to adapt to changing demands and conditions, thus improving coverage and capacity; (iii) Path computation, where AI/ML optimizes routing decisions based on network status, latency, and congestion, ensuring efficient data transmission; (iv) Traffic scheduling, which is intelligently managed by AI/ML algorithms, prioritizing critical data flows and managing bandwidth allocation, crucial for maintaining QoS and QoE in networks supporting diverse applications.

In the rest of this section, we present algorithmic solutions for the efficient implementation and management of the above AI/ML-based services within the framework that we have designed. Specifically,

- Section 6.1 exploits ML to provide robust solutions for joint communication and computational resource allocation satisfying performance and energy constraints;
- Section 6.2 proposes an approach leveraging the collective intelligence of BDIx agents, informed by precise localization information, to improve network management in 6G networks;
- Section 6.3 describes an energy-efficient traffic prediction method leveraging on RNN-BLSTM model;
- Section 6.4 focuses on optimal D2D topologies formation through BDIx Agents and Minimum-Weight Spanning Tree ML algorithms;
- Section 6.5 exploits ML to enhance scaling and migration of any given service function in the ADROIT6G infrastructure;
- Finally, Section 6.6 deals with the management of containerized AI/ML functions, tackling the problem of migrating such functions whenever AI/ML-based network services are instrumental to provide continuous time-critical services to mobile end users such as Unmanned Aerial Vehicles (UAVs), connected autonomous vehicles, and connected mobile robots.

6.1 Robust Algorithms for Joint Resource Management satisfying Performance and Energy Constraints

6G systems will be intelligent in various performance and management areas, including the RAN network slicing and resource allocation procedures [81]. In addition, 6G needs to serve a huge number of users requiring various service types and applications. The support of the required use cases and services can be significantly demanding and will add more complexity, layers, and domains, in addition to the technologies the network already has [82]. To this end, in ADROIT, we develop AI-based algorithms for joint communication and computational resource allocation that satisfy both performance and energy constraints.

6.1.1 ADROIT6G specific enhancements over state of the art

Although there are research works in the literature that target to address the network-slicing challenges with heuristic and genetic algorithms [83], [84], [85], [86] in 5G, Reinforcement Learning (RL) is a key part of beyond 5G and 6G. RL and DRL-based techniques evolve network management and resource allocation in the RAN sector [87]. The highly dynamic nature of 6G networks, specifically the RAN section, can remarkably benefit from an RL-based intelligent decision-making procedure [81]. RL has been considered as a promising approach for dynamic

resource management in a complex environment such as 6G networks, mainly due to the fact that RL is a model- and algorithm-deficient approach [88].

However, allocating bandwidth resource blocks is not enough for the realization of all types of services, and RAN communication links and entities, grouping and clustering, and other RAN topology specifications are affecting. The authors in [89] propose both communication and computation resource slicing for Ultra-Reliable and Low-Latency Communications (URLLC) users in an O-RAN-based system model. The proposed approach is also applied in two levels of decision-making and deploys a DDQN algorithm; however, only for one service type is considered. The network-management approach presented in [88] considers that the mobile network operator can change the weights of the RL reward function so that the agent adopts each service-type and fulfills the agreed QoS for that slice type. This work uses Transfer Learning (TL) in order to address the agent's convergence challenges after the modifications. Since we relate to the service realization, part of our management approach as a set of conflicting service types, using a shared set of resources in an isolated way, we try to benefit from Multi-Objective Reinforcement Learning (MORL) algorithms to improve our work.

Consequently, MORL-based approaches have gained interest with applications in wireless network fields. The approach presented in [90] deploys MORL to address the challenge of offloading an application consisting of dependent tasks in MEC. In addition, the authors in [91] introduce a MORL framework for load balancing and benefits from meta-RL to learn a general policy that can adapt to new trade-offs between objectives. Finally, the authors in [92] uses MORL with two objectives for resource allocation and energy consumption in Cloud-RANs (C-RAN).

In ADROIT6G, we go beyond the state of the art, by using MORL agent to assign communication links, resource blocks, and network entities to each user, asking for a specific service type. This service-type realization creates slices and clusters network entities for each request. The proposed approach targets joint communication and computational resource allocation, with significant low computational complexity. Considering service types as multiple contradictory objectives to be optimized simultaneously, we deploy a MORL agent that can be trained for all the objectives and can be used for any preferred service type. While by handling the trade-off between different service types the MORL agent improves the consistency compared to having multiple single agents for each service type, it also significantly reduces the number of agents, neural networks, and training processes.

6.1.2 Functional Architecture

System model and proposed MORL-based algorithm

The proposed approach not only considers low-level agents that manage the computational resource assignment, but also the resources of the communication links. This means that in this work, each agent is able to consider effective metrics and KPIs for each service type (received from the service orchestration/management) and take a sequence of actions specific to that slice type (at the slice orcherstration / management level). By deploying that, one agent can implement computational resource allocation, RAN resource allocation, and DU assignment simultaneously. As a result, the need to have at least one single agent for each service type is eliminated, and the number of agents only depends on the number of tenants the network is serving. In order to achieve that, this work implements a MORL agent instead of a group of single RL agents. This agent tries to learn policies over multiple objectives that are competing. The agent does not know the relative importance of each objective in the training phase. After the initial learning, it can execute optimal policies for any chosen objective.

In MORL, the Markov Decision Process (MOMDP) includes a vector reward function (instead of one single value), a space of preferences (objectives or service types), and preference functions. If the space of preferences only consists of one value, MOMDP converts to the standard MDP. The improved low-level agent considers the DUs and the Radio Units (RUs) as the nodes, and the interfaces between them as the edges. The system is modeled as a binary tree with a depth equal to the sum of the number of DUs and RUs. In each level, choosing the right subtree means the chosen path includes the node in that level, and choosing the left subtree means this RU or DU is not included in the considered path. Each final node has an array of reward values for the parameters that

we consider for each service type or objective. This array includes one item for each service type. This value is formed in a way that important metrics for that service type are included. When there is a request for a service type, a path including all the levels that is the best choice for that service type is assigned.

The considered system architecture considers a set of RUs at the radio-edge, which are connected to the DUs through fronthaul links. Thus, in the edge part, the system is represented as a binary tree of the RUs, DUs, and the connecting fronthaul links. Figure 6.1 shows the system model represented in this work. All the final nodes can be represented as the binary form of numbers {0, 1,…, 2^n}, with *n* being the depth of the tree. The binary form of the number consists of *n* digits, each representing one entity. Since groups of RUs are served by each DU, the higher-valued positions (starting from the left) are mapped to the DUs. Each position can be {0,1}, meaning the chosen path includes that DU or not. The reminded spaces represent the RUs. The 2^n-th level, is represented by a vector of rewards, each element showing the value of the path for each parameter or objective. To explain more, each service type is an objective for the agent. The objectives are represented by parameters that show the value of each path regarding that service type. Thus, the depth of *n* represents 2^n arrays each consisting of *m* elements, where *m* is equal to the number of the supported service types.

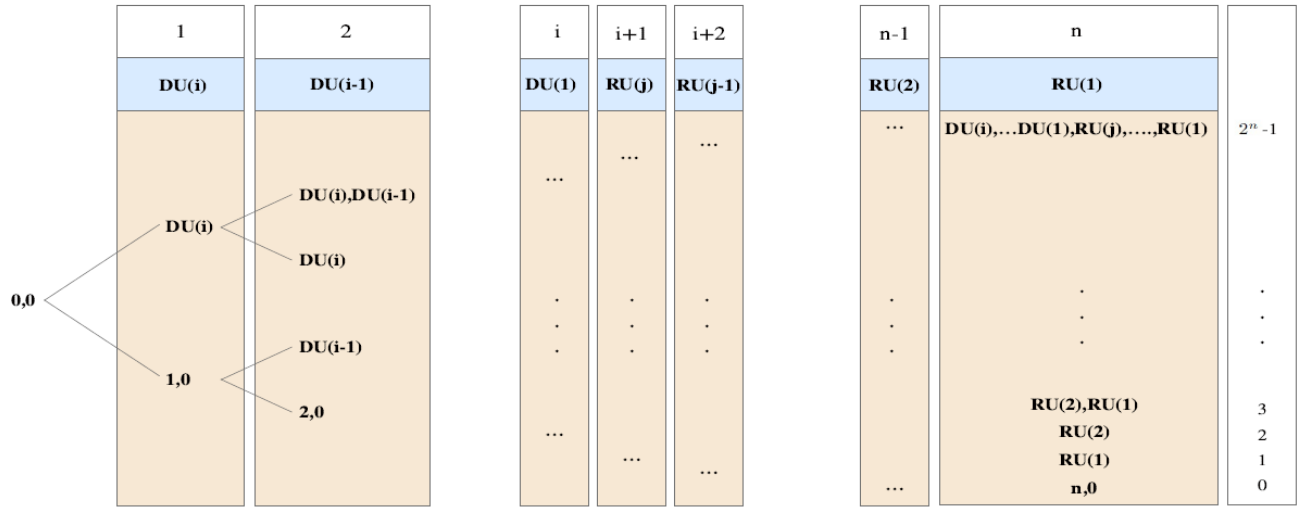


Figure 6.1. Binary tree representation of the resource orchestration system model in Low-part.

The agent in this work uses the Enveloped Q-learning proposed in [93], which can learn a single policy network being the optimized space of preferences. The novelty of the algorithm is having vectorized value functions. This means that instead of using only one preference when updating the value function, utilizes a convex envelope of the solution frontier to update parameters. Thus, by generalizing the Bellman equation, the agent can learn a single parametric representation of the optimal policies for all the preferences. When deploying this algorithm in our environment, in the training phase, the relative importance of the service types is unknown to the agent, and thus, it learns optimal policies over the space of all service types. After the learning phase, the agent will execute the optimal policy for any chosen service type.

One of the main reasons for deploying two levels of management, assignment, and time scales in is to distribute the complexity. In addition, the MARL algorithm plays a crucial role in the system's scalability. To continue, the MORL algorithm in this work was chosen considering its scalability, sample efficiency, and policy adaptation. Our system model in this work is represented as a binary tree with a depth of 8, which can be considered a large state space comparatively. The state space can grow with a greater number of network elements. The optimal policy set (i.e., all the available paths in the system) also grows when having more nodes and paths. In addition to that, the number of objectives is directly related to the number of the considered service types and will grow if more slice types are added to the services. According to [84], the employed algorithm in this work can address the scalability issues related to large state space and objective space better than Multi-Objective Fitted Q-Iteration (MOFQI) and Conditional Neural Network with Optimistic Linear Support (CN+OLS), respectively.

Moreover, the envelope version used in this work outperforms the Scalarized Q-update with Double Q-learning (Scalarized version) in larger state spaces and has a better sample efficiency.

Thus, while the improved approach inherits all the positive points in the scalability context (e.g., has two levels of time and performance and scalable MARL algorithm), It improves the scalability and computational complexity by deploying a smaller number of agents and neural networks. It is worth mentioning that instead of training all the single RL agents, only one training process is needed for the MORL agent.

In the next period, we aim at finalizing the proposed algorithm and develop an evaluation procedure to demonstrate its performance.

6.2 Traffic and Energy-Efficient BDIx agent Configurations

The configuration of Belief-Desire-Intention extended with ML (BDIx) agents plays a critical role in optimizing energy and power consumption within 6G networks. These agents, by virtue of their sophisticated AI-driven framework, are designed to autonomously make decisions based on the current state of the network, their programmed goals (desires), and the available strategies (intentions) to achieve these goals. The configuration of BDIx agents directly influences their ability to efficiently manage network resources, thereby significantly impacting the network's overall energy and power usage. The BDIx agents are configured with specific beliefs about the network's state, including data on traffic patterns, node connectivity, and energy usage metrics. These beliefs are continuously updated with real-time information, enabling the agents to make informed decisions that reflect the current network conditions. The desires of these agents are aligned with the overarching goals of minimizing energy consumption while maintaining or enhancing network performance and reliability. The intentions, which are actionable plans derived from a combination of fuzzy logic rules and ML predictions, guide the agents in selecting the most energy-efficient actions [94], [95], [96], [97].

The impact of BDIx agent configuration on energy and power consumption in 6G networks is profound. By optimizing the decision-making process of these agents, networks can achieve several energy-efficient outcomes:

- **Dynamic Resource Allocation:** BDIx agents can dynamically adjust the allocation of network resources, such as bandwidth and power, to match the real-time demand. This prevents over-provisioning and underutilization, which are common sources of energy waste in telecommunications networks.
- **Intelligent Traffic Management:** Through the analysis of traffic data and patterns, BDIx agents can predict and mitigate potential bottlenecks or congestion in the network. This ensures data is routed through the most efficient paths, reducing the energy required for data transmission and processing.
- **Network Infrastructure Optimization:** BDIx agents can control the operational status of network components, such as base stations and relay nodes. By intelligently switching off or reducing the operational capacity of underutilized components, significant energy savings can be realized without compromising the network's quality of service.
- **Self-organizing Networks**: The autonomous nature of BDIx agents supports the development of self-organizing networks that can adapt to changes in user behavior, device mobility, and traffic demand. This adaptability minimizes manual reconfiguration and ensures that the network always operates at peak energy efficiency.
- **Predictive Maintenance and Operation:** Leveraging ML capabilities, BDIx agents can forecast future network conditions and preemptively adjust network settings to avoid energy-intensive operations. This predictive approach to network management not only conserves energy but also enhances the overall user experience by ensuring high-quality connectivity.

The configuration of BDIx agents, particularly their ability to integrate and act upon complex datasets using ML and fuzzy logic, embodies a significant advancement in the pursuit of energy and power efficiency in 6G networks. By enabling a more intelligent, responsive, and adaptive network infrastructure, the strategic

deployment of these agents supports the dual goals of reducing the environmental impact of telecommunications and meeting the growing demand for high-performance mobile connectivity.

6.2.1 ADROIT6G specific enhancements over state of the art

A novel ML-based Distributed AI (DAI) framework that aims to achieve the ambitious goals of integrating AI and ML into 6G telecommunications networks is created as DAI Framework. The DAI framework stands out due to its independent, adaptable, and versatile execution, utilising BDI agents that are empowered with ML capabilities, referred to as BDIx agents, which are located within mobile devices or UEs. This novel framework employs a component-based methodology, similar to constructing LEGO blocks, allowing for the creation and utilisation of execution plans by flexibly combining and arranging ML algorithms. The achievement of modularity is facilitated by a Multi-Agent System, where each BDIx agent is connected with Fuzzy Logic to improve the agents' sensory and cognitive capabilities. BDIx agents provide autonomous intercommunication and cooperation among mobile devices, enhancing the network with multiple advantageous features. These encompass enhanced performance metrics such as the time it takes to execute network control and the efficiency of message exchange, quick response capabilities for dynamic network changes, self-organizing network functionalities, and the ability to serve as a fundamental platform for implementing diverse intelligent approaches to meet the complex technical requirements of 5G/6G networks [94],[95],[96],[97].

The BDIx agents are working under Device to Device (D2D) communication as a key application area to showcase the versatility of the DAI framework in tackling various D2D difficulties. The study explores the implementation of Data Analytics and Intelligence by using example Plan Libraries and improved metrics (as shown in the Figure 6.2). Its objective is to meet the defined needs for D2D communication in 6G networks. In addition, this text thoroughly examines the problem of D2D transmission mode selection, starting from defining the problem and continuing with the solution technique called DAIS, as well as the technical details of its implementation. This investigation involves a comparative assessment of DAIS in relation to different procedures, such as unsupervised ML techniques, centralised control techniques, and random selection methods. The results demonstrate that DAIS not only improves the efficiency of mobile network spectral utilisation and reduces power consumption, but also facilitates more efficient cluster formation and greatly minimises delays in control decision-making. The accomplishments are made possible by the Traffic and Energy-Efficient BDIx agent Configurations, which are described in JSON format and backed by an Adaptive Neuro-Fuzzy Inference System (ANFIS) pre-trained model. The DAI framework has the potential to greatly transform 5G/6G network operations by providing a strong, intelligent, and highly flexible solution for the future of mobile telecommunications [94], [95], [96], [97].

Focusing exclusively on the realm of 6G, the proposed DAI framework, leveraging ML-enhanced BDIx agents, presents a significant advancement in addressing the complex and dynamic requirements of 6G networks. This comparison with the current state-of-the-art technologies and methodologies reveals the innovative edge and potential of the DAI framework in driving forward the evolution of 6G telecommunications.

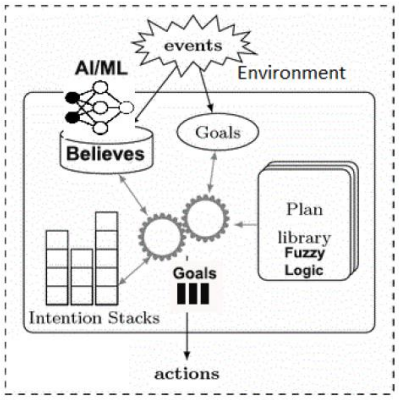


Figure 6.2: BDIx architecture.

Comparison with Current SoTA in 6G

The SoTA in 6G network technologies is characterized by an emphasis on URLLC, enhanced mobile broadband, and massive machine-type communications, alongside new paradigms such as the integration of AI for network optimization, the use of sub-terahertz (THz) frequencies for communication, and the exploration of novel network architectures like cell-free massive Multiple Input Multiple Output (MIMO). The list below outlines the current state of the art for BDIx agents:

Innovative Edge of DAI Framework:

- **Autonomy and Flexibility:** Unlike traditional network management solutions that may rely on centralized control mechanisms, the DAI framework introduces a high degree of autonomy and flexibility. BDIx agents residing on mobile devices can make decentralized decisions, allowing for real-time optimization of D2D communications.
- **Dynamic Network Adaptation:** The framework's use of ML within BDI agents enables dynamic adaptation to changing network conditions, user behaviours, and traffic patterns. This capability goes beyond the static or semi-static network configurations solutions, offering a more responsive and efficient network management approach.
- **Enhanced Spectral and Energy Efficiency:** By focusing on improved mobile network spectral efficiency and power consumption, the DAI framework addresses two critical aspects of 6G networks that are paramount for supporting the anticipated increase in data traffic and the need for sustainable network operations. While SoTA technologies aim to improve these metrics, the DAI framework's novel approach to autonomous, real-time optimization provides a potentially more effective solution.

Challenges and Opportunities related to the ADROIT6G:

- **Integration with Existing and Emerging Technologies:** Integrating the DAI framework with other ADRPOIT6G technologies, such as Crowd Sourcing AI, UE-VBS Computing, etc., poses challenges but also offers opportunities to enhance network performance further. The modular nature of the DAI framework suggests potential synergies with these technologies, enabling more comprehensive network optimization strategies.
- **Scalability and Deployment:** While the DAI framework demonstrates significant potential, its scalability and practical deployment in large-scale 6G networks will require thorough validation. This includes assessing the framework's performance in diverse environments and its compatibility with ADROIT6G 6G architecture.

In conclusion, the proposed DAI framework introduces a pioneering approach to network management in 6G telecommunications, emphasizing the strategic use of any AI and ML that can improve the BDIx agent's decision within a Distributed Artificial Intelligent (DAI) BDIx agent-based architecture. By comparing its features and capabilities with current SoTA solutions, it's evident that the DAI framework offers a promising avenue for addressing the intricate demands of future 6G networks. However, realizing its full potential will necessitate addressing integration, scalability, and deployment challenges, paving the way for next-generation telecommunications solutions that are both intelligent and sustainable.

The proposed DAI architecture, which incorporates ML-enhanced BDIx agents onto mobile devices, is a key step towards achieving CrowdSourcing AI within the context of 6G networks. In this context, CrowdSourcing AI utilises the collective intelligence and computational resources of various UEs in the network to dynamically and efficiently optimise network functionalities. This method utilises the capabilities of distributed computing and artificial intelligence to tackle intricate network issues, specifically in improving energy efficiency and optimising traffic management. The proposed solution actively helps to the achievement of CrowdSourcing AI at the following:

- **Distributed Intelligence:** The proposed method involves incorporating BDIx agents into mobile devices, which enables the decentralisation of the decision-making process. This allows for the distribution of

intelligence across the network. Every gadget, equipped with artificial intelligence capabilities, can examine data within its immediate vicinity, make determinations, and operate independently to enhance its network interactions. The idea of CrowdSourcing AI is exemplified by this decentralised form of intelligence, in which solutions arise from the combined decisions and actions of numerous agents throughout the network.

- **Dynamic Network Optimisation:** The combined data and knowledge produced by individual BDIx agents contribute to a thorough comprehension of the network's condition, encompassing traffic patterns, node connectivity, and energy consumption. The utilisation of crowdsourced information allows the network to adaptively modify its configuration and allocate resources in order to enhance efficiency and performance. Employing CrowdSourcing AI offers a crucial advantage by enhancing the network's responsiveness to changing conditions through the ability of BDIx agents to learn and adjust their methods based on real-time data.
- **Energy and Power Consumption:** An essential component of the proposed approach is its emphasis on optimising energy and power usage throughout the network. By integrating the ANFIS paradigm and strategically configuring BDIx agents, the network can achieve substantial energy savings. The bots collaborate using fuzzy logic and ML predictions to precisely optimise network operations, minimising energy waste while ensuring optimal service levels. The utilisation of CrowdSourcing AI in energy management exemplifies its capacity to address a significant and urgent issue in contemporary telecoms.

The proposed DAI system utilises the combined capabilities of BDIx agents to efficiently manage and optimise network traffic and formation (as shown in Figure 6.3). The system utilises CrowdSourcing to control traffic at the network's edge, where data is both generated and consumed. This approach guarantees more precise and detailed traffic forecasts and decisions.

The utilisation of CrowdSourcing AI in addressing intricate network optimisation challenges results in enhanced spectral efficiency, decreased latency, and superior overall network performance, hence demonstrating its practical advantages. The suggested system possesses inherent scalability and flexibility due to its modular and distributed architecture. This allows for optimal deployment across networks of different sizes and complexities. The ability to adapt is essential for addressing the varied and changing demands of 6G networks, demonstrating the effectiveness of CrowdSourcing AI as a resilient and flexible framework for managing next-generation networks.

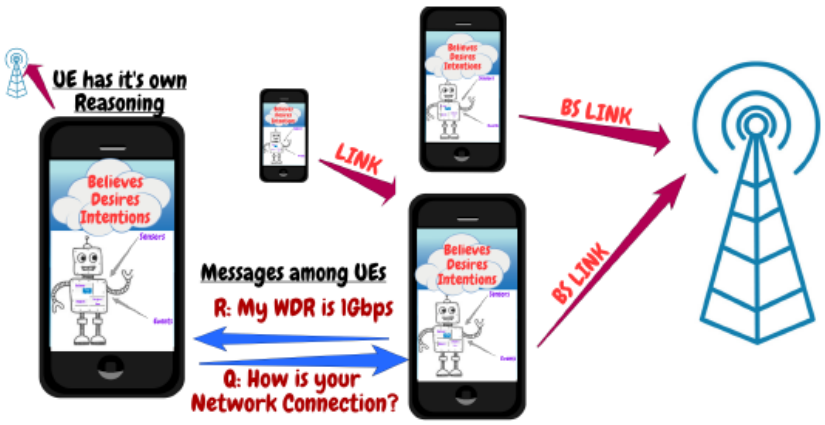


Figure 6.3: A D2D Communication Network Example Using BDIx Agents.

6.2.2 Functional architecture

In the context of a functional architecture aimed at enhancing Resource Orchestration and Management within 6G networks, the proposed solution introduces a unique component designated for generating configurations

for the BDIx agents and interfacing with the Localisation module that holds the tree structures of the D2D communications networks of each gNodeB under the ADROIT6G (as shown in Figure 6.3). This component is pivotal for the dynamic adaptation and optimization of the network based on real-time conditions. The description of how this component integrates within the ADROIT6G architecture, along with its specified inputs and outputs, is provided below. With reference to the AI/ML architecture shown in Figure 3.1.

- This unique component will be placed within the "Resource Orchestration and Management" component and act as a central intelligence hub. It is responsible for configuring the BDIx agents, which are distributed across the network and embedded within user equipment or network nodes. The component leverages data from the Localization module to inform its decisions, ensuring that the configurations it generates for BDIx agents are optimized based on the current network topology and conditions.
- This component has a direct connection to the Localization module (as described in Section 4.2.2), allowing it to receive up-to-date information regarding the positions and movements of nodes within the network. This data is crucial for tailoring the BDIx agents' configurations to the network's dynamic environment, facilitating efficient resource management and network optimization.
- This component includes ML Models stored in the AI/ML Model storage that helps the BDIx agent to decide locally.
- This component includes Policies that the telecom provider will set for the Panning library to be configure. These policies can be stored as meta-data in the AI/ML repository.
- This component is sharing its services with other components using REST services

Inputs needed:

- **Real-time Localization Data:** Information on the current positions, movements, and connectivity of network nodes provided by the Localisation module. This data is essential for understanding the network's physical topology and dynamics, which influence the optimal configuration of BDIx agents.
- **Network Performance Logs:** Historical and real-time performance data collected from the BDIx agents. This includes metrics related to network efficiency, resource utilization, traffic patterns, and any observed challenges or bottlenecks within the network.

Outputs provided:

- **BDIx Agent Configurations:** Optimized configuration parameters for each BDIx agent, generated based on the analysis of real-time localization data and network performance logs. These configurations dictate how each agent should behave in terms of data routing, resource allocation, and other network optimization tasks.
- **Configuration Update Requests:** Instructions sent to BDIx agents to update their configurations according to the newly generated parameters. This ensures that the network continuously adapts to changing conditions and maintains optimal performance.
- **Log Request Commands:** Commands sent to BDIx agents to transmit their logs to the component for analysis. This feedback loop allows the component to refine future configurations further and enhance its decision-making process.

Functional Process Flow:

- **Data Collection:** The component receives real-time localization data from the localisation module and performance logs from BDIx agents across the network.
- **Analysis and Optimization:** Utilizing the received data, the component analyzes the current state of the network, identifies areas for improvement, and generates optimized configurations for the BDIx agents.
- **Configuration Deployment:** The component sends out configuration update requests to BDIx agents, instructing them to adjust their operations according to the new parameters.

- **Feedback and Adjustment:** BDIx agents send back logs in response to log request commands. The component uses this data to evaluate the effectiveness of the current configurations and make necessary adjustments.

By integrating this component into the Resource Orchestration and Management framework, the proposed solution ensures that the network can dynamically adapt to changing conditions, optimize resource utilization, and maintain high levels of performance and efficiency. This approach leverages the collective intelligence of BDIx agents, informed by precise localization data, to achieve superior network management and optimization in 6G environments.

Figure 6.4 illustrates the functional architecture of the ADROIT6G system, showcasing the interactions among its components, including the integration of BDIx agents within the AI/ML Orchestrator framework. This comprehensive architecture optimizes network operations through cooperative and distributed intelligence, leveraging advanced AI/ML techniques.

To summarise, the proposed DAI framework, by employing BDIx agents and ML in a novel manner, greatly enhances the implementation of CrowdSourcing AI in the telecoms industry. The solution optimises network operations by distributing intelligence across the network and utilising the collective insights and computational capacity of several devices. It is particularly effective in improving energy efficiency and traffic control. This strategy not only coincides with the lofty objectives of 6G networks but also showcases the revolutionary capacity of CrowdSourcing AI in influencing the future of network management and optimisation.

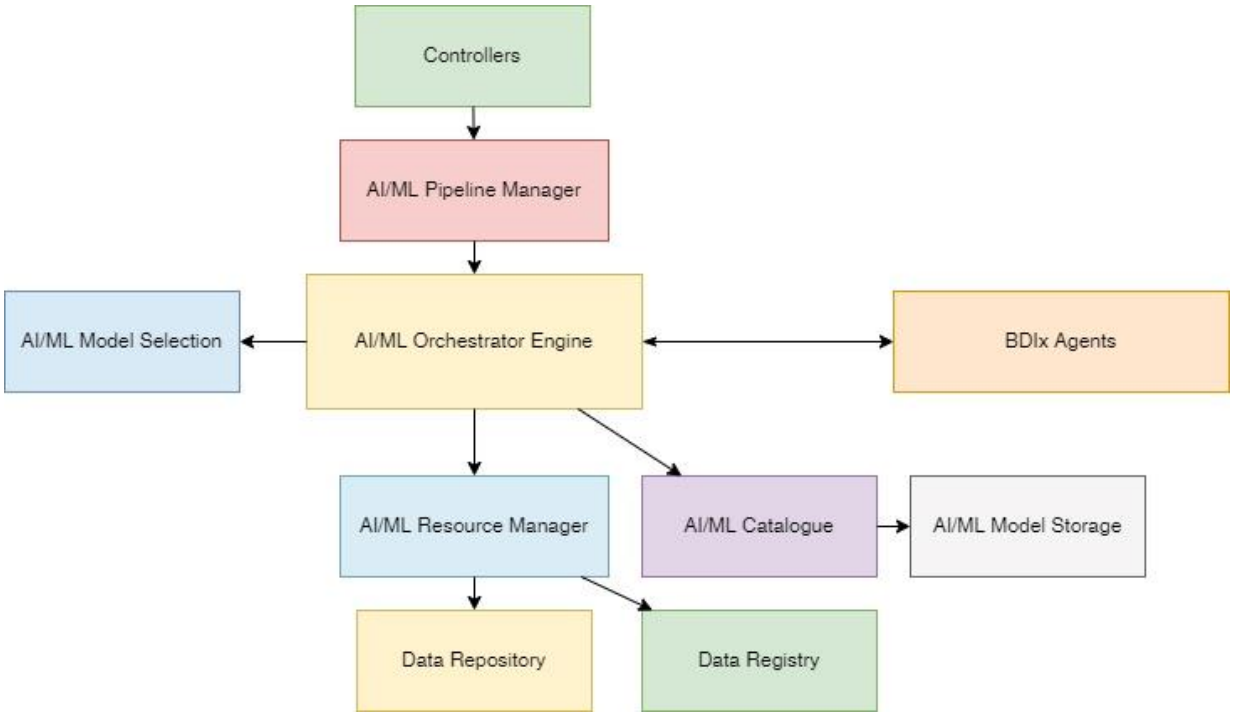


Figure 6.4: Interaction of all ADROIT components with BDIx agents

6.3 Traffic predictor

Traffic prediction plays a critical role in modern network management, particularly in optimizing the performance of User Equipment as Virtual Base Station (UE-VBS) Controllers at each base station. With the surge in mobile internet traffic, networks are under increasing pressure to efficiently manage resources to maintain high QoS. This challenge is compounded by the dynamic nature of internet traffic, where different traffic classes—each

© ADROIT6G, 2023

with its unique QoS requirements—can suddenly increase demand on the network. Traditional approaches to traffic management, which often involve reactive adjustments based on current conditions, fall short in addressing these challenges effectively. The introduction of traffic prediction mechanisms, particularly through advanced AI models like RNN-BLSTM, offers a solution. By analyzing patterns from past and current traffic data, these models can forecast future traffic trends across different QoS classes with remarkable accuracy. In this context, the prediction extends over a horizon of more than 7 hours, achieving an average accuracy of 97.7% through the analysis of data collected via deep packet inspection (DPI) in operational networks [98].

This predictive capability allows the UE-VBS Controller to anticipate which traffic classes will demand higher bandwidth and resources soon. Armed with this foresight, the controller can undertake proactive measures such as enabling caching operations for anticipated high-demand content or adjusting the priority of QoS on the UE-VBS to align with the predicted demand of these classes. Such preemptive adjustments ensure that the network can better manage its resources, preventing congestion before it occurs and ensuring a smoother, more efficient service delivery that aligns with the specific needs of different traffic classes. the RNN-BLSTM's achieves a suitability for dynamic environments where traffic conditions change rapidly. Despite its slightly higher energy consumption, the RNN-BLSTM's superior accuracy and execution time make it an invaluable tool for network service providers aiming to leverage dynamic link utilization more effectively. This approach not only enhances the operational efficiency of the network but also significantly improves the user experience by ensuring that network resources are allocated in a manner that meets the evolving demands of different QoS classes. By incorporating such advanced traffic prediction frameworks, network service providers can navigate the complexities of traffic heterogeneity and dynamic internet application requirements, setting a new standard for proactive network management.

6.3.1 ADROIT6G specific enhancements over state of the art

The advancement in energy-efficient traffic prediction algorithms signifies a notable leap forward, integrating cutting-edge ML technologies to enhance accuracy and efficacy in forecasting network traffic. Traditional methodologies like ARIMA and LSTM networks have been foundational but are increasingly challenged by modern data traffic's voluminous and complex nature. The advent of RNN-BLSTM models represents a crucial evolution, offering insights into traffic patterns by evaluating the bidirectional data flow over prolonged periods. This capability is pivotal for precisely anticipating the QoS requirements and ensuring network resources are allocated efficiently, avoiding unnecessary energy consumption. The integration of DPI with RNN-BLSTM models elevates the prediction process by permitting a detailed examination of network traffic at the packet level, thus refining forecasts of QoS categories and traffic volumes. This combination empowers network service providers to dynamically optimize. Despite its marginally higher power usage, the RNN-BLSTM model's superior precision, speed, and energy efficiency make it indispensable for navigating the dynamic landscape of network traffic with constantly changing patterns.

Employing RNN-BLSTM alongside DPI marks a transformative step in applying AI for proactive network management and optimization, shifting from a reactive stance to a proactive one in resource allocation. It leverages crowdsourced data to predict future traffic trends and discern the specific QoS needs, allowing AI to make well-informed resource distribution decisions. This strategy not only boosts network efficiency and user satisfaction but also aligns with green computing efforts by reducing energy wastage and supporting environmental sustainability goals. The utilization of RNN-BLSTM and DPI in network management underscores the potential of specialized AI models to significantly improve network operations with a forward-thinking approach to resource distribution, emphasizing the importance of QoS in creating a more effective, adaptive, and sustainable network infrastructure.

6.3.2 Functional architecture

Integrating a proposed energy-efficient traffic prediction model with a focus on QoS and base station specificity entails developing a functional architecture designed for predictive resource allocation tailored to each base station's anticipated QoS requirements. This method dynamically manages network resources to cater to the

diverse demands of various services at each base station, enhancing network performance and user satisfaction. The functional architecture comprises several components, each playing a pivotal role in the predictive modeling process. The QoS and Base Station Data Collector is tasked with collecting comprehensive traffic data from each base station, categorized by QoS classes. It gathers metrics like data rates, service request volumes, and bandwidth usage, with data collection occurring in real-time, typically at 5-minute intervals. This ensures a continuous stream of up-to-date information for accurate prediction. Following data collection, the Data Preprocessing Module processes the gathered data to ensure consistency and readiness for analysis. Given the varied nature of the data across different base stations and QoS classes, this module standardizes formats, normalizes values, and resolves any discrepancies, setting the stage for effective predictive modeling (as shown in Figure 6.5).

The ML Model Storage acts as a repository for various predictive models, including RNN-BLSTM models specifically designed for different QoS classes and base stations. Stored models are primed for deployment, awaiting real-time data inputs to address predictive requirements as they arise. At the core of the architecture is the Prediction Engine per QoS and Base Station, which selects and deploys the most suitable predictive model from the ML Model Storage. This selection is based on the data's origin (the specific base station) and its QoS class. Capable of running multiple model instances concurrently, the engine facilitates parallel predictions for various QoS classes and base stations, ensuring a comprehensive and simultaneous forecasting capability across the network. This architecture not only streamlines the process of predicting network traffic but also optimizes resource allocation at each base station according to the specific QoS requirements, ensuring high network performance and enhanced user experience.

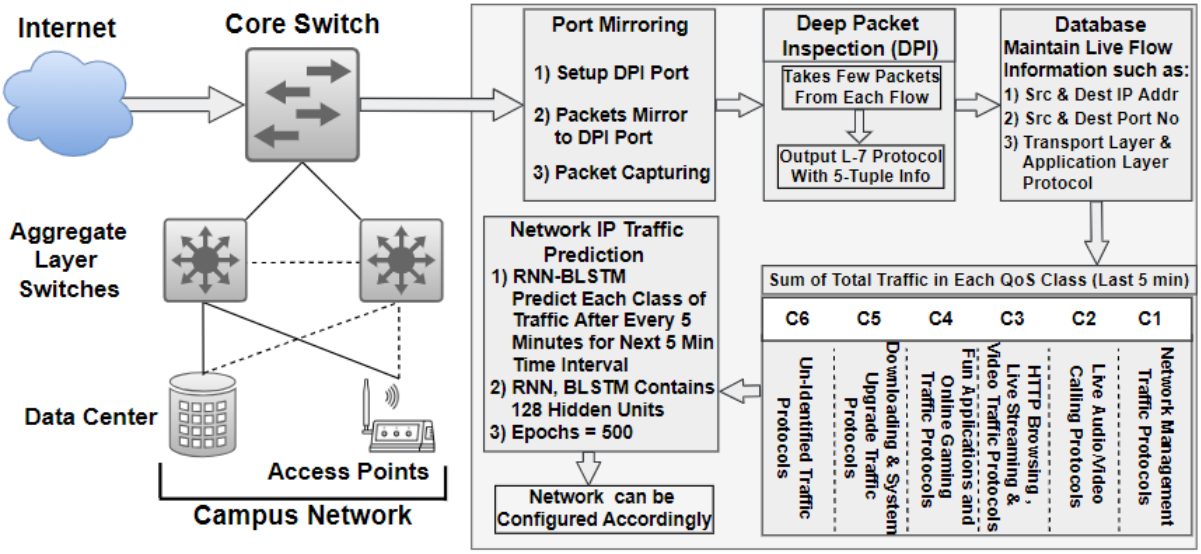


Figure 6.5: System model for Traffic prediction of 6G multivariate heterogeneous mobile network.

Inputs:

- **Real-time Traffic Data per QoS and Base Station:** The specific traffic metrics are collected at 5-minute intervals from each base station, segmented by QoS classes. This granularity allows for precise modeling of traffic trends and requirements.
- **Model Parameters:** The configurations or hyperparameters for each predictive model, tailored to the characteristics of specific QoS classes and base stations. These parameters are optimized during the model training phase.

Outputs:

- **Traffic Predictions per QoS and Base Station for the Next 5 Minutes:** The key output, providing forecasts of traffic volumes and patterns for each QoS class at individual base stations. This level of detail supports resource allocation decisions.
- **Resource Allocation Recommendations:** Based on the predictions, this output offers actionable insights for dynamically adjusting network resources at each base station to optimally meet the anticipated demand for each QoS class.

Process Flow:

1. **Data Collection:** Traffic data, segmented by QoS class, is collected in real-time from each base station.
2. **Preprocessing:** The data undergoes preprocessing to ensure accuracy and consistency for predictive analysis.
3. **Prediction:** For each base station and QoS class, the corresponding predictive model is applied to the preprocessed data, forecasting traffic for the next 5 minutes.
4. **Resource Allocation:** Utilizing the traffic predictions, the network dynamically allocates resources at each base station, tailored to the predicted needs of different QoS classes.

By focusing on both QoS requirements and base station specifics, this approach enables a highly responsive and efficient network management system. It allows for proactive adjustments in resource allocation, ensuring that the network can maintain optimal performance levels and QoS for all users, regardless of the variability in traffic patterns across different locations and services. This tailored, predictive resource management strategy exemplifies a sophisticated application of CrowdSourcing AI in telecommunications, leveraging detailed data and advanced modeling techniques to enhance network responsiveness and sustainability.

6.3.3 Preliminary results

Regarding the prediction performance of RNN-BLSTM, the framework utilizes the LSTM neural network to predict for each QoS class, the traffic within the Mobile network The results of the multivariate time series prediction for various QoS classes are detailed through figures, contrasting the actual time series traffic with the predicted traffic. Specifically, Figure 6.6 and subfigures (a – f):

a. Shows the actual network management traffic versus the predicted traffic within the total network, represented by a black solid line for the actual traffic and a red dotted line for the predicted traffic.
b. Compares the actual and predicted live audio/video calling traffic within the total network traffic.
c. Illustrates the actual versus predicted HTTP traffic, which includes live streaming and video traffic within the total internet traffic.
d. Compares the actual and predicted traffic for fun applications and online gaming within the total network traffic.
e. Presents the actual and predicted downloading and system update protocol traffic.
f. Displays the actual and predicted traffic for a QoS class that encompasses unidentified traffic.

These subfigures collectively provide a comprehensive overview of the prediction performance of the RNN-BLSTM model across various types of network traffic.

To summarise, the pursuit of energy-efficient traffic prediction involves a constant search for advanced, precise, and environmentally friendly solutions. The current cutting-edge technology, exemplified by RNN-BLSTM models and the strategic implementation of DPI, provides a preview of the future of network management (as shown in Figure 6.6). Thus, networks not only can adjust to the constantly evolving requirements of internet traffic, but

they also prioritise energy efficiency and environmental sustainability. The ongoing development of emerging technology and innovative methodologies will inevitably further expand the limits of what can be achieved in energy-efficient traffic prediction as the area progresses.

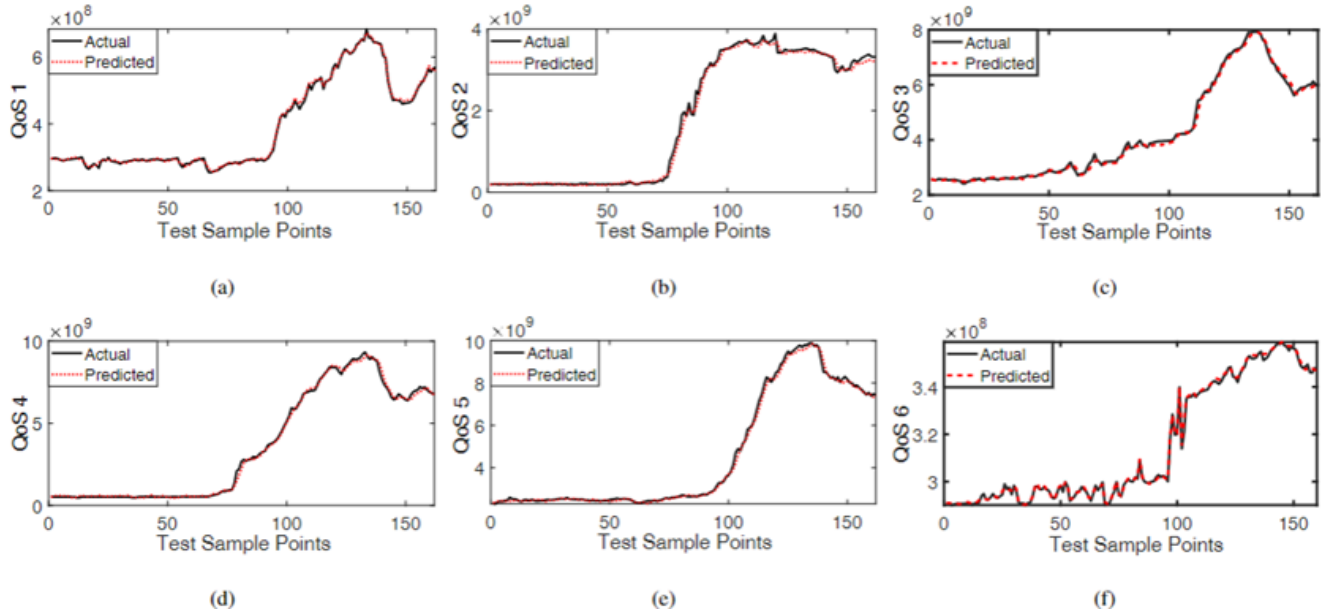


Figure 6.6: Prediction of (a) Network management traffic (b) Live audio/video traffic (c) HTTP and live streaming and video traffic (d) Fun applications and online gaming traffic (e) Downloading and system update traffic (f) Unidentified traffic.

6.4 Optimal D2D Topologies Formation

Central to this innovative service is the optimization of subnet trees through the integration of BDIx Agents and the application of state-of-the-art Minimum-Weight Spanning Tree ML Algorithms, utilizing the capabilities of Graph Neural Networks. The utilization of BDIx agents, grounded in the BDI framework and augmented with ML, addresses the intricate weighted tree structures essential for the effective governance of 6G networks. This setup provides a depiction of the network's current state, including node positions and connection quality, offering critical insights for the Minimum Spanning Tree algorithms. These algorithms are meticulously designed to enhance the physical infrastructure of the network, focusing on minimizing latency by selecting paths with short distances and maximizing throughput to ensure optimal data rates.

The service's dynamic adaptability is a key feature, enabling real-time adjustments to network configurations in response to changes in conditions, user movements, and traffic patterns. This ensures that the network consistently operates at peak efficiency. Built with scalability and flexibility at its core, the service can be seamlessly deployed across a wide range of network sizes and complexities, making it a versatile solution for the next generation of wireless networks. The advantages of implementing this AI/ML-driven service in 6G networks include significant enhancements in network performance through improved data transmission efficiency, leading to reduced latency and increased throughput. This results in adaptive network management capable of maintaining optimal performance in various scenarios, ultimately optimizing network resources and reducing operational costs. The service is especially beneficial in high-demand scenarios, such as densely populated urban areas, large-scale events, and in remote or rural locations where maximizing coverage and connectivity is a priority.

© ADROIT6G, 2023

6.4.1 ADROIT6G specific enhancements over state of the art

The integration of a localization module into the network optimization particularly for base stations, significantly enhances the AI/ML-driven optimization service by providing detailed structural information about the network's physical topology. This information includes the precise locations and connectivity of D2D-Relays (UE-VBS) and other network nodes (D2D-Multi Hop Relays), essential for tailored, predictive resource allocation based on anticipated QoS requirements. The localization module serves as a crucial element, feeding critical structural data into the optimization process and aiding Graph Neural Networks (GNN) and BDIx agents in effectively executing Minimum-Weight Spanning Tree algorithms. BDIx integration facilitates enhanced decision-making for subnet management under gNodeB by optimizing network topology, considering key metrics such as delay minimization and throughput maximization. The benefits brought by the localization module include dynamic network adaptation, allowing real-time adjustments to physical changes, user movement, and traffic patterns. This ensures that the network is optimized for current conditions. Additionally, optimized resource allocation becomes more efficient, leading to energy savings and improved QoS. The module also improves the accuracy of predictive models by supplying them with precise, up-to-date information, enhancing scalability and flexibility for deployment in diverse network environments.

Incorporating this module enriches the data for AI/ML processes, enabling the optimization service to make more accurate and effective decisions and it can be applied to all network structures. This approach aligns with the realization of CrowdSourcing AI in D2D communication networks, optimizing network architectures for enhanced spectrum efficiency and power reservation. By leveraging crowdsourced data, the proposed solution improves spectral efficiency through direct optimization under base stations, adaptively modifying network configurations based on real-time data and structural insights from the localization module. Furthermore, the solution contributes to power allocation and optimization, addressing the energy consumption challenges in modern networks. It employs AI/ML optimization techniques to identify energy-saving strategies, optimizing the network's physical and operational parameters. This process minimizes unnecessary data transmissions and redirects traffic through energy-efficient routes, significantly reducing overall energy consumption. The proposed solution exemplifies the power of CrowdSourcing AI, utilizing extensive network data to continually adapt to changing conditions and optimize network topologies. This collective intelligence approach underlines the transformative potential of AI/ML in network management and optimization, showcasing a proactive, data-driven strategy to enhance network performance, reliability, and user satisfaction.

6.4.2 Functional architecture

The proposed AI/ML-driven optimization solution is a crucial component aimed at enhancing network management at BSs. This solution finds its place within the AI/ML Models Storage, where it aids BDIx agents in optimizing the network structures they generate. It functions as part of a closed-loop governance system that relies on real-time data from a localization module to continuously inform and refine the network optimization efforts. The Models Repository serves as a central hub for AI models, making the optimization model readily available across the network management system for efficient reuse and deployment. This repository ensures that models can be easily accessed and utilized wherever needed within the network to streamline optimization processes.

Operating within a closed-loop governance framework, the solution engages in ongoing monitoring, decision-making, and adjustments of network configurations. It utilizes both real-time data and established policies to maintain the network at peak performance, efficiency, and reliability levels. The localization module, a key component of this closed-loop system, provides essential real-time data on the network's physical structure under the BS. This includes the location of nodes, their connectivity, and other pertinent topological information. Such detailed input enables the AI/ML model to execute informed optimization decisions, ensuring the network is configured optimally. This comprehensive integration underscores the importance of the proposed solution in the broader context of telecommunications architecture, highlighting its role in perpetuating a cycle of continuous improvement and management of network structures for enhanced performance and efficiency.

© ADROIT6G, 2023

Inputs to the Optimization Solution:

- **Network Structure Data from Localization Module:** The primary input for the optimization model is the detailed structure of the network under the BS, as provided by the localization module. This includes data on the positions of network nodes, their interconnections, and other physical attributes of the network topology.

Outputs from the Optimization Solution:

- **Optimized Network Tree:** The main output of the model is an optimized network tree, which represents the most efficient configuration of the network structure under the BS. This optimized tree is designed to enhance spectral efficiency and power conservation while ensuring high levels of service quality and network performance.

Process Flow:

1. **Data Collection:** The localization module continuously collects and updates data on the network's physical structure under the BS, ensuring that the optimization model has access to the most current information.
2. **Optimization Execution:** Utilizing the network structure data, the AI/ML model within the Models Repository processes this information to identify the optimal network configuration. This involves applying Minimum-Weight Spanning Tree algorithms and other optimization techniques tailored to the network's specific needs and constraints.
3. **Network Configuration Update**: Once the optimized network tree is determined, the model outputs this configuration, which is then implemented within the network's physical infrastructure. This optimized structure is aimed at improving network efficiency, reducing latency, maximizing throughput, and ensuring robust connectivity.
4. **Continuous Monitoring and Adjustment**: The closed-loop governance system monitors the performance of the optimized network and, using real-time data from the localization module, makes adjustments as necessary to maintain optimal network conditions.

This integration outlines a sophisticated approach to network optimization, leveraging the synergy between real-time localization data, advanced AI/ML models, and the dynamic capabilities of BDIx agents. By continuously refining the network structure under the BS based on up-to-date information, the proposed solution ensures that the network remains efficient, responsive, and capable of meeting the evolving demands of users and services.

To summarise, the suggested AI/ML-based optimisation service greatly enhances the use of CrowdSourcing AI in telecommunications networks. The solution improves spectral efficiency and power reservation by optimising network structures under base stations. It utilises the collective intelligence of the network to enhance performance, sustainability, and efficiency. This represents a significant advancement in intelligent network management and optimisation.

6.5 Predictors for Real-time VNF Migration, Placement and Scale-out

Network function virtualization (NFV) technology has been a key enabler not only for the 5G but also for the upcoming B5G/6G networks, coming to meet the growing demands of mobile network operators and infrastructure owners for openness, scalability, and granularity of the deployed functions and services [99]. Real-time Virtual Network Function (VNF) migration, placement, and scale-out are critical aspects of NFV and essential for building flexible and adaptive networks capable of meeting the dynamic requirements of modern applications and services. Hence, this section delves into our contributions for orchestrating these capabilities of network services (VNFs), showcasing our enhancements, based on intelligent algorithms, to state-of-art solutions and their alignment with ADROIT6G's innovative architecture.

6.5.1 ADROIT6G specific enhancements over state of the art

NFV is a promising paradigm, which, alongside with the network programmability, will allow to implement one of the ADROIT6G's objective for dynamic reconfiguration of slices across multiple technological domains, enabling the VNF of service functions to adapt dynamically based on the network conditions. According to the ESTI standard, the scheduling of VNF lifecycle is controlled by a central entity that has the role of the NFV orchestrator [100] and consists of three key operations: i) service initialization and onboarding, during which the VNF graph is embedded; these operations happen once in order to allocate VNFs to a specific domain or different domains (controlled by computing nodes with a fixed maximum capacity and satisfying some service-related SLA constraints), and, periodically, ii) service scale-out, and iii) service scale-in are executed [101].

A challenging issue in resources' scaling is to attain a good trade-off between performance (activating adequate resources) and energy consumption, namely, to activate adequate resources but no more than the one is needed, to meet the service performance requirements specified in Service Level Agreement (SLA) [100]. Scale-in reduces the VNF instances of a given service chain during underutilization periods to avoid overprovisioning, while scale-out increases the instances of a given VNF service chain such that load balancing is ensured. The latter is typically triggered when the allocated resources (namely, CPU, memory or disk usage) utilization is increased upon increased traffic [102]. To further optimize the VNF placement under scale out/in triggering events, live service migration [103] is performed as well. VNF migration is responsible for redeploying a VNF instance from one node in a specific domain to a node in a different domain (inter-domain) ensuring parallel service continuity and user experience. Hence, efficient VNF placement and orchestration mechanisms are needed to address the challenges of continuously changing network dynamics and complexity, advanced service requirements (e.g., low latency) and potential user mobility.

Scaling operations are not straightforward, and their triggering event(s) should be optimally identified. The decisions related to such events can be made either reactively or proactively; the former relies on threshold policies observing if the selected metrics' values satisfy or exceed a predefined threshold value, while the latter is based on the predicted future value(s) of a metric in a time horizon (short-term or long-term). The leverage of Deep Learning (DL) and ML on proactive approaches [102], [103], [104], [105], [106], [107] design will help to make intelligent decisions autonomously based on historical data of monitored metrics that reflect network status.

Such methods, in ADROIT6G, can be used to train efficient models for predicting the future value(s) of important metrics using the collected data from resources (computational, network, etc.) monitoring; their output can then feed as input to a local agent in a specific domain which may interact with agents in a different domain enhancing scaling and migration of any given service function against the default threshold-based systems. Moreover, by incorporating Reinforcement Learning (RL) combined with advanced neural architectures into NFV-enabled ADROIT6G infrastructure, it will be possible to promote the network functionality via algorithmic innovation to achieve both the required domain objectives (e.g., for reducing cost, minimizing the end-to-end latency, reducing energy consumption, etc.) and the level of intelligence and automation in VNF operations scheduling. A state-of-the-art solution for making an energy-aware, dynamic SFC placement in URLLC services with strict performance requirements has been presented in [108] and was extended [109], where distributed DRL agents interact with the environment and learn optimal VNF placement strategies taking into account domain- and/or node-specific performance metrics (latency, bandwidth, energy) that feed the so-called reward functions of these agents. A decentralized mechanism based on Voting makes the final decision in terms of the optimal placement. The concept of RL in combination with advances in DQNs, the so-called Double DQNs with Dueling Architecture [110], will be utilized to improve migration decision and VNF placement, and thus optimize ADROIT6G functionality and lay the foundation of the proposed approach.

The novel turning points of our approach in ADROIT6G are highlighted by the following selections:

RL Agent: it demonstrates the fundamental ability for intelligent decision-making in VNF placement, optimizing network performance and resource utilization in real-time. The agent is designed with advanced Dueling

Network Architectures on Double DQNs for learning optimal actions regarding VNF placement based on the current state of the system. This involves the use of experience replay and target networks to stabilize learning. Dueling network architectures represent a significant advancement in the field of DRL, particularly for value-based methods like DQNs. Furthermore, to push the boundaries beyond current benchmarks, we incorporated additional innovations such as prioritized experience replay, which optimizes the learning process by focusing on more informative experiences, and the aforementioned dueling network architectures, which separately estimate state values and the advantage of each action, leading to more precise policy evaluation.

Cloud-Native Orchestration: it successfully integrates multiple container runtimes and Kubernetes for container management and orchestration. This allows for the flexible deployment and management of VNFs across different nodes within the network.

Dynamic Resource Management: Our approach aims for efficient resource management, where the DDQN RL agent dynamically decides on the placement of VNFs based on their resource requirements (CPU, memory, disk usage) and the current load on different nodes. This is critical for optimizing resource utilization and ensuring high service quality.

System Synchronization and Coordination: The orchestration logic includes mechanisms for synchronizing actions across different nodes, as demonstrated by the checkpoint synchronization and voting process for decision-making. This ensures a coordinated approach to VNF placement and migration, which is essential for maintaining network stability and performance.

Performance Monitoring and Adaptation: The system is designed to continuously monitor its performance. This allows the system to adapt to changing network conditions and service demands, optimizing VNF placements over time to meet SLAs and enhance user experience.

Scalability and Flexibility: The use of cloud-native technologies coupled with the modular design of the DRL agent and orchestration logic serves as a cornerstone regarding system scalability as the latter can be extended to accommodate different network environments and requirements.

Operational Efficiency: Through the intelligent automation of VNF placement decisions, we aim to increase automation, minimize latency by optimizing service locality, and prevent SLA violations, thereby improving operational efficiency.

6.5.2 Functional architecture

The ADROIT6G architecture proposes an **AI-driven Management & Orchestration Framework**, specifically designed for 6G networks. Within this framework, the proposed predictor serves as an essential component of **Resources Management & Orchestration** block for managing the scaling operations under the problem of VNF placement based on DQNs. Therefore, we conceptualize the process of interaction between the agent of DQN and the environment as a Markov Decision Process -MDP (S, A, R, P, γ) . In this process:

- State $s \in S$ (with S being the state space) encapsulates the current allocation of computational resources across the network, including server CPU cores, memory, and storage, along with bandwidth and latency of the domain links.
- Actions $a \in A$ (with A being the actions space) represent the various possible placements of VNFs, each with an associated Confidence metric that quantifies the system's readiness to host the VNF.
- The reward function R captures the agent's goal during action selection and, in ADROIT6G, is designed to balance the throughput and latency across the network, directing the agents towards a cooperative and optimal orchestration of resources.
- The policy π denotes the distribution that maps state to action while $Q(s, a)$ is the action-value function under policy π.
- γ is a discount factor in the range [0,1]
- P is the transfer probability for moving from current state s to next state s'

Solution tool based on Dueling Architecture on Double DQNs

The design of the optimal policy will be built upon DDQN, an improvement upon DQN architecture to mitigate their inherent overestimation bias of action values. In a conventional DQN, the same network estimates the value of taking each action and then selects the action with the highest value, which can lead to overoptimistic value estimates. DDQN improves upon DQN by decoupling the selection of the action from its evaluation. It does this by employing two separate neural networks (as shown in Figure 6.7) that decouple the selection of the action from its evaluation.

- The Q-Network (Policy Network): This network is responsible for selecting actions given the current state of the environment
- Target Q-Network: This network is used to evaluate the action chosen by the Q-Network by providing a stable target for the Q-value updates. It is a copy of the Q-Network that gets updated less frequently to ensure stability in the learning updates.

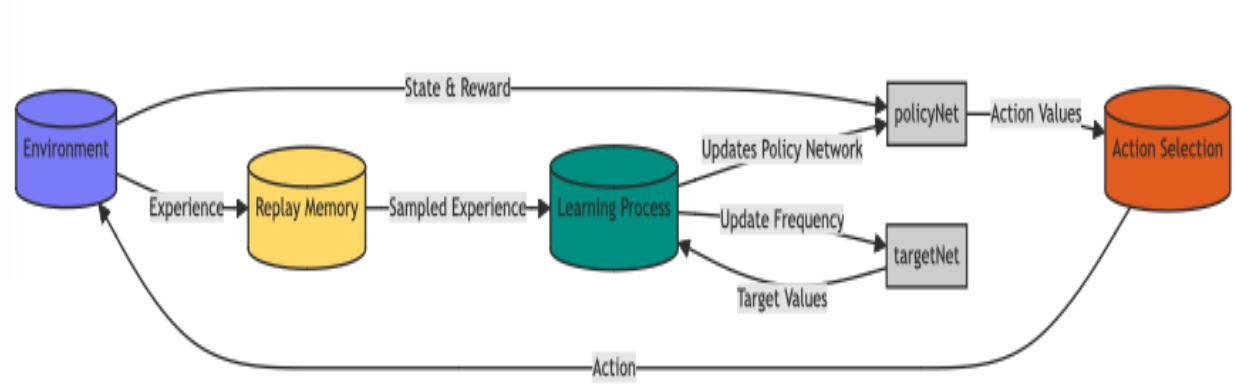


Figure 6.7. Double DQN agent.

Moreover, DDQN incorporates advanced features like Dueling Network Architecture that enhances the ability to learn state values and the advantages of each action separately. This architecture is designed to provide a more sophisticated understanding of the value of each state, independent of the actions taken, which can lead to more efficient and effective policy learning, especially in scenarios where the choice of action does not significantly affect the outcome. The key innovation in the Dueling Network Architecture is the separation of the neural network into two distinct pathways or streams after an initial common feature learning layer:

- Value Stream: This stream estimates the overall value of being in a given state, $V(s)$. It assesses the underlying value of the state itself, regardless of the actions available.
- Advantage Stream: This stream calculates the advantage of each action, $A(s, a)$, which is the extra value the agent gets by choosing a particular action a in state over other actions. The advantage function essentially measures the potential superiority of some actions over others from the same state.

The outputs of these two streams are then combined to produce a single Q-value function, $Q(s, a)$, for each action a in the given states. This is typically done using a special aggregation formula that ensures the learned Q-values make sense in terms of both the value and advantage estimations. For a refined approach of the action-value function, this one is formulated as $Q(s, a) = V(s) + A(s, a)$. Then, the two functions $V(s)$ and $A(s, a)$ are approximated using two neural networks (NN). The NN of V(s) chooses actions and the NN of $A(s, a)$ called target network evaluates those choices. The Dueling Network Architecture enhances our model providing a slightly distinct assessment of the network states (through the Value Stream) and the relative value of actions (via the Advantage Stream). These modifications are codified into the RL agent class, which uses a sequential model with layers optimized for capturing the complexity of the state-action-reward relationship. A common

approach is to subtract the mean of the advantages from each action's advantage to ensure that the Q-values are properly normalized and that the value estimates do not overly influence the action advantages.

Figure 6.8 presents the algorithm for training the Dueling DDQN-based agent. At its core, the agent initializes with defined observation and action spaces, reflecting the network's current state and potential VNF placements. It incorporates a dual-model approach, consisting of a policy model for action selection and a target model for evaluating those actions, both constructed using a sequential neural network framework. Through an epsilon-greedy algorithm, the agent dynamically balances between exploring new actions and exploiting known strategies, ensuring a comprehensive understanding of the network environment. The inclusion of a memory buffer for experience replay allows the agent to learn from past decisions, enhancing its decision-making process over time. By periodically updating the target model's weights towards the policy model, the agent maintains a stable learning procedure. This sophisticated combination of DDQN with strategic memory replay and model updating mechanisms equips the RL agent to intelligently navigate the complexities of network optimization through MDP, achieving efficient and effective VNF placement decisions.

In VNF placement, the Dueling Double DQN (DDDQN) architecture demonstrates a profound capability for optimizing network performance and resource utilization in real-time. By intelligently deciding where to place VNFs within the network, DDQN helps in enhancing the efficiency and reliability of network services, showcasing the practical implications of advanced deep reinforcement learning techniques in real-world scenarios.

Algorithm Dueling-DDQN

Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$
Initialize replay memory D to capacity N
Set batch size $|B|$, ϵ, ϵ_{min}, ϵ_{decay}, γ, τ, learning rate l_r
for episode m = 1 to M **do**
    Observe initial state s
    for time t = 1 to T **do**
        Select a from s using an ϵ-greedy policy based on $Q(s, a; \theta)$
        Take action a, observe reward r and next state $s^{'}$
        Store transition $(s, a, r, s^{'}$, done$) \in D$
        if D is sufficiently populated **then**
            Sample a random minibatch of transitions from D
            for each minibatch transition **do**
                if episode has ended **then**
                    The target value is set to the immediate reward
                else
                    Use the current Q network to select the next action from the new state
                    Use the target Q network to evaluate the Q value for the next state and the action selected by the current Q.
                    Calculate the target value as the immediate reward plus the discounted Q value obtained from the target Q network.
                    Perform a gradient descent step to update the weights of the current Q network.
                    Every C steps, update $\hat{Q}$ by copying weights from Q
                end if
            end for
        end if
    end for
end for

Figure 6.8. Algorithm for training Dueling DDQN

Integrating DDDQN instead of single DQNs represents an extension basis of the [108], [109] on optimizing migration strategies. The original DQN model, while effective in learning policies for migration based on a single agent's interaction with the environment, is susceptible to overestimation of action values due to its inherent design. This overestimation can lead to suboptimal decision-making, where the agent might prefer actions that appear beneficial in the short term but are less advantageous overall. This enhancement addresses the overestimation bias inherent in DQN by decoupling the action selection from its evaluation, thereby refining the accuracy of Q-value estimations. It is expected to get more accurate results regarding migration in a multi-domain and multi-node environment, leading to improved policy decisions, by refining the agent's efficiency that

can lead to innovative solutions beyond state of the art. This approach significantly mitigates unexpected results and sets a precedent for future research in applying advanced RL techniques to distributed environments.

Integrating Recurrent Neural Networks (RNNs) into the DDDQN architecture could improve the agent's ability to represent and interpret complex network states. This enhanced state representation is crucial in dynamically changing environments, where the agent must adapt its strategy in response to evolving network conditions. By capturing the sequential nature of state transitions, RNNs can provide a richer context for each decision point, potentially improving the accuracy of Q-value estimations. RNNs can also contribute predictive capabilities to the RL agent, enabling it to forecast future network traffic patterns and resource demands. Such forecasts can inform more proactive VNF placement strategies, allowing the network to prepare for anticipated load increases or shifts in traffic distribution. This proactive approach can improve resource utilization and maintain high levels of service quality even under fluctuating network conditions.

The interaction of DDDQN model with the ADROIT6G network infrastructure is outlined in the following**:**

- **AI/ML Orchestrator**: This is a crucial component for the Dueling DDQN algorithm which aggregates the learning outcomes from the edge nodes by evaluating and combining state-action values (Q-values) to form a cohesive strategy for VNF placement. It is responsible for distributing the optimized policy back to the edge nodes, facilitating the convergence of the learning model while adapting it to local network conditions.
- **AI/ML Pipeline Manager**: This manager oversees the broadcasting of the updated policy from the AI/ML Orchestrator to the edge nodes and ensures robust and efficient transmission of the policy updates. By utilizing Redis, a fast and reliable in-memory data structure store, the AI/ML Pipeline Manager can handle real-time data synchronization and messaging between edge nodes and the central orchestrator. Redis supports rapid data access and can manage the state of the distributed system, facilitating the aggregation of local Q-values from each agent and ensuring the convergence of the optimization process.
- **Resources Monitoring:** This component provides crucial insights into the status of resources (utilization) at each edge node participating in the distributed learning process. It supplies data that the AI/ML Orchestrator can use to mitigate the influence of nodes with low or negative contributions, thus enhancing the overall performance of the global model.
- **Performance Metrics Evaluation:** The evaluation of key performance metrics, such as average latency and resource utilization. Here is supported detailed analysis and monitoring of the model's performance throughout the learning process, enabling further refinements and optimizations.
- **Edge Nodes**: These are the locations where the actual training of the RL models occurs. Each edge node performs local computations using the DDDQN steps, enabling the local model to better represent the node’s specific resource conditions. Edge nodes calculate the estimated advantages of actions and value functions, enhancing the convergence process through localized optimization.

VNF placement strategy: The action determination does not necessarily lie under the responsibility of a single agent. Instead, after each agent proposes an action based on its DDDQN assessment, a Voting mechanism takes over. This mechanism aggregates the Confidence metrics from all proposals, selecting the action with the highest collective confidence. This democratic approach ensures that the final decision reflects a consensus among agents, considering diverse perspectives and leading to a more robust VNF placement strategy.

- **Auction Initiation**: The system highlights the next service VNF and initiates an auction among the distributed domains.
- **Distributed Operation**: The distributed Agents of the domains generate their action and bid with their confidence to host this specific VNF locally in their cluster, ensuring minimum data transfers.
- **Global Operation**: The distributed agents exchange the Confidence Metrics and the highest bidder or the domain with the maximum Confidence receives the VNF currently in auction. The Auction Mechanism notifies the candidate domain with an acknowledgment response.

- **Orchestration**: If the candidate domain is different from the current domain that hosts the VNF, the inter-domain migration is initiated. Contrariwise, the domain agent performs an intra-domain migration to the internal cluster node of the destination domain with the highest Confidence. If the VNF is already running in the same node, the procedure of migration is declined.
- **Iteration**: The procedure is repeated indefinitely.

Key Performance Indicators for performance assessment. The proposed solution actively meets the requirements of QoS and QoE. Nonetheless, given the nature of the RL techniques employed in the proposed solution for handling VNF placement, two tiers of KPIs are considered. Besides to computational- and networking-related KPIs such as ultra-low user experience latency, peak throughput, and Energy Efficiency (EE), RL techniques must evaluate some internal KPIs. A list of the most relevant ones is outlined below.

1. The learning convergence speed, which refers to the time required for learning-based algorithms to converge.
2. Sample complexity, which indicates the number of samples necessary for the algorithm to achieve scalability.
3. The impact of the problem size impacts the convergence speed and thus it is crucial to develop an algorithm with convergence speed that is as less sensitive as possible to it.
4. Adaptability to a dynamic environment is of high importance. Therefore, the robustness in a dynamic environment is evaluated, which represents how effectively a learning agent can quickly comprehend its environment and subsequently leverage its rewards.

In the next period, we aim at finalizing the proposed algorithm and develop an evaluation procedure to demonstrate its performance.

6.6 Migration of Containerized AI/ML-based Applications

Stateful migration has emerged as the dominant technology to support microservice mobility at the network edge while meeting the end users' QoE requirements. In this context, within the project we started addressing the two pivotal challenges of implementing and orchestrating the migration of containerized AI/ML-driven microservices. In the present reporting period, we have focused on developing an orchestration framework that efficiently realizes stateful migration and effectively orchestrates the migration process by fulfilling both network and application KPI targets. We remark that stateful migration is used whenever keeping track of an MS state is essential to guarantee service continuity. In fact, despite the current trend favouring the development of stateless MSs, stateful MSs are still extremely common due to the complexity in refactoring legacy monolithic applications. Furthermore, according to service-oriented architecture patterns, some essential stateful utility services will always be required, even if stateless service implementation will become dominant. On-going research activity is investigating the performance obtained in AI/ML-driven applications in mobile scenarios such as object tracking through UAVs.

6.6.1 ADROIT6G specific enhancements over state of the art

Implementing an effective and efficient migration framework at the edge is challenging since:

- the duration of service disruption due to the migration process depends on a number of factors, such as coding optimization, communication protocols, and limited nodes' computational capabilities, thus its minimization is not trivial;
- upon statefully migrating an microservice, the established connection thereof with the mobile end user needs to be preserved;
- the stateful migration process needs to be independent of the specific microservice and the underlying edge technology;
- to effectively actuate and configure migration, it has to be envisioned an orchestrator that can collect and aggregate all relevant metrics.

Our solution, named Migration Orchestration framework for microservices at the Edge (MOSE), faces the above challenges by implementing stateful microservice migration at the edge and orchestrating the migration process to ensure minimal impact on the user's QoE. Specifically, the main enhancements of MOSE with respect to the state of the art are as follows:

- It enables the implementation of stateful migration of a generic MS at edge scale, while preserving its connection with the mobile end user
- Its agnostic design enables a seamless and efficient integration on top of already existing edge platforms,
- It features several optimizations of the migration workflow, thus minimizing the experienced service disruption duration
- Leveraging the PAM model [111], it effectively configures the migration process, so that the target migration KPIs and the vertical's objectives are met, even in the presence of strict reliability and robustness requirements.

6.6.2 Functional architecture

We consider a containerized stateful AI/ML-based microservice to be supported at the edge or far edge and that needs to be migrated for load balancing or to "follow" an end user thus ensuring consistently low bandwidth consumption or/and latency. To perform such migration, the Service Orchestration/Management module should run a proper algorithm that determines the destination host and instructs the AI/ML Resource Manager to move the AI/ML microservice there. To do so, the following flow should be implemented:

1. The AI/ML Orchestrator receives instructions from the Service Orchestration/Management module to relocate the AI/ML microservice;
2. The AI/ML Pipeline Manager relays the instructions to the AI/ML Resource Orchestrator. The latter will instruct the nodes that act as source and destination hosts (i.e., hosting the source and destination agents, respectively) and reserve the necessary network resources, and the computing resources at the destination host, while it releases computing resources at the source host;
3. Once the migration is completed, the agent at the destination node informs the AI/ML
4. Resource Manager that notifies the AI/ML Orchestration, which in its turn notifies the Service Orchestration/Management module.

The framework that allows the migration of the AI/ML-based microservice operates as follows. A migration agent runs at the mobile end user receiving the service and at each edge host; we refer to source (destination) agent as the instance running at the source (destination) host. It consists of the following main modules. The MOSE Migration module is responsible for implementing the migration process according to the configuration provided by the Service Orchestration/Management module. The MOSE Overlay Management module is in charge of managing and configuring the overlay network, thus enabling migration. The functionality of these modules strongly relies on two fundamental sets of RESTful APIs, namely, Podman and OvS. While the former is used to interact with the Podman container engine, the latter is used to configure the overlay network. The migration protocol is implemented via pub/sub/query mechanisms provided by Zenoh. It regulates: (i) the signaling of migration tasks, (ii) the actuation of the steps of the migration process, and (iii) the transfer of checkpoint images.

As an example, Figure 6.9 illustrates the communication flow between the AI/ML Resource Manager and the agents when an Iterative PreCopy migration task is performed.

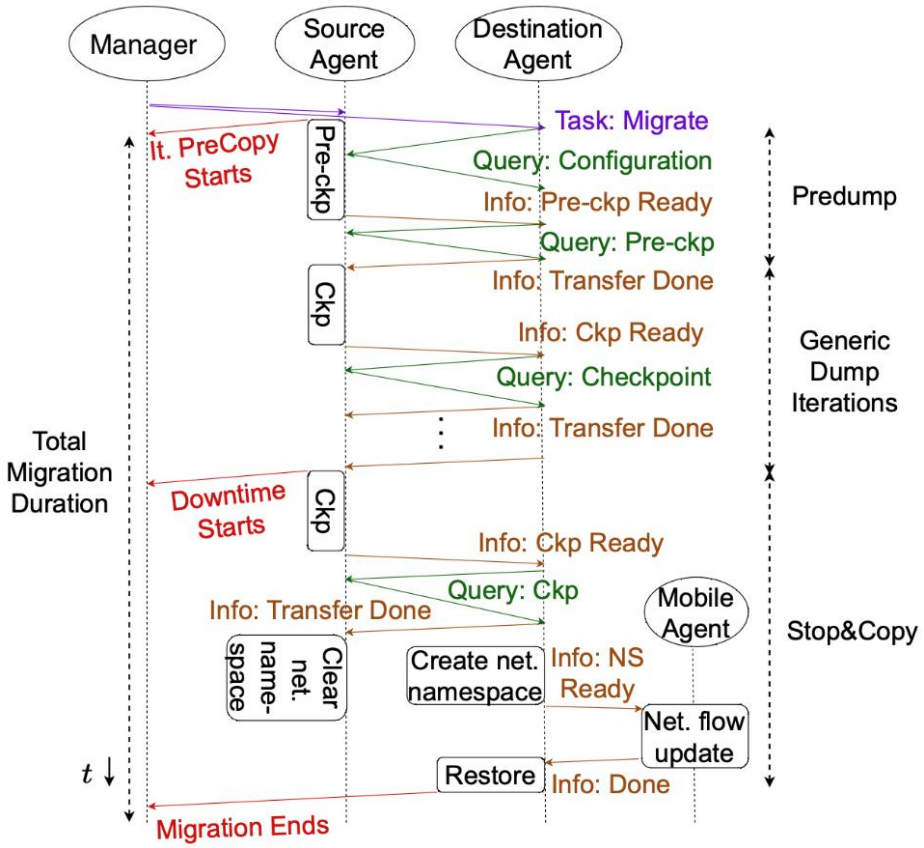


Figure 6.9: Migration protocol under MOSE.

The migration task is published on a Zenoh topic and is accessed by both source and destination agents. First, the source agent informs the AI/ML Resource Manager that migration has begun. Then, while the source agent starts the pre-checkpoint procedure, the destination agent queries the required information to restore locally the microservice, e.g., the microservice's network namespace configuration setting. After the pre-checkpoint, the source agent asks the destination agent to gather the related pre-checkpoint image. Similarly, the source agent runs the checkpoint stage and notifies the destination host, which queries the corresponding image. After the check-point image is transferred, the microservice's network namespace at the source host is cleared, while the namespace is recreated and configured at the destination host. Upon being informed of the migration of the microservice's network namespace, the mobile agent updates the client's network flow towards the agent, thus preserving the microservice established connection with the client application. Finally, the destination agent restores the microservice and notifies the AI/ML Resource Manager accordingly.

We remark that our communication solution enables an automated migration with no need for additional remote-control protocols such as SSH, thus minimizing the migration latency.

In the upcoming months, the activity will focus on validating and deriving performance results.

7 Conclusions

This deliverable reported the results of the research activity within WP3. Notably, the project has defined the CrowdSourcing AI framework and its interaction with the AI-driven M&O framework. It has designed the algorithmic solutions to be implemented at the AI/ML Orchestrator, which controls the AI/ML pipeline and, specifically, orchestrates collaborative ML models training and inference tasks in an efficient manner. The activity has also developed mechanisms for the management of multimodal data and their generation at the processing entities in charge of training ML models. Furthermore, the document presents relevant AI/ML-based network services that can leverage the CrowdSourcing AI framework and how they can be adapted to the dynamic operational conditions of network system.

A preliminary performance evaluation has shown that the above results achieved within WP3 can effectively contribute to the following target KPIs:

- *TO3 KPI 1: x10 improvement of energy efficiency (under light load)*, through energy-efficient training and resource management (Sections 4.1, 4.5, 4.6, 4.8, 6.1, and 6.2);
- *TO3 KPI 2: 70% more types of devices contributing to learning and inference processes*, through distributed learning and inference (Section 4);
- *TO3 KPI 3: 50% reduction in data transferred for decision making,* through transferred learning and synthetic data generation (Sections 4.2, 4.4, 4.7, 5);
- *TO4 KPI 1: >50% reduction in network bandwidth occupation,* through efficient distributed learning (Sections 4.3, 6.2, 6.3, 6.4, 6.5, 6.6);
- *TO4 KPI 2: >30% increased robustness to node faults*, through synthetic data generation (Section 5).

We plan to further expand the research activity within WP3 so that the presented solutions can be refined, and their performance can be thoroughly assessed. Based on such an evaluation, we intend to select the most performing techniques and integrate them within the CrowdSourcing AI framework. This will be instrumental to (i) determine the mechanisms to be implemented within WP5, and (ii) to quantify how the selected solutions contribute to the performance gain highlighted by the project target KPIs. Results from Testing Cycle 1 (M20-M25) will allow for potential further improvements of the designed solutions.

© ADROIT6G, 2023

8 References

[1] https://hexa-x.eu/wp-content/uploads/2023/05/Hexa-X_D5.3_v1.0.pdf
[2] https://www.prefect.io/opensource
[3] https://argoproj.github.io/workflows/
[4] https://airflow.apache.org/
[5] https://www.kubeflow.org/
[6] 3GPP TS 23.288 V18.4.0 (2023-12) Architecture enhancements for 5G System (5GS) to support network data analytics services (Release 18)
[7] TR 23.700-82 Study on application layer support for AI/ML services (Release 19)
[8] A. M. Abdelmoniem, A. N. Sahu, M. Canini, S. A. Fahmy, “Resource-efficient Federated Learning,” arXiv preprint arXiv:2111.01108, 2021.
[9] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, K. Chan, “Adaptive federated learning in resource constrained edge computing systems,” IEEE JSAC, 2019.
[10] H. Wu, P. Wang, “Fast-convergent federated learning with adaptive weighting,” IEEE Transactions on Cognitive Communications and Networking, 2021.
[11] F. Malandrino, G. di Giacomo, A. Karamzade, M. Levorato, C. F. Chiasserini, "Tuning DNN Model Compression to Resource and Data Availability in Cooperative Training," IEEE/ACM Transactions on Networking, doi: 10.1109/TNET.2023.3323023
[12] T. Linjordet, K. Balog, “Impact of training dataset size on neural answer selection models,” ECIR, 2019.
[13] S. Iwata, “Submodular function minimization,” Mathematical Programming, 2008.
[14] M. Groetschel, L. Lovasz, A. Schrijver, “The ellipsoid method and its consequences in combinatorial optimization,” Combinatorica, 1981.
[15] T. Itoko, S. Iwata, “Computational geometric approach to sub-modular function minimization for multiclass queueing systems,” International Conference on Integer Programming and Combinatorial Optimization, 2007.
[16] W. A. Aziz, I. Ioannou, M. Lestas and V. Vassiliou, "Dual-Step Transfer Learning-Based Prediction Model for Next-Generation Intelligent Cellular Networks," 2023 IEEE International Conference on Communications Workshops (ICC Workshops), Rome, Italy, 2023, pp. 260-265.
[17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Artificial intelligence and statistics. PMLR, 2017, pp. 1273–1282.
[18] M. Mohri, G. Sivek, and A. T. Suresh, “Agnostic federated learning,” in International Conference on Machine Learning. PMLR, 2019, pp. 4615–4625.
[19] Y. Deng, M. M. Kamani, and M. Mahdavi, “Distributionally robust federated averaging,” Advances in neural information processing systems, vol. 33, pp. 15 111–15 122, 2020.
[20] M. M. Amiri and D. Gündüz, “Over-the-air machine learning at the wireless edge,” in 2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). IEEE, 2019, pp. 1–5.
[21] J. Du, B. Jiang, C. Jiang, Y. Shi, and Z. Han, “Gradient and channel aware dynamic scheduling for over-the-air computation in federated edge learning systems,” IEEE Journal on Selected Areas in Communications, vol. 41, no. 4, pp. 1035–1050, 2023.
[22] M. Badi, C. Ben Issaid, A. Elgabli, and M. Bennis. "Balancing Energy Efficiency and Distributional Robustness in Over-the-Air Federated Learning," ICMLCN 2024 - IEEE International Conference on Machine Learning for Communication and Networking.
[23] 3GPP TS 38.470, “F1 general aspects and principles, (Rel. 17), 2022.
[24] B. Brik, K. Boutiba, A. Ksentini, “Deep Learning for {B5G} Open Radio Access Network: Evolution, Survey, Case Studies, and Challenges,” IEEE Open Journal of the Communications Society, vol. 3, pp. 228-250, 2022, doi: 10.1109/OJCOMS.2022.3146618.

© ADROIT6G, 2023

[25] S. Tripathi, C. Puligheddu, S. Pramanik, A. Garcia-Saavedra, C. F. Chiasserini, “Fair and Scalable Orchestration of Network and Compute Resources for Virtual Edge Services,” IEEE Transactions on Mobile Computing, pp. 1-17, 2023, doi: 10.1109/TMC.2023.3254999

[26] M. Zawish, N. Ashraf, R. I. Ansari, S. Davy, “Energy-Aware AI-Driven Framework for Edge-Computing-Based IoT Applications,” IEEE Internet of Things Journal, vol. 10, no. 6, pp. 5013-5023, 2023, doi: 10.1109/JIOT.2022.3219202

[27] Z. Zhang, N. Wang, H. Wu, C. Tang, R. Li, “MR-DRO: A Fast and Efficient Task Offloading Algorithm in Heterogeneous Edge/Cloud Computing Environments,” IEEE Internet of Things Journal, vol. 10, no. 4, pp. 3165-3178, 2023, doi: 10.1109/JIOT.2021.3126101

[28] S. Chen, L. Rui, Z. Gao, W. Li, X. Qiu, “Cache-Assisted Collaborative Task Offloading and Resource Allocation Strategy: A Meta-reinforcement Learning Approach,” IEEE Internet of Things Journal, vol. 9, no. 20, pp. 19823-19842, 2022, doi: 10.1109/JIOT.2022.3168885

[29] K. Min, Y. Kim, H.-S. Lee, “Meta-Scheduling Framework With Cooperative Learning Toward Beyond 5G,” IEEE Journal on Selected Areas in Communications, vol. 41, no. 6, pp. 1810-1824, 2023, doi: 10.1109/JSAC.2023.3273698

[30] Z. Jiang, Y. Xu, H. Xu, Z. Wang, J. Liu, Q. Chen, C. Qiao, “Computation and Communication Efficient Federated Learning With Adaptive Model Pruning,” IEEE Transactions on Mobile Computing, pp. 1-18, 2023, doi: 10.1109/TMC.2023.3247798

[31] Z. Jiang, et al., “Computation and Communication Efficient Federated Learning with Adaptive Model Pruning,” IEEE Transactions on Mobile Computing, 2023, doi: 10.1109/TMC.2023.3247798

[32] N. Sanghi, “Deep Reinforcement Learning with Python", *Apress Berkeley*, ISBN 978-1-4842-6809-4, New York, NY, 2021

[33] E. Diao, J. Ding, V. Tarokh, “HeteroFL:Computation and Communication Efficient Federated Learning for Heterogeneous Clients,” ICLR, 2021.

[34] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, “Scaffold: Stochastic controlled averaging for federated learning,” in International conference on machine learning. PMLR, 2020, pp. 5132–5143.

[35] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, “Federated learning with compression: Unified analysis and sharp guarantees,” in International Conference on Artificial Intelligence and Statistics. PMLR, 2021, pp. 2350–2358.

[36] M. Safaryan, R. Islamov, X. Qian, and P. Richtarik, “FedNL: Making Newton-type methods applicable to federated learning,” in International Conference on Machine Learning. PMLR, 2022, pp. 18 959–19 010.

[37] A. Elgabli, C. B. Issaid, A. S. Bedi, K. Rajawat, M. Bennis, and V. Aggarwal, “FedNew: A communication-efficient and privacy-preserving Newton-type method for federated learning,” in International Conference on Machine Learning. PMLR, 2022, pp. 5861–5877.

[38] C. T. Dinh, N. H. Tran, T. D. Nguyen, W. Bao, A. R. Balef, B. B. Zhou, and A. Y. Zomaya, “DONE: Distributed approximate Newton-type method for federated edge learning,” IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 11, pp. 2648–2660, 2022.

[39] A. Agafonov, D. Kamzolov, R. Tappenden, A. Gasnikov, and M. Takac, “Flecs: A federated learning second-order framework via compression and sketching,” arXiv preprint arXiv:2206.02009, 2022.

[40] A. Elbakary, C. Ben Issaid, M. Shehab, K. Seddik, T. ElBatt, and M. Bennis, "Fed-Sophia: A Communication-efficient Second-order Federated Learning Algorithm," ICC 2024 - IEEE International Conference on Communications.

[41] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, “signSGD: Compressed optimisation for non-convex problems,” in Proceedings of the 35th International Conference on Machine Learning, vol. 80. PMLR, 10–15 Jul 2018, pp. 560–569.

[42] J. Wangni, J. Wang, J. Liu, and T. Zhang, “Gradient sparsification for communication-efficient distributed optimization,” in Proceedings of the 32nd International Conference on Neural Information Processing Systems, ser. NIPS’18, 2018, p. 1306–1316.

[43] T. Chen, G. Giannakis, T. Sun, and W. Yin, “Lag: Lazily aggregated gradient for communication-efficient distributed learning,” in Advances in Neural Information Processing Systems, vol. 31, 2018.
[44] Y. Malitsky and K. Mishchenko, “Adaptive gradient descent without descent,” in Proceedings of the 37th International Conference on Machine Learning, ser. ICML’20. JMLR.org, 2020.
[45] M. Krouka, A. Elgabli, C. Ben Issaid and M. Bennis, "Communication-Efficient Second-Order Newton-Type Approach for Decentralized Learning," 2023 IEEE Wireless Communications and Networking Conference (WCNC), Glasgow, United Kingdom, 2023, pp. 1-7.
[46] A. Ghalkha, C. Ben Issaid, A. Elgabli and M. Bennis, "DIN: A Decentralized Inexact Newton Algorithm for Consensus Optimization," ICC 2023 - IEEE International Conference on Communications, Rome, Italy, 2023, pp. 4391-4396.
[47] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, “Q-GADMM:Quantized group ADMM for communication efficient decentralized machine learning,” in ICASSP, 2020, pp. 8876–8880.
[48] J. Zhang, Q. Ling, and A. M.-C. So, “A newton tracking algorithm with exact linear convergence for decentralized consensus optimization,” IEEE Transactions on Signal and Information Processing over Networks, vol. 7, pp. 346–358, 2021.
[49] A. Mokhtari, Q. Ling, and A. Ribeiro, “Network Newton distributed optimization methods,” IEEE Transactions on Signal Processing, vol. 65, no. 1, pp. 146–161, 2016.
[50] C. Campolo, A. Iera, and A. Molinaro, "Network for Distributed Intelligence: A Survey and Future Perspectives," in IEEE Access, vol. 11, pp. 52840-52861, 2023.
[51] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. "Pruning convolutional neural networks for resource efficient inference." arXiv preprint arXiv:1611.06440 (2016).
[52] Kim, Hyeji, Muhammad Umar Karim Khan, and Chong-Min Kyung. "Efficient neural network compression." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 12569-12577. 2019.
[53] F. Malandrino, G. Di Giacomo, A. Karamzade, M. Levorato, and C. F. Chiasserini. "Matching DNN compression and cooperative training with resources and data availability." In IEEE INFOCOM 2023-IEEE Conference on Computer Communications, pp. 1-10. IEEE, 2023.
[54] Jia, Yongzhe, Bowen Liu, Wanchun Dou, Xiaolong Xu, Xiaokang Zhou, Lianyong Qi, and Zheng Yan. "CroApp: a CNN-based resource optimization approach in edge computing environment." IEEE Transactions on Industrial Informatics 18, no. 9 (2022): 6300-6307.
[55] O. Chukhno, G. Singh, C. Campolo, A. Molinaro, and C.F. Chiasserini, 2023, October. Machine Learning Performance at the Edge: When to Offload an Inference Task. In Proceedings of the 2nd Workshop on Networked Sensing Systems for a Sustainable Society (pp. 180-186).
[56] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” IEEE Commun. Surv. tutorials, vol. 19, no. 4, pp. 2322–2358, 2017.
[57] S. Bi, L. Huang, and Y.-J. A. Zhang, “Joint optimization of service caching placement and computation offloading in mobile edge computing systems,” IEEE Trans. Wirel. Commun., vol. 19, no. 7, pp. 4947–4963, 2020.
[58] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, “Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency,” IEEE Trans. Wirel. Commun., vol. 19, no. 1, pp. 235–250, 2019.
[59] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, “Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization,” IEEE Trans. Veh. Technol., vol. 68, no. 2, pp. 1079–1092, 2018.
[60] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, “Offloading in mobile edge computing: Task allocation and computational frequency scaling,” IEEE Trans. Commun., vol. 65, no. 8, pp. 3571–3584, 2017.
[61] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, “Learning-based computation offloading for IoT devices with energy harvesting,” IEEE Trans. Veh. Technol., vol. 68, no. 2, pp. 1930–1941, 2019.

[62] Y. Liu, H. Yu, S. Xie, and Y. Zhang, “Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks,” IEEE Trans. Veh. Technol., vol. 68, no. 11, pp. 11158–11168, 2019.
[63] J. Li, H. Gao, T. Lv, and Y. Lu, “Deep reinforcement learning based computation offloading and resource allocation for MEC,” in 2018 IEEE wireless communications and networking conference (WCNC), IEEE, 2018, pp. 1–6.
[64] J. Zhang, J. Du, Y. Shen, and J. Wang, “Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach,” IEEE Internet Things J., vol. 7, no. 10, pp. 9303–9317, 2020.
[65] L. Xiao, X. Lu, T. Xu, X. Wan, W. Ji, and Y. Zhang, “Reinforcement learning-based mobile offloading for edge computing against jamming and interference,” IEEE Trans. Commun., vol. 68, no. 10, pp. 6114–6126, 2020.
[66] Y. Wei, F. R. Yu, M. Song, and Z. Han, “Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor–critic deep reinforcement learning,” IEEE Internet Things J., vol. 6, no. 2, pp. 2061–2073, 2018.
[67] J. Du, F. R. Yu, G. Lu, J. Wang, J. Jiang, and X. Chu, “MEC-assisted immersive VR video streaming over terahertz wireless networks: A deep reinforcement learning approach,” IEEE Internet Things J., vol. 7, no. 10, pp. 9517–9529, 2020.
[68] L. Huang, S. Bi, and Y.-J. A. Zhang, “Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks,” IEEE Trans. Mob. Comput., vol. 19, no. 11, pp. 2581–2593, 2019.
[69] Y. Sun, S. Zhou, and J. Xu, “EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks,” IEEE J. Sel. Areas Commun., vol. 35, no. 11, pp. 2637–2646, 2017.
[70] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, “Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing,” IEEE Trans. Commun., vol. 67, no. 6, pp. 4132–4150, 2019.
[71] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, “Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems,” IEEE Trans. Wirel. Commun., vol. 16, no. 9, pp. 5994–6009, 2017.
[72] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, “Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks,” IEEE Trans. Wirel. Commun., vol. 20, no. 11, pp. 7519–7537, 2021.
[73] B. Mustapha, G. Franzese, and P. Michiardi. "Multi-modal latent diffusion." Entropy, 26(4):320, 2024.
[74] P.E. Kloeden and E. Platen, Stochastic differential equations (pp. 103-160). Springer Berlin Heidelberg, 1992.
[75] A. Jolicoeur-Martineau, K. Li, R. Piché-Taillefer, T. Kachman, I. Mitliagkas, “Gotta Go Fast When Generating Data with Score-Based Models”, https://arxiv.org/abs/2105.14080, 2024.
[76] T. Karras, M. Aittala, T. Aila and S. Laine, “Elucidating the Design Space of Diffusion-Based Generative Models”, Neurips 2022.
[77] F. Giulio, S. Rossi, L. Yang, A. Finamore, D. Rossi, M. Filippone, and P. Michiardi, "How Much Is Enough? A Study on Diffusion Times in Score-Based Generative Models" Entropy 25, no. 4: 633, 2023.
[78] H. Zheng, P. He, W. Chen and M. Zhou, “Truncated Diffusion Probabilistic Models and Diffusion-based Adversarial Auto-Encoders” https://arxiv.org/abs/2202.09671
[79] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-Resolution Image Synthesis with Latent Diffusion Models”, CVPR 2022.
[80] G. Di Giacomo, G. Franzese, T. Cerquitelli, C. F. Chiasserini, and P. Michiardi, “Multi-View Latent Diffusion,” IEEE International Conference on Big Data, 2023.
[81] J. Wang, J. Liu, J. Li, and N. Kato, “Artificial intelligence-assisted network slicing: Network assurance and service provisioning in 6g,” IEEE Vehicular Technology Magazine, vol. 18, no. 1, pp. 49–58, 2023.

[82] A. Filali, B. Nour, S. Cherkaoui, and A. Kobbane, “Communication and computation o-ran resource slicing for URLLC services using deep reinforcement learning,” IEEE Communications Standards Magazine, vol. 7, no. 1, pp. 66–73, 2023.
[83] B. Xiang, J. Elias, F. Martignon, and E. Di Nitto, “Joint network slicing and mobile edge computing in 5g networks,” IEEE ICC, 2019.
[84] A. T. Ajibare and O. E. Falowo, “Resource allocation and admission control strategy for 5g networks using slices and users priorities,” in Proc. IEEE AFRICON 2019.
[85] X. Yang, Y. Wang, I. C. Wong, Y. Liu, and L. Cuthbert, “Genetic algorithm in resource allocation of ran slicing with qos isolation and fairness,” in Proc. IEEE LATINCOM, 2020.
[86] A. A. Khan, M. Abolhasan, W. Ni, J. Lipman, and A. Jamalipour, “An end-to-end (e2e) network slicing framework for 5g vehicular ad-hoc networks,” IEEE Transactions on Vehicular Technology, vol. 70, no. 7, pp. 7103–7112, 2021.
[87] J. A. Hurtado S´anchez, K. Casilimas, and O. M. Caicedo Rendon, “Deep reinforcement learning for resource management on network slicing: A survey,” Sensors, vol. 22, no. 8, p. 3031, 2022.
[88] A. M. Nagib, H. Abou-zeid, and H. S. Hassanein, “Accelerating reinforcement learning via predictive policy transfer in 6g ran slicing,” IEEE Transactions on Network and Service Management, 2023.
[89] A. Filali, B. Nour, S. Cherkaoui, and A. Kobbane, “Communication and computation o-ran resource slicing for urllc services using deep reinforcement learning,” IEEE Communications Standards Magazine, vol. 7, no. 1, pp. 66–73, 2023.
[90] F. Song, H. Xing, X. Wang, S. Luo, P. Dai, and K. Li, “Offloading dependent tasks in multi-access edge computing: A multi-objective reinforcement learning approach,” Future Generation Computer Systems, vol. 128, pp. 333–348, 2022.
[91] A. Feriani, D. Wu, Y. T. Xu, J. Li, S. Jang, E. Hossain, X. Liu, and G. Dudek, “Multiobjective load balancing for multiband downlink cellular networks: A meta-reinforcement learning approach,” IEEE Journal on Selected Areas in Communications, vol. 40, no. 9, pp. 2614–2629, 2022.
[92] S. Sharma and W. Yoon, “Multiobjective reinforcement learning based energy consumption in c-ran enabled massive mimo,” Radioengineering, vol. 31, no. 1, pp. 155–163, 2022
[93] R. Yang, X. Sun, and K. Narasimhan, “A generalized algorithm for multi-objective reinforcement learning and policy adaptation,” Advances in neural information processing systems, vol. 32, 2019.
[94] I. Ioannou, C. Christophorou, V. Vassiliou, and A. Pitsillides, “A novel Distributed AI framework with ML for D2D communication in 5G/6G networks,” Comput. Networks, vol. 211, pp. 1–1, 2022, doi: 10.1016/j.comnet.2022.108987.
[95] I. Ioannou, C. Christophorou, V. Vassiliou, and A. Pitsillides, “A distributed AI/ML framework for D2D Transmission Mode Selection in 5G and beyond,” Comput. Networks, vol. 210, Jun. 2022, doi: 10.1016/j.comnet.2022.108964.
[96] I. Ioannou, C. Christophorou, V. Vassiliou, and A. Pitsillides, “Performance Evaluation of Transmission Mode Selection in D2D communication,” Jan. 2021, doi: 10.1109/ntms49979.2021.9432648.
[97] I. Ioannou, C. Christophorou, V. Vassiliou, and A. Pitsillides, “Performance Evaluation of Transmission Mode Selection in D2D communication,” in 2021 11th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2021, Jan. 2021. doi: 10.1109/NTMS49979.2021.9432648.
[98] W. A. Aziz, I. I. Ioannou, M. Lestas, H. K. Qureshi, A. Iqbal and V. Vassiliou, "Content-Aware Network Traffic Prediction Framework for Quality of Service-Aware Dynamic Network Resource Management," in IEEE Access, vol. 11, pp. 99716-99733, 2023, doi: 10.1109/ACCESS.2023.3309002.
[99] Attaoui, W., Sabir, E., Elbiaze, H., & Guizani, M. "Vnf and cnf placement in 5g: Recent advances and future trends." IEEE Transactions on Network and Service Management, 2023.
[100] Ferdosian, N., Moazzeni, S., Jaisudthi, P., Ren, Y., Agrawal, H., Simeonidou, D., Nejabati, R. Ferdosian, "Autonomous Intelligent VNF Profiling for Future Intelligent Network Orchestration." IEEE Transactions on Machine Learning in Communications and Networking, 2023.

[101] Sarrigiannis, I., Ramantas, K., Kartsakli, E., Mekikis, P. V., Antonopoulos, A., Verikoukis, C., "Online VNF lifecycle management in an MEC-enabled 5G IoT architecture." IEEE Internet of Things Journal 7.5, pp. 4183-4194, 2019.

[102] Zalokostas-Diplas V., Makris, N., Passas, V., Korakis, T., "Experimental Evaluation of ML Models for Dynamic VNF Autoscaling." IEEE Conference on Standards for Communications and Networking (CSCN), 2022.

[103] Jeong, S., Van Tu, N., Yoo, J. H., Hong, J. W. K. Jeong, "Proactive live migration for virtual network functions using machine learning," IEEE International Conference on Network and Service Management (CNSM), 2021.

[104] Moradi, M., Ahmadi, M., Nikbazm, Moradi, R., Mahsa, M. A., Rojia N. "Comparison of machine learning techniques for VNF resource requirements prediction in NFV." Journal of Network and Systems Management, 2022.

[105] Bunyakitanon, M., Da Silva, A. P., Vasilakos, X., Nejabati, R., Simeonidou, D. Bunyakitanon, "Auto-3P: An autonomous VNF performance prediction & placement framework based on machine learning." Computer Networks 181, 2020.

[106] Subramanya, T., Riggio, R., "Machine learning-driven scaling and placement of virtual network functions at the network edges," IEEE Conference on Network Softwarization (NetSoft), 2019.

[107] Rahman, S., et al. "Auto-scaling VNFs using machine learning to improve QoS and reduce cost," IEEE ICC, 2018.

[108] Dalgkitsis, A., L. A. Garrido, P. -V. Mekikis, K. Ramantas, L. Alonso, C. Verikoukis, "SCHEMA: Service Chain Elastic Management with Distributed Reinforcement Learning," IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 2021.

[109] Dalgkitsis, A., et al., "SCHE2MA: Scalable, Energy-Aware, Multidomain Orchestration for Beyond-5G URLLC Services," in IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 7, pp. 7653-7663, July 2023.

[110] Chen Geng, et al., "Slicing resource allocation based on dueling DQN for eMBB and URLLC hybrid services in heterogeneous integrated networks," Sensors 23.5, 2023.

[111] A. Calagna, Y. Yu, P. Giaccone, C. F. Chiasserini, "Design, Modeling, and Implementation of Robust Migration of Stateful Edge Microservices," IEEE Transactions on Network and Service Management, 2024.