Quality-aware Predictive Modelling & Inferential Analytics at the Network Edge

Natascha Sabrina Harth

Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

School of Computing Science

College of Science and Engineering
University of Glasgow

June 2021

Abstract

The Internet of Things has grown by an enormous amount of devices over the later years. With the upcoming idea of the Internet of Everything the growth will be even faster. These embedded devices are connected to a central server, e.g. the Cloud. A major task is to send the generated data for further analysis and modelling to this central collection point. The devices' network and deployed system are constrained due to energy, bandwidth, connectivity, latency, and privacy. To overcome these constraints, Edge Computing has been introduced to enable devices performing computation near the source.

With the increase of embedded devices and the Internet of Things, the continuous data transmission between devices and Central Locations reached an infeasible point in which efficient communication and computational offloading are required. Edge Computing enables devices to compute lightweight algorithms locally to reduce the raw-data transmission of the network. The quality of predictive analytics tasks is of high importance as user satisfaction and decision making depend on the outcome. Therefore, this thesis investigates the ability to perform predictive analytics and model inference in Edge Devices with communication-efficient, latency-efficient, and privacy-efficient procedures by focusing on quality-aware results.

The first part of the thesis focuses on reducing data transmission between the device and the central location. Two possible energy-efficient methodologies to control the data forwarding are introduced: prediction-based and time-optimised. Both data forwarding strategies aim to maintain the Central Location's quality of analytics by introducing reconstruction policies.

The second part provides a mechanism to enable edge-centric analytics towards latency-efficient network optimisation. One aspect shows the importance of locally generated analytical models in Edge Devices embracing each device's data subspace. Furthermore, two possible ensemble-pruning methods are introduced that allow the aggregation of individual models at the Central Location towards accurate query predictions.

The conclusion chapter presents the importance of privacy-efficient local learning and analytics in Edge Devices. With the aid of Federated Learning, it is possible to train analytical models for privacy-preserving data locally. Furthermore, for continuous changing environments, the parallel deployment of personalisation and generalisation for quality-aware predictions is highlighted and demonstrated through experimental evaluation.

Acknowledgement

Finishing this thesis has been a journey that would not have been possible without the support and help of some extraordinary people. My first thank goes to Dr. Christos Anagnostopoulos, who helped me start and finish this PhD as a great supervisor and provided even some funding from EU/H2020/GNFUV Project (Grant#645220). Thanks to my second supervisor Prof. Dr. Dimitrios Pezaros, for his feedback and support during the PhD. A special thank you to my examiners Dr. Fani Deligianni (University of Glasgow) and Dr. Konstantinos Gantis (University of Oxford), for their time and the inspiring discussion during the viva.

A big thank you to Stefanie Haeckel, my previous manager at Hewlett-Packard GmbH, who facilitated my work position to remote so I could pursue my M.Sc. in Glasgow and start the PhD. This fantastic journey could never have ended without BMW Group's support and scholarship. My thank you goes to the two essential people from BMW: Dr. Hans-Joerg Voegel and Martin Arend. Thanks for introducing me to the world of autonomous cars and industrial research. You two supported me in so many aspects that would not fit in here.

Also, a big thank you to all the people I met and formed the fantastic time I had in Glasgow during my PhD. It is impossible to mention all of them, but here are some: Anders, Antoine, Grimur, Francesco, Goezel, Paddy, Tom, William, and Frances. Thanks for the game nights, cooking challenges and many more memorable moments.

Blair and Jena, thank you for the hospitality, fun, and kindness you always have whenever I am around. I am happy to have you as friends!

A huge thank you goes to Iulia, who became the sister I never had. Without the moments you made me laugh and gave me pretzels when I struggled, I would never complete this journey. No words ever could express how grateful I am for having you in my life.

Rico, you are the person that joined me halfway through this journey and stood with me till the end. I told you it would not be easy, but you took the challenge, and I am so happy you did. Your patience, support, encouragement and unconditional love kept me going and gave me strength. Thank you for the fantastic years and everything you did. Even though this journey ended, ours just started. I am looking forward to spending the rest of my life having new adventures with you.

The most important people I owe my thanks are my mum and dad. You always believed in me and formed me into the person I am now. When I came with the crazy idea to do a PhD, you supported me and told me that you are at my side no matter what. I love you so much, and I could not have wished for better parents.

The achievement of one goal should be the starting point of another.

Alexander Graham Bell

Table of Contents

List of Tables

List of Figures

List of Algorithms

Abbreviations

A

API: Application Programming Interface
AR: Autoregressive
ARIMA: Auto-Regression Integrated Moving Average
ASM: Adaptive Smoothing Model

C

CCPA: California Consumer Privacy Act
CH: Cluster Head
CL: Central Location
CV: Coefficient of Variation

D

DPD: Dual Prediction Design
DS: dataset
DTW: Dynamic Time Warping

E

ED: Edge Device
EFM: Evolving Federated Model
EG: Edge Gateway
EPE: Expected Prediction Error
EQE: Expected Quantisation Error
ERM: Empirical Risk Minimisation
EWMA: Exponentially Weighted Moving Average

F

FL: Federated Learning
FM: Federated Model

G

G: Global Model
GD: Gradient Descent
GDPR: General Data Protection Regulation
GPS: Global Positioning System

H

HTOFS: Hybrid-Time-Optimised Forwarding Strategy

I

i.i.d.: independent and identically distributed
IAM: Input-space Aware top-$\mathcal{K}$ Model
IDM: instantaneous decision making
IEAM: Input/Error-space Aware top-$\mathcal{K}$ Model
IoT: Internet of Things

K

KDD: Knowledge Discovery in Database
KDE: Kernel Density Estimation
KL: Kullback-Leibler

L

L: Local Model
L^2*-norm*: Euclidean distance
LAN: Local Area Network
LFM: Local Federated Model
LM: Linear Regression Model
LMS: Least-mean-squared

M

MA: Moving-Average
MAE: Mean absolute error
MD-DTW: multidimensional Dynamic Time Warping

N

N-DTW: Normalised Dynamic Time Warping
NFC: Near Field Communication

O

OST: Optimal Stopping Theory

P

PCA: Principal Component Analysis
PDF: Probability Density Function

Q

QEPD: Quality-Efficient Prediction Design
QoS: Quality of Service

R

RFID: Radio-frequency Identification
RMSE: Root-Mean-Squared-Error

S

SAN: Sensing and Actuating Node
SBSM: Similarity-based selection Model
SGD: Stochastic Gradient Descent
SLW: Sliding Window
SM: Smoothing Model
SMA: Simple Model Aggregation
SMAPE: Symmetric mean absolute percentage error
SPD: Single Prediction Design

T

TOFS: Time-Optimised Forwarding Strategy
TOSM: Time-Optimised Switching Model

U

UK: United Kingdom

W

WAN: Wide Area Network
WLAN: Wireless Local Area Network
WSN: Wireless Sensor Network

Chapter 1

Introduction

1.1 Motivation & Challenges

The development from traditional client-server architecture towards more advanced distributed systems, including small devices such as cars, smartphones, and other Internet of Things, has been grown in the last years. Gartner predicted 5.8 billion devices to be connected and deployed, just in the enterprise and automotive industry [1]. These devices continuously monitor the environment through sensor technology, permitting the collection of valuable data. This data enables applications to perform highly personal and customised analytics to the needs of the user. The variety of assembled applications using the Internet of Things increased over the years, from automotive self-driving cars, smart cities, or healthcare deployments to intelligent stores, smart homes, or even manufacturing with Industry 4.0 deployments.

With the increase of applications, the considerable amount of data collected from these devices evoked multiple constraints, such as bandwidth, storage, and energy. The need for efficient methods to overcome those constraints has precipitated performing analytics, inference, and learning as close as possible at the data source. In the last years, the hardware size decrease enabled the devices to become extremely powerful and perform computational tasks directly inside themselves. Edge Computing is the research area that empowers the capability of devices to perform computation inside the devices and therefore near the generated data. This closeness to the data source tries to overcome the issue of transferring enormous amounts of data over the network and enable real-time decision making. Using Edge Computing as a proxy to combine machine learning and analytics can create knowledge and insights from the environment and people in real-time.

Several challenges emerge in Edge Computing when focusing on performing analytics and machine learning in devices. Most devices are operated on batteries emphasising an efficient usage of these resources when deploying analytical functions, especially when these devices are deployed in unreachable locations. The most energy consumption's critical part is sending the measured data via the network connection to a central collection point. Therefore a significant research area of Edge Computing uses the computational power of devices to enable intelligent decision making in delaying or selecting the data forwarded to this Central Locations. The constraints of devices in this environment is, besides the battery, also limited storage and computational power. These additional constraints restrict the usage of low complex algorithms and machine learning models inside the device. The environment the devices are placed in is mostly changing and continuously evolving. Therefore, algorithms and models using continuous learning and inference are of considerable importance. Deploying inference and learning at the device allows personalised real-time analytics and inference and the ability to adapt immediately to rapidly changing environments. This thesis is consequently contributing to very fast, continuously sensing and changing environments or applications. Crucial for most applications deployed is the accuracy and reliability of the analytical outcome as decisions on behaviour and user satisfaction are made based on the prediction outcome. Emerging over the last years as a crucial aspect in the machine learning area is data protection and privacy. As devices are applied in user personal environments, most of the collected data require privacy protection.

Based on these highlighted constraints and emerging areas, this thesis adds towards using Edge Devices' capacity to selective forwarding and delaying data transmission. Data transfer reduction has been encountered using low complex mathematical and statistical algorithms to allow selective intelligent forwarding by maintaining qualitative analytics at the central collection location. Moreover, using the possibility to place intelligence at Edge Devices and adapt to changing environments motivated the work to enhance the research on locally deployed analytics for real-time decision making by using personalised local models instead of generalised central. A model retraining and forwarding methodology is proposed, as the deployment of local models inside devices must be efficient and energy balanced. Additionally, the problem of combining multiple learners (each device is a local learner) through ensemble pruning using quantisation and similarity of the data space is explored and evaluated. The introduced local learning emphasised exploring the necessity of privacy through local, personalised model deployment in devices. Personalised and local learning allows transferring parameters instead of raw data. In the thesis, a methodology is proposed that uses privacy and efficient local learning for continuous changing environments to qualitatively select the best model inside the device to predict future behaviour.

1.2 Thesis Statement

Performing analytical tasks and machine learning over data from Edge Devices, such as the Internet of Things experience latency, bandwidth, and privacy constraints due to communication overhead and data processing at a Central Locations. This thesis provides a communication, latency, and privacy-efficient methodology through enabling analytics at the source of the data - the edge. Intelligent decision-making mechanisms combined with collaborative intelligence between the edge and the Central Locations are presented to reduce the communication and empower local learning. This thesis further concentrates on the quality of analytics in privacy-preserving environments with real-time local learning and inference.

Therefore this thesis is researching the following three Hypotheses:

Hypothesis 1: Pushing computational intelligence of advanced decision-making in data forwarding to the edge of the network will overcome energy and bandwidth constraints due to the deployment of efficient communication methodologies. Combining this with intelligent reconstruction at a collection point leads to highly accurate analytical tasks and reconstruction of the imputed values.

Hypothesis 2: Enabling machine learning and predictive analytics locally at Edge Devices will empower real-time applications that can adapt intelligently to concept drifts and changes of the continuous data arriving. These locally learned (trained) models can be selected through qualitative model selection methodologies at central coordinators, e.g., Cloud.

Hypothesis 3: Generalised models over privacy-preserved data by only transferring analytical model parameters over the network will not provide qualitative results in constantly changing and heterogeneous environments. Using the prospect of locally learning models with an intelligent model selection and weighting mechanism for personalisation and generalisation in Edge Devices enables data privacy and qualitative prediction results.

1.3 Contributions

The research contributions in this thesis are structured based on the specific chapters where the contribution is placed. Generally, this thesis presents methodologies for advancing the quality of analytical tasks performed at Edge Devices by focusing on efficiency regarding energy and bandwidth. To this end the following aspects are contributed:

Chapter 3 Contributions:

1. A comprehensive evaluation of state-of-the-art prediction-based data forwarding strategies with the identical deployment of models in sensors and Edge Devices (collection point) to minimise the communication overhead and the ability to reconstruct the missing data.
2. Exploring the computational ability of sensors by introducing a quality-efficient prediction based forwarding strategy inside them with investigating the upper bound of the given threshold to minimise the quality difference towards reconstruction.
3. Introducing a time-optimised data forwarding strategy based on Optimal Stopping Theory (OST) [2], to find the optimal time to send the measurements based on historical decisions and a defined rewarding function.
4. Extensive experimental evaluation on actual data to evaluate the trade-off between reconstruction and analytical functions regarding their quality by reducing communication through the implemented intelligence mechanism and reconstruction methods at the collection point.

Chapter 4 Contributions:

1. Evaluation of research gaps in edge-centric localised analytics and model forwarding strategies combined with the absence of model selection criterion in edge environments.
2. Introducing a computational and communication efficient model retraining and forwarding approach based on familiarity and degree of change for edge environments with resource constraints to aim for qualitative predictions at the Central Locations.
3. Developing an efficient ensemble pruning strategy for model selection over locally generated models provides highly accurate results for query-driven user requests based on input/error-space quantisation and similarity-based clustering.

4. Real data set evaluation and extensive experiments of latency-efficient model re-training, forwarding and selection using the proposed techniques and approaches to identify the trade-off between accuracy and efficiency.

Chapter 5 Contributions:

1. A complete assessment of current research in private-efficient local learning over resource constraint environments regarding personalisation and continuously evolving data, mainly focusing on Federated Learning concepts which provides more privacy by design.
2. Exploring the ability to perform local model training inside devices to incorporate the heterogeneity of environments and provide qualitative predictive analytics inside the device and centrally.
3. Introducing the importance of balancing personalisation and generalisation in changing environments, such as the Internet of Things, focusing on adaptive weighting and optimising the time of swapping with low complex deployments for efficient implementations inside the devices.
4. Experiments and evaluation on a real dataset to show the improvements of local, personalised Federated Learning with the introduced strategies over edge environments.

1.4 Publications

The research presented in this thesis is entirely the author's own work. This thesis exploits only the parts of these papers that are directly attributed to the author. The following publications are related to the results and the work presented in this thesis. Especially, some proofs and mathematical concepts have already been presented in the peer-reviewed published work and only re-adjusted for this thesis:

- Natascha Harth, Hans-Joerg Voegel, Kostas Kolomvatsos and Christos Anagnostopoulos. ”Local Learning at the Network Edge for Efficient & Secure Real-Time Predictive Analytics” arXiv preprint arXiv:2109.12375, 2021 [3]
- Natascha Harth, and Christos Anagnostopoulos. ”Edge-centric efficient regression analytics.” IEEE International Conference on Edge Computing (EDGE 2018). San Francisco, CA, USA, July 2018, pp. 93-100 [4]
- Natascha Harth, and Christos Anagnostopoulos. ”Quality-aware aggregation & predictive analytics at the edge.” In: IEEE International Conference on Big Data (IEEE Big Data 2017). Boston, MA, USA, Dec.2017, pp. 17-26 [5]
- Natascha Harth, Christos Anagnostopoulos, and Dimitrios Pezaros. ”Predictive intelligence to the edge: impact on edge analytics”. Evolving Systems, 9(2), Aug. 2017, pp. 95-118. DOI:10.1007/s12530-017-9190-z [6]
- Natascha Harth, Kostas Delakouridis, and Christos Anagnostopoulos. ”Convey intelligence to edge aggregation analytics.” New Advances in the Internet of Things. Springer, Cham, June 2017. pp. 25-44. DOI: 10.1007/978-3-319-58190-3_2 [7]

1.5 Organisation of the Thesis

The structure of this work is presented as follow:

Chapter 2 presents an extensive literature review of relevant work in distributed systems, machine learning and Edge Computing. This chapter starts with the evolving from client-server architecture to the current distributed systems, including the Internet of Things and Ubiquitous Computing. The chapter further highlights the different machine learning types and current research topics of machine learning over continuous and distributed data processing. Finally, the combination of distributed systems and machine learning in Edge Computing is illustrated with the current work and constraints this deployment has.

Chapter 3 introduces efficient and quality-focused communication reduction of data between sensing devices and Edge Devices. Current state-of-the-art research of intelligent decision making using predictive forwarding are characterised. Besides one additional intelligent data-forwarding method using prediction-based forwarding, an additional data-forwarding strategy of time-optimisation is introduced in this chapter. Furthermore, different reconstruction policies towards qualitative imputation of non-forwarded values will be accentuated.

Chapter 4 enhances the intelligence of Edge Devices by introducing latency-efficient model forwarding strategies. This chapter emphasised the edge device capabilities of locally performing analytics, inference and machine learning tasks. The data-forwarding mechanisms of Chapter 3 are enhanced in this chapter by introducing familiarity based model-forwarding methodologies. Moreover, ensemble pruning techniques for model selection based on input/error quantisation and similarity-based are introduced. The introduced strategies will enable efficient communication between the Edge Device and Edge Gateway and improve real-time action and decisions over changing environments.

Chapter 5 departs from the efficient latency-efficient model-forwarding and selection of Chapter 4 towards data privacy concern in Edge Computing. A methodology of privacy-efficient and qualitative local learning over continuous evolving environments is presented in this chapter. A model selection method to enable personalised learning while maintaining a global generalisation using the fundamentals of Federated Learning is highlighted throughout the chapter.

Chapter 6 provides an overview of the conducted work in this thesis with revising the thesis statement and hypotheses, highlighting future work and some limitations that have not been considered for this thesis.

Chapter 2

Literature Review

2.1 Chapter Overview

The area of Edge Computing and analytics in which this thesis is placed has its fundamentals in two parts: distributed systems and machine learning. This literature review introduces these two main fields with their definitions and currently deployed techniques relevant for edge analytics.

Distributed systems evolved from essential technologies such as client-server architecture to the area of Pervasive Computing. The first part of this chapter (Section 2.2) highlights the evolution of distributed systems and introduces the terminology used throughout the presented work. The main focus is on Ubiquitous or Pervasive Computing, which aims to generate context-aware and real-time systems using everyday objects, including the Internet of Things (IoT) and Wireless Sensor Network (WSN). From the enormous increase of applications, the need emerged to perform computation and analytics nearer the data source. In this section, the possibility is highlighted to use different data-processing levels within distributed systems, such as Cloud Computing and the emerging technology of Edge Computing.

Besides the system perspective, the thesis employs techniques and methods of the area of machine learning and predictive modelling to enable the use of edge learning and analytics. Therefore, the second part of this chapter (Section 2.3) introduces the definition and rationale of machine learning with all required formulations. It further presents the different variations existing and the usage in deployed applications. In real-time distributed systems, machine learning is deployed under unique characteristics. Especially in small computing devices, analytics and machine learning require unique distributed and continuous learning techniques to be deployed efficiently and scalable. Therefore, this chapter highlights the current work in these areas and explains the underlying techniques, such as window-based learning and model selection over distributed learning.

The last section of this chapter (Section 2.4) focuses on combining the fundamentals of machine learning and distributed systems towards Edge Computing and analytics. It highlights machine learning challenges in Edge Device (ED) and their benefits towards real-time and low latency applications. Finally, currently deployed methodologies for edge inference and learning are introduced, which are fundamental work this thesis builds on.

2.2 Distributed Systems and Evolution

2.2.1 Definitions and Characteristics

Computer systems arise in the 1950s in which all applications and computations are designed to function on a centralised system, meaning one computer processes and operates independently. Centralised computer systems tend to be large and powerful but also expensive. The evolving nature of personal computers and workstations and the invention of network technologies such as the Local Area Network (LAN) distinguished the need to share resources such as printers, data, software, or storage with multiple personal computers. With this demand, the research area of distributed systems emerges. A centralised system perspective is defined as everything is stored and processed on a single machine. In comparison to this, a distributed system should be "a collection of independent computers that appears to its users as a single coherent system" [8]. Additionally, a distributed system should be a system "in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages" [9]. It is essential to accentuate that traditional distributed systems are autonomous components that do not share any memory and do not have a global clock or time synchronisation. Therefore independent failures occur, but also concurrent tasks are possible.

From these definitions of distributed systems, seven main characteristics and challenges can be derived [8, 10]:

1. Resource Sharing is the primary impulse and means that within the system, all resources should be available to every user or component at any time, e.g., hardware, software, or data;

2. The challenge of heterogeneity in distributed systems defines that components vary throughout the system in the form of operating systems, programming language, hardware or software;

3. The characteristic of openness of a system identifies that the system should be accessible to add new components anytime to the network;

4. The importance of concurrency derives from the previous characteristics, an emphasis that the system has to guarantee, even with resource sharing, that the underlying data or service is consistent at any time;

5. Scalability is a characteristic and challenge, which relates to the openness of the system, intending to highlight that with the increase of added components, the performance should not decrease;

6. The ability of fault tolerance highlights that with the failure of one component, the system is still running and ideally, the user should not notice that one component is not working so that the characteristic of continuous availability is guaranteed;

7. The last challenge is transparency accentuates that the communication between the components is not visible for the user, which is the main point of defining a distributed system.

Inside a distributed system, different entities are connected to communicate with each other forming a network [11, 12]. These entities are called nodes, expressing any device inside a network [13, 14]. A network is how components interact and communicate with each other to share resources. The crucial difference between a computer network and a distributed system is that the user is exposed to the actual machines without any attempt to use transparency. Therefore a computer network is the underlying infrastructure of a distributed system without the software that will provide non-transparency of the framework for the user.

A distributed system can be designed using an architecture model defining the autonomous components' structure, order, and interaction. Therefore a system architecture can be seen as a formal representation of a system with the components, interactions, and functionalities ordered.

Distributed Computing is often associated with the term distributed system. However, the definition of a distributed system differs slightly from the computing paradigm. Distributed Computing means to solve an algorithm, software, or program using a distributed system [15, 16, 17]. Modern distributed systems are primarily implemented to perform Distributed Computing tasks.

2.2.2 Contemporary Distributed Systems

Nowadays, deployed distributed systems are of complex order and often include sub-systems. Understanding the underlying architecture is key to master the current deployed distributed systems. A distributed system architecture can be divided into two classical components representing a centralised and decentralised structure of the autonomous components [9, 18]:

1. Client-Server: this type of centralised architecture is deployed when each entity (client) is connected to a central server that does the computation, coordination, or service, whereas the client is just requesting this service.

2. Peer-to-Peer: this type of decentralised architecture allows each entity to act as a client and as a server. Each entity of the system is interconnected; no central coordination or computation exists.

The two classical architecture models expand over time towards more complex system architectures of combinations between a peer-to-peer and client-server model. This combination of architecture styles results in a hybrid model and the deployment of systems in systems. With the complexity of the systems, the client-server model itself evolved. This architecture model has been branched into a logical functionality orientated architecture and a physical tiered orientated architecture. The logical architecture is mainly divided into three or four main functionalities represented in most current distributed systems [19, 20]: perception level, network layer, processing level, and application level. The physical architecture introduces tiers that allow multiple components to deploy, acting as either server or client. Most commonly implemented is a three-tier architecture, in which each of the tiers is dedicated to one specific logical functionality. Modern distributed systems are merging the levels into multiple tiers to provide an optimal and efficient system.

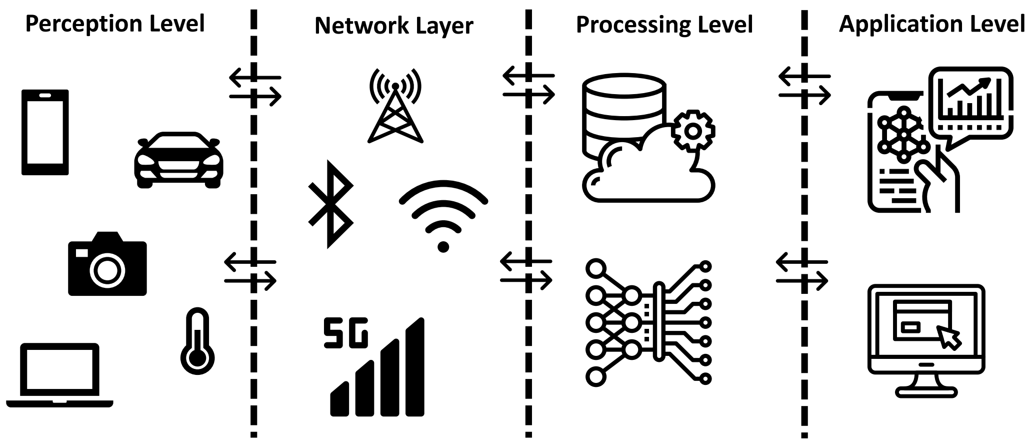


Figure 2.1: Example architecture of a contemporary distributed system.

In Figure 2.1[1], an overview of a complex distributed system is displayed using the four-level architecture style. This illustration makes it possible to identify a complex contemporary distributed system's main functionalities and perspectives. The first applicable level is the

[1]Some illustration icons used from [21]

perception or device level, typically collecting data through sensors that monitor the environment. These sensor systems and networks can form a peer-to-peer sub-system to interact and communicate with each other. The following logical functionality perspective is the processing and computation level. This level is responsible for storing, analysing, and performing any computation and processing of the data. The third perspective and its functionality illustrated in this figure is designed for the user or application. This level presents the user, the data analysis, or any other application. Finally, all of the above levels are connected through a transport or network perspective [22, 23, 24]. The network layer is responsible for forwarding the generated data and connecting the devices using any communication technology appropriate for the range, such as WiFi, Bluetooth, or 5G.

2.2.3 Application Perspective

Modern distributed system applications and use cases are gigantic and diverse. In the previous section (Section 2.2.2), one broadly used modern system design focusing on data-driven and context-aware applications [25, 26] has been highlighted. The generated data by sensors and devices deployed everywhere can be processed using the distributed system infrastructure to make the user's application outcome visible. These applications require different communication types between the devices and the processing or application level [27]. These communication types can be divided into three categories (note that this thesis is focusing on continuous data):

1. Continuous: the devices are constantly reporting towards the central service after receiving or sensing the measurements. This constant transmission can be periodically or permanently.
2. Sporadic: the device reports the data at any time a connection can be established or if a certain level of energy is reached at the node. At these times the connection is established and the data is transmitted from the device to the processing level.
3. On-Demand, this communication type is further divided into:

   Event-Based: the devices and sensors only report data to the processing level after locally or collaboratively identifying a pre-defined event.

   Query-Based: the devices only report data after a user or the system is centrally requesting the data in the form of a query arriving at the device level.

Considering a type of contemporary system, the applications and use cases deployed can be classified into seven categories. These categories evolved from simple monitoring using sensor networks [28, 29, 30, 31] towards intelligent and smart applications acting based on the generated data using an IoT infrastructure. These seven application categories are:

1. Home: home applications can include smart grid and energy applications with smart meters providing an intelligent system for monitoring or regulating energy usage, or applications for securing the house such as intruder detection or children monitoring and alerting, or towards personalised entertainment such as music, television, gaming, or applications towards simplifying daily life tasks such as shopping [32, 33, 34].

2. Health: applications implemented for health vary from health monitoring [35, 36, 37, 38] for treatments or prevention towards elderly-care. Health applications can be for patients, hospitals, insurances, or physicians with many possible use cases [39].

3. Cities: application deployed for smart cities [40, 41] can vary from monitoring buildings or infrastructure [42, 43] towards smart traffic, pollution control [44, 45] or intelligent waste management.

4. Mobility: its movement characteristic mainly defines an application's belonging to the category of mobility, including aeroplanes [46], cars [47], public transportation, emergency services, or unmanned vehicles.

5. Environmental: in this category, applications such as general environmental monitoring such as habitat monitoring [48], fire detection [49], agriculture [50, 51], volcano monitoring [52], or underwater monitoring are included [53].

6. Industry: applications for industry use cases are primarily deployed for machine-to-machine communication and the usage of Industry 4.0 [54, 55] areas such as smart production, supply chain management, retailing, logistics, and many more.

7. Military: military applications could be the usage of unmanned vehicles, or applications combining environmental monitoring, health monitoring for people, or vehicle monitoring (ships, cars, planes) to identify attacks, crime, or optimise processes [56].

2.2.4 Device and Perception Perspective

2.2.4.1 Pervasive and Ubiquitous Computing

The previous section has identified that modern distributed systems evolved from simple clients represented by personal computers towards objects that can interact as autonomous components with a server. The area of distributed systems emerges from architecture and components designed for a stable network and connection approaching unstable connection and mobile components. The technology evolution of Local Area Networks (LANs) to Wide Area Networks (WANs) using wireless communication technologies, as well as the downsizing of computational devices results in the possibility of embedding computation into physical objects. These objects allow the mobile use of computers. This research area of distributed embedded systems is also known as ubiquitous or pervasive systems [57, 58, 15]. Departing from the classical distributed system in which the nodes or components are stable in a wired connection and fixed at a specific location, pervasive systems are mobile and rely on a wireless connection, which is unstable. Ubiquitous Computing has been introduced by Weiser et al. [59]. The authors elaborate their vision of embedded devices into physical objects and how their interaction with users should be invisible. The following requirements can characterise ubiquitous and pervasive systems [60]: (1) Human interaction should be hidden and reduced as much as possible so that entities operate autonomously; (2) The system should be context-aware and able to adapt and act based on the sensed surrounding using intelligent decision-making or other forms of intelligence.

2.2.4.2 Sensor Networks

In the previous section (Section 2.2.4.1), the research field of Pervasive or Ubiquitous Computing as part of a distributed system has been introduced and defined. This research area emphasised that devices are embedded in objects able to sense their surrounding environment. Implementing these concepts, the infrastructure of sensor networks appeared. Sensor networks can be divided into wireless or wired connected setups. This chapter focuses on Wireless Sensor Networks (WSNs) [61, 62, 63], as they are widely used in contemporary distributed systems. A WSN is organised by several sensor nodes called Sensing and Actuating Nodes (SANs). These nodes are embedded computing devices capable of sensing their surrounding environment and translating this information into data. They can vary in size but are mostly small and of low cost. In most WSNs, hundreds or even thousands of nodes are positioned.

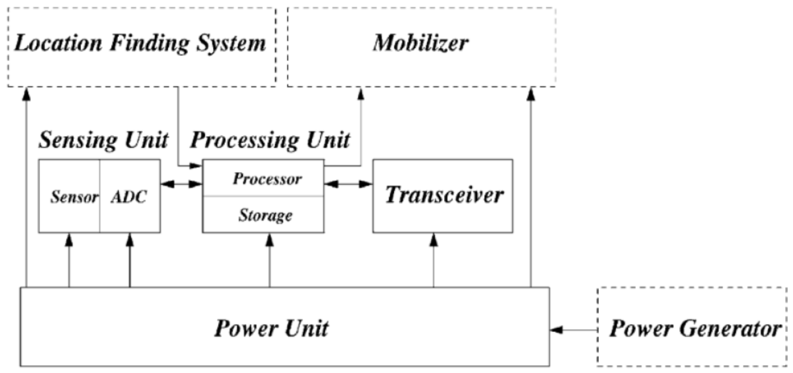


Figure 2.2: Example of hardware components used for Wireless Sensor Network devices [64].

The build-in components of these nodes can change depending on the price and the application. However, they are equipped with a power unit (battery), a sensor, a processing unit containing storage and a processor, and a transceiver with an antenna to communicate with other nodes. Some of these nodes can also have Global Positioning System (GPS) units to identify their location or a power generator for additional power or even some kind of mobiliser to move the node to a different location. In addition, different types of sensors are deployed to collect a variety of environmental data. These sensor types can be thermal such as temperature or humidity, visual such as lightning, acoustic for noise level, infrared, radar for vehicle movements, magnetic for flaw detection in industry, and seismic for motion intruder detection. An overview of a typical sensor node deployed in WSNs with its hardware components is illustrated in Figure 2.2.

As highlighted, SANs are deployed to collect data and forward this input to the base station. The base station, sink node, or collection point usually has computational power and energy and can perform aggregation of the collected data it received from all connected sensing nodes. Another responsibility made to the base station is forwarding this aggregated information to a Central Location (CL) such as a database, server, or Cloud. At this Central Location, further processing and analysis of the data is carried out. Figure 2.3 shows a typical network architecture of a WSN in which the sensor nodes are ordered in a tree network [65]. This results that SANs deliver the data first to the base station, which can communicate with the Internet to forward the data to a processing location.

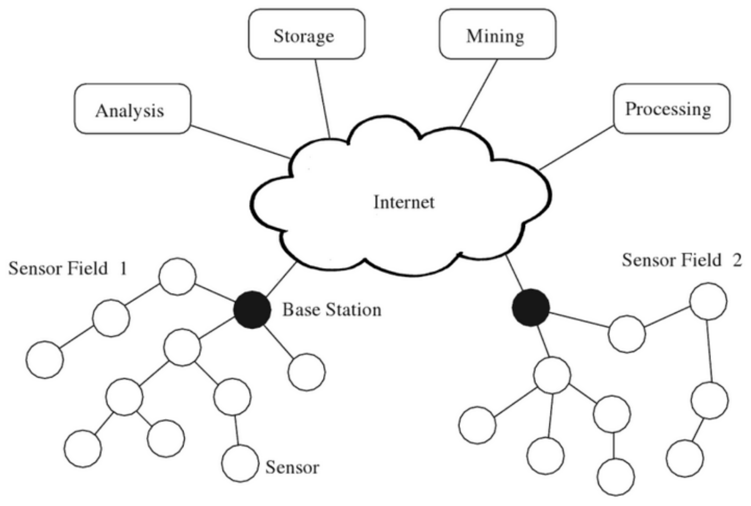


Figure 2.3: Representative Wireless Sensor Network architecture [66].

Even though the possibilities of deploying such low budged sensors in a distributed system network using wireless connections seem to be infinite, some key features and challenges need to be considered while designing a WSN as they provoke problems and limitations [64, 27, 29, 67]. It should be noted, that these features are relevant for many deployed applications but not for all or in different magnitude.

As the number of SANs can be thousands, the high scalability is the first and critical feature of a WSN. SANs can either be deployed in an organised way or at random. New devices should be able to join the network at any given time. This flexibility results in the next feature, the high heterogeneity of hardware, software, and resulting performance. Not only is it possible to position different sensor types, processing units, or power units, but also during the lifetime, heterogeneity can occur if one node performs more processing than another node so that the energy is lower than of the other peer nodes. This heterogeneity of sensing nodes is a significant challenge, but also a huge opportunity. An already stated critical feature of WSNs is the power consumption. The constraints prevail as mostly small batteries are used as energy components, or the devices are deployed in unreachable locations, so it is impossible to recharge them easily. Therefore its lifetime has to be maximised with energy-efficient operations. Not only do sensor nodes have low energy resources but they are also constraint in computation. The embedded processing units can compute some functions, but they have to be of low complexity. The transmission unit faces constraints in its range of sending and receiving messages. Primarily deployed are radio waves such as Bluetooth or WLAN, resulting in limited data throughput and possible disconnections. All of the above-mentioned critical features of WSNs, resulting in high node failures, highlighting another aspect of consideration while designing a WSN.

2.2.4.3 Internet of Things

Ashton [68] introduced the expression of Internet of Things (IoT) in 1999. IoT has emerged from the Ubiquitous Computing paradigm in which everyday objects should be enriched with embedded computational power towards monitor the environment without any user interaction. IoT systems aim to monitor everyday objects in real-time and over the Internet to collect, process, and analyse the data to generate knowledge [69]. In particular, Cloud Computing for analysis and further processing is used as the primary Central Location for IoT systems [70]. The physical world should be digitised with the usage of IoT. The IoT paradigm is developed with the emergence of Cloud Computing and the already existing technology of WSNs and Radio-frequency Identification (RFID) chips that allow identification, tracking, and monitoring objects over the Internet [30, 71, 72]. Summarised the three key components of IoT are [73]:

- Each device is identifiable through RFID or any other digital name;
- Devices need to communicate among themselves and over a network;
- Devices need to interact with the environment using sensors and should act based on the knowledge generated

WSNs can be part of IoT systems, but not vice versa. To include WSNs into IoT systems, they can be enriched with a gateway or other computational device that can communicate, process, and send the data over the Internet to a Central Location for further processing [74, 75]. Comparing the IoT and WSNs, the critical difference is the size of deployed devices. In a WSN, thousands of devices are usually connected, whereas, in IoT networks, billions of devices are expected to be connected [69]. These devices can be wired or wireless, and they have to be connected to the Internet. In a WSN, these devices are external sensing devices attached or placed to monitor the environment mainly. In contrast, IoT devices are physical objects with embedded sensors, such as a fridge or other home appliances, a watch, or a vehicle that is not only to be there to monitor, but further to analyse the data and act based on the knowledge generated. IoT devices are equipped with the ability to communicate, compute, store, and sense data.

2.2.5 Network and Transportation Perspective

Continuing from Figure 2.1 detailing the layers of a contemporary system, the network layer connects the device level with the processing level and application level using leading technologies for communication in modern distributed systems and networks such as NFC, Bluetooth, ZigBee, WLAN, and 5G. Depending on the range of communication and throughput rate, one of the mentioned technologies can be chosen. However, the higher the throughput and broader the range, the more energy the communication requires from the device that performs the transmission. Figure 2.4 shows an overview of the different wireless technologies and their capacities with throughput clustered into four categories. The authors in [76] provide a good overview of technologies and challenges for IoT environments.

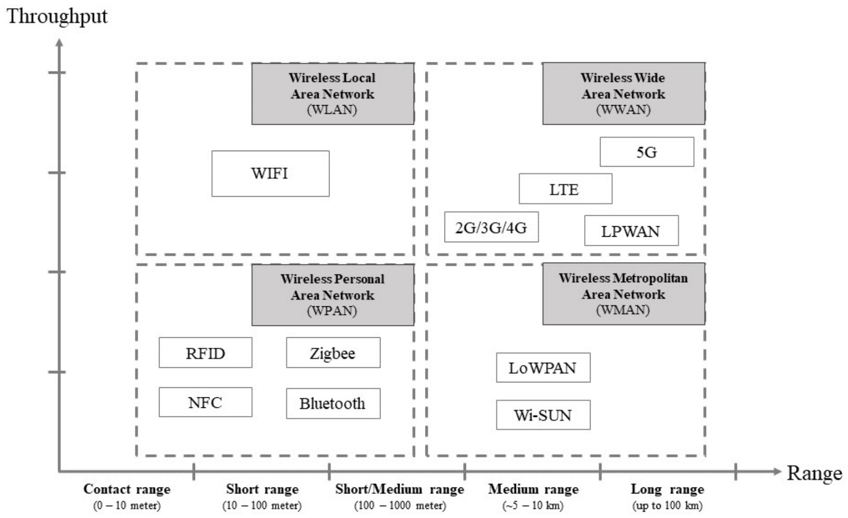

Throughput
Wireless Local Area Network (WLAN)
WIFI
Wireless Wide Area Network (WWAN)
5G
LTE
2G/3G/4G
LPWAN
Wireless Personal Area Network (WPAN)
RFID
Zigbee
NFC
Bluetooth
Wireless Metropolitan Area Network (WMAN)
LoWPAN
Wi-SUN
Range
Contact range (0 – 10 meter)
Short range (10 – 100 meter)
Short/Medium range (100 – 1000 meter)
Medium range (~5 – 10 km)
Long range (up to 100 km)

Figure 2.4: Overview of wireless communication technologies sorted by their range and throughput grouped in categories.

The network layer's key aspect is forwarding the data in the form of raw sensed data, representations of the data, or decisions. Facilitating this responsibility, establishing a network topology of how the nodes in a network are connected is essential. Topology describes how the nodes are arranged to interact and communicate with each other in a network. The most deployed network topologies are [22, 23, 24]: bus, star, ring, mesh, tree, and other hybrid forms. In Figure 2.5, an overview of the different topologies is illustrated, in which a circle represents an individual node in a network, e.g. a SAN.

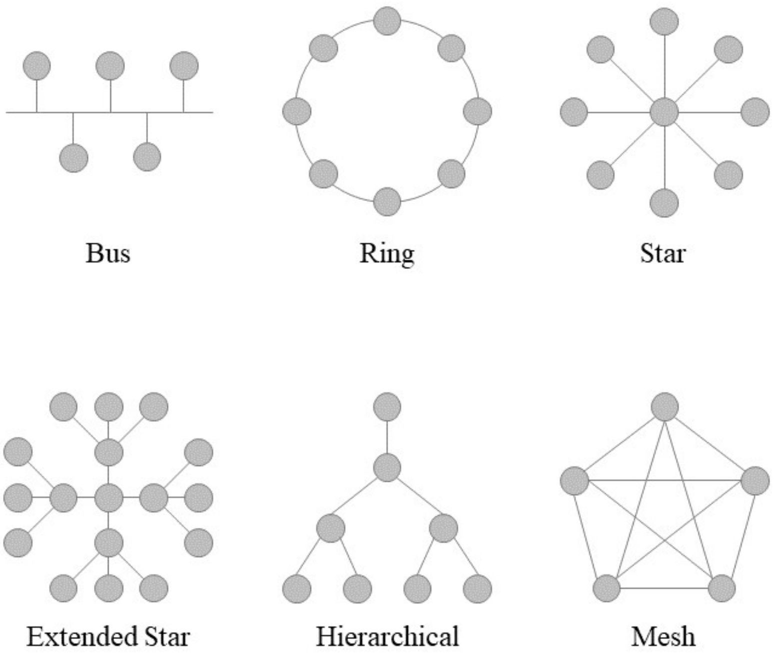


Figure 2.5: Overview of example network topologies.

Given the design of the network topology, it is possible to define the compatible routing protocol. A routing protocol contains information of the delivering path of the data or information transported and communicated from one node to others or the Central Location. It mostly takes the constraints of the network, system, and device constraints into consideration, enabling intelligent routing mechanisms with advanced technologies.

2.2.6 Computation and Data Processing Perspective

2.2.6.1 Cloud Computing

In the last 20 years, data centre deployment costs increased, especially when companies try to maintain the newest technology. The increase of data collection and the need to perform analysis, requires storage and computational power during the run time. This computational power is not needed afterwards, enabling the need for on-demand storage, computation, and software. The terminology of Cloud Computing is defined as: “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [77]. As the definition states, Cloud Computing is an enabler to allow access to resources everywhere and anytime. Resources can be software, infrastructure, or a platform that can be used as a service. Thus, Cloud Computing is a form of Utility Computing that allows users or companies to use resources based on their needs.

Two leading Cloud technologies are currently deployed [78, 79]: the Private Cloud and the Public Cloud. As the names are already projecting, the Public Cloud is a pay-as-you-go service for every user that needs some service organised and managed by the Cloud provider. Private Cloud systems are specially designed for companies that need their infrastructure and data to be secure without any other person, company or system accessing it. Therefore, the resources are build and provided only for a specific company without any sharing or access from outside the allowed circle. It should be noted that there is also the possibility of hybrid or community Cloud technologies, representing a merge of Public and Private Cloud systems in the form of different magnitudes of accessibility.

When connecting Cloud Computing and distributed systems, it is possible to place Cloud Computing’s system infrastructure towards Grid Computing or Cluster Computing [80, 81]. Grid and Cluster Computing aims to distribute tasks that require high computational resources towards a cluster of computing devices or server. They are formed dynamically over a network able to share the resources needed to compute the given task. In traditional Grid Computing, the infrastructure is owned by a company. In contrast, Cloud Computing is managed centrally by an external provider and is service-oriented. Distributing a specific task by forming a cluster of computing resources is also given in Cloud Computing, which involves different servers clustered to perform the user’s service. Thus, Cloud Computing uses a different business model than traditional Grid Computing systems by having an extensive infrastructure and renting it to users by their needs.

Overall, Cloud Computing [82, 83] has the advantage from a user perspective of using only needed resources and extending and reducing these resources anytime without high costs and time for installation. Further, the payment is usage-based and straightforward, allowing increased flexibility. Not only is the reliability guaranteed by the provider, but it also allows a manageable and mobile use of the services, only using the Internet, resulting in having access anywhere and anytime. However, besides the advantages of Cloud Computing, it also faces significant disadvantages such as security, bandwidth, and latency.

2.2.6.2 Edge Computing

Edge Computing is a terminology that arose in the last years with a strong relation to the disadvantages of Cloud Computing and the enormous increase of IoT devices. Most deployed IoT architectures are of centralised order [84], resulting in transmitting all the generated and sensed data towards the Cloud to process, analyse, and develop knowledge. However, with the expansion of IoT devices [69], this transmission with processing and storing at a centralised system is infeasible, even for Cloud systems. The main limitation of a Cloud system can be highlighted by applications and systems that require low latency and real-time actuating. The need emerge to overcome these issues and perform and act instantly in IoT devices. Not only is latency an essential accelerator for device-located processing, but also the limitation of bandwidth resulting from transmitting all generated data from all IoT devices. As devices tend to monitor sensitive data to enable context-aware applications, privacy concerns about the data occur with the need and regulations to limit data transmissions towards the Cloud and centralised systems. Therefore, it is essential to increase the computation by pushing intelligence towards the edge of the network [85, 86, 87], forming decentralised processing of the data.

Different technologies in the research community have proposed to achieve this push of computation and intelligence to the edge (device level). One of them is cloudlets, a localised Cloud Computing server [88]. It is sometimes referred to as a micro data centre. Cloudlets aim to introduce a small Cloud server more localised to enable a lower latency but still provide a reasonable amount of computational power. This localisation can be made applicable by the following example. Instead of using the USA's cloud system for a service performed on in Germany deployed devices, one cloudlet can be implemented in Germany as a layer between device and cloud (in the USA) to pre-process the data. Fog Computing [89, 90, 91] aims to process the data another layer down and even closer to the device level than cloudlets or micro data centres. A fog server would be deployed in each city in the previous example, whereas a cloudlet would be countrywide. Fog Computing and cloudlets are very similar by deploying the computation and data processing even nearer to the device by introducing another tier or level of pre-processing and computation. In some literature, Fog Computing and Edge Computing are merged into one. However, for this thesis, Edge Computing is defined as processing the data at the IoT-device level. Not anticipating a WSN but the gateway connected to the Internet, the smartphone, the car, or any other computational equipped devices capable of processing data.

In Figure 2.6[2], the relationship between Cloud, Fog, and Edge Computing combined with IoT devices and a WSN is illustrated. It should be noted that the fog/cloudlet server is in this figure only one level, but this can be extended towards multiple levels or tiers, getting a

[2]Some illustration icons used from [21]

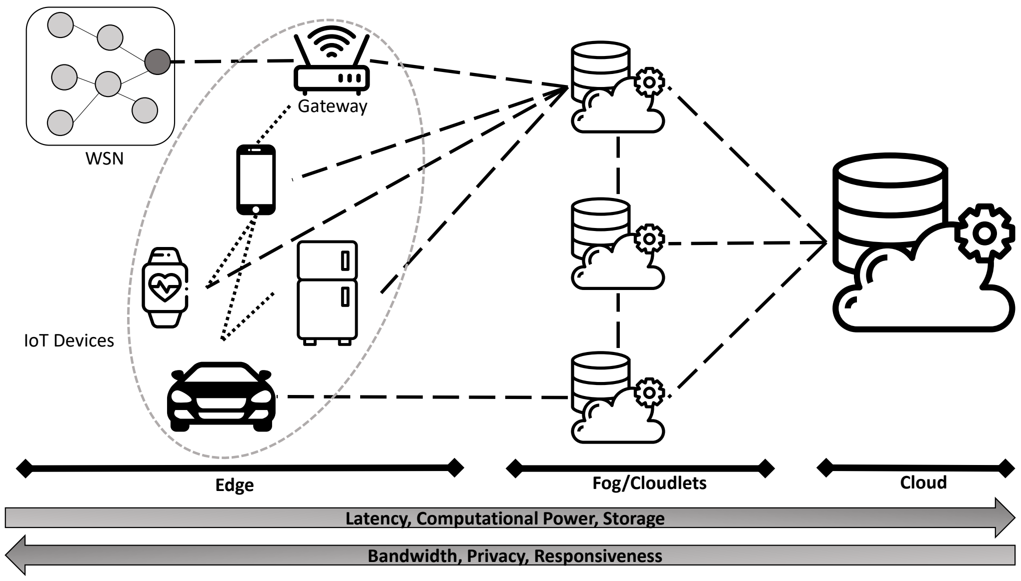

WSN
Gateway
IoT Devices
Edge
Fog/Cloudlets
Cloud
Latency, Computational Power, Storage
Bandwidth, Privacy, Responsiveness

Figure 2.6: Example architecture of Edge Computing using Internet of Thingss and Wireless Sensor Networks.

complex hierarchy of the architecture. Also, highlighted in this figure moving processing and computation further towards the devices level, the latency, the computational power, and the storage decreases. The opposite applies to bandwidth, privacy, and responsiveness. These features increase with pushing the intelligence and processing towards the devices instead of the central server. However, enabling computation, decision making, and intelligence at the devices leads to multiple challenges: energy constraints, privacy, latency, reliability, and computational power are only the main. Section 2.4 highlights and introduced more towards Edge Computing and the benefits and constraints.

2.3 Data-driven Predictive Modelling

The previous section (Section 2.2) highlighted that modern distributed systems collect massive amounts of data through devices deployed in everyday objects. This enormous collection of data can provide individualisation/customisation (e.g. personal profile for driving assistance), optimisation (e.g. demand forecasting for car sharing), or event/anomaly detection (e.g. predictive maintenance) in services and applications to identify risks, profitable possibilities and increase customer satisfaction. However, this has to be done automatically as the volume of data can not be processed and analysed by a human anymore. Therefore it requires processing the generated data with data analytics, pattern recognition, or machine learning techniques. This section is introducing the definitions, types, and methods related to data-driven predictive modelling. These fundamentals are used in the thesis as experimental implementation and baselines for edge-centric analytics and learning.

2.3.1 Definition and Rationale

Data processing describes a collection of methods that enable the creation of knowledge from the collected data. This collection of methods is known as data analysis with cleaning, transforming, and modelling the data. Data Analysis can also be named data mining when referring to Big Data and databases, or sometimes data analysis is referred to as Knowledge Discovery in Database (KDD) [92]. Their difference prevails in the volume of data on which the methods are used [93, 94].

Data analytics is using statistical models and algorithms to model and generate information from the data. "An algorithm is a sequence of instructions that are carried out to transform the input to the output" [95]. With respect to data analysis, data analytics can be seen as the modelling part of the data analysis process and can be divided into three categories: descriptive, predictive, and prescriptive. Descriptive analytics uses the history of the data applying statistics to identify insights and patterns for reporting and exploring. Predictive analytics is additionally trying to model the future based on historical data. Prescriptive analytics is even further a method to provide recommendations and decisions based on the data.

Machine learning techniques are part of predictive analytics, but these methods and techniques can also be used for descriptive analytics. If the instruction or the transformation from the input to the output is unknown, it is possible to use data to learn this transformation by approximating. This approximation of learning from data is defined as machine learning: "a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty" [96]. The key functionality of machine learning is to train a model that can then predict future data. It is essential to differentiate between learning and inference. The inference is applying the trained model to generate information and perform the prediction. In contrast, learning is the process to develop the machine learning model by using the underlying data and train the algorithm.

Machine learning and data analytics are aiming to generate knowledge and intelligence from the data. Knowledge can be defined as understanding and information gain through learning [97]. Intelligence, on the other hand, is defined as acting based on learning and achieved information [98, 99]. This thesis is not contributing towards the intelligence part on how to act based on the analytical findings. The presented work focus only on providing methodologies to enable intelligence.

2.3.2 Machine Learning Methodology

To generate knowledge and intelligence, machine learning algorithms learn from data a model to make predictions. This data can be categorised into two types: quantitative and qualitative [93]. Data classified as quantitative is mostly numeric with either continuous or discrete values. The qualitative data can be nominal or ordinal, representing a description of the data in an ordered or unordered group (e.g., hair colour, grade, or cloth size), also sometimes referred to as categorical. Besides these primitive data types, complex data types can also be used for machine learning, such as text, speech, video, image, graph, or time series [96]. Independent on the kind of data, the dataset should be divided into at least two but sometimes required three different datasets (train, validate and test) to perform a machine learning algorithm [98]. This splitting of the data will support the identification and evaluation of the learned model's actual performance. The majority of values should be inside the training dataset, on which the model is learned and parameters fitted. The validation dataset can be used to tune the model's parameters that minimise the error or select the best model. The final performance of the generated model can be obtained by using a test set. This test dataset is an independent and unseen selection of data to evaluate the learned model using different performance metrics. The rationale behind separating the data into blocks is to generate a generalisation of the model, preventing over-fitting or under-fitting and providing a measurement of its certainty. Over-fitting is a term which characterise a model that is learned and adapted to close to the underlying data and cannot predict unseen data well. In contrast to over-fitting is under-fitting, in which the model cannot capture the underlying data structure accurately [93, 96].

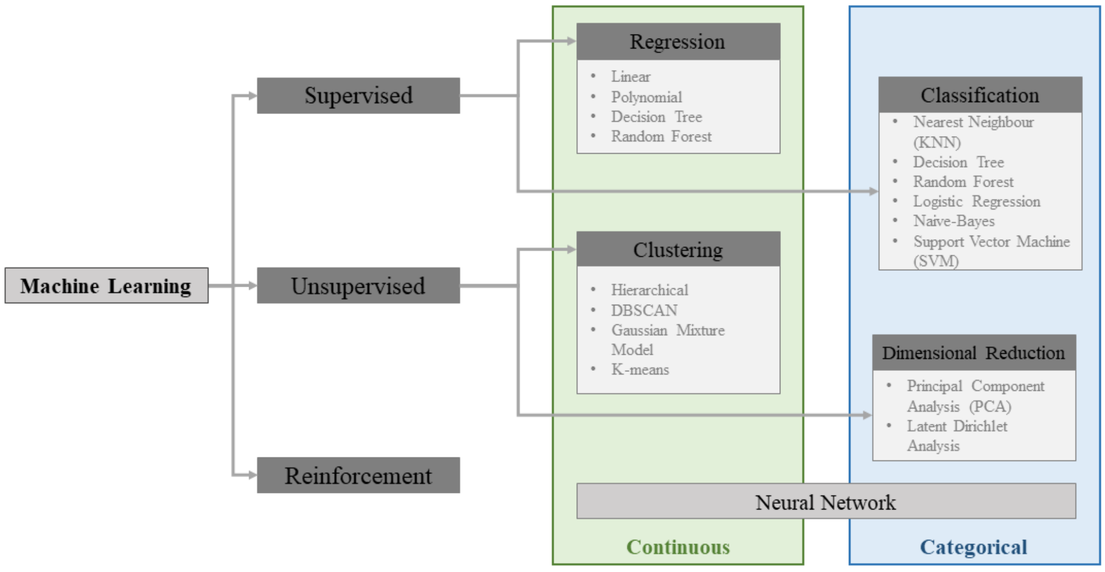


Figure 2.7: Overview of different machine learning algorithms clustered by type and usage.

Machine learning techniques can be classified into three different model types supervised, unsupervised, and reinforcement learning [98, 100, 101]. Figure 2.7 shows an overview of the classes and the comprised algorithms separated into continuous and categorical data. Neural networks are applicable over all types and all data categories.

2.3.2.1 Supervised Learning

In supervised learning, the algorithm processes data during the training of the model, in which the outcome is already known in advance given a label. The most commonly used supervised learning models are classification and regression. The input data from an input space X is defined in supervised learning as $\mathbf{x} \in X$ that contains one or multiple attributes, variables, or features resulting in $\mathbf{x}^T \in [x_1, ..., x_d]$ with d representing the input data dimension. The output data from the output space Y is defined as $y \in Y$and is in supervised learning known to the model while training. The goal is to find a model or function $f \in \mathcal{F}$ that can map the input data $\mathbf{x}$ to the output data y so that $f : X \to Y$. If the output of the function $f(\mathbf{x})$ is nominal, then the used algorithm is a classification; else, if the output is numeric, the model used is a regression. To find the model $f \in \mathcal{F}$ a training set of data is needed. The training set D of length Ncontaining input-output pairs $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N} \subseteq X \times Y$ that are forming an unknown probability distribution of $p(x, y)$. During the learning period the algorithm is trying to minimise a objective function $\mathcal{J}$ which contains a loss function $\mathcal{L}$ and a regularisation term $R(\mathbf{w})$. The objective function $\mathcal{J}$ can be defined as the expected risk of a function f:

$$\mathcal{J}(\mathbf{w}) = \mathbb{E}[\mathcal{L}(\mathbf{x}, y; \mathbf{w}) + R(\mathbf{w})] = \int \mathcal{L}(\mathbf{x}, y; \mathbf{w}) p(\mathbf{x}, y) d(\mathbf{x}, y) + R(\mathbf{w}) \tag{2.1}$$

The regularisation term R aims to avoid over-fitting. Commonly used is the L^2 with $R(\mathbf{w}) = \frac{\lambda}{2}||\mathbf{w}||_2^2$ or the L^1 with $R(\mathbf{w}) = \lambda||\mathbf{w}||$. The loss function, also named error function, cost function, or utility function, calculates the discrepancy between the real output value y and the fitted value of $f(\mathbf{x})$ on the training dataset D over a set of parameters $\mathbf{w} \in \omega$, with ω represents the parameter space. Depending on the supervised learning model, the loss function $\mathcal{L}$ can be defined for classification using either the logistic loss $\mathcal{L} = \log(1 + e^{-yf(\mathbf{x})})$ or the hinge loss $\mathcal{L} = max\{0, 1 - yf(\mathbf{x})\}$. For regression, the loss function can be defines as the absolute loss $\mathcal{L} = |f(\mathbf{x}) - y|$, squared loss $\mathcal{L} = (f(\mathbf{x}) - y)^2$, or ϵ-insensitive loss $\mathcal{L} = max\{0, |f(\mathbf{x}) - y| - \epsilon\}$. Summarised in supervised learning aim is to find the optimal parameters $\mathbf{w}^*$ that minimise the objective function $\mathcal{J}$ (see Equation 2.2) [102, 96].

$$\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathcal{F}} \mathcal{J}(\mathbf{w}) \tag{2.2}$$

As the distribution of $p(\mathbf{x}, y)$ is unknown, only an approximation of the expected risk using the method of calculating the empirical risk is possible to use. Using Empirical Risk Minimisation (ERM) [103] over the training's set D of length N, the approximate $\mathcal{J}(\mathbf{w})$ can be defined as:

$$\mathcal{J}(\mathbf{w}) \approx \hat{\mathcal{J}}_N(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}) + R(\mathbf{w}) \tag{2.3}$$

2.3.2.2 Unsupervised Learning

Unsupervised differs from supervised learning in the sense that no target values are known, and the model has to identify patterns based on the relation of the variables. The formulation of the training data is defined as $D = \{\mathbf{x}_i\}_{i=1}^{N}$. Models for unsupervised learning are clustering, density estimation, dimension reduction (e.g., principal component analysis). The loss function $\mathcal{L}$ in unsupervised learning can be defined as $\mathcal{L}(\mathbf{x}, f(\mathbf{x}))$.

2.3.2.3 Reinforcement Learning

In reinforcement learning, an agent interacts with the environment and learns with rewards based on its feedback. Each time the agent has to decide his action based on maximising its reward. Reinforcement learning is related to game theory and is often associated with artificial intelligence and deep learning [104]. The basic approach of reinforcement learning can be modelled using a Markov decision process, in which the assumption of complete knowledge of the environment is made [98]. Also, evolutionary algorithms are part of reinforcement learning.

2.3.3 Learning under Continuous Data

Performing learning on continuous data or streaming data has gain attention over the last years. The massive increase of data provoking storage and processing issues characterised by the ability to frequently generating new data has led to developing techniques on learning over data that continuously arrive. The presented machine learning techniques in the previous section rely on one-time learning in which access to the entire training data D is assumed. In distributed machine learning introduced in Section 2.3.4, the information is split into subsets. Still, one-time learning with access to this subset is adopted, resulting in batch learning for each subgroup. One area of continuous data is streaming data, in which the training data D is only available at arrival as only limited storage for historical data is available. Moreover, in some continuous data streams, a temporary or entirely temporal dependency exists between historical data points. In general, continuous data is defined as data vectors $\mathbf{x}$ generated from an input space X over a discrete-time sequence of $t \in \mathbb{T} = \{1, \ldots, t, \ldots, T\}$. A new data vector $\mathbf{x}_t$ is generated at each time instance.

2.3.3.1 Time Series Analytics

Time series and sequential data analytics are a particular form of continuous data analytics. The input variables $\mathbf{x}_t$ depend on the previously received variables $\mathbf{x}_{t-j}$ with $j = \{1, 2, ..., t-1\}$. Most relevant for time series analysis is modelling this dependency between adjacent observations and forecasting upcoming observations based on their history [105, 106, 107]. Time series can be stationary and non-stationary. Time series is stationary if the joint probability distribution of any two time sequences is the same. This probability distribution is defined by its mean μ, variance σ, and autocorrelation values. Meaning that the distribution is not changing if the modelling horizon is shifted backwards or forwards in time. Most commonly for stationary time series is modelling the underlying observations by performing the Autoregressive (AR) process. AR assumes that the current value at a time t can be expressed with a relationship of the p previous values and a random white noise a, resulting in the following equation:

$$x_t = \left(1 - \sum_{i=1}^{p} w_i\right) \mu + w_1 x_{t-1} + ... + w_p x_{t-p} + a_t \tag{2.4}$$

Another modelling method widely used is the Moving-Average (MA), which models the current value x_t by the mean μ and the randomly generated noise a_t over the last q period, so that the predicted value using MA can be defined as:

$$x_t = \mu + a_t - w_1 a_{t-1} - ... - w_q a_{t-q} \tag{2.5}$$

MA and AR are both assuming a stationary series. In real data, the stationary of a series is not always applicable. Mostly, time series appears with a trend or some seasonality, resulting in a non-stationary series. If the underlying data is non-stationary, it is possible to integrate (I) the series by performing a difference of an order g of the values with:

$$x_t = \nabla^g x_t \tag{2.6}$$

Combining the different methods of MA, AR and I results in Auto-Regression Integrated Moving Average (ARIMA), the most frequent modelling and forecasting method for time series analysis. ARIMA can be defined with the parameters p for AR, q for MA, and g for I, deriving into the notation ARIMA(p, q, g).

Besides the well-established statistical modelling and forecasting method of ARIMA, other machine learning techniques for especially time-dependent series have been introduced in the last years. The algorithm is mainly based on similarity measurements or entropy values when comparing multiple time series. The similarity measurements can be divided into shape-based, feature-based, edit-based, and model-based. Widely used are only shape-based and model-based similarity measures. Clustering, classification, segmentation, anomaly detection, and prediction are the most commonly used learning tasks [108, 109].

2.3.3.2 Online Machine Learning

Online machine learning rests on the assumption that the data is not accessible for the algorithm so that training must be performed at each instance t when the data occur. Incremental learning is related to online learning; their difference relies on the exception that some data buffer is available to train the algorithm. Online machine learning is an algorithm specially developed for continuous data and Big Data applications with the inability to access and process the data in memory [110, 111, 112]. The described time series analysis assumes that modelling the data and forecasting is based on a distribution drawn from the data with ordered values. In online machine learning, the assumption of a prior distribution and ordered values is not made. However, online machine learning can be applied to time series data if the order is guaranteed from the underlying system. Multiple machine learning algorithms have been introduced for online or incremental use in recent years, such as clustering, classification, decision trees, or regression [113, 114, 115, 116]. Generally, in supervised online machine learning, a new input vector $\mathbf{x}_t$ with the corresponding output label y_t is observed at each time instance t. At each time, the already fitted model $f_t(\mathbf{x})$ is using inference to predict the value y_t. The error or loss can be calculated by comparing the expected $f_t(\mathbf{x}_t) = \hat{y}_t$ with the actual y_t using a loss function $\mathcal{L}$. The aim is to minimise the regret between the sequentially defined model loss and the loss fitting over the entire data. The regret is defined as the difference between the cumulative sum of the loss function $\mathcal{L}$ for each time up to the current time t and the loss function at hindsight up to this time [117, 118, 119].

$$Regret(\mathbf{w}) = \sum_{t=1}^{T} \mathcal{L}(\mathbf{x}_t, y_t; \mathbf{w}_t) - \sum_{t=1}^{T} \mathcal{L}(\mathbf{x}_t, y_t; \mathbf{w}) \tag{2.7}$$

In online learning, the minimisation of the regret and loss function is only possible by approximation. The most widely implemented approximation method to find the parameters $\mathbf{w}$ of Equation (2.2) is Gradient Descent (GD). GD is an iterative optimisation algorithm that tries to find the minimum of the approximate objective function $\hat{\mathcal{J}}_N(\mathbf{w})$ by estimating w at each iteration k. GD is possible to apply if the function f is differentiable and convex. The derivatives of the optimisation function $\mathcal{J}$ for the parameter vector $\mathbf{w}$ of the function f is

$\nabla \mathcal{J}(\mathbf{w}) = (\frac{\partial \mathcal{J}}{\partial w_1}, ..., \frac{\partial \mathcal{J}}{\partial w_d})$. In GD, the algorithm calculates for each iteration k over the entire training's data the new gradients and updates the function parameters [95, 102, 120]. As applicable in Equation (2.8) at each iteration k, the gradients are weighted with a learning rate η and then subtracted from the previously estimated function parameters w_k. The learning rate η is critical to define as it converges very slowly to the global minimum of the convex function or jumps around without any convergence [121].

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \nabla \hat{\mathcal{J}}_N(\mathbf{w_k}) \tag{2.8}$$

As GD is of high computational complexity because of the iteration over the entire data, other forms of GD emerged. Stochastic Gradient Descent (SGD) [122] uses only a subset of length B of the available data N to calculate the gradient. Depending on B, either performed on a batch if $B > 1$, or online if $B = 1$, using a single data point for the gradient [123]. The computational complexity of GD can be improved from $O(NdT)$ up to $O(d)$, with $B = 1$. In Equation (2.9), the SGD is defined for the following parameter at iteration $k + 1$.

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \nabla \hat{\mathcal{J}}_B(\mathbf{w}_k) \tag{2.9}$$

Besides SGD, other optimisation algorithms are introduced using GD [124]. A detailed comparison is provided in [125]. Even though the assumption on independent and identically distributed (i.i.d.) exists (stationary), current research has revealed that SGD is also effective on non-i.i.d. data [126].

2.3.3.3 Window-Based Learning and Concept Drifts

Window-based learning is a possible method to handle continuous data with a temporary temporal dependency or non-stationary data. Machine learning using windowing can be deployed by either a timestamp-based window or a Sliding Window (SLW) [127]. Generally, a window is a temporary storage of historical data. In a timestamp-based window, a certain sequence of time instances is considered between two timestamps. In contrast, in a sliding window, the newly arrived observation decays the oldest observation stored in the window. In many applications, this type of deployed analytics and machine learning is used. A SLW $\mathcal{W}$ of size $M > 0$ at time t is able to store $t - M$ recent observations. With a new observation, the oldest observation will be discarded in favour of the new observation. The SLW $\mathcal{W}$ is defined at time t as $\mathcal{W}_t = (\mathbf{x}_{t-M}, \mathbf{x}_{t-M+1}, \ldots, \mathbf{x}_{t-1})$. Extensions on SLW approaches such as adaptive size regularisation [128] or multiple windows of parallel or distributed data [129], or recurrent windowing [130] exist, but are not relevant for this thesis. Most applications with windowing usage answer a query or monitor statistics (e.g., `MIN`, `MAX`, or `AVG`) over a continuous data stream [111, 131]. When dealing with continuous data and temporal dependencies, window-based learning is the most used technique of handling this.

Traditional online learning techniques assume or work best under the stationary assumption. However, in real-world data and time-dependent data, this assumption can not be made. This issue can be overcome by using the window-based learning approach for concept drifts or non-stationary data. In some applications identifying this concept drift and acting based on it is of high importance. A concept drift is broadly defined as a learned function f that changes over time. Formally is a concept drift defined for supervised learning with input variables $\mathbf{x}$ and target variables y, that the joint probability distribution at a time t and time $t+1$ are not identically $p_t(\mathbf{x}, y) \neq p_{t+1}(\mathbf{x}, y)$ [132]. In the literature, several ways of how this change can occur are classified. Overall, four types of concept drifts can be categorised: sudden/abrupt, incrementally gradual, and reoccurring. An illustration of these different types is shown in Figure 2.8. It is important to differentiate a concept drift from an outlier or anomaly. Considerable research on handling concept drifts has been proposed in the literature, mostly relying on window-based machine learning [133, 134, 135, 136].

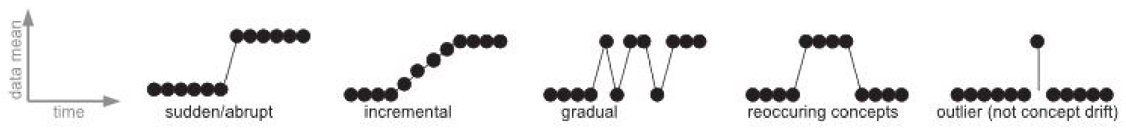


Figure 2.8: Examples of existing concept drifts types in online learning environments [132].

2.3.4 Distributed Machine Learning

The introduced machine learning models in Section 2.3.2 are engineered on centralised data collection assumption. The learned algorithm is considered to run once using supervised or unsupervised approaches on centrally stored data that fit during the computation and calculation of the model into the memory. It is important that all data is available at the training time inside the Central Location. The enormous increase of IoT devices and the continuous collection of data highlights the limitations regarding the storage and computational capacities of standard machine learning methods. Algorithms cannot be performed on centralised data and through one-time learning. Possible approaches to distribute a machine learning algorithm is either using data distribution or model distribution. A machine learning algorithm based on data distribution is performed by deploying a model on separate parts of the datasets resulting in multiple trained models. In contrast, the model distribution is learning on the same data but separate parts of the model [137]. Both approaches aim to train a central model, usually done by one-shot or one-time learning. These two approaches of distributed machine learning are illustrated in Figure 2.9.

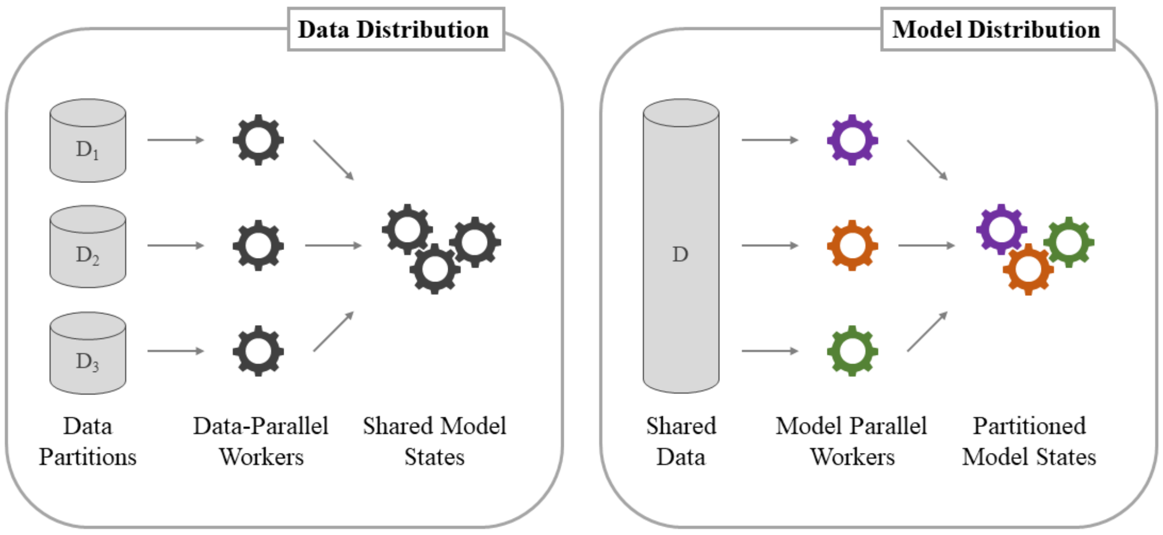


Figure 2.9: Overview of distributed machine learning types.

2.3.4.1 Data Distribution

As highlighted above, data distribution is one possible method of solving large-scale learning by using different datasets to train a model. This training can be performed in a centralised or decentralised fashion. The centralised approach aims to have a central coordinator that merges the separated model related to the server-client model architecture from a distributed system perspective [138]. In contrast, the decentralised approach works in a peer-to-peer way. All clients are connected without any global knowledge or coordination [139]. Generally, the data normally stored on a central server denoted as D is distributed into k subsets of equal size so that $D \equiv \{D_1, D_2, ..., D_k\}$. Each distributed system member performs over this data subset the minimisation of the loss function by finding the parameters w of the model. All members are using the same model so that $D_i \perp D_k | w, \forall i \neq k$. In a centralised way, the model is communicated to the coordinator or parameter server. In the decentralised deployment, the circulation is through peer-to-peer communication. The training to find the minimisation of the loss function over multiple subsets of data can be achieved by deploying online, batch, and incremental learning techniques. SGD has been introduced in the previous section (Section 2.3.3.2) and is suitable for distributed machine learning on distributed data. Long-established is the parallel SGD and other convex optimisation algorithms for iterating over data subsets [140, 141]. Multiple platforms in Big Data rely on the distributed data machine learning technique with advanced success, e.g., Hadoop Spark or Tensorflow [142, 143].

2.3.4.2 Model Distribution

The second possible distribution of machine learning is using a model distribution. In this method, the model parameters w are split across the system members. Each learner has full access to the dataset D but trains only certain parts of the overall model so that the final model parameters are $w \equiv \{w_1, ...w_k\}$. Each trained model on a learner can be defined as $w_i \not\perp w_k|D, \exists(i,k)$. The model parameters' training can be achieved again by performing online, batch, or incremental machine learning with convex optimisation algorithms such as SGD. The difficulty of distributed model deployment is to combine the independent generated models with a central model. Combining or selecting models has been extensively studied in research by introducing ensemble learning techniques. Most notable are bagging, boosting, and stacking [144]. The relation behind each of these methods is training 'weak learner' and combining their results in a strong learner.

In bagging or bootstrap aggregating [145], each model is trained in parallel on an individual learner. The Central Location or parameter server then combines each model by averaging each learner's predictions or parameters towards the final prediction or model. For the classification task, a majority voting mechanism is used for selecting the final decision. More formal bagging is defined as $f(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} f_i(\mathbf{x})$.

Boosting or the more commonly known AdaBoost [146, 147] algorithm differs from the bagging method by not running in parallel on each learner. Instead, the algorithm is training the model sequentially, nourishing the pre-trained model towards the next learner so that boosting can be defined as the weighted sum of all models with $f(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i f_i(\mathbf{x})$.

Stacking extends the bagging and boosting method by introducing heterogeneous learners [148]. In boosting and bagging, the assumption is made that each learner uses the same algorithm to train the data. In stacking, multiple algorithms are trained in parallel and feeding a meta-learner that aims to minimise the generalisation error based on its inputs.

2.3.5 Optimal Stopping Theory

The problem of choosing and identifying the best time to make a decision or perform an action is the research area of Optimal Stopping Theory (OST) [2]. The fundamentals are highlighted in this section and used in detail throughout Chapter 3 and Chapter 5. The aim of OST is to minimise an expected cost or maximise an expected revenue to find the best time to take action or stop the process while observing a sequence of random variables. The most famous application is the secretary problem or similar the discounted secretary problem, which has been researched well [149, 150, 151]. The aim is to identify the best candidate for a single open position given a number of applicants. Their qualification can rank all applicants in order from best to worst. The order in which they are interviewed is random.

The observance is sequential with a number of finite elements N, with N known in advance. Sequentially defines that one candidate or element is observed at a time. Each time a new element (candidate) is observed or in the secretary problem interviewed, the decision has to be made immediately on accepting or rejecting the candidate. In some implementations, the decision can be reconsidered with a discounted value added. However, the classical approach is that accepting or rejecting the candidate is irrevocable. After the element is observed, the decision is made by ranking the just seen element based on the previously seen ones. The aim is then to select the candidate with the highest probability of being the best of all. It has been shown and well known that the optimal time to select a candidate is tending to a probability of approximately $\frac{1}{e}$ [152]. Therefore, the approach is simple in interviewing the first $m \approx \frac{N}{e}$ candidates and then hiring the next applicant that exceeds the best of the first m candidates. Multiple other extensions have improving the $\frac{N}{e}$ probability [153, 154].The privacy aspect is implemented in FL by using the data of the ED only on the device itself and train the global model by sharing only the updated model weights over the network. The basis FL deployment does, however, allow possible security and privacy-related attacks. Not only is it possible to insert through a malicious user some data to poison the trained algorithm [155], but also to identify user data from the model updates sent over the network [156, 157]. Research on advancing the basic FL with security and privacy-enhancing techniques is an ongoing field [158, 159, 160]. Important FL can only provide complete security and privacy over any attacks when used with additional security, e.g. the encryption as mentioned earlier.

2.4 Machine Learning at the Edge

The continuously massive data volume generated from Edge Devices, such as smartphones, cars, or the Internet of Things, enables data-driven decisions and prediction by using machine learning. However, to analyse and draw conclusions from this data, most applications transfer the data to a Central Location using Cloud or Fog Computing. This approach allows high computational power at the central server but restrains the application with a delay of the central calculated inference from the learned algorithm. Additionally, issues of transferring massive amounts of data concerning bandwidth, privacy, and storage arise. Edge Computing provides the ability to perform machine learning in Edge Devices close to the data source, allowing to build low-latency and location-aware applications with keeping the privacy of the data as highlighted in Section 2.2.6.2 [161]. The following section highlights machine learning deployments in Edge Devices with their general constraints and advantages enabling intelligence. Edge intelligence can be performed by inference or training on devices, both are elaborated in the following section. Generally, the data from Edge Devices and the gained intelligence through machine learning can be used to either support and improve the underlying infrastructure and management or to enable applications and services

[162] on devices. An overview of the different possibilities to use the generated data of Edge Devices by performing machine learning is highlighted in Figure 2.10[3]. This figure shows the development of edge inference to edge training in contrast to the centralised procedure of current implemented applications.

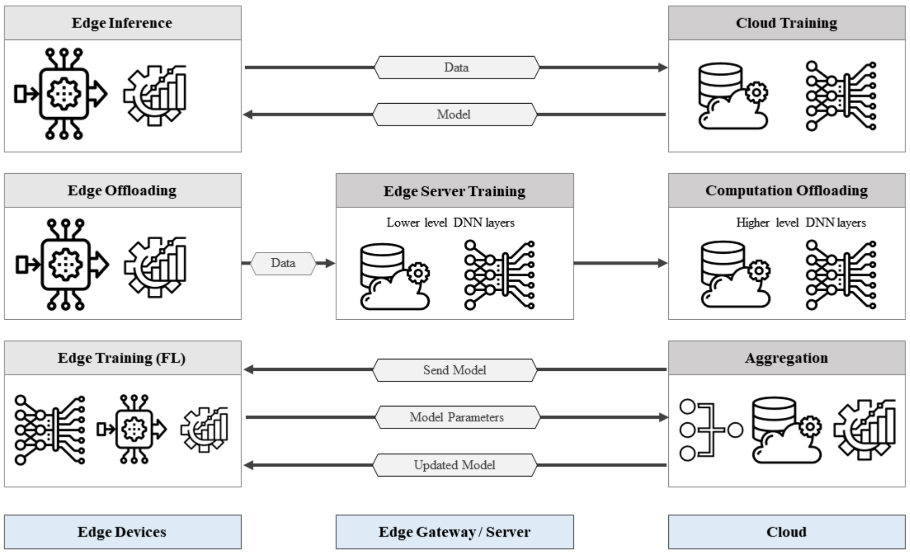


Figure 2.10: Overview of different levels for computation and model training in using edge inference and edge learning.

2.4.1 Constraints & Challenges Leading to Efficiency

Deploying machine learning with inference or training on Edge Devices is limited through multiple constraints. Illustrated in Section 2.2.4, Edge Devices rely primarily on batteries as power sources. With respect to the battery lifetime, the limited energy highlights the significant restraint of all analytics, machine learning, or computational tasks performed at the devices. Additionally, through the device's nature of being a small physical item, the computational capability in terms of storage, memory, and processing power is a constraint [163]. However, in the field of Internet of Things (IoT), the devices occur in different variations causing heterogeneity in computational capabilities, memory and processing power. Considering a car or smartphone, it is clear that these devices can perform more computation, having more memory and storage than a simple device placed inside a fridge or television. Nevertheless, in all IoT energy and resource-efficient deployment of computational tasks are critical as they are all function on batteries. Additionally, with the increase of the IoT, the enormous data collection and the unreliable and bounded network become a critical challenge for deploying intelligence in Edge Devices [76]. Even with the deployment of 5G, the

[3]Some illustration icons used from [21]

restriction on uploading and downloading, when transferring all raw data towards a Cloud or central server and waiting for the inferred results, will increase latency and, therefore, delay intelligence and decisions. Also, 5G and other wireless wide area network technologies are costly. Integrating Edge Devices and sensors into everyday objects to generate Ubiquitous Computing raises concerns about personal data that these devices collect and analyse. The topic of privacy connected to the machine learning of Edge Device data and collection is one of the hottest topics nowadays and created legalised regulations, such as General Data Protection Regulation (GDPR) [164] or California Consumer Privacy Act (CCPA) [165], that challenge the current deployments of centralised machine learning.

Performing machine learning on data coming from Edge Devices can be used to either overcome hardware constraints and supporting infrastructure and management limitations but also to enable applications and services for customised and real-time predictions [162]. Key component is efficiency when deploying learning or inference on devices. Efficiency with respect to use the resources of the devices in a light-weight manner to reduce communication, bandwidth, storage, computation and energy. Efficiency of resources has not only gained interest in edge environments but also on data centre level as resources tend to be limited with the tremendous increase of continuous data [166]. Latency and communication is the major issue from the management perspective of deploying machine learning. Overcoming these issues has been proposed by deploying intelligent clustering mechanisms, data forwarding strategies in devices [167, 168], or data reconstruction models [169] as possible solutions. Widely implemented for efficient communication reduction and resolving latency involves performing machine learning algorithms to cluster the devices. From the clustering algorithm results, Cluster Heads (CHs) can be assigned to generate a hierarchical structure of the network [170, 171]. These implemented and deployed structures and architecture generated through machine learning and analytics are the baselines for deploying edge intelligence towards an application and service perspective. Efficiency is implemented through local inference and training at the device level. However, advanced analytics in Edge Devices performing services towards user behaviour or customisation with the aim to overcome the constraints mentioned above can lead to latency issues when inefficient implemented algorithms are used and not updated for changing environments.

2.4.2 Benefits of Edge Intelligence

Limitations and the importance of efficient machine learning and analytics at the Edge Device level have been elaborated on in the previous paragraph. Efficiency is the key factor when deploying machine learning to overcome resource limitations such as computational power, storage, memory, and energy. Machine learning on Edge Device data enables enormous advantages and possibilities. The main benefit is the ability to derive autonomous decision-making and knowledge creation without latency effects. Pushing analytics and machine learning towards the edge facilitates the application to perform real-time decisions and actions. Significant is this advantage to time-sensitive critical applications and services. The most gained application using Edge Computing with machine learning techniques are autonomous driving services and use cases. The transmitted data exceeds the network's possible bandwidth and the ability to perform inference or learning on a Central Location. Each car generates terabytes of information in a single day. Services and applications of autonomous vehicles rely on real-time decision-making and intelligence to guarantee the vehicle's safe and correct driving behaviour [172]. Edge intelligence is an enabler to provide these services in time-sensitive use cases as latency and bandwidth constraints are eliminated. The locally performed inference and training further benefits from improving data privacy of the users. Especially when using Edge Device data to customise and personalise applications and services, local training and inference enable the possibility to overcome legal restrictions to provide such a service. Highlighted in Section 2.2.4, deployment of heterogeneous sensors and diverse IoT application devices is generally seen as a constraint and challenge. When empowering edge-centric machine learning, this disadvantage is changing to an advantage. Combining the data of the different sensing devices with their given contextual information and their location-awareness enables tremendous information gain for services and applications. The combined data of all heterogeneous systems connected can, especially by using advanced machine learning algorithms, provide highly accurate and customised prediction models. Accuracy of the deployed models in changing environments is essential, as wrong decisions and predictions can decrease customer satisfaction or induce serious disturbance. Most devices collect and generate continuous data of their surrounding environment, which not only changes over time but can also lead to concept drifts. In Section 2.3.3.3, the definition of concept drifts has been highlighted and the importance to continuously learn and update the machine learning model. Edge intelligence can be beneficial to provide and perform these updates and identify concept drifts.

Summarised Edge Device intelligence's benefits towards services and applications are location-aware, privacy-sensitive, and real-time performance.

2.4.3 Edge Inference

Edge intelligence's benefits are highlighted in the previous paragraph, and the importance of efficiency towards the deployed algorithms and techniques has motivated the following section. Inference of trained machine learning models in Edge Devices is one possible solution to enable real-time action and decision making. Most applications and deployed Edge Computing paradigm are considering the usage of edge inference [173, 174]. Training the machine learning model is done in a Central Location, e.g., the Cloud or Fog. The generated model is then distributed towards the Edge Devices. The device performs intelligence in real-time by applying the received model locally on the newly sensed and received data. Only the output of the model alongside some meta-data is sent towards a Central Location. In there, the model can be updated and retrained. The retrained model can then be again distributed towards the Edge Devices.

Deploying machine learning models that require excessive memory on performing the inference is not feasible on Edge Devices. Not only can the memory lead to resource constraints when deploying the algorithm locally, but storage is also one of the bottlenecks as the size of the model can exceed the device's capacity. Recent research investigates the ability to deploy deep learning with neural network at Edge Devices [175, 176]. Current implementations consider pruning or compression of the model as a possible solution to overcome the limitations mentioned earlier of memory, processing, and storage in Edge Devices. It is possible to further calculate the energy and time to be used performing the inference locally in comparison to the offloading the task towards other devices, server, or micro-data centre to optimal schedule computing tasks in terms of efficiency [177, 178].

Edge inference is limited with the possible delay towards updating the algorithm to new data due to the retraining at a Central Location. This can result in less accurate predictions and wrong decisions of the service. Not only does this method require additional communication resulting in overhead but also in massive latency towards adapting of concept drifts. Additionally, edge inference requires raw data transfer towards a Central Location during the model training period. Especially when privacy is essential for the considered application, edge inference can not provide the necessary data protection. Therefore, some services use the pre-trained model from the Central Location to deploy locally at the device and use online learning methods to retrain and update the model continuously.

2.4.4 Edge Training

Departing to advance the Edge Devices with further intelligence, training on Edge Devices has become an increased research area in recent years. The deployment and training of intelligence through deep learning has especially gained interest [179, 180, 181]. Training can be implemented using the mechanism elaborated in Section 2.3.4 of online learning and distributed machine learning. The implementation can be done using a central coordinator or only peer-to-peer training [182]. However, one of the advantages and constraints of using machine learning in Edge Device environments is their highly heterogeneous nature. This heterogeneous nature of devices results in non-independent and identically distributed (i.i.d.) data and occurs as devices either sense different features, are placed in environments with diverse behaviour or different geo-locations. Most algorithms assume an i.i.d. of the data when learning on distributed datasets, which is in environments such as the IoT not realistic.

However, solutions working with non-i.i.d. [126, 183, 184] have already been proposed, as mentioned in Section 2.3.3.2. The most notable work of edge training in a heterogeneous environment called Federated Learning (FL) [185] has been recently proposed and shows promising directions [186, 187, 157].

Federated Learning has been introduced within the context of Edge Computing, aiming to solve edge learning under privacy sensitive, non-i.i.d., and unbalanced data generated from massively distributed devices using the technique of machine learning highlighted in Section 2.3.4. Federated Learning is defined as: ”a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider. Each client’s raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective” [188].

Each device or client is defined as k, holding a subset D_k with a length of n_k of the overall training data D with length n. Federated Learning aims to minimise the central objective function defined in Equation (2.1). This optimisation is made by minimising each client’s objective function $\mathcal{J}_k$ over its subset of data D_k. Each client k sends its local model parameters to the Central Location, which performs an aggregation over the received parameters using FedAvg [189, 190]. Each client k, gets randomly selected from the pool of connected devices. After the selection, a download of the current central model to the device is induced. The local training of the model using online SGD or similar optimisation methods is performed over batch or streaming data. Then the updated model parameters are uploaded to the Central Location. The centralised aggregation of individual objective functions can be defined as:

$$\mathcal{J}(\mathbf{w}) = \sum_{i=1}^{k} \frac{n_i}{n} \mathcal{J}_i(\mathbf{w}) \tag{2.10}$$

The primary aggregation of FedAvg and FedSGD has been improved in recent years to different techniques optimising the communication between the Central Location and Edge Device [191] and providing other security mechanisms to ensure the privacy of the uploaded models [158].

2.5 Chapter Summary

In this section, distributed systems’ evolution has been highlighted with the corresponding different perspectives (application, device, network and processing). The application perspective shows the diversity of deployed systems clustered into seven areas and their three different communication types. The devices and perception perspective of distributed systems gave an overview of sensor networks and the IoT. The network layer introduced the technologies of transporting the data and the possible routing strategies. The data processing and computation layer of contemporary distributed systems have been introduced by the techniques of Cloud Computing and Edge Computing. Having the foundation of distributed systems of Section 2.2, fundamentals of machine learning, and data analytics have been elaborated in Section 2.3. Starting with highlighting machine learning and definitions, the section focused on distributed machine learning and continuous data analytics. This section's remainder shows machine learning in Edge Devices connecting the fundamentals of both introduced research fields, machine learning and data analytics and modern distributed system, particularly IoT and Edge Devices. Performing analytics in devices is limited to constraints, such as computation, memory, storage, and energy. The critical element of machine learning in Edge Devices is the efficiency of using these limited resources. However, Edge Computing enables possibilities, including privacy, real-time decision-making, personalisation, reduced network load and data transfer. The newly introduced research areas of inference and learning on devices has shown promising methods to stimulate local learning and overcome limitations and constraints of currently deployed centralised methods. This thesis contributes to efficient machine learning in these highly distributed systems of IoT and Edge Devices. The goal is to maintain the quality of the performed analytics while efficiently using the device's local resources.

Chapter 3

Quality-Efficient Data Forwarding

3.1 Chapter Overview

IoT applications and Pervasive Computing become part of our daily lives, aiming to use the ability of the generated data from battery-powered sensors and devices towards analytical tasks. Machine learning and predictive analytics emerge to acquire knowledge, essential for applications introduced in Section 2.2.3. The introduction and literature review sections emphasised efficient energy consumption usage as an important aspect for sensors and sink nodes. Most of these devices rely on batteries or are deployed in unreachable locations. The input data on which the analytics is performed must be of good quality but not draining the sensor's lifetime through continuous sensing. Studies have been shown that data transmission and reception drains the maximum of power and energy from a battery, while data processing inside a sensor can be neglected [64]. Considering that most of the energy is used to transmit the generated data to a Central Location (CL), communication reduction can be seen as energy-efficient.

Therefore, the coming chapter introduces a quality-aware and communication-efficient data forwarding mechanism implemented in Wireless Sensor Networks (WSNs) to reduce the energy consumption of sending data. First, in this chapter is the state-of-the-art quality and energy-efficient data forwarding strategies highlighted, followed by presenting intelligent decision-making methodology in sensing nodes and reconstruction ability inside sink nodes. Reconstruction is needed as implementing a communication reduction of the transferred data induces missing values at the collection point. These missing values are not random and known to the sink node. The first introduced methodology highlights the current deployment of prediction-based forwarding mechanisms. This implementation is improved by combining sensors' and sink nodes' computational capabilities towards a quality-efficient data forwarding strategy. Departing from the classical instantaneous decision making (IDM), a time-optimised forwarding strategy will be imposed for quality-aware data forwarding from sensors to sink nodes. The thesis distinguishes from the current research by introducing quality-aware data forwarding strategies not only on focusing on investigating the reconstruction ability of selective forwarding but also on analytical tasks and their quality. Therefore, this chapter's remainder conducts a comparative assessment of state-of-the-art methods with the newly introduced quality-efficient and time-optimised data forwarding strategies. Overall, this chapter contributes to the first hypothesis defined in Section 1.2:

> *Hypothesis 1:* Pushing computational intelligence of advanced decision-making in data forwarding to the edge of the network will overcome energy and bandwidth constraints due to the deployment of efficient communication methodologies. Combining this with intelligent reconstruction at a collection point leads to highly accurate analytical tasks and reconstruction of the imputed values.

3.2 State-of-the-Art on Data Forwarding Strategies

Wireless Sensor Networks (WSNs) are facing significant constraints of limited energy and bandwidth. Their primary function is to sense and monitor the environment. This sensing results in a constant data transfer from the sensing device towards a central location where the raw data analysis is performed. The majority of the energy is used for the transmission of data. Therefore, research on energy-efficient data forwarding strategies in routing protocols has been introduced in the research community [65, 192, 193, 194]. Routing protocols design the communication path between the source of information and the target, meaning between the sensor and the gateway or sink node. The aim of routing protocols is to extend the sensor network's lifetime by reducing the transmission range or throughput towards the sink node.

Routing Protocols for WSNs are divided into six categories: location-based, data-centric, hierarchical, mobility, multipath, and Quality of Service (QoS). Besides these six categories, the simplest routing method instructs each sensing node to deliver the data towards the sink node directly. This address-centric approach is the baseline for comparison for all energy-improved routing protocols listed in the six categories. Besides these categories, consideration must be made if the sensors are placed in a stationary or non-stationary environment considering particular mobility routing protocols. In this chapter, only static and stationary sensors are considered. The location-based methods of routing protocols use GPS signals to generate the closest path by grouping the sensors based on their minimal distance. On the other hand, data-centric protocols are mostly query-based, coherent-based, or negotiation-based protocols in which the sink node sends queries to the sensors and waits for the requested data. QoS protocols in WSNs aim to find the optimal path within the network towards the best throughput, being fault-tolerant and having high reliability. Related to the

QoS protocols is also the multipath protocol which splits the data from the sensors towards the k-shortest path for transmission.

With the increase of WSNs connected via the Internet, the hierarchical network structure became standard. Therefore, the deployment of different inter-domain and intra-domain routing protocols or topologies become unavoidable. Intra-domain topologies consider the overall network structure and deployment of a routing protocol throughout all hierarchy levels. Inter-domain, on the contrary, structures the protocol for only one single level of the hierarchy. The hierarchical routing protocol clusters the sensor nodes using their range of transmission, their types of sensors, their energy levels, or their geographical location. The most popular method, and first using a hierarchical routing, is LEACH [195]. The WSN generates Cluster Heads (CHs) that coordinate the data collection of surrounding sensor nodes and forward the fused and aggregated data towards a higher layer of the cluster or the sink node directly. This type of protocol does not need any global knowledge of the network, but requires a setup phase for generating the CH. Based on LEACH, further developed protocols are introduced, such as PEGASIS [196], HEED [197], TEEN [198], and APTEEN [199]. LEACH can be declared as proactive network routing, whereas TEEN is reactive. As already stated in Section 2.2.3, different type of applications are existing. Proactive networks require continuous data transmission, whereas reactive networks only send data by an event or query. APTEEN is combining LEACH and TEEN towards a hybrid protocol. However, most routing protocols deal only with one type of transmission, either reactive or proactive.

Alongside computational power upgrading at sensor nodes and the rapid deployment of WSNs, ideas of supplementary energy-efficient strategies appear. Three relevant aspects are introduced, in-network processing, data aggregation, and compression. Compression reduces the size of the transmitted data by different techniques and has been extensively studied [200, 201]. Nevertheless compression techniques are out-of-scope for this thesis but they can additionally applied on the proposed methods. The idea of in-network processing and data aggregation is mostly deployed by using prediction-based data reduction methodologies, which have been reviewed in [202] and [203]. The prediction-based approach aims to forecast and predict the future measured value with an implemented function in either the sensor, the CH, or both. The application or user is pre-defining a certain threshold used as forwarding decision making of a measurement. If the predicted value exceeds that predefined threshold, a transmission of the measurement to the CH or sink node is made. Besides the prediction-based data reduction, a delay-tolerant approach has been introduced in the literature, primarily by reducing the frequency or using aggregation of the data [204].

3.3 Definition and Architecture

The abstracted architecture is based on Figure 2.6 introduced in Section 2.2.6.2. Generally, each Edge Device (ED) monitors through Sensing and Actuating Nodes (SANs) the environment. The SANs and EDs are forming a WSN using a tree-like topology. Most communication-efficient routing protocols utilise the implementation of clusters, in which one SAN is selected as a CH. Their tasks are to collect its surrounding SANs measurements and to forward the aggregation of all data towards a higher hierarchy in the network. The proposed architecture uses the notation of a CH as ED with further hierarchy orders below and above. However, any level of cluster tree-like topology can apply the later proposed architecture and methods.

In the later sections, different data forwarding strategies are introduced. At this point, an overview of the rationale and architecture across all methods is given. Each SAN k measures contextual data of its surrounding environment continuously each time instance t over a d-dimensional space. A discrete-time domain of $t \in \mathbb{T} = \{1, 2, ...\}$ is defined with $t = 1, .., T$ and $T \in \mathbb{T}$ and used throughout the complete thesis. At each time t, this sensing generates into each SAN k a contextual raw-data vector $\mathbf{x}_t = [x_{1t}, ..., x_{dt}] \in \mathbb{R}^d$. The vector $\mathbf{x}_t$ is then sent to the ED i, receiving overall K contextual data-vectors of each SAN k at each instance t. Each SAN k is connected to a unique ED i so that the set of SANs is defined as $\mathcal{K}_i = \{1, \dots, k, \dots, K\}$ with $k \in \mathcal{K}_i$.The EDs can then decide on aggregate or merely forwarding the information to a Central Location (CL) or an in-between deployed Edge Gateway (EG). In using prediction-based forwarding or other routing mechanisms, each SAN k is equipped with a function $f(\mathbf{x})$, generating a prediction of the next to be seen value $\hat{\mathbf{x}}_{t+1}$. Based on the predicted value $\hat{\mathbf{x}}$ and the real sensed value $\mathbf{x}$, the SAN is tolerating a predefined error bound of θ in which it is deciding not to send the observed values. The ED i is deployed with a function $g(\mathbf{u})$ that can reconstruct the missing value calculating $\tilde{\mathbf{x}}_{t+1} = g(\mathbf{u}_{t-1}, ..., \mathbf{u}_{t-M})$. Depending on the decision in SAN k, $\mathbf{u}$ can be defined as:

$$\mathbf{u}_t = \begin{cases} \mathbf{x}_t & \text{Case 1 with } \theta > \text{error bound}, \\ \tilde{\mathbf{x}}_t & \text{Case 2 with } \theta \leq \text{error bound}. \end{cases} \tag{3.1}$$

Figure 3.1 shows the highlighted architecture for both cases over multiple SANs and the transmission to one ED.

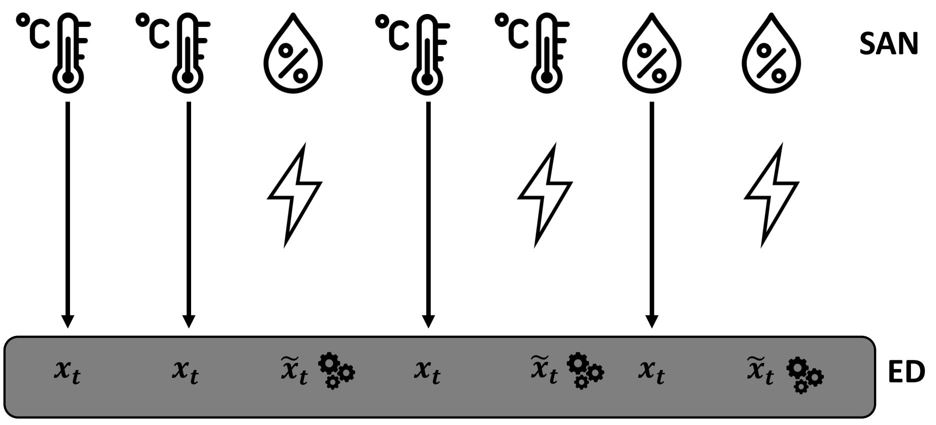


Figure 3.1: Architecture of the introduced selective data forwarding strategy and reconstruction.

3.4 Prediction-Based Decision Forwarding

Highlighted in the previous section, in-network processing using predictions is one solution towards quality efficient data forwarding in edge networks such as WSN. In prediction-based data forwarding, a function is used to impute missing or predict future measurements of SANs. The function depends on the prediction method that is implemented and used to compute the values. It relies mostly on other sensed values forming a regression, aggregated values, or weighted historical values. Related work [202] highlights two predominant implemented prediction-based data forwarding methods, either a **Single Prediction Design (SPD)** or a **Dual Prediction Design (DPD)**. This thesis proposes another implementation of prediction-based data forwarding based on the advantages of SPD by using resource efficient forwarding strategies and DPD by eliminating its disadvantages of communication and training overhead. The proposed method is further noted as **Quality-Efficient Prediction Design (QEPD)**.

3.4.1 Single Prediction Design

The first method for prediction-based data forwarding is the SPD and relies on deploying a prediction function in the SAN or the ED (it can be the CH as well if multiple hierarchies exist). Only one single function is used to impute or predict values, either $f(\mathbf{x})$ at the SAN or $g(\mathbf{x})$ at the ED. Suppose the function is deployed in the ED, more computational power and resources are available. In that case, the functions are mostly deployed to answer queries without asking the SANs for the current values, design a network topology using a grouping of SANs with similar measurements, or analyse the quality of the received sensed measurements [205, 206]. Deploying a prediction function at the SAN and not at the ED can be used for communication reduction in deciding when to send the measurements. This communication reduction is mostly incorporated with comparing the values sensed with neighbours, and therefore a peer-to-peer communication is assumed [202]. SPD's advantage is that no communication and synchronisation between ED and SAN is required to deploy the prediction function. Each SAN or ED can independently decide to deploy a prediction function or not. However, suppose the prediction function is only deployed at the SAN. In that case, a quality reduction in accuracy at the ED is provoked, as the ED cannot reconstruct the value. Comparing to deploy a prediction function at the ED, it is only useful for query-driven applications and can provoke missing crucial information if the prediction function is too accurate.

3.4.2 Dual Prediction Design

Resolving the limitations of SPD, the second possible prediction-based data forwarding method is the DPD, in which the SAN and ED deploy each of them identical functions. There are three possible learning strategies to achieve the DPD. The first strategy is to generate the model in the SAN and transmitted it to the ED. The second strategy is the model construction at the ED and distributing it to the SANs. The last possibility is to generate it inside the SAN and ED in parallel. If the model is set up in the SAN, this will require further computation from the sensors during the learning period and limit the use to only low complex algorithms. Additionally, the model needs to be shared with the ED for synchronisation, which requires additional bandwidth and energy consumption for the SAN. The deployment of the same function in SAN and ED using a parallel or ED setup procedure is only possible throughout an initialisation or training phase. During this time, all measurements are transmitted from the SANs to the ED. After the initialisation period, the dual function implementation has identical functions in the ED and the SAN. After the training period, DPD aims to reduce the transmission of measurements from the SANs and to be able to reconstruct the SANs values through the known error at the ED using the same function. Compared to the SPD, the DPD can provide a trade-off between the quality of measurements and energy consumption that can be pre-defined at the setup using an upper error bound.

Relaying on related work of DPD and SPD, most algorithms and techniques using time series forecast methods such as MA, AR, ARIMA, and EWMA [207, 208] (see the following paragraph for more details on EWMA and Section 2.3.3.1 for ARIMA). The authors in [209] compared the different methodologies in DPD and identified as most appropriate the EWMA and the naive method for archiving the best results. The naive method is a particular case of EWMA with $\alpha = 1$ and means that the predicted value $\hat{\mathbf{x}}_t = \mathbf{x}_{t-1}$. Despite time series forecast methods, other machine learning methods for online learning are proposed in the literature too [161].

3.4.3 Quality-Efficient Prediction Design

Departing from the SPD and DPD, this thesis aims to achieve a high quality of reconstructed measurements and analytical tasks while efficiently using as little energy and bandwidth as possible. The following work has been partly published in [6, 7]. The idea behind DPD techniques is used as a selective data forwarding strategy, improved with the advantages of an SPD deployment to reduce the communication overhead and enable the capacity and power at each level. In QEPD, intelligence and decision making is divided into SAN and ED. Contrary to DPD, their intelligence is independent of each other. However, QEPD requires a short initialisation period, equivalent to the DPD. During this time, an upper bound of error tolerance is defined by the system or user by expressing a threshold θ.

3.4.3.1 Sensor Node Intelligence

The sensor node intelligence is integrated by implementing at each SAN k besides the ability to receive and sense the environment a function $f(\mathbf{x})$ to predict a data vector. This prediction function's main focus is on efficiency, which reduces communication and focuses on other aspects, such as energy and storage. Therefore, only functions of low complex, light computational power and using as little storage as possible can be considered to implement inside the SAN as a prediction model. The QEPD method can be defined in more detail, that at each time t inside the SAN the actual data vector $\mathbf{x}_t$ is collected, and a predicted data vector $\hat{\mathbf{x}}_t$ is computed through the function $f_k(\mathbf{x})$. After computing the predicted data vector $\hat{\mathbf{x}}_t$ and receiving the actual data vector $\mathbf{x_t}$, a discrepancy in the form of the local prediction error e_t is calculated as:

$$e_t = d^{-0.5} \parallel \mathbf{x}_t - \hat{\mathbf{x}}_t \parallel \tag{3.2}$$

The assumption of the data space is that all values in $\mathbf{x_t}$ are normalised so that $\mathbf{x} \in [0,1]^d$. Ensuring the same scalability of e_t, a normalised factor of $d^{-0.5}$ has been added in Equation (3.2) to generate $e_t \in [0,1]$. The prediction function $f_k(\mathbf{x})$ is deployed inside each SAN k to generate the decision to forward the data vector $\mathbf{x}_t$ based on the calculated prediction error e_t. Given the calculated local prediction error, each SAN k is deciding to send if:

Case 1: the predicted data vector $\hat{\mathbf{x}}_t$ differs from the actual sensed $\mathbf{x}_t$ concerning a decision threshold $\theta \in [0,1]$, i.e., $e_t > \theta$, then the SAN k sends the real $\mathbf{x}_t$ to the ED i.

Case 2: Otherwise, i.e., $e_t \leq \theta$, the SAN k does not send $\mathbf{x}_t$ to the ED. In this case, the ED i is responsible for reconstructing the vector locally for further processing.

For the proposed QEPD, the prediction function $f_k(\mathbf{x})$ inside each device is based on the Exponentially Weighted Moving Average (EWMA), also known as Exponential Smoothing, making a one-step-ahead prediction. Using EWMA as a prediction function inside the SAN k is leading to the computational complexity of $O(d)$, and further only requires to store the current data vector $\hat{\mathbf{x}}_t$ inside each device. Using EWMA as prediction model inside SAN k leads to the following prediction at time t for the upcoming $t+1$:

$$\hat{\mathbf{x}}_{t+1} = (1-\alpha)\sum_{\tau=0}^{\infty} \alpha^{\tau}\mathbf{x}_{t-\tau} = (1-\alpha)\mathbf{x}_t + \alpha\hat{\mathbf{x}}_t \tag{3.3}$$

The parameter α with $\alpha \in [0,1]$ represents the relationship between the historical measured data and the current data. The higher the value of α is, the more influence is given to the historical values. Besides the EWMA, another classical time series forecasting model is Auto-Regression Integrated Moving Average (ARIMA). Substituting in Equation (3.3) the local prediction error e_t of Equation (3.2) leads to:

$$\mathbf{x}_t - \mathbf{e}_t = (1-\alpha)\mathbf{x}_{t-1} + \alpha(\mathbf{x}_{t-1} - \mathbf{e}_{t-1}),$$

. The change in times series is expressed by

$$\Delta\mathbf{x}_t \;\; = \;\; \mathbf{e}_t - \alpha\mathbf{e}_{t-1}. \tag{3.4}$$

By taking $\mathbf{e}_t$ to be a time series of independent $\mathcal{N}(\mathbf{0}, \sigma_e^2\mathbf{I})$ variables, the adopted EWMA recursion in (3.3) is deduced from the ARIMA with only one past vector, $p = 1$. Performing ARIMA with $p > 1$, the computational complexity inside the SAN increases to $O(d^2p)$, and data vectors over the past p times has to be stored. Aiming for the most efficient implementation in terms of scalability, storage, and computational complexity, the QEPD method is proceeding with EWMA as prediction function $f_k(\mathbf{x})$ inside the SAN k.

3.4.3.2 Edge Device Intelligence

In a DPD setup, the function deploy in SAN k and ED i is identical $f(\mathbf{x}) = g(\mathbf{u})$. In this thesis proposed QEPD, the assumption of identical functions in SAN and ED is not made. Further, in QEPD the capability of EDs is used for a more advanced reconstruction function improving the quality of imputed values. In contrast, the SAN function is deploying an energy-efficient method to focus on quality-efficient communication reduction. The reconstruction function deployed at the ED should also be light in computational complexity being energy efficient, even though it is not as limited as inside the SAN. In the abstracted application system, a network of SANs that are sending each time t its raw-data vector $\mathbf{x_t}$ towards the ED is considered. Therefore the frequency of the send data is known by the ED. At time instance t, the ED is either receiving a data vector $\mathbf{x}_t$ or nothing from an individual SAN k. If the ED receives the actual data vector, it saves $\mathbf{x}_t$ and computes its analytics with the received measurements. If the SAN k decided not to send the value, the ED is equipped with a reconstruction function $g(\mathbf{u})$, with $\mathbf{u}_t$ representing the ED data vector. Specifically, the data vector $\mathbf{u}_t$ either contains the received data vector $\mathbf{x}_t$ of SAN k or function's $g(\mathbf{u})$ reconstructed vector $\tilde{\mathbf{x}}_t$, highlighted in Equation (3.1).

The QEPD deployment is leading to different prediction errors in SAN and ED. The following should present the relationship between the local prediction error e_t (see Equation (3.2)) at the SAN k and the ED's performed reconstruction function $g(\mathbf{u})$. The implemented functions causing a reconstruction error at the ED with respect to the elaborated cases of SANs decision on sending or not sending, see Equation (3.1). This decision is leading to the reconstruction error r value inside the ED of :

$$r_t = \begin{cases} 0 & \text{Case 1,} \\ \|\mathbf{x}_t - \tilde{\mathbf{x}}_t\| & \text{Case 2.} \end{cases} \tag{3.5}$$

Aiming to analyse the local conditional expectation of the prediction error $\mathbb{E}[e_t|e_t \leq \theta]$ conditioned on event (Case 2): $\{e_t \leq \theta\}$ and its relation with the expected reconstruction difference $\mathbb{E}[r_t]$. By identifying the association of the reconstruction policies and the local prediction in the SAN k, it is possible to adapt the different policies based on the knowledge for quality improvements towards the analytical task performed at the ED.

First, concentrate on the SAN k using the prediction function $f_k(\mathbf{x})$ and the corresponding ED with the reconstruction function $g(\mathbf{u}) = \tilde{\mathbf{x}}$. Based on this derived data vector, it is possible to monitor the evolution of the errors generated at the SAN k with the prediction $\hat{\mathbf{x}}_t$ and the centralised prediction $\tilde{\mathbf{x}}_t$. Constructing a time series representing the difference between these predicted values based on the decision the SAN k made, given a dependency on θ, the threshold to decide on sending the actual value $\mathbf{x}_t$ or not to the ED, the following Equation can be derived:

$$\xi_t = \begin{cases} 0, & e_t > \theta \\ \|\hat{\mathbf{x}}_t - \tilde{\mathbf{x}}_t\|, & e_t \leq \theta \end{cases} \tag{3.6}$$

As shown from Equation (3.6), two cases of this time series can be identified, both relying on θ and the condition if $e_t \leq \theta$ or not. A detailed explanation of both cases and their relation to the errors at the SAN and ED will be shown in the following paragraph.

Case A: $\hat{\mathbf{x}}_t = \tilde{\mathbf{x}}_t$. In this case, the predicted data vector at SAN k is the same as the ED's reconstructed vector. This occurs when SAN and ED are adopting the same algorithms for prediction and reconstruction. The DPD is performing this implementation, as explained in Section 3.4.2 with $f(\mathbf{x}) = g(\mathbf{u})$. If the DPD is implemented, then the ED's expected reconstruction difference $\mathbb{E}[r]$ is bounded by the expected prediction error $\mathbb{E}[e]$ at the SANs, i.e., $\mathbb{E}[r] \leq \mathbb{E}[e]$. The proof of the expected reconstruction difference $\mathbb{E}[r]$ of the reconstruction difference in Equation (3.5) is bounded to $\mathbb{E}[e]$ of Equation (3.2) can be defined as:

$$\begin{aligned}
\mathbb{E}[r] &= \mathbb{E}[\|\mathbf{x} - \tilde{\mathbf{x}}\| | e \leq \theta] P(e \leq \theta) \\
&= \mathbb{E}[\|\mathbf{x} - \hat{\mathbf{x}}\| | e \leq \theta] P(e \leq \theta) \\
&= \mathbb{E}[e | e \leq \theta] P(e \leq \theta) \\
&= \int_0^\theta e p_i(e) de \leq \mathbb{E}[e] \qquad (3.7)
\end{aligned}$$

In Equation (3.7) is $P(e \leq \theta)$ the probability of Case 2, where no data vector is delivered from SAN k to the ED, w.r.t., θ. Further is $p_i(e)$ defined as the probability distribution of the local prediction error at SAN k. As proven above, the predicted vector at SAN k is the same as the reconstruction vector at ED for Case A. Therefore, the expected reconstruction difference $\mathbb{E}[r]$ is bounded by the expected prediction error $\mathbb{E}[e]$. This demonstrates that the evolution of the reconstructed data vectors at the ED is known to the SAN k. If the SAN k produces those vectors from its local predictor $f_k(\mathbf{x})$, it knows the upper bound of the expected reconstruction error that the ED will experience. This enables the SAN to adjust the decision threshold θ towards satisfy the accuracy needs of the launched IoT analytics application. The expected prediction error $\mathbb{E}[e]$ can be adjusted at the SAN k by incrementally approximating the error mean $\tilde{e}_t$ as: $\tilde{e}_t = \tilde{e}_{t-1} + \frac{1}{t}(e_t - \tilde{e}_{t-1})$ for a large $t > 0$.

Case B: $\hat{\mathbf{x}}_t = \tilde{\mathbf{x}}_t + \boldsymbol{\rho}_t$, where $\boldsymbol{\rho}_t$ is the vector discrepancy of the predicted vector and the reconstructed vector, given that $e_t \leq \theta$ at SAN k, with $\mathbb{E}[\|\boldsymbol{\rho}\|] < \infty$. In this case, the expected reconstruction difference $\mathbb{E}[r]$ at the ED is bounded by the expected prediction error $\mathbb{E}[e]$ at SAN and the expectation of ξ (see Equation (3.6)), i.e., $\mathbb{E}[r] \leq \mathbb{E}[e] + \mathbb{E}[\xi]$. This bound of the expected reconstruction difference $\mathbb{E}[r]$ in Equation (3.5) for Case B can be proven as follow:

$$\begin{aligned}
\mathbb{E}[r] &= \mathbb{E}[\|\mathbf{x} - \tilde{\mathbf{x}}\| | e \leq \theta] P(e \leq \theta) = \mathbb{E}[\|(\mathbf{x} - \hat{\mathbf{x}}) + (\hat{\mathbf{x}} - \tilde{\mathbf{x}})\| | e \leq \theta] P(e \leq \theta) \\
&\leq \mathbb{E}[\|\mathbf{x} - \hat{\mathbf{x}}\| | e \leq \theta] P(e \leq \theta) + \mathbb{E}[\|\boldsymbol{\rho}\| \big| e \leq \theta] P(e \leq \theta) \\
&= \int_0^\theta e p_i(e) de + \mathbb{E}[\|\boldsymbol{\rho}\| \big| e \leq \theta] P(e \leq \theta) \\
&\leq \mathbb{E}[e] + \mathbb{E}[\|\boldsymbol{\rho}\| \big| e \leq \theta] P(e \leq \theta). \qquad (3.8)
\end{aligned}$$

Inferred from this, the expected reconstruction difference is bounded at least by the expected prediction error, known at SAN k, and the conditional expectation discrepancy $\mathbb{E}[\|\boldsymbol{\rho}\| \big| e \leq \theta] P(e \leq \theta)$ derived by the intrinsic difference of the reconstructed and predicted vectors.

This conditional expectation refers to the expected value $\mathbb{E}[\xi]$ of the time series ξ_t defined in Equation (3.6) to this:

$$\mathbb{E}[\xi] = \mathbb{E}[\|\hat{\mathbf{x}} - \tilde{\mathbf{x}}\| | e \leq \theta] P(e \leq \theta) = \mathbb{E}[\|\boldsymbol{\rho}\| | e \leq \theta] P(e \leq \theta) \tag{3.9}$$

The times series evolution ξ_t can be monitored in a training phase where the ED sends the reconstructed vectors $\tilde{\mathbf{x}}_t$ to the SAN k. After this training phase, the SAN k is aware of the expected discrepancy by approximating the time series's mean $\tilde{\xi}_t = \tilde{\xi}_{t-1} + \frac{1}{t}(\xi_t - \tilde{\xi}_{t-1})$. Based on this learned evolution of the time series ξ, the SAN k knows the upper bound of the expected reconstruction error that the ED will experience and can adjust the application-specific decision threshold θ. Moreover, during this training phase, the SAN k can send the pairs $(\mathbf{x}_t, \hat{\mathbf{x}}_t)$ to the ED to locally approximate both the expected prediction error and the expected discrepancy. In this context, the ED can adjust the current reconstruction policy (Policy 1, 2, 3, 4, or 5) by selecting the policy that corresponds to the minimum $\mathbb{E}[r]$.

The proposed methodology of QEPD introduces five possible reconstruction functions $g(\mathbf{u})$ defined as policies, which can recreate the ED's missing values. The Algorithm 1 highlights the different policies for QEPD towards reconstructing the values that the SANs decided not to send based on their predictive error tolerance.

Policy 1: The first policy is to replace the missing data vector with the previously received values of a specific SAN k as the current data vector, i.e. $\mathbf{u}_t = \mathbf{u}_{t-1}$. This leads to the possibility that the last seen actual data vector is duplicated multiple times and used for analytics over a long period.

Policy 2: The second policy uses the average value of previously received values over a time horizon M to generate the missing data vector $\mathbf{u}_t$, leading to a reconstruction of $\mathbf{u}_t = \frac{1}{M}\sum_{i=t-M}^{t-1} \mathbf{u}_i$. Using the average data vector is requiring a computational complexity of $O(Md)$.

Policy 3: The third reconstruction function is of light computational complexity requiring only $O(d)$. It is an efficient variation of reconstructing time series, as proof has been given in the previous section (Section 3.4.3.1); Exponential Smoothing is used to replace $\mathbf{x}_t$ with the calculated value that is considering the historical data and most recent measurement in weighting. This influencing of past and current is leading to the calculation of the smoothed value $\mathbf{s}_t$ with $\mathbf{s}_t = \alpha\mathbf{x}_t + (1-\alpha)s_{t-1}$. If the data vector $\mathbf{x}_t$ has not been sent to the ED, the function $g(\mathbf{u}_t)$ is using Exponential Smoothing so that $\tilde{\mathbf{x}}_t = \mathbf{s}_{t-1}$.

Policy 4: Departing from a function with low complexity and computational power, such as EWMA, towards a more advanced algorithm to reconstruct the missing data vector. The fourth policy is using adaptive filtering with the Least-mean-squared (LMS) method based on the work of [210]. LMS is used as it requires the least computational power of all other adaptive filter methods and generates good prediction values. LMS requires storing the last M values and a vector of weights w for the last M values. These weights are updated based on the calculated error each time t. Therefore its computational complexity is $O(2M+1)$ for multiplication, $O(2M)$ for addition, and $O(M)$ for memory. The predicted value $\tilde{\mathbf{x}}_t = \sum_{i=1}^{M} \mathbf{w}_i \mathbf{x}_i$. For more details on adaptive filter implementation, see [211, 212].

Policy 5: ARIMA is of high complexity depending on p; however, the deployment inside the ED is more feasible than inside the SAN. Therefore, the fifth policy is reconstructing the missing values by deploying an ARIMA function inside the ED.

Algorithm 1 Overview algorithm for reconstruction policies.

```
Data: Collected data in ED from each SAN $k$
Result: Reconstruction of data vector at CL for each connected ED $i$
 1: for SAN $k$ do
 2:     if $e_t > \theta_t$ then
 3:         SAN $k$ sends $\mathbf{x}_t$ to ED
 4:         $\mathbf{u}_t = \mathbf{x}_t$
 5:     else
 6:         Reconstruction using Policy 1
 7:         $\mathbf{u}_t = \mathbf{u}_{t-1}$
 8:         Reconstruction using Policy 2
 9:         $\mathbf{u}_t = \frac{1}{M} \sum_{k=t-M}^{t-1} \mathbf{u}_k$
10:         Reconstruction using Policy 3
11:         $\mathbf{s}_{t-1} = \alpha \mathbf{x}_{t-1} + (1-\alpha)\mathbf{s}_{t-2}$
12:         $\mathbf{u}_t = \mathbf{s}'_{t-1}$
13:         Reconstruction using Policy 4
14:         $\mathbf{u}_t = LMS(\mathbf{x}_{t-1}) = \sum_{i=1}^{M} \mathbf{w}_i \mathbf{x}_i$
15:         Reconstruction using Policy 5
16:         $\mathbf{u}_t = ARIMA(\mathbf{x}_{t-1})$
17:     end if
18: end for
```

3.5 Time-Optimised Decision Forwarding

In the previous section, the prediction-based data forwarding strategies imply an instantaneous decision making (IDM) approach at the SAN. These methods implemented through DPD and QEPD only considers the current local prediction error to decide on forwarding the data vector $\mathbf{x}_t$ or not. In this section, time-optimised decision making is presented that incorporates the past local prediction errors into the forwarding decision. The presented work in this section has been published in [5]. Using the IDM even with advanced intelligence at the SAN causes major drawbacks at the ED. The following section will highlight and define these drawbacks by proposing an optimisation at the SAN decision making using a time-optimised quality-efficient forwarding intelligence.

3.5.1 Limitations of Prediction-Based Forwarding

Using IDM inside the SAN results in two possible drawbacks. The first drawback occurs if the prediction function implemented at the SAN is generating very accurate predictions. This results in low local prediction errors and a continuously non-sending of the data vector $\mathbf{x}_t$ towards the ED. Even though this is drastically reducing the communication, the ED faces the issue of not reconstructing the values accurately. Moreover, the ED is unable to follow the data stream and distribution of the measurements. The second drawback occurs if only outliers and novel data are sent to the ED. This appears if the SAN is only sending values that excessively exceed the threshold θ. These outliers and novel data are then interpolated into the reconstruction function $g(\mathbf{u})$ at the ED. On the other hand, the ED models the data with only or many outliers, which results in low accuracy of the reconstructed and imputed values and therefore minimises the quality of analytical tasks.

Figure 3.2 highlights these two drawbacks of IDM implementations, e.g. the presented DPD and QEPD, showing the actual sensed data at the SAN and the predicted and reconstructed data from the ED. This figure shows that IDM can have significant drawbacks that result into low qualitative analytical tasks and tremendous information loss at the ED.

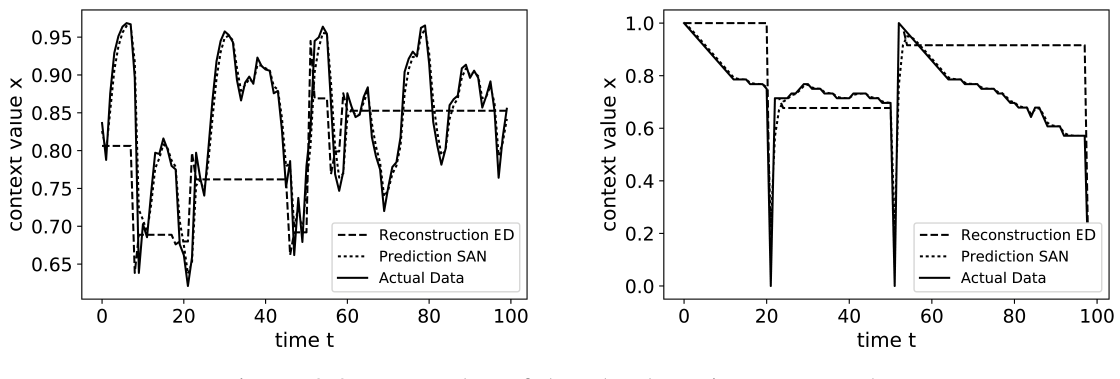


Figure 3.2: Examples of drawbacks using IDM at the ED.

3.5.2 Optimal Stopping Time Forwarding Strategy

Under consideration of the defined drawbacks of instantaneous decision making (IDM) with the introduced methodologies of Dual Prediction Design (DPD) and Quality-Efficient Prediction Design (QEPD) as presented in Section 3.4, the hereinafter section is solving the drawbacks mentioned above. The coming paragraphs introduce the strategy of optimal time forwarding. This approach uses Optimal Stopping Theory (OST) [2] to identify the optimal time t^* at the SAN k to forward the vector $\mathbf{x}_t$ towards the ED. The fundamentals of OST and its assumptions have been highlighted and introduced with the secretary problem in Section 2.3.5. This theory of problem is used as basis for this chapter. In the proposed optimal-time forwarding strategy, the past decisions of forwarding are considered for the upcoming decision. The decisions are relying on a prediction-based data forwarding mechanism similar to the IDM methodologies. A strategy is developed that generates a trade-off between communication to the ED and quality of the analytical tasks by maximising the delay tolerance.

As proven in Equation (3.8), the SAN prediction error's conditional expectation is an upper bound of the expected reconstruction error at the ED. Based on this, it is possible to define the stochastic indicator Z_t whose value depends on $e_t = \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|$:

$$Z_t = \begin{cases} \lambda\theta & \text{if } e_t > \theta, \\ e_t & \text{if } e_t \leq \theta. \end{cases} \tag{3.10}$$

In the DPD and QEPD methodology, the indicator Z_t is developed at each time and if $e_t > \theta$, the measurement at time t is directly sent without any consideration of Z_{t-1}. Incorporating the history of Z_t values the cumulative sum of the generated Z_t, with $\tau < t$, can be defined as

$$S_t = \sum_{\tau=0}^{t} Z_\tau. \tag{3.11}$$

The quantity of S_t provides information to the SAN to decide if it is best to delay further or send. This can be summarised in a reward function Y_t with $\beta \in (0, 1)$ indicating the delay tolerance level and discount factor:

$$Y_t = \beta^t S_t = \beta^t \sum_{\tau=0}^{t} Z_\tau, \tag{3.12}$$

As the events of S_t and consequently Z_t are random, the SAN tries to find the optimal time t^* to forward the measurements to the ED by maximising the expectation of Y_t, $\mathbb{E}[Y_t]$ with a fixed β and θ. This is formally defined as finding the supremum of the expectation of Y_t:

$$\sup_{t \geq 0} \mathbb{E}[Y_t]. \tag{3.13}$$

Proofing that the optimal time t^* exists at SAN can be shown using the fundamentals of OST. Two conditions need to be satisfied: (C1) $\limsup_t Y_t \leq Y_\infty = 0$ is surely true and (C2) $\mathbb{E}[\sup_t Y_t] < \infty$. C1 implies that with the elapse of time ($t \to \infty$), the reward should go to zero, i.e., $Y_\infty = 0$. Since no delivery of a data vector over an indefinite horizon is useless due to extremely high reconstruction error at ED, $Y_\infty = 0$ represents the reward of an endless non-delivery phase. The supremum limit of Y_t is notated by $\limsup_t Y_t$, i.e., the limit of $\sup_t Y_t$ as $t \to \infty$ or $\lim_{t\to\infty}(\sup\{Y_j : j \geq t\})$. As Z_t is non-negative and using the strong law of numbers $(\frac{1}{t}\sum_{j=1}^{t} Z_j) \to \mathbb{E}[Z]$ it is possible to derive:

$$Y_t = t\beta^t(S_t/t) = t\beta^t(1/t)\sum_{j=1}^{t} Z_j \sim t\beta^t\mathbb{E}[Z] \overset{\text{a.s.}}{\to} 0, \tag{3.14}$$

This results to $\lim_{t\to\infty}\sup_t Y_t = 0$. As $Y_\infty = 0$ is by definition true, it is possible to declare C1 as satisfied. C2 implies that the expected reward under any policy is finite. Therefore, C2 can be shown as:

$$\sup_t Y_t = \sup_t \beta^t \sum_{j=1}^{t} Z_j \leq \sup_t \sum_{j=1}^{t} \beta^j Z_j \leq \sum_{j=1}^{\infty} \beta^j Z_j. \tag{3.15}$$

This results into satisfying C2 with,

$$\mathbb{E}[\sup_t Y_t] \leq \sum_{j=1}^{\infty} \beta^j \mathbb{E}[Z] = \mathbb{E}[Z]\frac{\beta}{1-\beta} < \infty. \tag{3.16}$$

As both conditions are satisfied and proven, it can be shown that the optimal time t^* for forwarding the measurements in Equation (3.13) exists.

Proofing of the existence of the optimal time t^* desires to find that optimal time inside the SAN which enables it to decide on forwarding the measurement by maximising the trade-off between communication and accuracy.

Since Y_t is non-negative, the Equation (3.13) is monotone [213] so that the optimal time t^* is obtained by the one-stage look-ahead optimal rule (1-sla):

$$t^* = \inf\{t \geq 1 | Y_t \geq \mathbb{E}[Y_{t+1}]\}. \tag{3.17}$$

The adoption of 1-sla is optimal since $\sup_t Y_t$ has a finite expectation (equal to $\mathbb{E}[Z]\frac{\beta}{1-\beta}$) and $\limsup_t Y_t = 0$, as proved in Equation (3.14). Consequently, t^* is estimated through the principle of optimality. Presume $S_t = s$ when a SAN decides that it is optimal to forward a vector. Then, the current reward of $\beta^t s$ is at least as large as any expected $\mathbb{E}[(\frac{\beta}{1-\beta})^{t+\tau}(s + S_\tau)]$, which means that: $s(1 - \mathbb{E}[(\frac{\beta}{1-\beta})^\tau]) \geq \mathbb{E}[(\frac{\beta}{1-\beta})^\tau S_\tau]$ for all times τ. This must hold true for all $s' \geq s$, so that the optimal time t^* for some s_0 must be of the form $t^* = \inf\{t \geq 1 | S_t \geq s_0\}$. Especially when the SAN forwards at the first time t for which $S_t \geq s_0$, then the tolerance for forwarding s_0 must be the same as the tolerance for continuing using the 1-sla, therefore the sum of tolerances is positive. That is, s_0 must satisfy the equation

$$s_0 = \mathbb{E}[(\frac{\beta}{1-\beta})^\tau (s_0 + S_\tau)], \tag{3.18}$$

with $\tau = \inf\{t \geq 1 | S_\tau > 0\}$. Since Y is non-negative, it is possible to obtain $\tau \equiv 1$ and $S_\tau \equiv Y$ [213] and then replacing with $s_0 = \frac{\beta}{1-\beta}\mathbb{E}[Y]$. This will finally result in the definition of the optimal time t^* to forward the measurement to the ED with:

$$t^* = \inf\{t \geq 1 | \sum_{k=1}^{t} Z_k \geq \frac{\beta}{1-\beta}\mathbb{E}[Z]\}. \tag{3.19}$$

Given the Equation (3.19), it is possible to observe the events $\{e_t \lesseqgtr \theta\}$, which evaluates Z_t and S_t at the SAN each time t. Whenever Equation (3.19) holds true, the SAN forwards $\mathbf{x}_t$ to the ED and resets the sum to zero, starting a new optimal time forwarding period. However, to prompt Equation (3.19) it requires the knowledge of $\mathbb{E}[Z]$ at the SAN, which is associated with the conditional expectation of the prediction error $\mathbb{E}[e|e \leq \theta]$ as discussed in Section 3.4.3. $\mathbb{E}[e|e \leq \theta]$ is known to the SAN as proven in Equation (3.8). Estimating with an incremental mechanism $\mathbb{E}[Z]$ based on the SAN's expected prediction error, it is possible to obtain that

$$\begin{aligned} \mathbb{E}[Z] &= \mathbb{E}[Z|e > \theta]P(e > \theta) + \mathbb{E}[Z|e \leq \theta]P(e \leq \theta) \\ &= \lambda\theta - \int_0^\theta (\lambda\theta - e)p(e)de = \lambda\theta - \mathcal{I}(\theta), \end{aligned} \tag{3.20}$$

where $\mathcal{I}(\theta) = \int_0^\theta (\lambda\theta - e)p(e)de$ and $p(e)$ is the Probability Density Function (PDF) of the prediction error in the SAN. Note, $\mathcal{I}(\theta) \leq \lambda\theta$ because $\int_0^\theta ep(e)de \leq \int_0^\theta \lambda\theta p(e)de \leq \lambda\theta$. The criterion (3.19) is based on the estimation of $\mathcal{I}(\theta)$, which involves the estimation of $p(e)$. The approximation of $p(e)$ at time t, notated by $\hat{p}^{(t)}(e)$, is based on incremental Kernel Density Estimation (KDE) from the sequence $e_1, \ldots, e_t$:

$$\hat{p}^{(t)}(e) = \frac{1}{t}\sum_{j=1}^{t} K_h(e - e_j), \tag{3.21}$$

where $K_h(u)$ is a kernel function, unimodal, symmetric, non-negative that centers at zero and integrates to unity while the window h controls the degree of smoothing of the estimation. The PDF of e is then estimated incrementally as:

$$\hat{p}^{(t)}(e) = \frac{t-1}{t}\hat{p}^{(t-1)}(e) + \frac{1}{t}K_h(e - e_t) \tag{3.22}$$

with $\hat{p}^{(1)}(e) = K_h(e - e_1)$. The integral $\mathcal{I}(\theta)$ can be then incrementally estimated based on

$\hat{p}^{(t-1)}(e)$ and e_t at $t > 1$ based on the recursion:

$$\mathcal{I}^{(t)}(\theta) = \frac{t-1}{t}\mathcal{I}^{(t-1)}(\theta) + \frac{1}{t}\int_0^{\theta}(\theta-u)K_h(u-e_t)du \tag{3.23}$$

When the SAN obtains e_t, only the evaluation of $q^{(t)}(e) = \frac{1}{t}\int_0^{\theta}(\theta-u)K_h(u-e)du$ is needed to check the Equation (3.19) with an initial $\mathcal{I}^{(1)}(\theta) = q^{(1)}(e_1)$ at time $t = 1$. There are specific kernels K_h that can be adopted here, e.g., epanechnikov and gaussian kernel. The gaussian kernel is mostly used due to its convenient mathematical properties and, especially when dealing with PDF estimation. Therefore, this thesis adopts the gaussian $K_h(u) = \frac{1}{\sqrt{2\pi}h}e^{-\frac{1}{2}(\frac{u}{h})^2}$ where the optimal value of h is $h^* = 1.06\hat{\sigma}T^{-\frac{1}{5}}$ [214], where $\hat{\sigma}$ is the standard deviation of e, and T is the number of training error values. Based on K_{h^*}, SAN easily calculates $q^{(t)}(e_t)$ as follow:

$$\begin{aligned} q^{(t)}(e_t) = \frac{1}{\sqrt{2\pi}t}&[c(e_t-\theta) \\ &(erf(\frac{\sqrt{2}}{2h^*}(e_t-\theta)) - erf(\frac{\sqrt{2}}{2h^*}e_t)) \\ &h^*(\exp^{-\frac{1}{2}(\frac{e_t-\theta}{h^*})^2} + \exp^{-\frac{1}{2}(\frac{e_t}{h^*})^2})], \end{aligned} \tag{3.24}$$

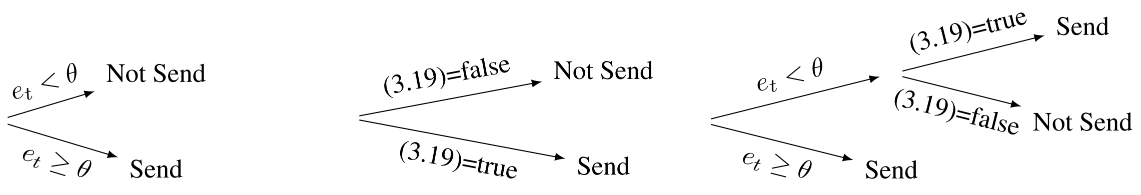
with $c = 1.25331$ and $erf(u)$ is the error function. The SAN can then evaluate Equation (3.19) for time-optimised forwarding in $O(1)$ through $\mathcal{I}(\theta)$ in (3.23). Please note, using a training phase during deployment, it is possible to compute the optimal time t^* given the above introduced formula.

3.5.3 Time-Optimised Forwarding Deployment

The deployment of the introduced time-optimised data forwarding strategy is based on two variants, which are depending on the value of λ in Equation (3.10). The following paragraph introduces the Time-Optimised Forwarding Strategy (TOFS) based on $\lambda = 1$ and the Hybrid-Time-Optimised Forwarding Strategy (HTOFS) with $\lambda = 0$.

With applying $\lambda = 1$, adoption of the Time-Optimised Forwarding Strategy (TOFS) is deployed. This variant uses θ as the upper bound value if the tolerance level exceeds e_t. In TOFS, the cumulative sum S_t as in Equation (3.11) continuously increases, even if the prediction function f_k in SAN generates an accurate forecast w.r.t. θ or not. This strict deployment takes into consideration even the relatively small prediction errors for deciding on data forwarding. Figure 3.3 (b) shows the TOFS decision tree. Comparing the previous prediction-based deployments of DPD and QEPD representing instantaneous decision making (IDM) as shown in Figure 3.3 (a), TOFS is purely based on S_t, and a forwarding decision is triggered w.r.t. Equation (3.19).

When using $\lambda = 0$, the Hybrid-Time-Optimised Forwarding Strategy (HTOFS) version is adopted. The deployment of HTOFS imposes a penalty towards S_t only when the prediction function $f_k(\mathbf{x})$ in SAN k forecasts the expected measurement with $e_t < \theta$. If the prediction error exceeds θ, HTOFS immediately sends the measurement to the ED. In this case, S_t does not always monotonically increase. HTOFS combines both the TOFS in the case where $\{e_t < \theta\}$, and the IDM in the case where the current prediction error exceeds θ. Using the TOFS by accumulating only the tolerances of S_t it is possible to address the first limitation highlighted in Section 3.5.1 (dealing with long absences of sending). Incorporating the IDM decision making into HTOFS, it is possible to capture immediately any significant event/outlier/novelty values resulting in resolving the other limitation introduced in the previous section. Figure 3.3 (c) shows the HTOFS decision tree represented by fusing decisions of TOFS and IDM.



(a) IDM (b) TOFS (c) Hybrid TOFS

Figure 3.3: The decision trees for IDM and variants TOFS and HTOFS. TOFS is triggered based on (3.19); HTOFS combines both decision trees of IDM and TOFS.

3.6 Performance Evaluation

3.6.1 Experimental Setup

Two real datasets (DSs) are used for evaluation to assess the performance of the introduced prediction-based and time-optimised data forwarding methodologies. The first dataset (DS1) has been made available through the Intel Berkeley Research Lab [215]. The DS1 contains a collection of $k = 54$ SANs measuring a $d = 4$-dimensional space of temperature, humidity, light, and voltage. The data has been collected every 31 seconds over 36 days, resulting in a dataset size of $N = 2,300,000$. However, some sensors show a high degree of missing values, so these sensors have been discarded during the experiments, leaving the DS1 with $k = 21$ SANs. Additionally, as the voltage dimension is relatively constant, only a 2-dimensional context vector $\mathbf{x}$ is considered throughout the experimentation containing temperature and humidity. The second dataset (DS2) contains 415 weather stations around the United Kingdom (UK) measuring the surrounding environment's contextual data. This data has been collected over the time horizon of December 2017 till March 2018 using the

API of Wunderground [216]. Each weather station represents a SAN ($k = 415$), which measures a 9-dimensional vector $\mathbf{x}$ containing temperature, dew point, humidity, wind-speed, wind-gust, wind direction, pressure, wind chill, and precipitation. For comparison, only a 2-dimensional data vector containing temperature and humidity are considered throughout the experiments. The data collected frequency is by every 5 minutes over the time horizon of 87 days, resulting in a dataset size of $N = 9,044,683$ with assembling roughly 250 values measured per SAN per day.

Independent on the dataset, each SAN is equipped with a function $f_k(\mathbf{x})$ to generate the prediction-based data decision forwarding or time-optimised forwarding. This function $f_k(\mathbf{x})$ is further deployed in the ED i, noted as $g(\mathbf{u})$, as a dual prediction deployment is considered. For simplicity, during the experiments, only one ED is set as a central collection point. However, this can be further expanded and is flexible and independent of the results. Guaranteeing a consistent behaviour of the functions $f_k(\mathbf{x})$ at the SAN and $g(\mathbf{u})$ at the ED, an initialisation or training period is required. DS1 uses an initialisation period of 8,000 measurements ($t = [0, 8000]$) and a testing period of over 36,000 ($t = [8000, 44000]$). In DS2, the time instance for training is set to $t = [0, 3000]$ and the testing period to $t = [3000, 38000]$. After the initialisation period, the methodologies explained in Section 3.6.3 will be carried out inside the SAN and ED.

The experiments using normalised data and is implemented so that each SAN k normalises and scales its context vectors, i.e., each parameter $x \in \mathbb{R}$ is mapped to $\frac{x-\mu}{\sigma}$ with mean value μ and variance σ and scaled in [0,1], thus vector $\mathbf{x} \in [0, 1]^d$.

3.6.2 Performance Metrics

The performance of the introduced methodologies will be evaluated with different metrics to provide an extensive comparison. These metrics can be divided into three distinct groups: *sensitivity*, *accuracy*, and *communication* metrics.

The quality or similarity of two given time series, one being the actual data and the other a reconstructed input, can be analysed through the *sensitivity*. The metrics used to measure the sensitivity in this section are (1) Kullback-Leibler (KL) divergence, (2) Coefficient of Variation (CV), (3) Euclidean Distance, and (4) Dynamic Time Warping (DTW). These four sensitivity metrics compare the actual data $\mathbf{x}$ measured in the SAN k with the reconstructed data $\tilde{\mathbf{x}}$ at the ED i for each of the introduced methodologies of DPD, QEPD, TOFS, and HTOFS summarised in Section 3.6.3.

An advanced metric for the sensitivity analysis is the Coefficient of Variation (CV), also known as relative standard deviation. CV is a standardised measure of the dispersion of the probability distribution $p(\mathbf{x}_t)$ of the time series $\mathbf{x}_t$. It is expressed as the ratio of the standard deviation σ_x to the absolute mean value $|\mu_x|$, i.e.,

$$CV(\mathbf{x}) = \frac{\sigma_x}{|\mu_x|}. \tag{3.25}$$

In this thesis, $CV(\mathbf{x})$ is adopted for the sensitivity analysis because the standard deviation σ_x of the time series $\mathbf{x}_t$ must be understood in the context of the mean of the time series compared with the $CV(\tilde{\mathbf{x}})$ of the reconstructed time series at the ED, i.e., $\frac{\sigma_{\tilde{x}}}{|\mu_{\tilde{x}}|}$. This thesis uses CV for comparing both the mean and the variance of the reconstructed time series in ED i with the actual time series in SAN k data. The CV's value is independent of the unit, i.e., it is a dimensionless number. The rationale behind investigating the sensitivity of the introduced mechanisms w.r.t. the CV is that many natural processes indeed show a correlation between the average value and the variation around it. By observing the coefficients of variation $CV(\tilde{\mathbf{x}})$ and $CV(\mathbf{x})$ of the reconstructed and actual time series, respectively, the discrepancy of the reconstructed time series at ED i w.r.t. the true time series at SAN k due to the adoption of specific mechanisms and decision threshold θ can be assessed. Ideally, the reconstructed mean and variance of the times series in ED i follows the actual mean and variance of the exact time series in SAN k.

Another baseline sensitivity metric is the Kullback-Leibler (KL) divergence. KL divergence from $p(\mathbf{x})$ to $p(\tilde{\mathbf{x}})$ denotes the information loss when attempting to reconstruct time series $\tilde{\mathbf{x}}$ for the actual time series $\mathbf{x}$, using $p(\tilde{\mathbf{x}})$ and $p(\mathbf{x})$ as the probability distribution functions, respectively. KL is defined as:

$$KL(p(\tilde{\mathbf{x}})\|p(\mathbf{x})) = \int_0^1 p(\tilde{\mathbf{x}}) \log \frac{p(\tilde{\mathbf{x}})}{p(\mathbf{x})} dx. \tag{3.26}$$

In this thesis, KL indicates the amount of information lost when ED i approximates the actual time series at SAN k based on the proposed methodologies. Ideally, $p(\tilde{\mathbf{x}})$ should be as close to the genuine $p(\mathbf{x})$ given certain reconstruction functions.

Further to the metrics mentioned above, analysing the real data's sensitivity towards the reconstructed data can be done by calculating the distance between them. This distance calculation can be done by the simplest form using Euclidean distance or L^2-norm. Euclidean distance (L^2-norm) calculates the distance between two time series $\mathbf{x} = (x_1, x_2, ..., x_t)^T$ and $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_t)^T$ by the length of the line of each of any two points in a plane. Both given time series have to be of the same length t. The distance $dist$ between each of the points inside the time series is then defined as:

$$\text{L}^2\text{-norm} = dist(\mathbf{x}, \tilde{\mathbf{x}}) = \sqrt{(x_1^2 - \tilde{x}_1^2) + ... + (x_t^2 - \tilde{x}_t^2)} \tag{3.27}$$

The Euclidean distance requires both time series lengths to be equal and compares only the points at one time instance with each other. As many time series occur to have the same distribution and being highly similar but, e.g., shifted by a time instance, a more dynamic metric allowing the flexibility has been introduced as similarity measurement of time series, named Dynamic Time Warping (DTW) [217, 218]. DTW calculates the similarity of the original data stream $\mathbf{x} = (x_1, x_2, ..., x_g, ..., x_n)^T$ and the reconstructed $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2, ..., x_h, ..., \tilde{x}_m)^T$ based on finding the optimal wrapping path $\mathcal{P}$ by generating a distance matrix using a distance function, e.g. $dist = |x_i - \tilde{x}_j|$, for each data-point. The optimal path $\mathcal{P}$ with $\mathcal{P} = p_1, p_2, ..., p_r, ..., p_R$ containing for element r an optimal point alignment with $p_r = (g, h)_r$. The optimal path $\mathcal{P}$ can be found by minimising under certain criteria (more details in Section 4.5.2) the cumulative distance for each path leading to:

$$dtw(\mathbf{x}, \tilde{\mathbf{x}}) = \min_p (\sum_{r=1}^{R} dist(p_r)) \tag{3.28}$$

Besides the knowledge about quality and similarity using sensitivity metrics, the performance in terms of *accuracy* using the reconstructed data stream's errors is of importance. The metrics used to assess the performance in terms of accuracy are: (1) Root-Mean-Squared-Error (RMSE), (2) Mean absolute error (MAE), and (3) Symmetric mean absolute percentage error (SMAPE).

The RMSE is showing the average deviation of the reconstructed time series $\tilde{\mathbf{x}}$ from the real values of time series $\mathbf{x}$ over a time horizon t,

$$\text{RMSE} = \sqrt{\frac{1}{T} \sum_{i=1}^{T} (\mathbf{x}_i - \tilde{\mathbf{x}}_i)} \tag{3.29}$$

Whereas the RMSE is giving higher penalty towards larger errors, the MAE does not. Therefore, in this experiment both metrics are considered for evaluation. The MAE is defined as:

$$\text{MAE} = \frac{1}{T} \sum_{i=1}^{T} |(\mathbf{x}_i - \tilde{\mathbf{x}}_i)| \tag{3.30}$$

SMAPE is used as accuracy metric through its unbiased properties [219] and the ability to represent the difference in a percentage value $[0, 100]$ defined with the equation:

$$\text{SMAPE} = \frac{100}{T} \sum_{t=1}^{T} \frac{\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|}{\|\mathbf{x}_t\| + \|\tilde{\mathbf{x}}_t\|} \tag{3.31}$$

The last group of metrics is analysing the communication. The *communication* is measured as the number of times the transmission of data occurs from g SAN to the ED.

$$c(T) = \sum_{t=1}^{T}\sum_{k=1}^{g} I_{k,t} \tag{3.32}$$

Equation (3.32) shows the metric for communication with $I_{k,t} = 1$ if SAN k sends its sensed value to the ED; otherwise $I_{k,t} = 0$. Evidently, the overall communication of g SANs over T sensed values, in the baseline method is $T \cdot g$, since $I_{k,t} = 1, \forall k, t$.The percentage of communication is then $\frac{c(T)}{T_g}$.

3.6.3 Comparative Assessment

The assessment of the data forwarding strategies can be divided into predictive-based data forwarding and time-optimised. Starting with the prediction-based strategies introduced in Section 3.7.1. Related work of dual prediction-based (DPD) data forwarding has been introduced with the proposed extension of quality-efficient strategies using different functions inside the SAN and ED. Based on the introduced methodologies, the experiments are structured in comparing different DPD methods with the proposed QEPD. Overall the following methods are evaluated for the decision on forwarding based on the prediction $\hat{\mathbf{x}}_t$ and the reconstructed value at the ED for $\mathbf{u}_t$ using the following functions:

1. The baseline of all experiments is the **NAIVE** method. This method does not require much storage or computational power as just the last measured value $\mathbf{x}_{t-1}$ is used as a replacement value inside the ED or value for deciding on forwarding inside the SAN.

2. The **MEAN** function is also of low complexity and computation and uses as $\hat{\mathbf{x}}_t$ or $\mathbf{u}$ the mean over all values inside a Sliding Window (SLW) of size M.

3. The next method of predicting future measurements has been extensively studied using the **ARIMA** model [220]. Already stated in Section 3.4.3, ARIMA requires a computational complexity of $O(d^2p)$.

4. The **EWMA** method is representing a special form of ARIMA and uses the theory of simple exponential smoothing ($\mathbf{s}_t = \alpha\mathbf{x}_t + (1-\alpha)s_{t-1}$). Resulting in the complexity of $O(d)$.

5. A more adaptive and continuous relearning method is by using **LMS** [210]. Resulting in the computational complexity of $O(2N+1)$ for multiplication and $O(2N)$ for addition (see for more details Policy 5 of Section 3.4.3).

Given the list of methods, it is possible to construct a matrix of combinations based on the methodologies in Section 3.4.2 and 3.4.3. Table 3.1 gives an overview of the combinations tested in the followed assessment section over different datasets. As the QEPD methods aiming to use the least possible computational complexity function inside the SAN, only NAIVE and EWMA are used inside the SAN, whereas the ED deploys all possible five introduced policies and functions.

		ED				
		NAIVE	ARIMA	EWMA	MEAN	LMS
SAN	NAIVE	DPD-N	QEPD-NA	QEPD-NE	QEPD-NM	QEPD-NL
	ARIMA	-	DPD-A	-	-	-
	EWMA	QEPD-EN	QEPD-EA	DPD-E	QEPD-EM	QEPD-EL
	MEAN	-	-	-	DPD-M	-
	LMS	-	-	-	-	DPD-L

Table 3.1: Overview of comparative assessment models for prediction-based data forwarding.

Based on the prediction-based data forwarding and the different introduced methods, the Time-Optimised Forwarding Strategy (TOFS) decides independently inside the SAN on the defined error bound θ when to forward the measurement. The methods using the prediction of $\hat{x}$ inside the SAN are based on the introduced functions above for DPD and the proposed QEPD. They form the baseline to compare against different methods of time-optimised decision forwarding. This results in the combination for Time-Optimised Forwarding Strategy (TOFS) shown in Table 3.2. Similar combining the five policy and functions towards the Hybrid-Time-Optimised Forwarding Strategy (HTOFS) results into the combinations highlighted in Table 3.3.

		ED				
		NAIVE	ARIMA	EWMA	MEAN	LMS
SAN	NAIVE	TOFS-N	TOFS-NA	TOFS-NE	TOFS-NM	TOFS-NL
	ARIMA	-	TOFS-A	-	-	-
	EWMA	TOFS-EN	TOFS-EA	TOFS-E	TOFS-EM	TOFS-EL
	MEAN	-	-	-	TOFS-M	-
	LMS	-	-	-	-	TOFS-L

Table 3.2: Overview of comparative assessment models for optimal-time data forwarding strategies.

3.6.4 Parameter Configuration

The experiments performed in this chapter require different parameter settings for the introduced prediction-based forwarding strategy (DPD and QEPD) and the time-optimised forwarding with TOFS and HTOFS. In Table 3.4 an overview shows the setting of these parameters for DS1. Most influencing parameters are the values of θ, highlighting the upper error

		ED				
		NAIVE	**ARIMA**	**EWMA**	**MEAN**	**LMS**
SAN	**NAIVE**	HTOFS-N	HTOFS-NA	HTOFS-NE	HTOFS-NM	HTOFS-NL
	ARIMA	-	HTOFS-A	-	-	-
	EWMA	HTOFS-EN	HTOFS-EA	HTOFS-E	HTOFS-EM	HTOFS-EL
	MEAN	-	-	-	HTOFS-M	-
	LMS	-	-	-	-	HTOFS-L

Table 3.3: Overview of comparative assessment models for hybrid optimal-time data forwarding strategies.

bound for forwarding the measurements, set to be $\theta = \{0.001, 0.003, 0.008, 0.01, 0.02, 0.03\}$. Deploying the time-optimised forwarding strategies inside the SANs, the delay tolerance level β has to be set to $\beta = \{0.05, 0.08, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$. The functions implemented for $f(\mathbf{x})$ and $g(\mathbf{u})$ require the setting of parameters for the EWMA policy using the weighting factor $\alpha = 0.7$ and the ARIMA to be $(1, 1, 4)$. The Sliding Window (SLW) $\mathcal{W}$ size the aggregation analytics is performed on is set to $M = 300$ and for the precitive analytics task is a learning rate η for SGD used with $\eta = 0.0001$.

	θ	β	α	**ARIMA (p,q,g)**	**SLW $\mathcal{W}$**	**SGD η**
Parameters	[0.001, 0.003, 0.008, 0.01, 0.02, 0.03]	[0.05, 0.08, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7]	[0.7,1]	(1,1,4)	300	0.0001

Table 3.4: Overview of the parameter setting for the data forwarding strategies.

3.6.5 Performed Analytical Tasks in Edge Computing

The reconstruction of the real data is of utter importance as otherwise the data can not be used for analysis or decision making in central locations and qualitative results are impossible to generate. This reconstruction discrepancy is focused on most of the related work mentioned in the previous sections. This thesis departs from this work by highlighting the importance of the quality of the performed analytics using reconstructed data. As EDs become more powerful and able to perform analytics inside the network, close towards the data collection, the impact of using reconstruction and data forwarding strategies towards analytical tasks is of high interest. In this section, analytical tasks performed inside a WSN at the ED will be presented. In particular, aggregation analytics and predictive analytics. As highlighted in Section 2.3.3.1 time series analysis is mostly performed on a Sliding Window, which is specified by a fixed size $M > 0$ and appends new context vectors by discarding older ones based on their appearance. At each time t, a Sliding Window (SLW) $\mathcal{W}$ is a sequence of all context vectors observed from $t - M$ to t, i.e., $\mathcal{W}_t = (\mathbf{x}_{t-M}, \mathbf{x}_{t-M+1}, \ldots, \mathbf{x}_t)$.

3.6.5.1 Aggregation Analytics

The aggregation functions used for the analytics tasks of a window $\mathcal{W}$ can be classified based on [221] into three categories: distributive, algebraic, and holistic. Let $\mathcal{W}$, $\mathcal{W}_1$, and $\mathcal{W}_2$ be windows. An aggregation analytics function $h : \mathcal{W} \to \mathbb{R}^d$ is distributive if $h(\mathcal{W}_1 \cup \mathcal{W}_2)$ can be computed from $h(\mathcal{W}_1)$ and $h(\mathcal{W}_2)$ for all $\mathcal{W}_1$, $\mathcal{W}_2$. An aggregation analytics function h is algebraic if a synopsis function σ for all $\mathcal{W}$, $\mathcal{W}_1$, and $\mathcal{W}_2$ exists so that: (1) $h(\mathcal{W})$ can be computed from $\sigma(\mathcal{W})$; (2) $\sigma(\mathcal{W})$ can be stored in constant memory; and (3) $\sigma(\mathcal{W}_1 \cup \mathcal{W}_2)$ can be computed from $\sigma(\mathcal{W}_1)$ and $\sigma(\mathcal{W}_2)$. An aggregation analytics function h is holistic if it is not algebraic. The standard aggregates `MAX` and `MIN` are distributive, `AVG` is algebraic, since it can be computed from a synopsis containing `SUM` and `COUNT`. The aggregates `QUANTILE` and `MEDIAN` are representatives of holistic functions. The `AVG` and `MAX` analytics functions can be defined as: $h^{avg}(\mathcal{W}) = \frac{1}{M}\sum_{j=t-M}^{t} \mathbf{x}_j$ and $h^{max}(\mathcal{W}) = [\max\{x_{1j}\}, \ldots, \max\{x_{dj}\}]_{j=t-M}^{t}$, respectively.

These aggregation analytics functions leading to build-in continuous analytics queries formulated inside the ED.

Example 1: The aggregation analytics query 'every minute find the average temperature and the maximum humidity over context streams 'temperature' and 'humidity' collected during the past hour', is formulated in Continuous Query Language [222, 223] involving `AVG` and `MAX` operators over a sliding window $\mathcal{W}, M = 60min$ as follows:

```
SELECT AVG(temperature), MAX(humidity)
FROM Context Streams [RANGE 60 MINUTES SLIDE 1 MINUTE]
```

Dynamic aggregates like `SUM`, `MIN` and `AVG` require a constant time $O(1)$ computation per value. However, more advanced aggregation analytics functions like outliers detection or concept drift detection in a sliding window $\mathcal{W}$ require multiple scanning of the $\mathcal{W}$. Besides the classical queries issued towards the ED, aggregation analytics functions can also be combined to infer certain events that might trigger decision making towards sending or acting inside the ED.

Example 2: The evaluation of a localised event stream, processing the past ten minutes as activation of a rule associated with `AVG` and `MAX` aggregation analytics functions over 'temperature' and 'wind-speed' using a sliding window from two corresponding SANs:

```
EVENT := IF AVG(temperature) ≥ 90 AND MAX(wind-speed)
∈ [10,20] WITHIN 10 minutes THEN ACTION is 'warning'
```

The aggregation analytics function h is running on each ED i for each sliding window $\mathcal{W}$ containing M received and/or reconstructed context vectors from the SAN $k \in \mathcal{K}_i$ depending on whether the data is sent from SAN to ED or not. Recall, in prediction-based data forwarding data is send only when the predicted value of the deployed function $f(\mathbf{x})$ differs above a threshold θ otherwise it is reconstructed at the ED with the function $g(\mathbf{u}) = \tilde{\mathbf{x}}$. In time-optimised forwarding, the prediction errors are accumulated and only when the optimal time of forwarding based on the rewarding function in Equation (3.19) is reached the actual measurements are sent. In the meantime the ED is reconstruction the values using as well $g(\mathbf{u}) = \tilde{\mathbf{x}}$.

Using an aggregation analytics function inside the ED on either prediction-based or time-optimised data forwarding, the importance is identifying the discrepancy between the reconstructed data inside the ED using the aggregation analytics function and the actual sensed data. Consider an ED i and its SAN $k \in \mathcal{K}_i$. The aggregation analytics difference δ_i between the analytics result on ED i derived from aggregation function h over the window $\mathcal{W}$ in the ED i and the actual analytics result derived from h over the window $\mathcal{W}^*$, which contains only the actual context vectors from SAN k to ED i (ground truth) is:

$$\delta_i = \|h(\mathcal{W}) - h(\mathcal{W}^*)\|. \tag{3.33}$$

The aggregation analytics difference δ_i denotes how much the aggregation results over the window $\mathcal{W}$ on ED i with context vectors $\mathbf{u}$ differ from the aggregation results over the window $\mathcal{W}^*$ with context vectors $\mathbf{x}$. Should SAN k have sent all context vectors to ED i, then $\delta_i = 0, \forall i \in \mathcal{K}_i$. Since the proposed methodologies allow SAN k to decide on sensing context vectors w.r.t. θ and ED i being able to reconstruct undelivered context vectors, then $\delta_i \geq 0$. The aim is to identify how the quality of analytics change in light of communication.

3.6.5.2 Predictive Analytics

Besides the aggregation analytics functions, further performing predictive analytics on ED can lead to knowledge and real-time decision making. One of the most essential models for predictive analytics is the multivariate linear regression approximation [224]. Highlighting the importance of using predictive analytics inside the ED, two examples will be introduced later. The first example focuses on using data from one SAN, whereas the second introduces the ability to combine multiple SANs for analysis.

Example 4: Consider SAN k with context vector $\mathbf{x} = [x_1, x_2, x_3]$ referring to the contextual parameters humidity, wind speed and temperature. The corresponding ED i is responsible for learning the statistical dependency $\mathbf{w}_i$ between temperature (dependent variable $y^{out} = x_3$) with humidity and wind speed (independent variables $\mathbf{x}^{in} = [x_1, x_2]$).

Example 5: Consider the SAN k and SAN ℓ with $k, \ell \in \mathcal{K}_i$ sensing context vectors $\mathbf{x}_k = [x_{k1}, x_{k2}]$ and $\mathbf{x}_\ell = [x_{\ell 1}, x_{\ell 2}, x_{\ell 3}]$, respectively. The ED i is responsible, e.g., for learning the linear dependency $y^{out} = x_{k2}$ and $\mathbf{x}^{in} = [x_{\ell 1}, x_{\ell 2}]$ between the contextual parameters from those SANs in $\mathcal{K}_i$.

These two illustrations show the case when all data is send to the ED i. However, using the introduced selective data forwarding methodologies of this section, it can be seen that in each SAN k a vector $\mathbf{x}_t = [\mathbf{x}_t^{in}, y_t^{out}] \in \mathbb{R}^d$ is produced, represented as input-output pairs $(\mathbf{x}_1^{in}, y_1^{out}), \dots, (\mathbf{x}_T^{in}, y_T^{out}) \in \mathbb{R}^{d-1} \times \mathbb{R}$, $T > 0$. At the ED i these pairs are either received if the condition of forwarding is fulfilled, or reconstructed by using one of the methods manifest earlier. Overall, ED i has the context vector of $\mathbf{u}_t = [\mathbf{u}_t^{in}, z_t^{out}] \in \mathbb{R}^d$, based on the Cases in Equation (3.1).

In Section 2.3 machine learning has been introduced with the aim to minimise a objective function $\mathcal{J}(\mathbf{w})$ as in Equation (2.3). Using the linear regression analytics function inside the ED i is leading to the task of estimating the coefficient vector $\mathbf{w}_i \in \mathbb{R}^d$. This vector interprets the dependencies between $\mathbf{u}_t^{in}$ and z_t^{out}. The approximation of the coefficient vector $\mathbf{w}_i$ inside the ED i using the Equation (2.3) is leading to the following objective function:

$$\mathcal{J}(\mathbf{w}_i) = \min_{\mathbf{w}_i \in \mathbb{R}^d} \frac{1}{T} \sum_{t=1}^{T} \left(z_t^{out} - (\mathbf{u}_t^{in})^\top \mathbf{w}_i \right)^2 + \lambda \|\mathbf{w}_i\|^2 \tag{3.34}$$

This objective function has to be locally computed inside the ED by a lightweight computational algorithm. Introduced in Section 2.3.3.2, SGD is the most common method to minimise the objective function and finding the coefficient vector $\mathbf{w}_i$ in an iterative and low computational complex manner. Not only is the introduced reconstruction difference and aggregation difference of interest when using intelligent decision forwarding mechanisms inside the SAN and reconstruction inside the ED. Also, the regression analytics difference between the real coefficient vector $\mathbf{w}_i^*$ if all measurements are transferred to the ED and the approximated $\mathbf{w}_i$ using the methods summarised in Section 3.6.3. The regression analytics difference γ_i is defined as the absolute difference of ϵ_i derived from the approximated regression line (coefficient vector $\mathbf{w}_i$) and ϵ_i^* derived from the actual regression line using the actual coefficient vector $\mathbf{w}_i^*$ trained by the actual SAN k's context vectors. ϵ_i and ϵ_i^* can be calculated with one of the accuracy metrics introduced above in Section 3.6.2 (mostly RMSE). Using the RMSE for illustration, it is possible to define $\epsilon_i = \left(\frac{1}{M} \sum_{m=1}^{M} (y_m^{out} - \hat{z}_m^{out})^2 \right)^{1/2}$ and $\epsilon_i^* = \left(\frac{1}{M} \sum_{m=1}^{M} (y_m^{out} - \hat{y}_m^{out})^2 \right)^{1/2}$ over a SLW $\mathcal{W}$ of size M, so that regression analytics difference γ_i is defined as:

$$\gamma_i = |\epsilon_i - \epsilon_i^*|. \tag{3.35}$$

3.7 Performance Assessment

3.7.1 Prediction-Based Performance

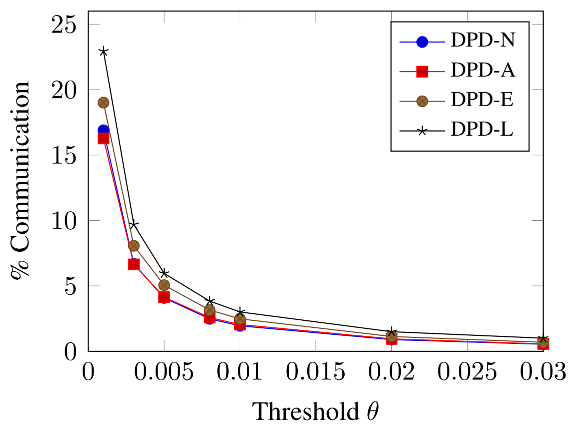
Using the mentioned DS1 and DS2 over the given models in Table 3.1 with the relevant parameter settings in Table 3.4, the latter section shows the results obtained and highlights key findings of the data forwarding strategies DPD and the improved QEPD. The following section is divided into first comparing the models' sensitivity and reconstruction ability, secondly investigating the quality of performing aggregation analytics on the reconstructed data analytical tasks presented in Section 3.6.5, and last the predictive analytics quality on the selective data forwarding methodologies. Please note that the results of DS2 can be found in Appendix A.

3.7.1.1 Sensitivity Analysis and Reconstruction Assessment

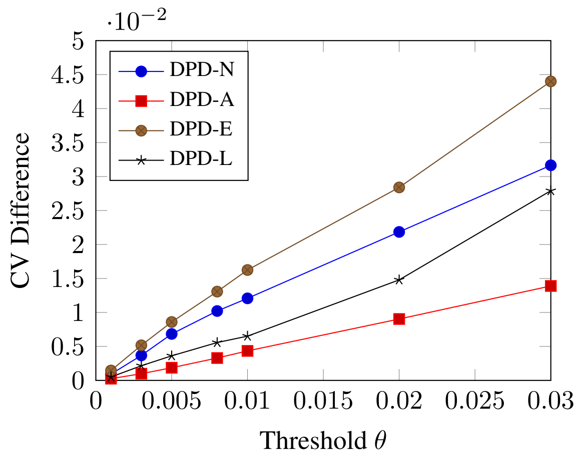
The ability to reconstruct the data stream at the ED when using the different DPD and QEPD models as data forwarding strategies are highly influenced by the application defined error-bound θ. Increasing $\theta \to 1$ results in less communication between ED and SAN, as the SAN only sends outliers and novel data.

This dependency over the DPD models, in which SAN and ED deploy the same model for prediction-based forwarding and reconstruction is highlighted in Figure 3.4 (a). The figure shows that with increasing the θ-value the remaining communication decreases between ED and SAN, measured with the metric of the percentage using $c(T)$ of Equation (3.32). The opposite effect is represented in Figure 3.4 (b), which shows that with increasing the θ-value, the CV defined in Equation (3.25) divergence towards the real data rises, indicating an increase of dissimilarity between μ and σ. As θ positively influences the sensitivity and communication, the trade-off has been highlighted in Figure 3.4 (c) and (d). These two figures show that increasing communication results in higher similarity and less information loss from the reconstructed stream to the real data stream; each point represents a different θ-value.

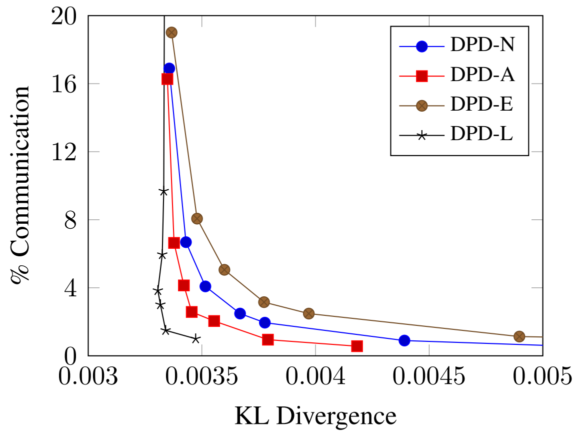
From these four figures in Figure 3.4, it is shown that communication, sensitivity, similarity, and θ-value are highly connected. It further demonstrates the different models for DPD compared to their efficiency. At this point, it should be noted that throughout all experiments, the proposed MEAN methodology resulted in poor results far away from the others. Therefore the following figures only compare the other four reconstruction policies. The model of DPD-L using LMS inside SAN and ED shows the best efficiency in the trade-off between information loss and communication, as presented in Figure 3.4 (c). Closely followed by the DPD-A using ARIMA as a prediction and reconstruction model. The worst



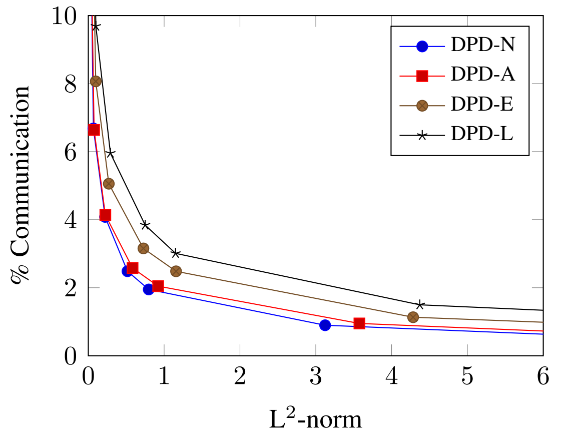
(a) Influence of θ with respect to the remaining communication over all DPD models.



(b) Influence of θ on the metric CV over all DPD models.



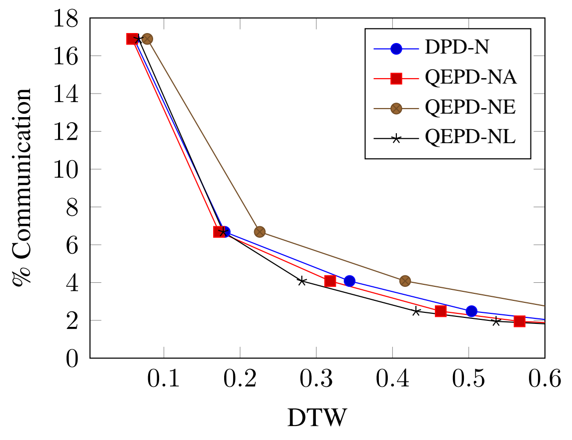
(c) Trade-off between KL divergence and communication over all DPD methods.



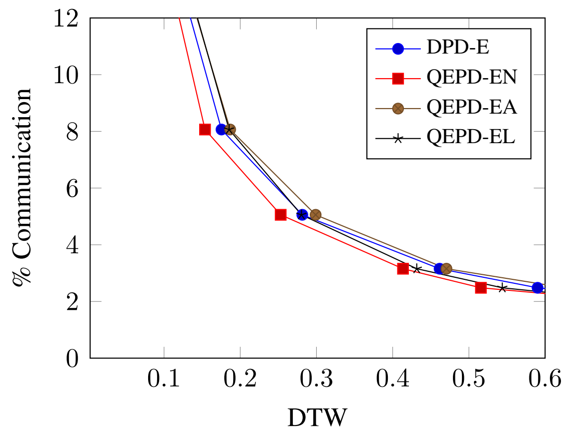
(d) Trade-off using the L^2-norm metric and communication over all DPD methods.

Figure 3.4: Evaluation of DPD methodology and the influence of θ and trade-off for evaluating the efficiency with information loss and similarity towards reduced communication.

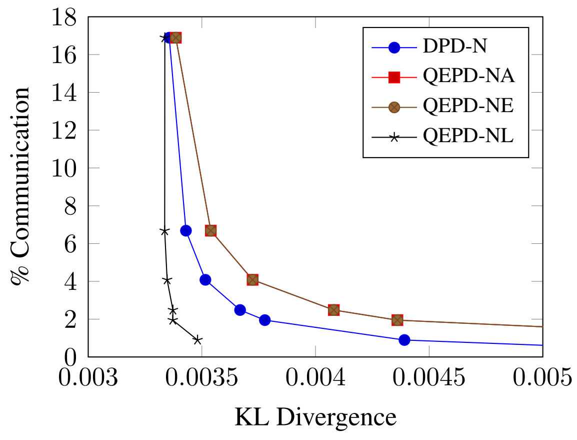
trade-off results in the DPD-E, which uses a simplistic version of ARIMA. In the middle is the NAIVE model, which is using the last seen measurement as prediction and reconstruction. Interestingly, when looking into the similarity metrics such as L^2-norm or DTW, and not the information loss metric CV and KL, it is derivable in Figure 3.4 (d) that the order of efficiency is nearly reverted. This effect occurs as the similarity metrics focus on each point of the time series and their distance between each other, while the metric of KL is investigating the Probability Density Function (PDF) over the data. Overall it is possible to observe from the figures that using one of the DPD methods decreases the communication to just 10% while being able to reconstruct the data stream to be nearly the same as the real data. This is true for both information loss and similarity. None of the models is showing any comparable behaviour above a 10% communication value. Therefore the range of observation for comparing the proposed models in their efficiency is set to be between 1% and 10% for $c(T)$ the remaining communication between SANs and EDs.



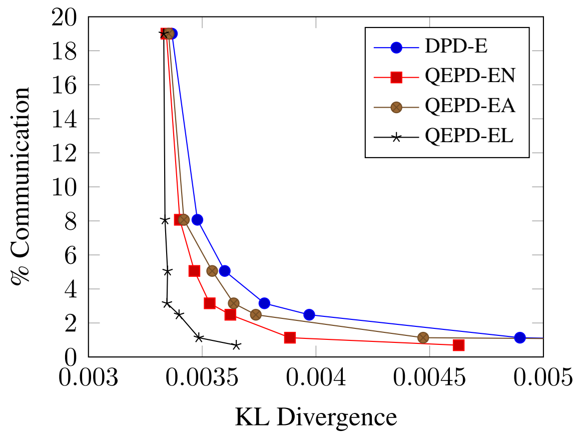
(a) Comparison of the trade-off using the DTW metric and remaining communication using the NAIVE function inside the SAN.



(b) Comparison of the trade-off using the DTW metric and remaining communication using the EWMA function inside the SAN.



(c) Comparison of the trade-off using the KL divergence metric and remaining communication using the NAIVE function inside the SAN.



(d) Comparison of the trade-off using the KL divergence metric and remaining communication using the EWMA function inside the SAN.

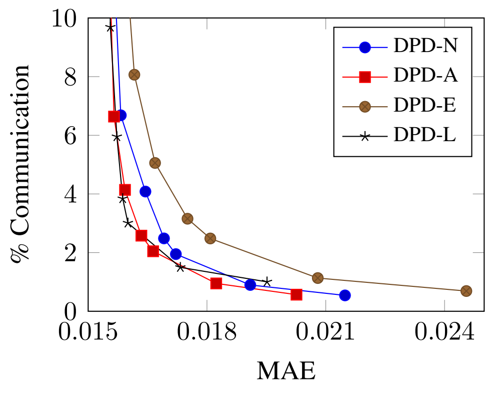
Figure 3.5: Evaluation of the trade-off between communication and sensitivity metrics KL/DTW using the QEPD methods in compare to the DPD equivalent method.

Continuing towards the evaluation of our proposed quality-efficient prediction based methodology, noted as QEPD. Figure 3.5 shows the results of QEPD with a comparison to the DPD models. It is possible to see that using independent models inside the SAN and ED is results in a much better efficiency towards sensitivity and communication trade-off than identical models. The models used inside the SAN is reduced to the NAIVE and the EWMA models as the aim is to limit the computational complexity towards a minimum for energy usage, and this is only fulfilled with NAIVE and EWMA implementations. From the analysis in Figure 3.5 above, it was possible to identify that the NAIVE method in DPD is the most efficient method for generating similar time series. However, illustrated in Figure 3.5 (a), it is possible to see that using the proposed QEPD-NL compared to the DPD-N, an improvement in the trade-off between communication and reconstruction similarity has been made, represented by the DTW metric. Figure 3.5 (b) shows the results for using EWMA as a decision method in the SAN. These results identify that using QEPD improves the trade-off as well, not with LMS inside the ED but with the NAIVE approach. From both figures, it is possible to conclude that using the QEPD reduces communication by around 1% with the same similarity of the datastreams. Departing from the similarity of the datastreams towards its information loss represented by the KL divergence metric. Figure 3.5 (c) shows the effects of QEPD and indicates that using the NAIVE prediction-based forwarding strategy inside the SAN and the LMS for reconstruction in the ED generates a communication improvement of up to 7% with the same information loss as using DPD-N. Similar behaviour is seen in Figure 3.5 (d) for the EWMA implementation. For EWMA, all models improve the information loss, whereas the LMS is the most efficient in terms of communication reduction with the same gain of information. Generally, achieving a reduction of communication with the same accuracy level using the QEPD strategy can validate that enabling the usage of a light prediction-based algorithms inside the SAN, while using the full capacity of the ED for more advanced reconstruction results, leads to good results and higher reconstruction performance. Even advanced methods are possible to implement inside the ED, involving machine learning models that could combine multiple SANs or further data analysis functions to reconstruct missing data with even less communication required between SAN and ED.

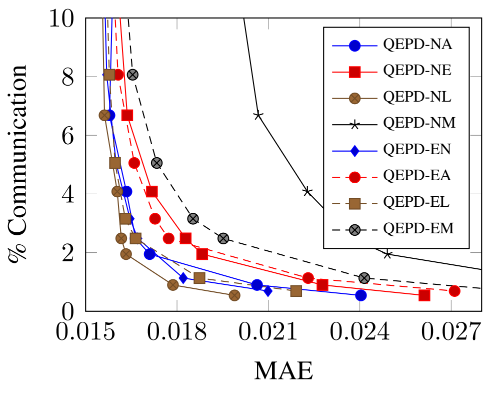
3.7.1.2 Aggregation Analytics Assessment

Besides the reconstruction assessment and the evaluation of the sensitivity, the main focus is on how the reconstruction is influencing the ability to make high-quality analysis inside the ED. Therefore, all three categories of aggregation function are investigated: algebraic, distributive and holistic. For this `AVG`, `MAX` and `Median` have been use respectively. In Figure 3.6, an overview is illustrated of the assessment with the accuracy metrics MAE, SMAPE, and RMSE for generating the aggregation discrepancy in Equation (3.33). The figure shows the three mentioned aggregation categories for the different models of DPD and QEDP over DS1. The assessment for DS2 can be found in Appendix A, along with the reconstruction results. Further should be noted that the abscissa (x-coordinate) values have been shifted for better illustration of the difference between the comparable models as the graphs perform an asymptotic behaviour towards a certain value.

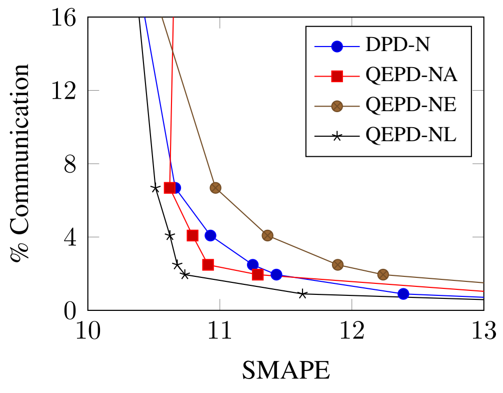
Generally, the highlighted results for reconstruction evaluated in Section 3.7.1.1 can be found in the aggregation analytics results as well. Examine the results of the `AVG` aggregation function in detail, Figure 3.6 (a) shows all DPD models are compared with their ability of trading communication and accuracy using MAE as discrepancy metric for δ. The order of the best



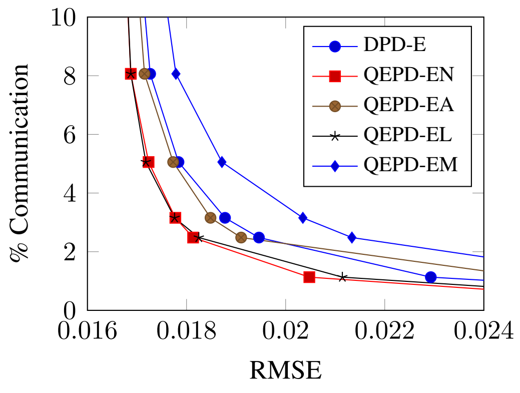
(a) Trade-off using the MAE and remaining communication over the aggregation function `AVG` for all DPD methods.



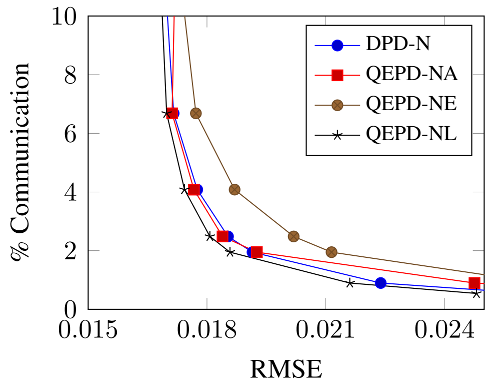
(b) Trade-off using the MAE metric and remaining communication over the aggregation function `AVG` for all QEPD methods.



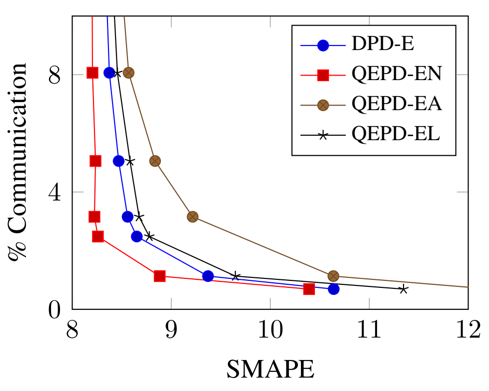
(c) Trade-off using the SMAPE and remaining communication using the NAIVE function inside the SAN over the aggregation function `AVG`.



(d) Trade-off using the RMSE metric and remaining communication using the EWMA function inside the SAN over the aggregation function `AVG`.



(e) Trade-off using the RMSE and remaining communication using the NAIVE function inside the SAN over the aggregation function `MEDIAN`.



(f) Trade-off using the SMAPE metric and remaining communication using the EWMA function inside the SAN over the aggregation function `MAX`.

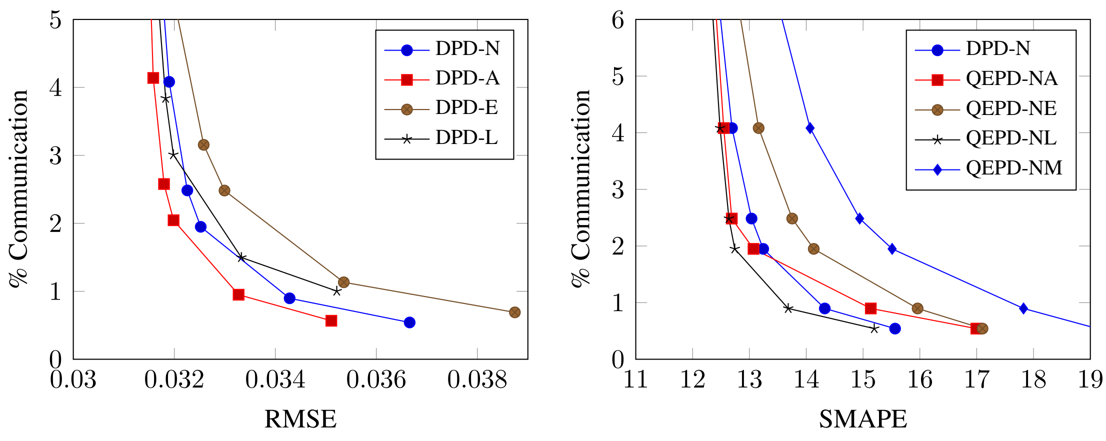
Figure 3.6: Evaluation of the trade-off between communication and accuracy metrics (RMSE, SMAPE, MAE) over the aggregation functions for the QEPD and DPD methods.

trade-off model is similar to the results of Figure 3.4 (c) of the reconstruction and sensitivity analysis in Section 3.7.1.1. The figure indicates DPD-L and DPD-A as most accurate and DPD-E as the least accurate for the aggregation analytic function. However, when performing QEPD, the assessment for aggregation shows significant improvement inefficiency. These results are highlighted in Figure 3.6 (b), indicating that QEPD-NL is the most accurate while saving the most energy by limiting communication. Figure 3.6 (c) inspects using the SMAPE metric on how DPD-N is improved by the different QEPD model implementations. This figure further highlights that using QEPD-NE results in no improvement towards the DPD-N. In contrast, all other QEPD methods perform better with a significant decrease of communication and the same accuracy using the NAIVE method in the SAN. In Figure 3.6 (d), the results for deploying the EWMA inside the SAN using the QEPD models is illustrated. For the QEPD using NAIVE inside the SAN, only the EWMA function inside the ED results in less accurate aggregations than the DPD methods. Contrary to the EWMA implementation inside SANs using QEPD, the LMS and ARIMA perform less accurate than the corresponding DPD method. However, as already seen in the reconstruction, using the NAIVE model in the ED with the EWMA in the SAN is greatly improving the DPD-E deployment. Showing the similarity towards the other two aggregation function categories, Figure 3.6 (e) is illustrating the assessment for the holistic function `MEDIAN` on the RMSE as an accuracy metric. Figure 3.6 (f) shows the results obtained from the distributive function `MAX` with the metric SMAPE. Both figures highlight that QEPD is improving the accuracy of aggregation discrepancy in trade-off with communication reduction considerably. Best results are realised by QEPD-NL.

3.7.1.3 Predictive Analytics Assessment

Similar to the conducted results of aggregation analytics and reconstruction, the predictive analytics performs with the same behaviour for DPD and QEPD. Using the real data to perform the prediction on temperature using humidity as an input variable results in a MAE of 0.012944163, a RMSE of 0.028857876, and a SMAPE of 6%.

Figure 3.7 represents the use of the DPD and QEPD methods showing the difference of Equation (3.35) for the metric SMAPE and RMSE. The ARIMA model performs the best over all DPD models, which is illustrated in Figure 3.7 (a). Comparing the NAIVE methods of QEPD and DPD, the SMAPE shows in Figure 3.7 (b) the impact using QEPD and its reduction of communication overhead while improving the accuracy. The best model is the QEPD-NL for predictive analytics. QEPD only improved the trade-off for LMS and ARIMA, whereas the EWMA and MEAN decrease the trade-off compared to the DPD method.



(a) Trade-off using the RMSE and remaining communication over the predictive function for all DPD methods.

(b) Trade-off using the MAE and remaining communication over the predictive function for all QEPD methods with NAIVE.

Figure 3.7: Evaluation performance of DPD and QEPD methods using the metrics RMSE and SMAPE for trade-off in predictive analytics over DS1.

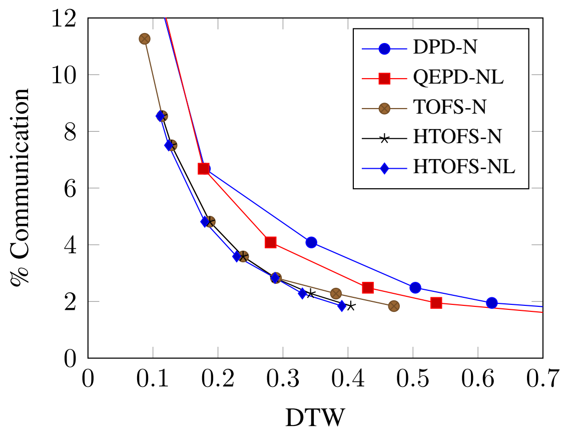
3.7.2 Time-Optimised Performance

The previous section showed the results of the experiments performed using prediction-based data forwarding. The following section presents the results of the introduced methodology of time-optimised data forwarding as introduced in Section 3.5. Setup for the evaluation is given by Table 3.2 and Table 3.3, as well as the parameters presented in Table 3.4. The following figures showing the improvements towards DPD and QEPD using TOFS and the hybrid version HTOFS. Similar to the assessment of the prediction-based data forwarding strategies, this section is divided into evaluating the reconstruction ability, the quality of using aggregation analytics, and finally, the predictive analytics performance. The results of DS2 can be found in Appendix A.

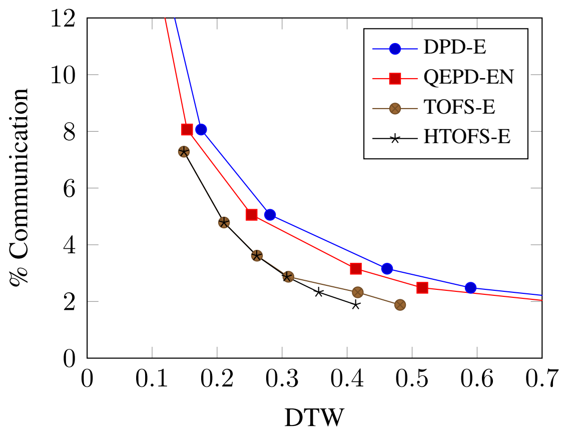
3.7.2.1 Sensitivity Analysis and Reconstruction Assessment

The ability to reconstruct the real datastream is highly dependent on the chosen β of TOFS and HTOFS from Equation (3.19). Compared to the DPD and QEPD in which θ is primarily influencing the accuracy and communication, in the time-optimised data forwarding β and not θ is essential. Increasing the value of $\beta \rightarrow 1$ decreases the communication and, therefore the reconstruction ability of the ED. The value of β is indicating the delay tolerance of the SAN to forward the data based on the calculated prediction error. The similarity between the real data and reconstructed is measured during the experiments using DTW. The information loss and distribution reconstruction ability is measured by the KL divergence.

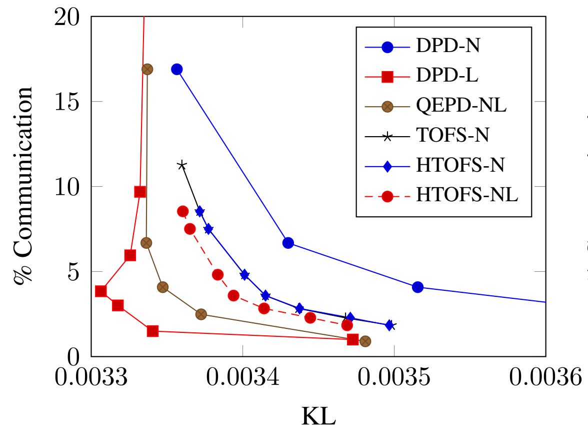
In Figure 3.8, these metrics are used to show the trade-off between communication reduction



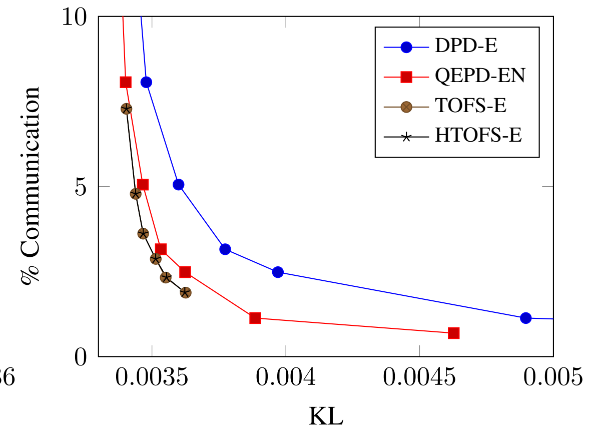
(a) Comparison of the trade-off using the DTW metric and remaining communication using the NAIVE function inside the SAN.



(b) Comparison of the trade-off using the DTW metric and remaining communication using the EWMA function inside the SAN.



(c) Comparison of the trade-off using the KL divergence metric and remaining communication using the NAIVE function inside the SAN.



(d) Comparison of the trade-off using the KL divergence metric and remaining communication using the EWMA function inside the SAN.

Figure 3.8: Evaluation of the trade-off between communication and sensitivity metrics KL/DTW using the TOFS/HTOFS methods in compare to the DPD/QEPD equivalent methods.

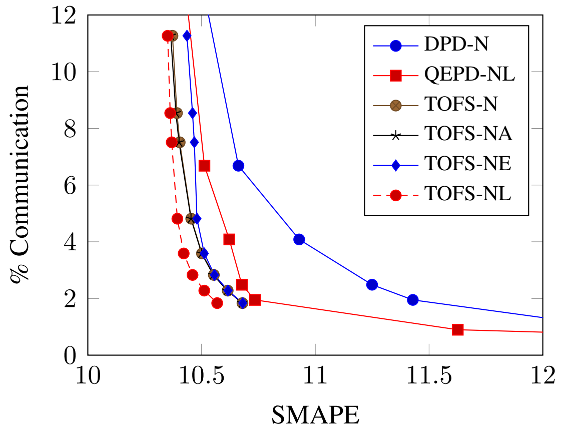
and similarity for deploying the EWMA and the NAIVE method inside the SAN. Figure 3.8 (a) indicates that using the DPD with the NAIVE method inside the SAN and ED generates a lot more communication than using the TOFS with a NAIVE prediction method in the SAN and ED. The same holds true when using the improved QEPD-NL method and the corresponding TOFS-NL. TOFS is improving the similarity considerably towards the real data with the same or less communication. As an example, with 4% of communication, using the TOFS strategy, the same similarity as QEPD-NL and DPD-N with over 6% is generated. Similar results are applicable in Figure 3.8 (b) for using EWMA inside the SAN. The hybrid version of TOFS marked with HTOFS inside Figure 3.8 shows an improvement towards the similarity in comparison to TOFS when reaching a communication below 3%. Otherwise, it is behaving analogue towards the TOFS for both methods, NAIVE and EWMA. Additionally analysed from the assessment shown in the figures, using different methods inside the ED, such as the QEPD deployed model variations, does not improve the trade-off using the TOFS or HTOFS methodology. It can be stated that using NAIVE in ED and SAN results in the same similarity as NAIVE and LMS using the time-optimised forwarding strategy. This highly differs from the assessment of DPD towards implementing QEPD.

Figure 3.8 (c) considers the assessment of the information loss using the KL divergence metric. Differing to the DTW metric, the improvement using TOFS only applies towards the DPD-N method, whereas the DPD-L and QEPD-NL show much less information loss than any TOFS and HTOFS implementations. Moreover, it can be seen in this figure that using the deployment of different models in SAN and ED is improving the trade-off, as shown for TOFS-NL in comparison to TOFS-N. In this figure the method of DPD-L is generating a jump below a 5% communication resulting in lower values for KL divergence. This jump might occur through generalisation and better fitting with less data and outlier reduction by identifying only relevant data with higher thresholds. It appears that only LMS is sensitive towards this. In Figure 3.8 (d), the EWMA deployment in SAN is illustrated, and contrary towards the NAIVE method, here a gain using TOFS is presented with improving the trade-off by 5% in compare to the DPD strategy.

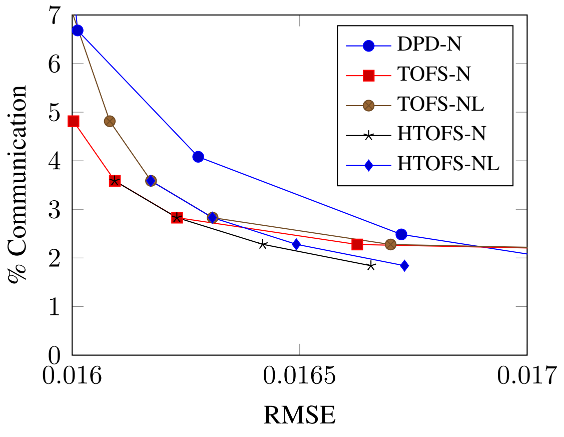
3.7.2.2 Aggregation Analytics Assessment

Investigating the behaviour of the aggregation discrepancy in Equation (3.33) for the time-optimised data forwarding strategies (TOFS and HTOFS), in Figure 3.9 the results for the assessment are illustrated.

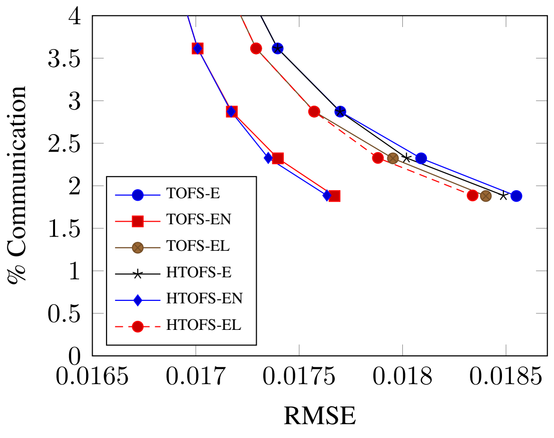
Figure 3.9 (a) shows the impact of TOFS for the algebraic aggregation analytics function `AVG`. All methods performed on a time-optimised data forwarding strategy are greatly improving the efficiency of the previous prediction-based methods. As already shown in Figure 3.6, the QEPD-NL method improves the DPD-N variant by up to 3%. Using the TOFS implementation of TOFS-NL, the QEPD-NL can be further improved by having 5% less communication with the same accuracy towards aggregation analytics as highlighted in Figure 3.9 (a). Figure 3.9 (b) illustrates the results for the distributive aggregation function `MAX`. In this figure, the TOFS is enhancing the efficiency by being qualitative equivalent towards DPD-N. Further on this figure, the effect of using HTOFS is illustrated. Similar to the reconstruction similarity and information loss assessed in Section 3.7.2.3, the impact of HTOFS only applies below 3% of communication. Below this communication level, the accuracy-efficiency trade-off tends towards better results towards the HOFTS. This effect is also seen in Figure 3.9 (c) for the holistic aggregation analytics function `MEDIAN`. In Figure 3.9 (d), the SMAPE of the `AVG` function using NAIVE is presented. In this figure, it is shown that in some cases, the HTOFS does not improve the TOFS, even in lower communication ranges.



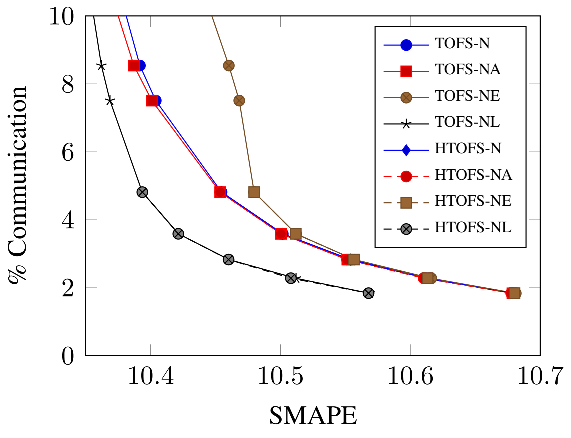
(a) Trade-off using the SMAPE and remaining communication over the aggregation function `AVG`.



(b) Trade-off using the RMSE and remaining communication over the aggregation function `MAX`.



(c) Trade-off using the RMSE and remaining communication over the aggregation function `MEDIAN`.



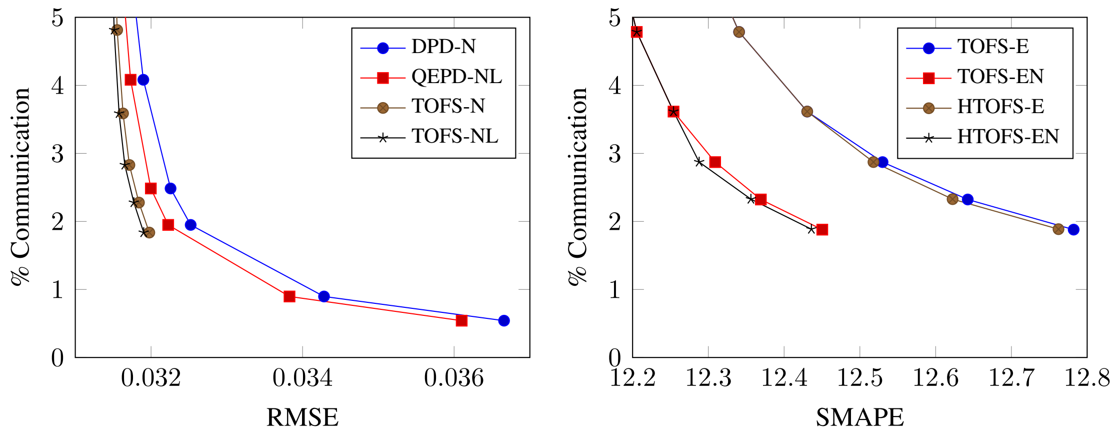
(d) Trade-off using the SMAPE and remaining communication over the aggregation function `AVG`.

Figure 3.9: Evaluation of the trade-off between communication and accuracy metrics (RMSE, SMAPE, MAE) over the aggregation functions using the TOFS/HTOFS methods in compare to the DPD/QEPD equivalent methods.

Overall, for TOFS deploying the NAIVE method inside the SAN with LMS inside the ED for reconstruction generates the best aggregation analytics accuracy while being efficient in communication overhead. TOFS is improving this trade-off considerably across all aggregation functions and all accuracy metrics. HTOFS only improve the trade-off below 3% of communication and only for algebraic and distributive aggregation functions.

3.7.2.3 Predictive Analytics Assessment

For the predictive analytics task, the results using TOFS and HTOFS over the predictive analytics discrepancy δ of Equation (3.35) are similar to the conducted results for the aggregation analytics tasks. The TOFS is generally improving the DPD and QEPD strategy greatly. The HTOFS in comparison only improves with a lower communication percentage than the TOFS. Overall the TOFS-NL performs best as model implementation inside SAN and ED. These results are illustrated in Figure 3.10 with Figure 3.10 (a) showing the results for comparing the DPD-N, QEPD-NL, and the corresponding TOFS implementations against each other. Figure 3.10 (b) shows the influence of HTOFS over the implemented EWMA function in SAN with the same deployment inside the ED and the NAIVE inside the ED.



(a) Trade-off using the RMSE and remaining communication over the predictive function for all methods with NAIVE function inside the SAN.

(b) Trade-off using the SMAPE metric and remaining communication over the predictive function for all methods with EWMA function inside the SAN.

Figure 3.10: Evaluation of the trade-off between communication and accuracy metrics (RMSE and SMAPE) over predictive functions using the TOFS/HTOFS methods in compare to the DPD/QEPD equivalent methods.

3.8 Chapter Summary

In this section, an extensive evaluation of current prediction-based data forwarding strategies represented by the Dual Prediction Design (DPD) models has been carried out. Embarrassing the computational abilities of SANs and EDs, the deployment of different models inside them has been proposed by the Quality-Efficient Prediction Design (QEPD) strategy. It has been shown, that this considerably improves the reconstruction and analytics quality. Departing from the instantaneous decision making (IDM) on a threshold or error-bound at a certain time t, this chapter introduced a time-optimised data forwarding methodology (including Time-Optimised Forwarding Strategy (TOFS) and Hybrid-Time-Optimised Forwarding Strategy (HTOFS)) to overcome the limitations of the prediction-based by using Optimal Stopping Theory (OST). The history of decisions is incorporated whenever the SAN decides on forwarding the data. From the extensive evaluation, it has been seen that the performance of

TOFS improves greatly the quality of reconstruction, aggregation and prediction analytics inside the ED with comparison to the IDM strategies. However, the HTOFS can show no significant improvement to the TOFS methodology. The shown experiments have only be performed on a special type of contextual data (weather data) and over a limited time horizon. The results presented in the performance assessment section sometimes show only small improvements or differences to the current implementation of DPD. Nonetheless, using other continuous data streams and contextual data over an infinity time, the improvements made in this chapter with using TOFS or QEPD can greatly increase towards the current research of DPD.

To sum up, using the intelligence of SANs by implementing a time-optimised data forwarding decision strategy and an advanced reconstruction function inside the ED, enables to reduce the data transmission to a minimum, while maintaining high-quality aggregation and predictive analytics results.

Chapter 4

Latency-Efficient Edge-Centric Analytics

4.1 Chapter Overview

Chapter 3 introduced quality-efficient data forwarding from sensing devices towards Edge Devices or sink nodes. Methodologies of prediction-based and time-optimised data forwarding have been proposed to considerably improve communication reduction through selective data forwarding. The influence of the proposed methods towards qualitative aggregation and prediction analytical tasks at the Edge Device and its impact was highlighted. In the following chapter, the research is departing from data forwarding strategies to qualitative and efficient edge-centric analytics. The coming chapter aims to improve latency by deploying machine learning and analytics at Edge Devices. Latency is defined by the computational time an algorithm needs to run or retrained, the network transfer of the data to the central collection point, and the transfer of the responding output from the central location back to the device. With Edge Computing and analytics, the latency can be reduced by eliminating the network transfer towards the central location and backwards to the device, which has been proven in other studies [225, 180, 226]. Therefore, with the introduced training and inference of analytics at the edge of the network, the latency can be by definition reduced in comparison to a centralised implementation and is consequently seen as a latency-efficient method. Chapter 2 discusses concepts of online learning and distributed machine learning and the usage of machine learning in Edge Devices. In this chapter, the importance of energy-efficient implementation of analytics and algorithms in EDs is highlighted as most of these devices are constrained by resources (e.g. battery, computation). Moreover, in Chapter 2 the importance of continuous retraining and adaption of machine learning models to concept drifts has been highlighted as essential for changing environments providing real-time (low latency) decision making. The following sections will introduce efficient model retraining and forwarding concepts at Edge Devices to preserve the devices resources and model selection of distributed learning at the Edge Gateways (EGs) for qualitative query-analytics. The attention lies in a qualitative and efficient implementation of edge-centric analytics to enable real-time analytics at the edge for evolving environments. Most of the research conducted in the following section has been published in [4]. Overall, this chapter contributes to the second defined research question in Section 1.2:

> *Hypothesis 2:* Enabling machine learning and predictive analytics locally at Edge Devices will empower real-time applications that can adapt intelligently to concept drifts and changes of the continuous data arriving. These locally learned (trained) models can be selected through qualitative model selection methodologies at central coordinators, e.g., Cloud.

4.2 State-of-the-Art on Edge-Centric Analytics

Centralised trained machine learning models are widely implemented in current Internet of Things (IoT) applications. Highlighted in Section 2.4, these models are distributed towards Edge Devices (EDs) to perform inference in real-time directly on the incoming data. Training is conducted over the entire data, so a continuous data transfer between devices and the Central Location (CL) is required. Edge Computing aims to push analytics and intelligence further towards the device (the data source), enabling the possibility to train machine learning models locally [227]. This local training or inferring of global initialised models inside EDs has gained interest and can be defined as the field of edge analytics and training. Recent work introduced approaches of edge-centric analytics that enable local computation in each EDs by using the available resources to perform model updates [185, 228, 229, 230, 231, 232]. All of these distributed methods focus on distributed estimation and minimisation of a global objective function aiming to archive the same performance as having all data at the centralised location. Retraining and updating the model at the CL is done through a periodical frequency or a request from the Central Location, which requires extra techniques and communication to synchronise the EDs and CL. Training an algorithm local requires additional resources of energy but also reduced the transmission of bandwidth through only transmitting analytical models. However, in an Edge Computing environment, only data forwarding and selection has been considered in the literature, which has been highlighted in the previous chapter. Efficient model retraining and forwarding has only very recently become of interest in the research area. Besides the most straightforward way of performing the model retraining by using a periodic or online method (each time a new input arrives), also complex algorithms to detect anomalies or changes in the data distribution have been widely implemented to trigger a model retraining. The work of [233] and [234] focus on model forwarding and retraining by extending the prior published work [4] of this chapter with time-optimised model forwarding strategies based on accumulating error of prediction values.

The approach of current implemented edge analytics applications lack real-time retraining and adapting towards changing environments, which is in IoT applications of high importance (highlighted in Section 2.3.3.3). As primarily a minimisation of one single central model is considered inside the ED for inferring or training, the local adaptation to the data and the individualisation of the data subspaces is neglected. The local model training and resulting diversity has not been studied in the context of edge analytics. However, as mentioned in Section 2.3.4, ensemble learning is mainly used to combine these models towards one single central prediction model when distributing machine learning models over different datasets. Model selection has been an ongoing research field, mostly focusing on selecting different orders and features of one model using accuracy metrics as selective criteria [235, 236]. Ensemble learning has been primarily used for classification tasks but has been considered recently in regression as well [237]. In the introduced theory of ensemble learning, all individual learners contribute towards the final prediction. However, especially in regression tasks, this approach is resolving into low accuracy. Therefore ensemble pruning aims to minimise the number of individual learners while maximising the accuracy of the prediction tasks [238]. It has been shown that selecting only a subset of individual learners results in more accurate results than using all [239]. Ensemble pruning emerged to reduce the computation cost of involving all learners and storing the data accordingly [240]. Pruning evolved into a method to efficiently selecting a subset of machine learning models that can be equal or higher in their accuracy as involving all models. Ensemble pruning can be classified into three categories [241]:

> Ordering: this category is aiming to order the ensemble of models via a particular criterion (e.g. error) and selecting the top-k models for the final ensemble prediction;
>
> Clustering: the ensemble of models are clustered into groups which are then used as selection to perform the prediction task;
>
> Optimisation: the ensemble of models is treated as an optimisation task using an algorithm to minimise an objective function (e.g. genetic algorithm [242]).

Exploring the local model diversity of each ED, a highly distributed training of models in energy and resource constraint devices has to be considered. This raises the need for efficient training and forwarding of the models and model consolidation or selection at the central location using ensemble pruning methodologies of low-computational complexity and resource-efficient in ED and EG to answer user queries.

4.3 Rationale & Problem Fundamentals

In the previous section, it has been highlighted that centralised predictive analytics increases the latency and cannot adapt to generate real-time decision making. Further, with the natural appearance of concept drifts, locally trained and updated machine learning and analytics models emerge. For example, considering an application with two EDs, each of them has its locally generated data. When performing a centralised model over the entire data, inaccurate predictions inside each ED will occur. Presenting a visual example of this, Figure 4.1 shows the influence of locally trained and individualised models at each ED, noted as f_1 and f_2, compared to a global generated function f_G.

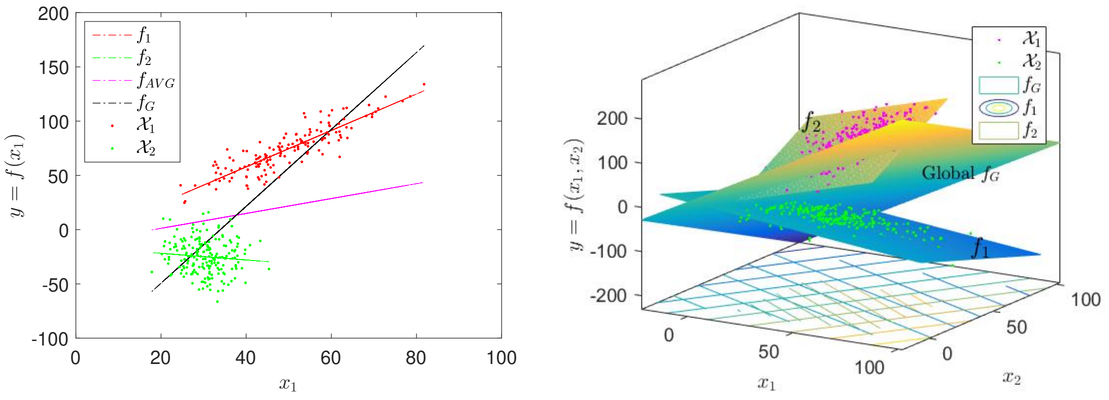


Figure 4.1: (Left) Local linear models f_1 and f_2 built over data $\mathcal{X}_1$ and $\mathcal{X}_2$ from ED$_1$ and ED$_2$, respectively, global model f_G build over $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$, and average model $f_{AVG} = \frac{f_1+f_2}{2}$; (right) Regression planes for local f_1, f_2 and global f_G models (f_{AVG} is not shown for readability) [4].

Based on the importance of local models enabling latency-efficient analytics in ED, this section abstracts the architecture as illustrated in Figure 4.2. In applications that require edge analytics, time series and continuous data for predictions are mainly used. Highlighted in Figure 2.7, using a regression algorithm is a possible solution for continuous data. Regression analytics can be performed over parametric and non-parametric algorithms. The non-parametric approaches require to store the entire input space X to generate predictions. As EDs are limited with storage and computational capacity this implementation is infeasible for edge-centric learning. On the other hand, parametric regression investigates the dependency between the input space X and the output space Y by adopting a function $y = f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^d$. Aim is to find the optimal model parameters $\mathbf{w}$ of the data X that minimise the loss function $\mathcal{L}$. This thesis section focuses on parametric regression analytics in a $(d+1)$-dimensional data space $(\mathbf{x}, y) \in \mathbb{R}^{d+1}$ over a continuous discrete time series with $t \in \mathbb{T} = \{1, \dots, t, \dots, T\}$ and $T \in \mathbb{T}$. The input variable $\mathbf{x}_t$ at time t is defined as $\mathbf{x}_t = [x_1, \dots, x_d]^T$ with the corresponding output y_t.

In the abstracted edge-centric analytics architecture each ED i learns online the regression model $y = f_i(\mathbf{x})$ over the input-output pairs $\{(\mathbf{x}, y)_i\} \in \mathbb{R}^{d+1}$. Each EG o is connected to s EDs with $\mathcal{S} = \{1, ..., i, ..., s\}$. This local learning introduces two problems: one assigned to the ED the other one to the EG. The first problem is placed at the device itself. The challenge on EDs is first to decide when to retrain and update the local model f_i and secondly when to forward the local models f_i to the EG. This forwarding methodology has to trade-off the quality of the performed analytics at the EG and the communication overhead through sending the model update over the network. The EG o caches the local models of each ED i inside its memory as f_i^o. The second problem of performing edge-centric analytics lies in ensemble the locally trained models cached in the EG defined as $\mathcal{F} = \{f_1^o, \ldots, f_s^o\}$. In a centralised setup, the application or user can issue a regression query represented as the point $\mathbf{q} \in \mathbb{R}^d$. It is possible to explore the behaviour of $f(\mathbf{x})$ around $\mathbf{q}$ and provide the prediction $\hat{y} = f(\mathbf{q})$ with prediction error $e(\mathbf{q}) = y - f(\mathbf{q})$. The EG has the challenge to find the best subset of $\mathcal{F}$ that maximises the given query's $\mathbf{q}$ accuracy.

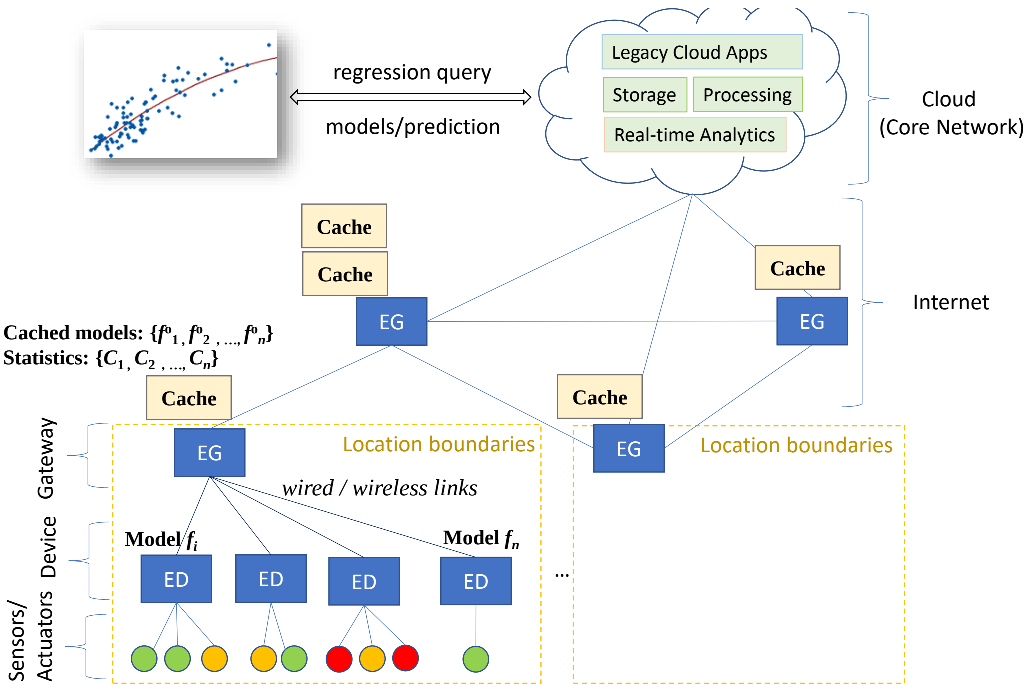


Figure 4.2: Physical world is divided in geographical units where IoT devices are deployed. Edge-centric regression analytics involve EGs, EDs, and SANs delivering cached models & sufficient statistics [4].

The idea to overcome the problems is to split the intelligence into Edge Device decision making towards retraining and forwarding and the Edge Gateway intelligence to aggregate and combine the locally generated regression models. Therefore, the following two cases can be defined as:

Edge Device Intelligence: Each ED i decides whether the pair $(\mathbf{x}, y)_t$ notably changes the prediction performance of the current local f_i or not. If this is the case, the ED i

appends $(\mathbf{x}, y)_t$ to the Sliding Window (SLW) $\mathcal{W}_i$ and discards the oldest pair. Further the ED retrains f_i based on the updated $\mathcal{W}_i$. If the input data is familiar and will not change the prediction performance, it forgets the measurements and does not retrain the model $f(\mathbf{x})_i$. In the case that the model is retrained, the ED i has additionally the task to decide if forwarding the updated model $f(\mathbf{x})_i$ towards the EG is necessary.

Edge Gateway Intelligence: Inside the EG the intelligence lies in deciding from the ensemble of locally trained models $\mathcal{F} = \{f_1^o, \dots, f_s^o\}$, which subset of models $\mathcal{F}' \subseteq \mathcal{F}$ to select towards maximising the accuracy. The aim is to be as accurate as of the global model f_G, which has been built over all collected data.

4.4 Retraining and Model Forwarding

The previous paragraph evaluated the importance of locally trained models inside Edge Devices (EDs). However, as highlighted, these devices are constrained in computational capacity and energy supply. Therefore, efficient retraining and model-forwarding of these local edge-centric analytics functions are of high interest. Moreover, the need of adapting the model not just based on time, but also on the quality of the performed predictions using a continuous evolving method is challenging, especially in an energy and resource constraint environment. Performing analytics on EDs to adapt to concept drifts and learn on continuous data streams can be achieved using window-based analytics as possible efficient implementation (highlighted in Section 2.3.3.3). This concept is designed to store the recent local data over a Sliding Window (SLW) $\mathcal{W}_i = \{(\mathbf{x}, y)_{t-M+1}, \dots, (\mathbf{x}, y)_t\}$: $\mathcal{W}_i$ consists of the most recent M observed input-output pairs $(\mathbf{x}, y)$. Given the architecture setup presented in Figure 4.2 a locally learned parametric regression model $f_i(\mathbf{x})$, e.g., $f_i(\mathbf{x}) = \mathbf{w}_i^\top(\mathbf{x})$ in each ED i. The data of the SLW $\mathcal{W}$ is used to build the local model f_i with its corresponding parameters $\mathbf{w}_i$. The collected models at the EG of each ED i are defined as f_i^o with the model parameters $\mathbf{w}_i^o$. The ED i only sends the parameters $\mathbf{w}_i$ towards the EG and not any raw data or measurements.

The decision on adjusting and sending the model has to be taken in real-time by sequentially observing input-output pairs. Each ED i captures at time t a new input-output pair $(\mathbf{x}, y)_t$. Simultaneous at this time t the ED i has to make a decision if the input-output pair $(\mathbf{x}, y)_t$ considerably changes the prediction accuracy of the current local f_i, or not. If it does not change the performance of the local model, it discards the pair and waits for the new time $t + 1$, in which a new pair arrives. If the new pair at time t is notably changing the current model, the ED does update the local f_i. Each ED i is responsible for updating the EG whenever there is a significant discrepancy of the prediction performance between the local f_i and cached f_i^o at the EG. Each ED i has a copy of f_i^o locally stored to derive the decision making of updating the local model f_i and sending the updated model to the EG. Based on the prediction performance towards the EG cached model f_i^o the ED decides to send the updated model. This decision of retraining and adapting the local model f_i is proposed by assessing the familiarity and novelty of input-output pairs measurements $(\mathbf{x}, y)_t$. This familiarity evaluation is done by an online methodology using an incremental adaption to the change of the incoming input-output pairs and the ability to react to possible concept drifts. The idea proposed in this section is to partition the input space of the values and associate a prediction error and performance measure to it. More formally, the input space of an individual ED i is quantised into K subspaces with $k = 1, \ldots, K$. The number of subspaces is generated throughout the runtime dynamically and is unknown a-priory at the ED i. Each subspace is represented by an input prototype $\mathbf{b}_k \in \mathbb{R}^d, k \in [K]$. The input prototypes $\mathbf{b}_k$ is mapped to an error prototype $u_k \in \mathbb{R}; k \in [K]$. This error prototype u_k is defined as the prediction error of the specific input prototypes $\mathbf{b}_k$ so that $e(\mathbf{x}) = y - f_i(\mathbf{x}) : k = \arg\min_{k \in K} \|\mathbf{x} - \mathbf{b}_k\|$. The intention of clustering the input space represented by $\mathbf{b}_k$ and associating the local prediction error represented by u_k with each other gives knowledge about the expected accuracy and allows fast action if these do not lie in the tolerance and therefore represents a novel input pair. Illustrating this quantisation of the input space, Figure 4.3 highlights the association of new input pairs towards the expected prediction error of the local model f_i and its novelty or familiarity.

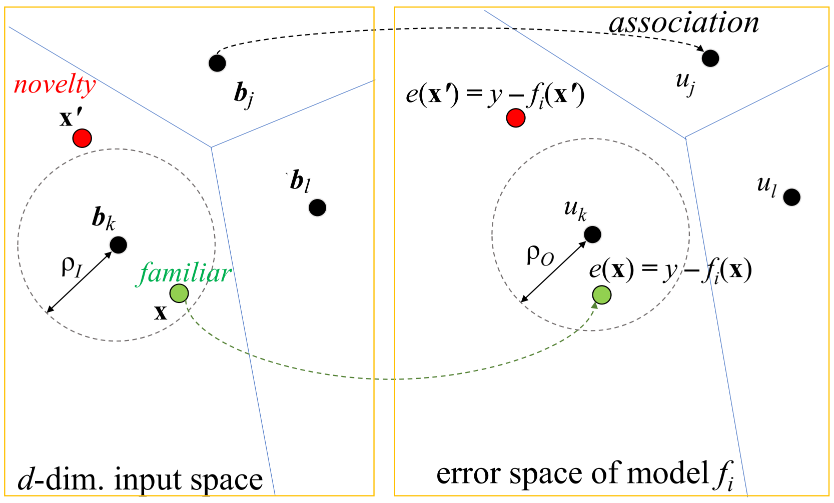


Figure 4.3: Statistics $\mathcal{C}_i$: association of input prototypes $\mathbf{b}_k$ with error prototypes u_k in input-error space determining the familiarity of $(\mathbf{x}, y)$ and $(\mathbf{x}', y')$.

The proposed fast and incremental input-error space quantisation at ED i with unknown number of prototypes K is an objective joint optimisation function that minimises the conditional Expected Quantisation Error (EQE) and the conditional Expected Prediction Error (EPE). The EQE is calculated through learning the best input prototypes $\mathbf{b}_k$, whereas the

EPE us based on the best error prototypes that capture the local model f_i accuracy performance. Overall, the condition of EQE and EPE has to be minimised based on the input/error prototypes $\mathcal{C}_i = \mathcal{B}_i \cup \mathcal{U}_i$, with $\mathcal{B}_i = \{\mathbf{b}_k\}$ and $\mathcal{U}_i = \{u_k\}$. This results into minimising the joint objective function of EQE and EPE with:

$$\mathcal{J}(\{\mathbf{b}_k, u_k\}) = \mathbb{E}\big[\lambda\|\mathbf{x} - \mathbf{b}_k\|^2 + (1-\lambda)|e(\mathbf{x}) - u_k|\big|\mathcal{A}_k\big] \tag{4.1}$$

In Equation (4.1) is $\mathcal{A}_k$ defined as $\mathcal{A}_k \equiv \{k = \arg\min_{l\in[K]}\|\mathbf{x} - \mathbf{b}_l\|^2\}$, $e(\mathbf{x}) = |y - f_i(\mathbf{x})|$ representing the absolute prediction error, and $\lambda \in [0,1]$ is a regularisation factor for weighting the importance of the input-error space quantisation. Defining $\lambda = 1$, refers to the known EQE [243], $\lambda \to 0$ indicates pure prediction-error based quantisation. The expectation is taken over input-error pairs $(\mathbf{x}, e(\mathbf{x})) \in \mathbb{R}^d \times \mathbb{R}$. Aim is to minimise the objective function given in Equation (4.1), using the prototypes $(\mathbf{b}_k, u_k) \in \mathcal{C}_i$ that need to be adapted and updated given a pair $(\mathbf{x}_t, y_t)$. This adaption of $\mathbf{b}_k$ and u_k each time t can be performed by the following Equation (4.2) with $\alpha_t \in (0,1)$ representing the learning rate: $\sum_{t=1}^{\infty}\alpha_t = \infty$ and $\sum_{t=1}^{\infty}\alpha_t^2 < \infty$, $e_t = |y_t - f_i(\mathbf{x}_t)|$, and $sgn(\cdot)$ is the signum function. The prototypes $(\mathbf{b}_k, u_k) \in \mathcal{C}_i$ converge to the centroid (mean vector) of inputs $\mathbf{x}$ and to the median of the absolute prediction error in the k-th input-error subspace.

$$\Delta\mathbf{b}_k = \alpha_t\lambda(\mathbf{x}_t - \mathbf{b}_k) \quad \Delta u_k = \alpha_t(1-\lambda)sgn(e_t - u_k) \tag{4.2}$$

The number of prototypes K is unknown to the ED i. Therefore the algorithm has incrementally to decide, based on the novelty of the input value and the local model performance, to add a new input-error prototype. The proposed evolving algorithm, which minimises the joint optimisation function presented in Equation (4.1), is summarised in Algorithm 2. This nature of evolving the prototypes during runtime allows the algorithm to be used without a training phase. However, to have a basis a training period is recommended.

At the initial start at time $t = 1$, the algorithm generates for $K = 1$ an input/error prototype pair $(\mathbf{b}_1, u_1)$ based on the first input values $(\mathbf{x}_1, y_1)$. For the duration of the application run time, two thresholds are defined that allow the algorithm to calculate the familiarity and novelty of a new input data $\mathbf{x}_t$. The algorithm's first threshold is defined as ρ_I, which indicates the closeness of the input prototype $\mathbf{b}_k$ to the new input $\mathbf{x}_t$. The second threshold ρ_O relates to the dynamically changing error tolerance for the current error $e(\mathbf{x_t}) = y - f_i(\mathbf{x}_t)$. After the initialisation of the first prototype-set, the algorithm uses the familiarity threshold ρ_I to indicate if the new value $\mathbf{x}$ has a corresponding input prototype $\mathbf{b}_k$. Suppose there is a b_k that is within the boundary of ρ_I. In that case, the new input $\mathbf{x}_t$ is classified as familiar if and only if the associated error prototype u_k and the resulted prediction error e are within

Algorithm 2 Online Local Algorithm at ED i.

```
Input: new pair $(\mathbf{x}, y)$
Output: familiarity; updated prototypes $\mathcal{C}_i$
 1: familiarity $\leftarrow$ FALSE
 2: closest input prototype $k = \arg\min_{\ell \in [K]} \|\mathbf{x} - \mathbf{b}_\ell\|$
 3: model prediction: $\hat{y} = f_i(\mathbf{x})$; absolute error $e = |y - \hat{y}|$
 4: if ($\|\mathbf{x} - \mathbf{b}_k\| \leq \rho_I$) then
 5:     prototype adaptation: $\Delta \mathbf{b}_k = \alpha\lambda(\mathbf{x} - \mathbf{b}_k)$
 6:     prototype adaptation: $\Delta u_k = \alpha(1-\lambda)sgn(e - u_k)$
 7:     if $e > \rho_O$ then
 8:         $\rho_O = \max(\frac{1}{2}\rho_O, \rho_O^*)$; adapt model $f_i$ w.r.t. $(\mathbf{x}, y)$
 9:     else
10:         familiarity $\leftarrow$ TRUE
11:     end if
12: else
13:     novelty (new prototype): $K = K + 1$, $\mathbf{b}_k = \mathbf{x}$, $e_K = e$
14:     if $e \leq \rho_O$ then
15:         $\rho_O = \max(\frac{1}{2}\rho_O, \rho_O^*)$; familiarity $\leftarrow$ TRUE
16:     else
17:         adapt model $f_i$ w.r.t. $(\mathbf{x}, y)$
18:     end if
19: end if
```

their tolerance ρ_O. If the error is within the range, the new input $\mathbf{x}_t$ is discarded, and the ED i waits towards the following input at time $t+1$. Both prototypes, $\mathbf{b}_k$ and u_k, are adapted towards the new input value as applicable in lines 5 and 6 of the algorithm. If the error of the input at time t is outside the range of ρ_O, this tolerance ρ_O decreases, denoting less tolerance in the error space for future inputs and triggers retraining of the local model f_i at the ED i. If the new input $\mathbf{x}_t$ is outside the range of the familiarity threshold ρ_I, a new prototype $\mathbf{b}_k$ is generated. Afterwards, a check regarding the error tolerance of ρ_O is made. If the error is within the range of ρ_O, ρ_O decreases, denoting less tolerance in the error space for future inputs. Still, the input $\mathbf{x}_t$ will be classified as familiar without retraining the model. Otherwise, if the input is outside the range of the closest $\mathbf{b}_k$ and the range of the associated error prototype u_k, direct retraining and updating of the local model f_i is triggered.

In short, the Algorithm 2 presents the intelligence of the ED i by highlighting the three main tasks of: (1) optimally quantising the input-error space by minimising Equation (4.1), (2) online deciding whether $(\mathbf{x}, y)$ is novel or not and used for triggering model adaptation and cache model update to the EG, and (3) incrementally evolving to identify new prototypes in $\mathcal{C}_i$. The outcome of the Algorithm 2 is the identification if a given pair $(\mathbf{x}, y)$ at time t is novel or familiar and if retraining of the local model is necessary.

The introduced algorithm of quantising the input/error space can lead to many prototypes that the ED is unable to store and handle for efficient computation. In avoiding this risk, multiple possible techniques can be used. One possible solution is applying a function that deletes or merges unused and old prototypes with the nearest. This function allows for continuous updating of the prototype space and keeps it to the only frequently used prototypes. Moreover, with the introduced parameters ρ_O and ρ_I, the radius and consequently the number of prototypes can be regulated. Furthermore, quantisation of the space can be difficult when dealing with high dimensions at the ED, overcoming this problem, multiple methods of dimensional reduction highlighted in related research can be applied before using the previously introduced method [244, 245]. One possible method is using Principal Component Analysis (PCA) to reduce the dimensions [246]. However, these methods are out of scope for this thesis.

The ED i has instantaneously to determine whether the new pair $(\mathbf{x}, y)$ is drawn from the input-output subspace defined by the pairs in the Sliding Window (SLW) $\mathcal{W}_i$ or not. If the new pair is interpolating within the current input-output data subspace, the values are considered as familiar and indicates that the current model f_i is expected to provide a good prediction $\hat{y}_t = f_i(\mathbf{x}_t)$ given the tth input $\mathbf{x}_t$. If the ED i does not need to retrain the local model as the accuracy of the model is not changing, so $|y_t - \hat{y}_t| \leq \epsilon$ for some accuracy threshold $\epsilon > 0$, then no communication between EG and ED occur. Whenever the input at time t is considered novel and some updating of the local model f_i is triggered, the decision to send the update to the EG or not has to be made at the ED. The parameter adaptation $\mathbf{w}_i$ upon a novel pair $(\mathbf{x}, y)$, is possible to deploy either a window-based batch retraining of f_i over window $\mathcal{W}_i$, or using the online Stochastic Gradient Descent (SGD) to incrementally update $\mathbf{w}_i$. In linear regression the use of a window-based batched updating can be formulate with the adapted parameters as $\mathbf{w}_i = (\sum_{l=1}^{N} \mathbf{x}_l^\top \mathbf{x}_l)^{-1}(\sum_{l=1}^{N} \mathbf{x}_l y_l)$ if a novelty pair $(\mathbf{x}, y)$ is inserted in $\mathcal{W}_i$ w.r.t. ordinary least squares optimisation. Comparing the adaption of $\mathbf{w}_i$ using the online and incrementally updating through SGD, the adaption of the parameters would be $\Delta\mathbf{w}_i = -\eta(y - f_i(\mathbf{x}))\mathbf{x}$; $\eta \in (0, 1)$.

Generally, three different cases are proposed to prompt the new local model f_i towards the EG. All of these strategies rely on the threshold θ which acts as tolerance and communication reduction measurement between ED and EG for the model update frequency. More specifically, the updated local model f_i is sent to the EG:

1. whenever the **prediction performance** of the local model f_i at the ED i shows a significant discrepancy towards the cached f_i^o model at the EG,
2. when the **fitting** of the input values stored in the SLW $\mathcal{W}_i$ towards the local model f_i compared to the cached model f_i^o shows a significant discrepancy,
3. or when the **model parameters** of the local model f_i show a significant discrepancy to the cached model f_i^o.

The first forwarding decision is based on the discrepancy of the prediction performance defined by the absolute difference of the prediction errors of f_i and f_i^o. The local prediction error is calculated with $e_i(\mathbf{x}_t) = |y - f_i(\mathbf{x})|$, whereas the prediction error of the cached model is calculated with $e_i^o(\mathbf{x}_t) = |y - f_i^o(\mathbf{x})|$. The error difference of the local prediction error e_i and the cached model error e_i^o is defined as $|e_i(\mathbf{x}_t) - e_i^o(\mathbf{x}_t)|$. If the error difference exceeds a threshold $\theta_e > 0$, then ED i updates the EG with the new model f_i and the locally stored cached model on the ED and EG is updated to $f_i^o = f_i$. If the difference of the performance predictions between both models is not exceeding the threshold θ_e, the models can be considered as similar and no update between EG and ED is required. Therefore the models of ED and EG can be seen as the equivalent in terms of prediction performance and the corresponding tolerance θ_e. The prediction performance forwarding decision method is highlighted in the Algorithm 3 as exemplary for the decision making of forwarding the updated model f_i to the EG.

Algorithm 3 Local Decision Making at ED i.

```
Input: input-output observed pair $(\mathbf{x}, y)$
 1: get pair $(\mathbf{x}, y)$ familiarity from Algorithm 2
 2: if $(\mathbf{x}, y)$ is novel (not familiar) then
 3:     append $(\mathbf{x}, y)$ in window $\mathcal{W}_i$; adapt/retrain model $f_i$
 4:     model prediction error: $e_i(\mathbf{x}) = |y - f_i(\mathbf{x})|$
 5:     cached model prediction error: $e_i^o(\mathbf{x}) = |y - f_i^o(\mathbf{x})|$
 6:     if $|e_i(\mathbf{x}) - e_i^o(\mathbf{x})| > \theta$ then
 7:         update EG with the new model $f_i$
 8:         update cache model $f_i^o \leftarrow f_i$
 9:     end if
10: end if
```

The second possible forwarding decision is proposed by the discrepancy of the model fitting towards the new input values in the SLW $\mathcal{W}_i$ of length M and the respective cached model. For this the R^2 is calculated using the local retrained model f_i resulting into $R_i^2 = \frac{\sum_{i=1}^{M}(y_i - f_i(\mathbf{x}))}{\sum_{i=1}^{M}(y_i - \bar{y})}$ and the EG cached model f_i^o resulting into $R_o^2 = \frac{\sum_{i=1}^{M}(y_i - f_i^o(\mathbf{x}))}{\sum_{i=1}^{M}(y_i - \bar{y})}$. If the absolute difference of R_i^2 and R_o^2 exceeds the pre-defined threshold $\theta_{r2} > 0$, then ED i updates the EG with the new model f_i and the locally stored cached model on the ED and EG is updated to $f_i^o = f_i$. If the difference of the fitting between both models is not exceeding the threshold θ_{r2}, the models can be considered as similar and no update between EG and ED is required. In this case both models can also be seen as equivalent in terms of fitting the values to the model and the corresponding tolerance θ_{r2}.

The last possible discrepancy measurement of deciding to update the EG model is based on the absolute difference of the model parameters $\mathbf{w}$ between the local model f_i and the cached model f_i^o. The difference of parameters between these two models can be calculated by $|w_i - w_i^o|$. If the error difference exceeds a threshold $\theta_w > 0$, then ED i updates the EG with the new model f_i and the locally stored cached model on the ED and EG is updated to $f_i^o = f_i$. Suppose the difference of the model parameters between the local $\mathbf{w}_i$ and cached model $\mathbf{w}_i^o$ is not exceeding the threshold θ_w. These models can be considered similar in that case, and no update between EG and ED is required. Then these models can be seen as equivalent in terms of model parameters concerning the corresponding tolerance θ_w.

4.5 Qualitative Model Selection

The previous paragraph introduced local model training at Edge Devices (EDs) using a familiarity and novelty incremental algorithm to retrain the model efficiently. Moreover, a selective model forwarding and updating strategy based on predictive, fitting, and parameter differences with respect to the Edge Gateway (EG) model has been presented. As stated earlier, the use of edge-centric learning and adaption leads to an ensemble of models collected at the EG. This is caused by the forwarding of each ED i generated local model f_i towards the EG. Applications in an IoT environment require a centralised knowledge for answering queries from a user or an application. Solving the problem of having an ensemble $\mathcal{F} = \{f_1^o, \ldots, f_s^o\}$ of cached local models on the EG is proposed by model selection techniques in this section. The aim is to define a model selection scheme that approximates the best subset of models $\mathcal{F}' \subseteq \mathcal{F}$, which are as accurate as the global model f_G, built over all collected data. In this section, two possible model selection methods are introduced. The first one is based on the familiarity calculations performed in the proposed ED intelligence for retraining the local model. The selection of the subset of models is performed by the input-error quantisation given from each ED. The second is based on the similarity of each local ED model using Dynamic Time Warping (DTW) as measurement and some clustering algorithm. Generalised, the first model selection strategy can be placed into ordering ensemble pruning. The second proposed methodology can be placed into the clustering pruning methods, which has been introduced in Section 4.2.

4.5.1 Input/Error-Space Quantisation

In Section 4.4, the usage of input-error space quantisation as a decision metric of retraining and adapting the local model in EDs has been highlighted. It has been shown that the ED i uses an online learning method to generate statistics over the received input values in the form of prototypes. Each of the input data prototypes, noted as $\mathbf{b}_k$, is associated with an error prototype, noted as u_k, that shows the prediction performance of the local model f_i. These two prototypes are represented inside the statistics $\mathcal{C}_i$. These statistics are generated inside each ED and can be sent along with the updated model parameters $\mathbf{w}_i$ towards the

EG. Each ED i decides independently of the other EDs on forwarding its local model f_i to the EG. This results into different up-to-date models inside the EG. Further, due to the local learning of each ED i, the EG stores s independent models in $\mathcal{F} = \{f_1^o, \dots, f_i^o, \dots, f_s^o\}$. The main tasks of the EG, as it is connected towards the Cloud (see architecture illustration in Figure 4.2), is to answer queries $\mathbf{q} \in \mathbb{R}^d$ coming from users or the application itself. So the main challenge of the EG is to select from the variety of models stored the best subset to answer the user queries $\mathbf{q} \in \mathbb{R}^d$. The answer to the query is needed in real-time and as accurate as possible, which discards the possibility of asking the EDs for an update of the local models. The proposed model selection inside the EG, is not only based on the parameters of the models from each ED i but also on the statistics $\mathcal{C}_i$ containing the input prototypes $\mathbf{b}_k$ and the associated error prototypes u_k of each ED i. These statistics can be used to guide the EG in selecting the appropriate subset of local models $\mathcal{F}' \subseteq \mathcal{F}$ to generate an ensemble prediction $\hat{y}$ towards the given query $\mathbf{q}$. This prediction $\hat{y}$ has to be of high quality concerning accuracy as applications derive decisions and actions from them. The ensemble model selection aims to generate the accuracy and prediction performance as close as possible towards the global function f_G, which would have been generated when all raw data would be available at the EG.

The proposed model selection inside the EG is based on the elaborated method of bagging (see Section 2.3.4), in which each learner's prediction is aggregated towards the final prediction, using majority voting or averaging. In the designed architecture continuous data prediction is assumed, which leads to the advanced modified bagging methodology shown in Equation (4.3). The ensemble prediction $\hat{y}$ of the EG is therefore made by the weighted sum of the individual predictions $\hat{y}_i = f_i^o(\mathbf{q})$ from the cached models:

$$\hat{y} = \sum_{i=1}^{s} f_i^o(\mathbf{q})\beta_i(\mathbf{q}). \tag{4.3}$$

The weight $\beta_i(\mathbf{q})$ in Equation (4.3) shows a function of $\mathbf{q}$ that can be interpreted as importance factor of the local model f_i^o at the EG to find the optimal subset of models $\mathcal{F}' \subseteq \mathcal{F}$. In the following paragraphs, three possible implementation and weighting strategies are introduced and highlighted to optimise selecting the best subset of local models.

Simple Model Aggregation (SMA): The first proposed method on aggregating and ensemble pruning the cached model inside the EG is without exploiting the statistics $\mathcal{C}_i$ from the EDs. The updated model f_i will be send from ED to EG based on the in Section 4.4 defined novelty and updating mechanism highlighted in Algorithm 2. In the model selection strategy SMA the EG simply aggregates the individual predictions $\hat{y}_i = f_i^o(\mathbf{q})$ for deriving the final prediction $\hat{y}$. The weighted function $\beta_i(\mathbf{q})$ is set to $\beta_i(\mathbf{q}) = 1/s$. In this case the ensemble subset $\mathcal{F}' \equiv \mathcal{F}$ so that the EG places an equal importance towards each local model f_i

into the final prediction of $\hat{y}$ to answer the query $\mathbf{q}$. This methodology is equivalent to the introduced bagging methodology of Section 2.3.4. Overall the formula to find the ensemble prediction using the SMA model selection strategy can be defined as:

$$\hat{y} = f_{AVG}(\mathbf{q}) = \sum_{i=1}^{s} \hat{y}_i \frac{1}{s} \tag{4.4}$$

Input-space Aware top-$\mathcal{K}$ Model (IAM): The second presented model selection strategy is incorporating the statistics C_i send from each ED i towards the EG. The EG is using these statistics to decide on the best $\mathcal{K}$ subset of local models f_i for a given query $\mathbf{q}$. More specifically IAM is using the input prototypes $\mathbf{b_{i,k}}$ over all cached models. Explaining the procedure of the IAM, first the value of $\mathcal{K}$ is set to $\mathcal{K} = 1$, representing the best top-1 model for generating the ensemble prediction $\hat{y}$. The selected model can be defined as f^* with $f^* \in \mathcal{F}$ so that the subset of the models only contain one single model, $\mathcal{F}' = \{f^*\}$. The selected model with using the strategy of IAM is based on the ℓ-th input prototype $\mathbf{b}_\ell^*$ that is the closest to query $\mathbf{q}$ compared to all input prototypes in $\mathcal{B} = \{\{\mathbf{b}_{1,k}\}_{k=1}^{k_1} \cup \cdots \cup \{\mathbf{b}_{s,k}\}_{k=1}^{k_s}\}$ from all s models:

$$\mathbf{b}_\ell^* = \arg\min_{\mathbf{b} \in \mathcal{B}} \|\mathbf{q} - \mathbf{b}\| \tag{4.5}$$

The from the EG selected best model f^* is considered to be the closest or most familiar to the given query q based on the input subspace (represented by $\mathbf{b}_\ell^*$) and should therefor be providing the best prediction. Therefore, with using IAM it is important that each ED i is updating and sending it's statistics C_i towards the EG. Otherwise the EG can not discriminate which model's input subspace is the most familiar with the given query point. In Equation (4.3) the ensemble prediction $\hat{y}$ is calculated through the prediction of the selected model f^* and some weighting factor $\beta_i(\mathbf{q})$. The function responsible for weighting each cached model f_i^o, highlighted in Equation (4.3), can be defined in IAM as:

$$\beta_i(\mathbf{q}) = \begin{cases} 1 & \text{if } \exists \mathbf{b}_{i,k} \in \mathcal{B}_i : \mathbf{b}_{i,k} = \mathbf{b}_\ell^* \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

Given the definition in Equation (4.6), $\beta_i(\mathbf{q})$ is defined to be 1 if the closest distance of $\mathbf{q}$ to the selected $\mathbf{b}_\ell^*$ exists, otherwise it is set to be 0. In the case of $\mathcal{K} = 1$ for IAM the EG only selects one model f^* with the closest input prototype for prediction so that $\hat{y} = f^*(\mathbf{q})$. After highlighting the approach using $\mathcal{K} = 1$, the approach needs to be generalised when $\mathcal{K} > 1$. In this case the EG ranks all prototypes $\mathbf{b} \in \mathcal{B}$ with respect to their distance towards the query $\mathbf{q}$. Those models $f_1^*, \ldots, f_\mathcal{K}^* \in \mathcal{F}' \subset \mathcal{F}$ whose closest input prototypes are ranked in the top-$\mathcal{K}$ closest distances are used for the ensemble prediction, which is then:

$$\hat{y} = \sum_{i=1}^{\mathcal{K}} f_i^*(\mathbf{q})\beta_i^*(\mathbf{q}) \tag{4.7}$$

The weighting factor $\beta_i^*(\mathbf{q})$ is normalised to [0,1] with respect to the top-$\mathcal{K}$ inverse distances:

$$\beta_i^*(\mathbf{q}) \quad = \quad \frac{e^{-\|\mathbf{q}-\mathbf{b}_{i,\ell}^*\|^2}}{\sum_{l=1}^{\mathcal{K}} e^{-\|\mathbf{q}-\mathbf{b}_{l,\ell}^*\|^2}}. \tag{4.8}$$

The influence of the distance $\|\mathbf{q}-\mathbf{b}\|$, i.e., the closer to $\mathbf{q}$ the higher the weight importance, is achieved by the exponential inverse squared distance weighting $e^{-\|\mathbf{q}-\mathbf{b}\|^2}$. For $\mathcal{K} = S$, the Weighted SMA (WSMA) is obtained, where the normalised weights reflect the (inverse) distance of $\mathbf{q}$ to the local closest input prototypes from each model.

Input/Error-space Aware top-$\mathcal{K}$ Model (IEAM): The last possible selection strategy for finding the optimal subset of models $\mathcal{F}' \subset \mathcal{F}$ is using not only the input prototype $\mathbf{b}_{i,\ell}$ of the statistics $\mathcal{C}_i, \forall i$ but also the associated performance reflected by the error prototype $u_{i,\ell}$. So the model selection of IEAM selects the best top-$\mathcal{K}$ models from $\mathcal{F}$, which are not only familiar to the queried input but also effective for providing accurate predictions based on their local prediction performance. This prediction performance is given through the error prototype $u_{i,\ell}$ connected to the closest input prototype $\mathbf{b}_{i,\ell}$. Using a combination of both directions, input space familiarity and associated prediction performance, allows the EG to generate a more sophisticated model selection. Similar to the IAM model selection the ensemble prediction $\hat{y}$ is generated with the prediction of the selected models and some weighting factor. The weight $\beta_i(\mathbf{q})$ for IEAM represents a degree of model closeness to an issued query taking into consideration the (inverse) closest input distance $\mathbf{b}_{i,\ell} \in \mathcal{B}_i$ and the associated median of the absolute prediction error $u_{i,\ell}$ around this subspace. Specifically, $\beta_i(\mathbf{q})$ interprets the relative closeness of model f_i to query $\mathbf{q}$ and can therefore be defined as:

$$\beta_i(\mathbf{q}) \quad = \quad \frac{e^{-\|\mathbf{q}-\mathbf{b}_{i,\ell}\|^2}(1-\bar{u}_{i,\ell})}{\sum_{l=1}^{\mathcal{K}} e^{-\|\mathbf{q}-\mathbf{b}_{l,\ell}\|^2}(1-\bar{u}_{l,\ell})}, \tag{4.9}$$

In Equation (4.9) is $\bar{u}_{i,k} = \frac{u_{i,k}}{\sum_{u\in\mathcal{U}} u}$ showing the normalised median of the prediction error of model f_i over the k-th input/error subspace among all error medians $\mathcal{U} = \{\{u_{1,k}\}_{k=1}^{k_1} \cup \cdots \cup \{u_{s,k}\}_{k=1}^{k_s}\}$ from all s models. Using the weighting factor of Equation (4.9) for the final prediction outcome $\hat{y}$, IEAM with $\mathcal{K} \geq 1$ generates a ranking of the models from $\mathcal{F}$ with the top-$\mathcal{K}$ having the highest degree of closeness by $\beta_i(\mathbf{q})$. So that $\hat{y} = \sum_{i=1}^{\mathcal{K}} f_i(\mathbf{q})\beta_i(\mathbf{q})$ where $\beta_i(\mathbf{q})$ is provided in (4.9).

4.5.2 Similarity-Based Clustering

In the previous section, the top-$\mathcal{K}$ best models close to the input space $\mathbf{b}_k$ or the input/error space using both statistics in $\mathcal{C}_i$ have been used to generate a selection of models to provide an ensemble prediction $\hat{y}$ towards a query q. These methods belong to the methodology of ordering ensemble pruning techniques. In the following paragraph, this strategy is advanced by introducing a similarity clustering-based pruning approach for model selection inside the EG.

In Section 3.6.2, the metric of Dynamic Time Warping (DTW) [218, 217] for highlighting time series similarity has been introduced. In dynamic environments such as IoT applications with EDs, the ability to have data streams of the exact same time length is infeasible. Therefore, using Euclidean distance (L^2-norm) as a similarity metric results in poor results as it only compares the points at time t of each time series with each other. Euclidean distance does not consider that two time series can have the same distribution but might be a shift by a specific time horizon. In Figure 4.4, the visual difference of DTW and L^2-norm as time series comparison metric is highlighted.

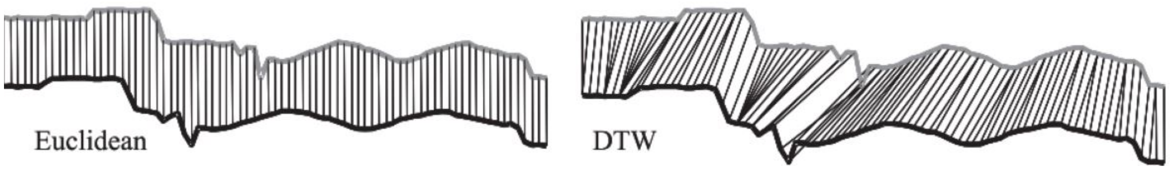


Figure 4.4: Comparison of DTW and L^2-norm for finding similarity in time series [218].

DTW calculates in an univariate data dimension setup the similarity of one input data stream $\mathbf{x} = [x_1, x_2, ..., x_g, ..., x_n]^T$ with another data stream $\mathbf{z} = [z_1, z_2, ..., z_h, ..., z_m]^T$. Aim is finding the optimal wrapping path $\mathcal{P}$ which minimises the distance between these two data streams by aligning the elements of each data stream. The distance can be expressed by a function $dist(g, h)$ for the elements at point g of the datastream $\mathbf{x}$ and h of the data stream $\mathbf{z}$. The most popular distance functions are: (i) $dist(g, h) = |x_g - z_h|$, using the absolute difference between two points or the magnitude; (ii) $dist(g, h) = (x_g - z_h)^2$, for the squared of the difference. Based on the calculated distances an n-by-m matrix is generated, in which each element (g, h) corresponds to the distance between point $dist(x_g, z_h)$. Given this distance matrix, it is possible to define a warping path $\mathcal{P}$ with $\mathcal{P} = (p_1, p_2, ..., p_r, ..., p_R), max(m, n) \leq R < m + n - 1$. The warping path is constraint to the following three conditions:

- Boundary conditions: The warping path has to start at $p_1 = (1, 1)$ and end at $p_R = (m, n)$;
- Continuity: The steps are restricted to the neighboring points. If warping path $p_r = (g, h)$, then $p_{r-1} = (g', h')$, where $g - g' \leq 1$ and $h - h' \leq 1$.

- Monotonicity: The points in the warping path must be monotonically ordered. Given $p_r = (g, h)$, then $p_{r-1} = (g', h')$, with $g - g' \geq 0$ and $h - h' > 0$.

Element r of the warping path $\mathcal{P}$ is defined as $p_r = (g, h)_r$. The optimal path $\mathcal{P}$ can be found by minimising the cumulative distance for each path (see Equation (4.10)). Therefore, the DTW measure can be defined as:

$$dtw(x, z) = \min_{p}(\sum_{r=1}^{R} dist(p_r)) \tag{4.10}$$

To calculate the minimum warping path of Equation (4.10), dynamic programming is used with the following recurrence of Equation (4.11). This equation defines the cumulative distance $\gamma(g, h)$ as the distance $dist(g, h)$ of the current element and the minimum of the cumulative distances of the adjacent elements, resulting into the following equation:

$$\gamma(g, h) = dist(x_g, z_h) + \min\{\gamma(g-1, h-1), \gamma(g-1, h), \gamma(g, h-1)\} \tag{4.11}$$

In applications of deployed EDs, the most common measurements and values are occurring in a multidimensional space. Using the DTW as a measurement for the similarity between different EDs, the DTW calculation must be adjusted towards a multidimensional usage. Several DTW extensions towards multidimensional space have been proposed in the literature [247, 248, 249, 250, 251]. Most common is the use of a p-norm of each dimension. It is important to normalise and scale the data of each dimension with a mean of zero. In Equation (4.12), the formula of the MD-DTW is highlighted. Each dimension can be weighted with λ_n towards its importance and influence. If $\lambda = 1$ for all dimensions d, each dimension is set to equal importance.

$$dist(g, h) = |\sum_{n=1}^{d} \lambda_n (x_{g,n} - z_{h,n})^p|^{\frac{1}{p}} \tag{4.12}$$

As this thesis contributes towards efficient and edge-centric data analytics over continuous data streams, it should be noted that the calculation of DTW can be extended by scaling the algorithm of DTW [252], introducing online and streaming DTW [253], or by performing piecewise calculations of DTW measurements [254]. It is out of scope for this thesis to evaluate these different methods and rather point the reader to the cited literature.

The introduced MD-DTW can be used as similarity measure inside the EG to group the cached local models $\mathcal{F} = \{f_1^o, \ldots, f_i^o, \ldots, f_s^o\}$ of all connected EDs s. Performing the grouping using the MD-DTW, a reconstruction of the actual multidimensional data stream of each ED i is required to extract in the EG. This requires additional computation from the

ED by performing an additional model $h_i(\mathbf{x})$ that represents the input data of ED i over a time-horizon M using a Sliding Window (SLW) $\mathcal{W}_i$ at the EG. Therefore, the ED i is sending additionally to its local model f_i, a model h_i that represents the input data. Highlighted in Section 2.3.3.1, ARIMA is one possible solution to model a time series. However, the authors in [255] show a communication efficient and qualitative approach of reconstructing the data of a node in a centralised location without transmitting data. The function h_i is cached inside the EG so a collection of $\mathcal{H} = \{h_1^o, \dots, h_i^o, \dots, h_j^o, \dots, h_s^o\}$ of reconstruction models is available. The EG is able to generate the distance measurement DTW for each ED i and ED j using the reconstruction functions h_i^o. A normalisation of this dtw value has to be performed to guarantee a possible comparison metric needed for further evaluation. Therefore, the $dtw(h_i^o, h_j^o)$ will be divided by the length of each ED i and j dataset, resulting in the metric of Normalised Dynamic Time Warping (N-DTW) as presented in Equation (4.13).

$$ndtw(i,j) = \frac{dtw(h_i^o, h_j^o)}{|h_i^o| + |h_j^o|} \tag{4.13}$$

Given the $ndtw$ value matrix of each combination of the EDs s, it is possible to perform grouping based on their similarity. Most commonly used for clustering over continuous data or grouping similarity-based time series [256] are the algorithms already presented in Figure 2.7. These can be hierarchical clustering, k-means, and density-based clustering with e.g. DBSCAN. The proposed grouping algorithm inside the EG is fundamental for the selection of the subset $\mathcal{F}' \subseteq \mathcal{F}$ of stored models from each ED i. In this thesis, the hierarchical cluster algorithm performs the grouping based on the $ndtw$ matrix. Not only is this algorithm parameter-free and deterministic, but also it is easy to interpret and understand. Primarily when further actions are performed based on the draw insights of the given groups, a hierarchical implementation allows easier exploring. Moreover, the architecture in IoT and edge-centric applications is mostly ordered over different communication levels, allowing this chosen cluster algorithm to be adapting this network architecture. The result of the hierarchical clustering is a dendrogram, showing the grouping over different levels, an example is visualised in Figure 4.5. Important is that after constructing this dendrogram, a cost threshold st has been defined to select the level of grouping.

Hierarchical clustering can be divided into two possible methods: divisive and agglomerative. The agglomerative clustering is using a bottom-up grouping method, in which each member (ED) is placed in one cluster. Then the most similar members are grouped together based on the given linkage function. In this thesis the members would be each ED i with its distance metric of the $ndtw$ (see Equation (4.13)). This results into starting with $\mathcal{A}$-cluster with $\mathcal{A} = s$ and s represents the number of EDs that are connect to the EG. The used linkage

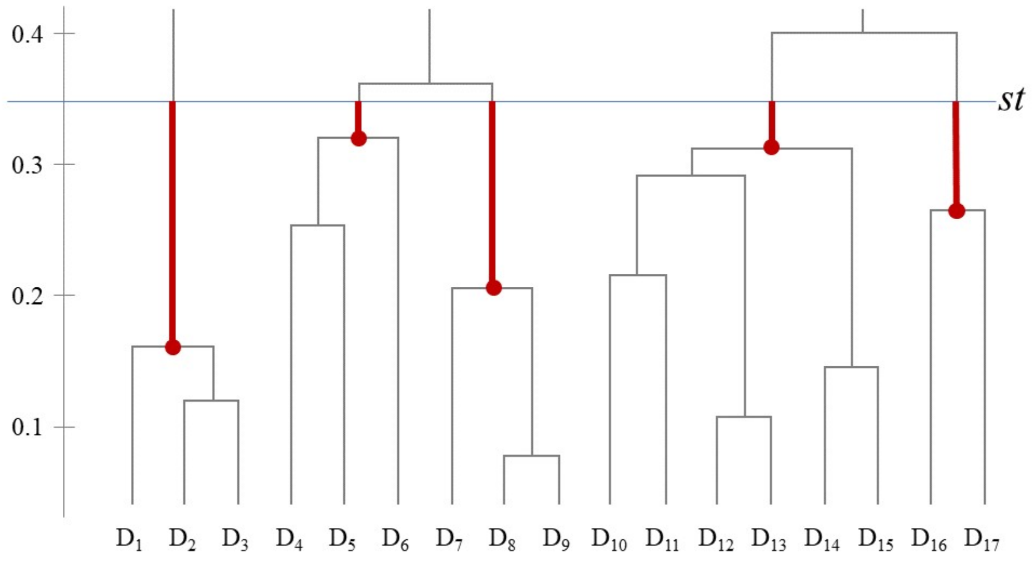


Figure 4.5: Dendrogram example for hierarchical clustering.

functions [96] available in the literature are:

- *Single*, grouping the members with the lowest (closest) distance and therefore dissimilarity. This is defined as: $dist(x, z) = \min_{i \in x, j \in z} dist(i, j)$
- *Complete*, grouping of the members with the highest (furthest) distance and therefore dissimilarity: $dist(x, z) = \max_{i \in x, j \in z} dist(i, j)$
- *Average*, grouping by using the average distance between two members: $dist(x, z) = \frac{1}{|x||z|} \sum_{i \in x} \sum_{j \in z} dist(i, j)$

Divisive clustering is using the approach of top-down by splitting the already grouped members based on their dissimilarity. Therefore, this method is starting with $\mathcal{A} = 1$ cluster and split towards $\mathcal{A} = s$ cluster or until some dissimilarity threshold is reached.

This thesis aims to propose a latency-efficient edge-centric analytic strategy that will not send any raw data over the communication channel from ED i to EG but instead only metadata and model parameters $\mathbf{w}_i$ of the locally updated model f_i. In the previous section, the statistics C_i of each ED are sent to the EG alongside the model parameters $\mathbf{w}_i$. These statistics are used to select the top-$\mathcal{K}$ best models to answer a query $\mathbf{q}$. The similarity-based model selection inside the EG receives besides $\mathcal{C}_i$ and f_i the data reconstruction function h_i from the EDs. The EG performs the calculation of $ndtw$ (see Equation (4.13)) and the agglomerative hierarchical cluster with an average linkage function based on a cost threshold st. This results into a cluster group $\mathcal{A} = \{A_1, \ldots, A_a\}$ mapping each ED i into a group $A_a \in \mathcal{A}$. So that the each of the ensemble models in the EG is mapped as: $\mathcal{F} = \cup_{i=1}^{a} A_i$ and $A_i \cap A_j = \varnothing$ for $i \neq j$. Issues the application or user now a query $\mathbf{q}$ to the EG, the query $\mathbf{q}$ is finding the closest $\mathcal{K} = 1$ input data prototype $\mathbf{b}_\ell^*$ that minimised the distance as defined for the IAM using Equation (4.5). Then the cluster group A of the associated ED i is identified and all members n of this group A are selected for $f_1^*, \ldots, f_n^* \in \mathcal{F}' \subseteq \mathcal{F}$. The ensemble prediction of Equation (4.3) for using the similarity-based model selection inside the EG over the best selected subset is defined as:

$$\hat{y} = \sum_{j=1}^{n} f_j^*(\mathbf{q})\beta_j^*(\mathbf{q}) \tag{4.14}$$

The weighted factor $\beta_j^*(\mathbf{q})$ is defined using the SMA with $1/n$.

4.6 Performance Evaluation

4.6.1 Experimental Setup

The dataset (DS) used in this chapter has been already introduced in Section 3.6. It contains 415 weather stations deployed around the United Kingdom (UK). Each of these stations represents an Edge Device (ED) that measures the surrounding environment's contextual data. This data has been collected over the time horizon of four-month (December 2017 till March 2018), using the API of Wunderground [216]. For simplicity only 30 stations from the overall 415 has been selected randomly to show the methodology of this chapter. It should be noted, that another dataset and the effectiveness has been published in previous work [4] and in related work advancing the introduced methodologies [233]. An overview of the location of the chosen weather stations can be found in Appendix B. Each weather station represents an ED i with an overall of $s = 30$ stations, each measures a 9-dimensional vector $\mathbf{x}$ containing temperature, dew point, humidity, wind-speed, wind-gust, wind direction, pressure, wind chill, and precipitation. The data collected frequency is every 5 minutes over the time horizon of 87 days, resulting in a dataset size of $N = 650,683$, assembling roughly 250 values measured per ED per day. All data is normalised and scaled, i.e., each parameter $x \in \mathbb{R}$ is mapped to $\frac{x-\mu}{\sigma}$ with mean value μ and variance σ and scaled in [0,1], thus vector $\mathbf{x} \in [0,1]^d$.

The experiments perform a parametric regression analytics over a $d = 9$ dimensional dataspace $\mathbf{x}, y)$. Each ED i is evolving a function $f_i(\mathbf{x})$ to generate a multivariate Linear Regression Model (LM) to learn the dependency between input $\mathbf{x}$ and output y. The input variables for the dataset representing $\mathbf{x}$ are the measurements: dew point, humidity, wind-speed, wind-gust, wind direction, pressure, wind chill, and precipitation. Furthermore, we aim to predict the output-variable y representing the temperature. This LM $y = f_i(\mathbf{x}) = \mathbf{w}_i\mathbf{x}^T$ is distributed by only the constructed model-parameters $\mathbf{w}_i \in \mathbb{R}^d$ over the network towards the

EG. Along with the regression function $f_i(\mathbf{x})$ the ED is equipped with a time series reconstruction function $h_i(\mathbf{x})$ that is able to represent each dimension for reconstruction. Both functions are evolving and learned over a Sliding Window (SLW) $\mathcal{W}_i$ with a time-horizon M. The setup for the experiments are a training time period of $t = 3,000$ and a testing period of $t = 35,000$. After the initialisation period, the methodologies explained in Section 4.6.3 will be carried out inside the ED and EG.

Assessing the accuracy of the methodologies, a type of cross-validation has been deployed to guarantee independent validation of the results. Randomly 24 time-points have been selected. At selected time the methodology implemented is stopped and the next M values of each station is used as query inputs $\mathbf{q}$ at the EG to analyse the performance of the different model selection strategies. More specific, the following time points have been chosen: 2018-01-02, 2018-01-05, 2018-01-12, 2018-01-22, 2018-01-25, 2018-01-29, 2018-02-01, 2018-02-05, 2018-02-07, 2018-02-08, 2018-02-09, 2018-02-11, 2018-02-14, 2018-02-16, 2018-02-17, 2018-02-18, 2018-02-19, 2018-02-20, 2018-02-21, 2018-02-22, 2018-02-23, 2018-02-24, 2018-02-25, 2018-02-26.

4.6.2 Performance Metrics

The conducted performance metrics rely on the already introduced metrics in Section 3.6.2. This chapter uses the Kullback-Leibler (KL) divergence to identify the information loss primarily used for the sensitivity measurement of the conducted analysis. The definition has been introduced in Equation (3.26). Further, to compare the predictive results, the accuracy metrics of MAE, RMSE and SMAPE are used. The calculation can be found in Equation (3.30), (3.29), and (3.31), respectively. Besides the information loss and accuracy, the methods presented rely on the importance of efficiency concerning communication reduction and energy saving. Therefore, the same calculation of communication as introduced in the previous chapter is used. The communication is measured as the number of times the transmission of data occur from the ED i over all EDs s to the EG.

$$c(T) = \sum_{t=1}^{T} \sum_{i=1}^{s} I_{i,t}, \tag{4.15}$$

The presented Equation (4.15) is defined with $I_{i,t} = 1$ if ED i sends its sensed value to the EG and $I_{i,t} = 0$ otherwise. Evidently, the overall communication of s EDs over T sensed values can be presented as baseline method and requires $T \cdot s$ communication, since $I_{i,t} = 1, \forall i, t$. The percentage of communication is then $\frac{c(T)}{T \cdots}$ when applying the proposed methodologies.

4.6.3 Comparative Assessment

Evaluating the performance of the latency-efficient edge-centric analytics strategies introduced in this chapter, the evaluation must be split into the model-retraining and model-forwarding procedures implemented inside the ED and presented in Section 4.4. This ED strategy of efficiency is mainly using the idea of familiarity and input-error space quantisation as decision mechanism to forward the model to the EG. Based on this idea, the following forwarding mechanism can be defined and tested in the following sections:

1. the first model-forwarding method relies on the **prediction** error tolerance, deciding to forward the model parameters $\mathbf{w}_i$ if the error between the current cached model at the EG and the updated model at the ED differs above a given threshold (θ_e);
2. the second decision to forward the model parameters to the EG, is based on the **fitting** of the model with respect to the data. The fitting is measured using the R^2 metric over the EG cached model and local updated model over a time horizon M. If the difference between them exceeds a threshold (θ_{r2}), the model is sent to the EG;
3. the last model-forwarding decision method is based on the divergence of the **parameters** $\mathbf{w}_i$ of the newly trained local model and the parameters of the cached model inside the EG. The divergence is measured by the angle between them represented with a degree value. If this degree is exceeding a certain value (θ_w), the local model is sent to the EG.

This chapter aims not only to reduce communication by deciding on when to forward the model from the ED to the EG, but also to enable edge analytics for real-time prediction. Therefore, the second evaluation focus is on qualitative model selection methodologies introduced in Section 4.5. The aim is to assess these methods based on their ability to qualitative predict a given query inside the EG. Overall the following methods are compared with each other in the assessment analysis:

1. The **global** method is representing the baseline with sending all raw data to the EG and generating a single function;
2. **Simple Model Aggregation (SMA)** introduces the method of aggregating all individual functions f_i generated by each ED at the EG by weighting the importance of each ED equally;
3. **Input-space Aware top-$\mathcal{K}$ Model (IAM)** represents the method of using the input space quantisation with the given prototypes $\mathbf{b}_k$ as a selection mechanism to find the optimal subset of the models to perform the ensemble query prediction at the EG;

4. **Input/Error-space Aware top-$\mathcal{K}$ Model (IEAM)** is extending the IAM methodology by further using the error-space. Therefore the entire statistics $\mathcal{C}_i$ are used to generate the ensemble query prediction inside the EG;

5. **Similarity-based selection Model (SBSM)** is introducing the method of clustering the EDs based on their similarity of the input data using DTW as metric and the hierarchical clustering to group the most similar EDs. The ensemble prediction for a certain query is generated by selecting the closest input prototype $\mathbf{b}_k$ and the EDs belonging in the same cluster.

4.6.4 Parameter Configuration

The previous methods and strategies highly depend on the defined parameters. In the following Table 4.1 all parameter configurations are listed. The threshold ρ_I for identifying the familiarity of the newly measured value $\mathbf{x}_t$ has been normalised for the input domain $[0,1]^d$, i.e., $\frac{\rho_I}{\sqrt{d}} \in (0,1)$. If the value of the familiarity threshold is close to 1, fewer prototypes of the input-space are generated with a broader quantisation. Whereas the value is towards 0 more prototype clusters are generated with a more detailed quantisation of the input space. The other threshold for deciding on model retraining given the previous input-space familiarity identification is ρ_O, which shows the familiarity with respect to the prediction ability. This parameter is set to be a portion of the MED_i, the median of the error differences $|e_i(\mathbf{x}) - e_i^o(\mathbf{x})|$ resulting into $\rho_O = \gamma MED_i$ for the initial value setting (highlighted in Algorithm 2). The final forwarding decision based on three parameters (θ_e, θ_{r2}, θ_w), depending on the chosen method. The values for θ_e decides on forwarding the model from the ED to the EG based on the predictive divergence of the models. The decision is made by choosing γMED_i. The value ranges for fitting and model parameter divergence as decision mechanism to forward the model is shown in the parameter setting with $\theta_{r2} = \{0.0000001, 0.0001, 0.001, 0.01\}$ and $\theta_w = \{0.01, 0.1, 0.5, 1\}$ respectively. For model selection inside the EG, the top-$\mathcal{K}$ parameter setting given with the values $\mathcal{K} = \{3, 5, 7\}$ are of importance. Further, the similarity-based clustering with using the linkage of average given the cost threshold values of $st = \{0.0017, 0.002, 0.0022, 0.0033\}$.

	Model-Retraining				Model-Forwarding			Model-Selection	
	Input-prototype Threshold	Error Tolerance Prototypes	Sliding Window Size	Learning Rate	Threshold Predictive Error	Threshold Fitting	Threshold parameter	Cluster Size	Hierarchical Cluster threshold
Notation	ρ_I	γ	SLW $\mathcal{W}$	α	θ_e	θ_{r2}	θ_w	$\mathcal{K}$	st
Value / Range	[0.01, 0.05, 0.1, 0.2]	[0.001, 0.005, 0.01, 0.05, 0.1]	[250, 500, 750, 1000]	[0.1]	[0.01, 0.001, 0.0001, 0.00001]	[0.0000001, 0.0001, 0.001, 0.01]	[0.01, 0.1, 0.5, 1]	[3,5,7]	[0.0017, 0.002, 0.0025, 0.0033]

Table 4.1: Parameter Setting for model forwarding and selection strategies

4.7 Performance Assessment

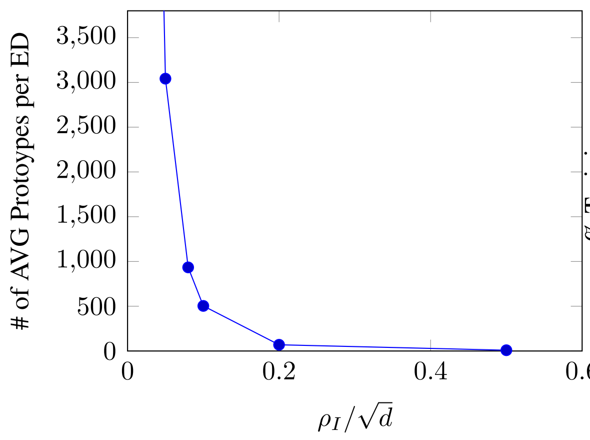
In the following section, the performance of the introduced methods and strategies is evaluated towards efficiency in terms of communication reduction and accuracy performance using the parameter settings and metrics presented above. This evaluation is divided into two assessments, the model-forwarding and settings at the ED i and the model-selection strategies at the EG over a given query $\mathbf{q}$ with the resulting prediction performance. The assessment is using the presented DS and cross-validation method of the randomly selected 24 time-points.

4.7.1 Model Forwarding Strategy

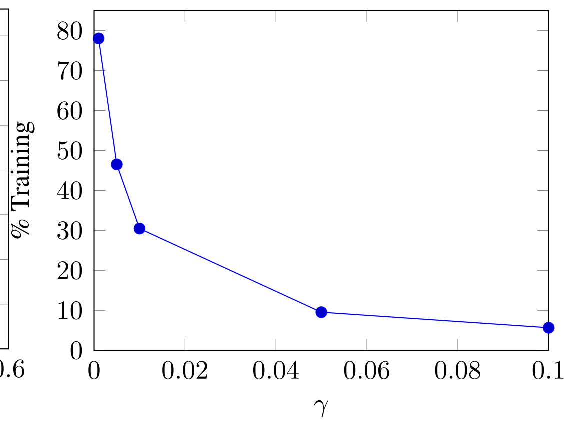
Section 4.4 highlights the model-forwarding strategies, which are analysed with respect to its performance in details throughout this section. The hypothesis of using a familiarity based retraining decision making that is based on input-space prototype quantisation inside the ED is analysed towards reducing communication and accuracy of prediction at the EG. Before comparing the model-forwarding mechanisms a sensitivity analysis is needed for the influence concerning the parameters and their impact on communication and accuracy.

In Figure 4.6 (a), increasing the radius of $\rho_I/\sqrt{d}$ to decrease the number of input-space prototypes $\mathbf{b}_k$ is presented. This figure shows that a value of $\rho_I/\sqrt{d} = 0.1$ results into approximately 500 input-space prototypes, whereas a value of $\rho_I/\sqrt{d} = 0.05$ increases the input-space quantisation towards $k = 3,000$. The value is the average number of prototypes per ED i in the introduced DS of 30 weather stations. Following this inside, Figure 4.6 (c) analyses the ratio of training the model inside the ED using the introduced familiarity based strategy and the number of times the model parameters $\mathbf{w}_i$ are sent to the EG using the error prediction forwarding method with threshold θ_e. This figure shows that the parameter ρ_0 and θ_e control the ratio between training and communication. Decreasing the threshold increases the ratio towards the optimum of 1. This optimum of 1 expresses that each time the model is retrained inside the ED based on the novel algorithm, it is also sent towards the EG. This avoids unnecessary retraining without sending.

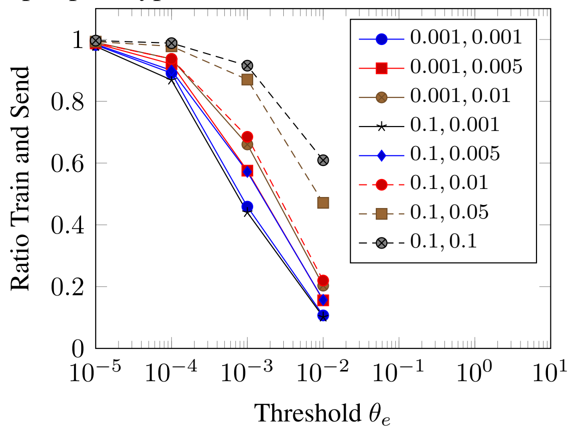
In Figure 4.6 (c), the importance of small θ-values is shown to guarantee a balanced ratio of training and sending and the significance of the value of ρ_0. Higher values with $\rho_0 \geq 0.1$ lead to less retraining and faster ratio convergence. This effect is further highlighted in Figure 4.6 (b). In this figure, the behaviour of ρ_0 represented by γ as $\rho_0 = \gamma MED_i$ shows the impact towards the amount of times the model is retrained inside the ED. It highlights that even with a small value of γ, the model's training can be decreased by 20%. The convergence towards no training inside the ED can be seen when γ tends towards 0.1 of the MED_i.



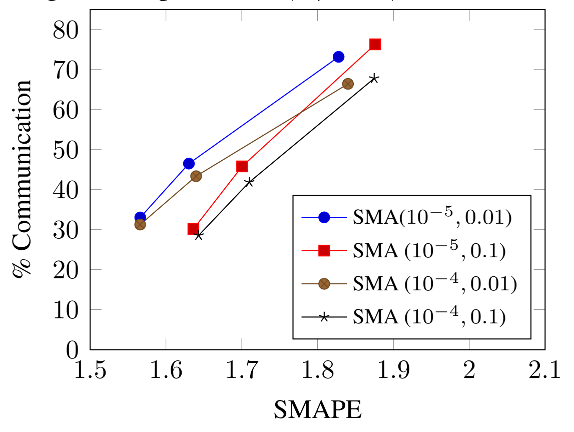
(a) Influence of $\rho_I/\sqrt{d}$ towards the average input-prototype number of each ED.



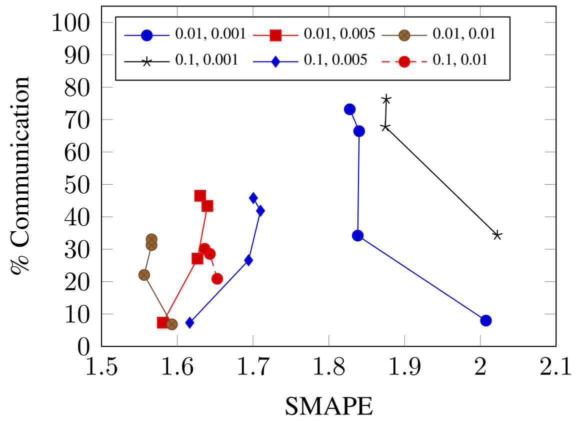
(b) Percentage of retraining the model at the ED given the parameter γ ($\rho_0 = \gamma MED_i$).



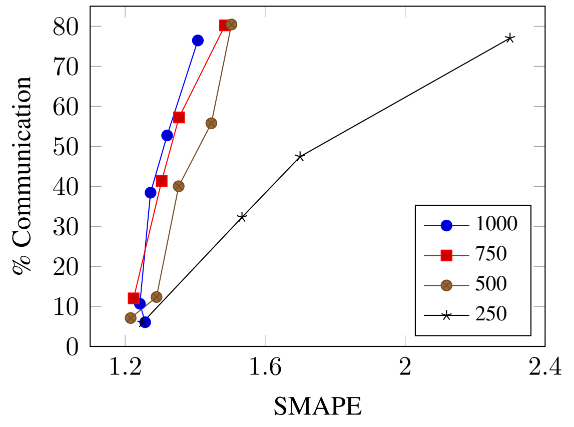
(c) Ratio of retraining and forwarding controlled by the error-tolerance ρ_0 and θ_e for the values of $\rho_I/\sqrt{d} = \{0.001, 0.1\}$.



(d) Communication accuracy trade-off for model selection SMA and $\rho_I = \{0.01, 0.1\}$ and $\theta_e = \{0.0001, 0.00001\}$.



(e) Communication accuracy trade-off for model selection SMA over increasing θ_e and fixed $\rho_I/\sqrt{d} = \{0.01, 0.1\}$ and $\gamma = \{0.001, 0.005, 0.01\}$.



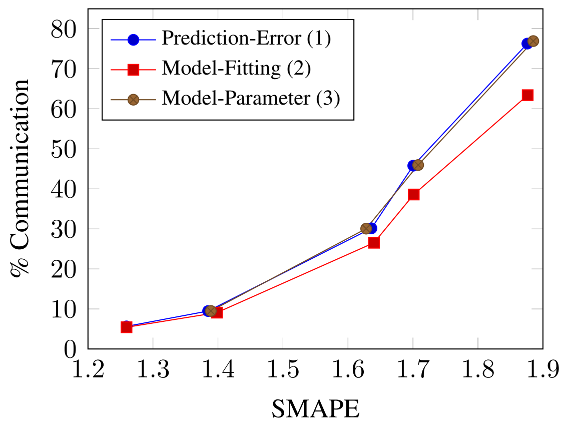
(f) Communication accuracy trade-off for model selection SMA over different SLW sizes M with $\rho_I = 0.05$ and $\theta_e = 0.00001$.

Figure 4.6: Evaluation of influence of different parameter settings at the ED i according to communication and retraining as well as its effect on accuracy for using SMA at the EG.

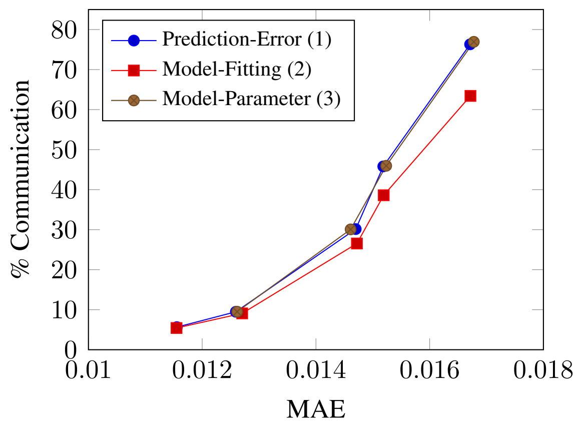
The impact of accuracy at the EG when using different values for the parameter settings of θ_e, ρ_I and γ is shown in Figure 4.6 (d). The accuracy for the chosen parameters are only considered for the SMA as model selection method inside the EG. All following figures in this subsection use SMA as model selection strategy measuring the accuracy of the model-forwarding methods. Further analysis between the different model-selection mechanisms can be found in the subsection that follows. In Figure 4.6 (d) the communication is controlled by γ with a fixed ρ_I and θ_e. Choosing the parameter for controlling the number of prototypes ρ_I to be small and using a small model-forwarding prediction error tolerance θ_e results in the best trade-off between communication and accuracy for considering only SMA as the model-selection method. However, choosing the value of $\rho_I = 0.01$ is developing an average number of prototypes $\mathbf{b}_k$ per ED of $k = 8,000$ whereas $\rho_I = 0.1$ only generates $k = 500$. Increasing the number of prototypes is require more storage capacity inside the ED and EG. Additionally, Figure 4.6 (d) shows that using a small threshold θ_e for model forwarding results in better accuracy than the larger values. Besides the value of γ, also the value for model-forwarding threshold θ is controlling the number of communication. This has been shown by the ratio training and sending in Figure 4.6 (c) but is further evaluated in Figure 4.6 (e). In the latter figure, the communication is fully controlled by the model-forwarding tolerance θ_e, in which the parameters of ρ_I and γ are fixed. The main insight of this figure is that θ_e is not influencing the accuracy at the EG. It is possible to have the same level of accuracy at the EG with 50% of remaining communication and the same with just 10%. This figure concludes that each θ_e, ρ_I and γ pair generates an upper-bound of accuracy level, which is independent of the communication frequency. Therefore, further analysis is assessed using the best training-sending ratio and controlling the communication by the value of γ with fixing θ_e to be small.

The last parameter influencing the communication and the accuracy is the size of the Sliding Window (SLW) $\mathcal{W}_i$. In Figure 4.6 (f), the influence of the size M towards the communication and accuracy is highlighted. The results of this figure are based on the parameter settings of $\rho_I = 0.1$ and using SMA as a model selection strategy. This illustration indicates that increasing the size of the SLW is improving the trade-off between communication and accuracy. More precisely, using 40% of communication with an SLW size of $M = 1000$ is resulting in a SMAPE of 1.28, whereas the SLW size of $M = 250$ is resulting in 1.6 for the SMAPE. The assumption of using large SLW sizes to improve the accuracy should be considered with the aspect of the storage capacity inside the ED to provide an appropriate efficiency towards battery and lifetime.

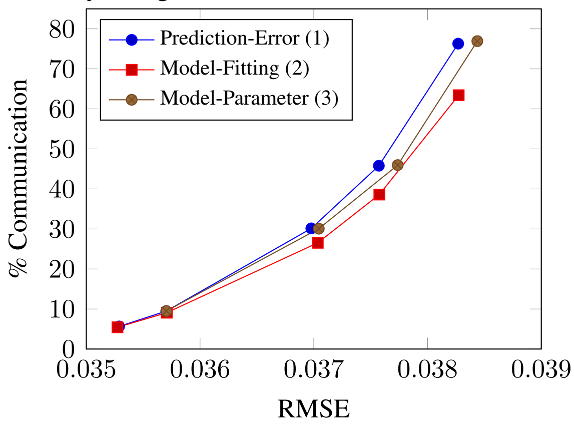
The setting and evaluation of the parameter values and their corresponding influence have been highlighted above. Choosing these settings, the proposed model-forwarding strategies are analysed. Given the three different model-forwarding strategies of prediction, fitting and parameters divergence as decision mechanism inside the ED, the aim is to identify which



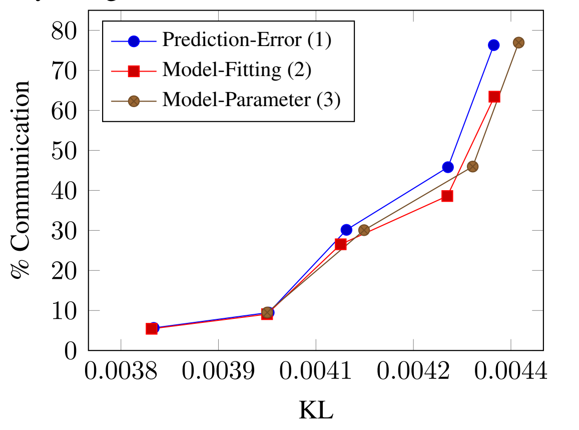
(a) Trade-off between communication and accuracy using the metric SMAPE.



(b) Trade-off between communication and accuracy using the metric MAE.



(c) Trade-off between communication and accuracy using the metric RMSE.



(d) Trade-off between communication and information loss using the metric KL divergence.

Figure 4.7: Evaluation of the proposed model-forwarding strategies at the ED and its influence on communication accuracy trade-off at the EG using SMA as selection method and the performance metrics SMAPE, RMSE, MAE and KL divergence.

methods prove to be most efficient by analysing the trade-off between communication reduction and quality of predictive results at the EG. At this point, only the forwarding mechanisms inside the EDs are investigated with respect to their performance, so that the model selection of SMA has been used for comparison only. Figure 4.7 shows the comparison of the three highlighted strategies using prediction, fitting and parameter divergence over the four performance metrics MAE, RMSE, SMAPE and KL. The setting of the parameters for the following analysis results is based on the previous of Figure 4.6. The size of the SLW $\mathcal{W}_i$ is set to be $M = 250$, due to limited storage capacity assumption and $\rho_I = 0.1$ for the retraining ratio. The error-prediction forwarding strategy is marked inside the illustrations with (1), the model-fitting with (2) and the model-parameter divergence strategy with (3). Comparing the three different strategies of model-forwarding, it can be seen in Figure 4.7 (a) that with respect to SMAPE, the model-forwarding method of error-prediction and model-parameter divergence generate similar results. The model-fitting method achieves with more communication the same accuracy and generates a lower trade-off consequently. Figure 4.7 (b) with the metric of MAE is equivalent to the results of the previous figure with SMAPE. In Figure 4.7 (c), with the performance metric of RMSE, the error prediction generates better results than the model-parameter divergence when increasing the remaining communication above 30%. Similar is the trade-off efficiency evaluation shown in Figure 4.7 (d) with the information loss represented by the KL divergence. The error-prediction model-forwarding strategy generates the n model-forwarding strategy.

Based on the analysis of the model-forwarding parameter settings and methodologies, comparison in the following performance assessment towards the model-selection at the EG is based on the error-prediction strategy using θ_e, and a relative small SLW size of $M = 250$ as the storage capacity is limited in IoT devices. Further, it should be noted that the values of $\rho_I = \{0.05, 0.1\}$ will be used and the communication reduction is controlled by γ only, using a fixed θ_e. This has been shown to generate the highest ratio between training and sending inside the ED with good accuracy results towards query predictions.

4.7.2 Qualitative Model Selection

The assessment of model-forwarding strategies inside the ED regarding the influence of parameter settings and accuracy level has been discussed in the previous section. The central focus of this section is to evaluate the qualitative model selection methodologies inside the EG towards efficient and qualitative query prediction. Efficiency is defined here with low communication overhead while maintaining or improving the accuracy level. The summary of comparative models has been highlighted in Section 4.6.3. The baseline for all introduced model selection strategies is the centrally generated model using the entire raw-data. Even though this is not an efficient and latency enabling methodology, the aim is to compare the methods against the accuracy of the central model for a full assessment. The comparison of all model selection strategies is against the Simple Model Aggregation (SMA), which means a simple aggregation of all models generated at the ED at time t, introducing the input-error space quantisation with the Input-space Aware top-$\mathcal{K}$ Model (IAM) and Input/Error-space Aware top-$\mathcal{K}$ Model (IEAM) and the extended similarity-based clustered method of Similarity-based selection Model (SBSM).

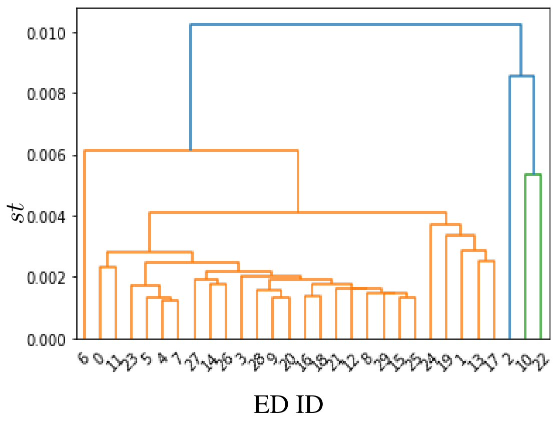
The methodologies of IAM, IEAM and SBSM require the setting of parameters that highly influence the accuracy-efficiency level. Therefore, in Figure 4.8 and Figure 4.9, the influence of the similarity threshold st for SBSM and selected cluster size $\mathcal{K}$ for IAM and IEAM are evaluated in detail. Figure 4.8 (a) shows the output of the dendrogram generated from the hierarchical cluster using Normalised Dynamic Time Warping (N-DTW) as the similarity function of each ED i. Based on this dendrogram, the similarity threshold st can be defined for generating the group $\mathcal{A} = \{A_1, \ldots, A_a\}$ in which each EDs i is mapped to a cluster. The detailed dendrogram in Figure 4.8 (a) indicates the range of the chosen similarity threshold of $st = [0.0017, 0.002, 0.0025, 0.0033]$. Highlighted is the strong dissimilar between ED $i = 2, 6, 10, 22$ towards all other EDs. Using the value of $st = 0.0017$ generates an overall of $a = 20$ cluster, using $st = 0.002$ only 15, with $st = 0.0025$ overall 11 and with $st = 0.0033$ the cluster size is reduced to 8. Connecting the information summarised in Table 4.2 with the dendrogram of Figure 4.8 (a) and the presented geographical information in Appendix B will provide an insight into the EDs relation. Overall it is accentuated from the dendrogram that some EDs are highly connected and similar in their measured input data $\mathbf{x}$. However, 4 to 5 evident visible groups can be seen from Figure 4.8 (a) with a very strong dissimilarity so that a value of $a \geq 4$ is appropriate.

		Station ID i																													
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
st	0.0017	1	14	20	11	3	3	17	3	10	8	18	2	10	12	5	10	9	13	9	15	8	10	19	4	16	10	6	7	8	10
	0.002	1	9	15	6	3	3	12	3	5	5	13	2	5	7	4	5	5	8	5	10	5	5	14	3	11	5	4	4	5	5
	0.0025	1	5	11	2	2	2	8	2	2	2	9	1	2	3	2	2	2	4	2	6	2	2	10	2	7	2	2	2	2	2
	0.0033	1	2	8	1	1	1	5	1	1	1	6	1	1	2	1	1	1	2	1	3	1	1	7	1	4	1	1	1	1	1

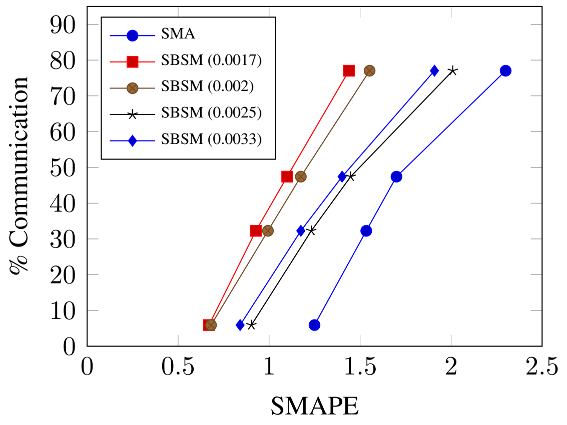
Table 4.2: Cluster allocation for different similarity thresholds st for each ED i at time instance $t =' 2018 - 02 - 05'$

Based on the information provided from the dendrogram, the communication-accuracy trade-off using SBSM over the selected four different similarity-thresholds st in comparison to SMA as model selection can be seen in Figure 4.8 (b). This figure shows a clear improvement with any selected value of st in contrast to SMA using as accuracy metric SMAPE. This figure is based on the model-forwarding strategy of the prediction-error threshold θ_e. Similar results are obtained with the other two model-forwarding methods but not shown due to limited space. Additionally, it is visible that choosing a small similarity threshold st with less EDs per group A results in better accuracy than using a larger number of EDs in one group indicated with higher values of st. This is seen for $st = 0.0017$ and a remaining communication of 40% in opposit to $st = 0.0033$ and the same amount of communication between ED and EG. The smaller st-value generates an improvement towards the accuracy of 0.5 in relation to the larger st-value. The best trade-off between accuracy and communication is seen for $st = 0.0017$. Therefore this similarity threshold is selected for all further performance assessments and model selection comparisons.

The assessment for the model selection methods IAM and IEAM depend on the parameter setting of the selected models $\mathcal{K}$. Figure 4.9 assess the performance between different values for $\mathcal{K}$ over the methods of IAM and IEAM. Figure 4.9 (a) highlights the impact of changing $\mathcal{K}$ on IEAM using the accuracy metric SMAPE. The results are compared against SMA using the parameter settings identified in the previous section. In this figure, it is possible to see that choosing the value of $\mathcal{K} = 3$ for IEAM improves the overall accuracy in contrast to the SMA model selection method. The values for $\mathcal{K} = \{5, 7\}$ only improve the accuracy below 50% of communication with respect to SMA. Figure 4.9 (b) uses the KL divergence as a metric



(a) Dendrogramm of hierarchical cluster over the DS and each ED i using the N-DTW as distance metric at time $t = 2018 - 02 - 05$.



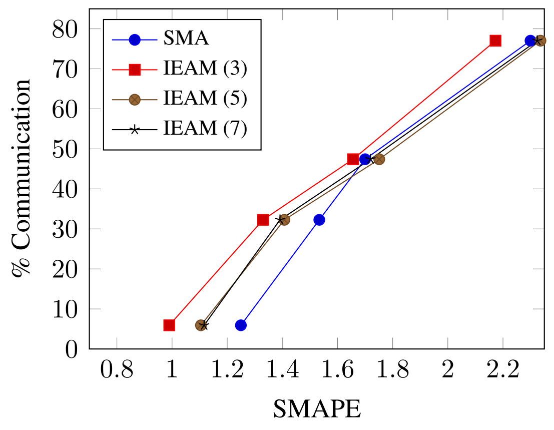
(b) Communication and accuracy trade-off using the SMAPE for different cost thresholds $st = \{0.0017, 0.002, 0.0025, 0.0033\}$.

Figure 4.8: Assessment for the similarity-bases model selection showing the dendrogram and cost-threshold st influencing the cluster group size a towards accuracy using SMAPE.

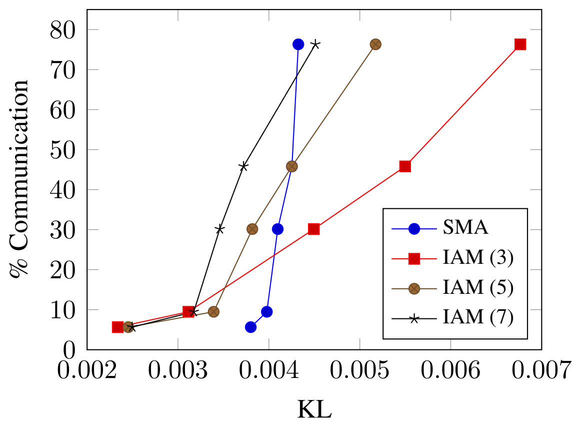
to identify the information loss between the different settings of $\mathcal{K}$ for IAM and the SMA. In this figure, the information loss for SMA is independent of the communication reduction and nearly constant at a value of 0.004. The information loss for IAM deviates considerably between the different values of $\mathcal{K}$ and a communication of more than 20%. Mainly, IAM with a value of $\mathcal{K} = 3$ shows immense information loss when the communication increases above 30%. The only value of $\mathcal{K}$ that improves the performance of SMA permanently is $\mathcal{K} = 7$. Interpreting these results as more EDs are used to generate the ensemble prediction for answering the query $\mathbf{q}$ the more information is lost, but higher accuracy is gained. For the remaining of the assessment, the value of $\mathcal{K} = 3$ is chosen for all accuracy metrics and $\mathcal{K} = 7$ for the information loss comparison. Dependent on the application, the focus can be shifted between accuracy or information loss.

After identifying the setting of the parameters for the SBSM, IAM and IEAM, the comparison between these methodologies and SMA as well as the global raw-data transferred model implementation is presented in the following figures. All following evaluations are based on the predictive error model-forwarding strategy with θ_e using $\rho_I = 0.1$ or $\rho_I = 0.05$, which has been identified as the best parameters in the previous assessment (see Section 4.7.1). Figure 4.10 shows the performance assessment over all four metrics, SMAPE, MAE, RMSE and KL divergence, using the selected model selection strategies.

Figure 4.10 (a) shows the impact between communication reduction and accuracy of all model selection methods with the metric of SMAPE. It is clearly visible that the global model using the raw-data results in the worst trade-off. All other proposed ensemble pruning model selection methodologies perform even with the same communication levels less than half of the prediction error. Overall the SBSM outperforms all other model selection strategies.



(a) Setting $\mathcal{K}$ for IEAM as model-selection using $\rho_I = 0.05$ and $\mathcal{K} = \{3, 5, 7\}$ showing the efficiency trade-off using SMAPE.



(b) Setting $\mathcal{K}$ for IAM as model-selection using $\rho_I = 0.1$ and $\mathcal{K} = \{3, 5, 7\}$ showing the efficiency trade-off using SMAPE.

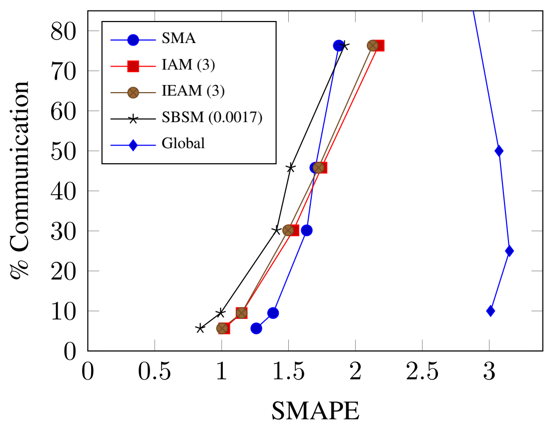
Figure 4.9: Influence of cluster size setting $\mathcal{K}$ for input-error space quantisation model selection IAM and IEAM towards accuracy-efficiency trade-off using SMAPE.

Followed by IEAM and IAM. If the communication is below 40%, the SMA outperforms the two model selection methods, IEAM and IAM.

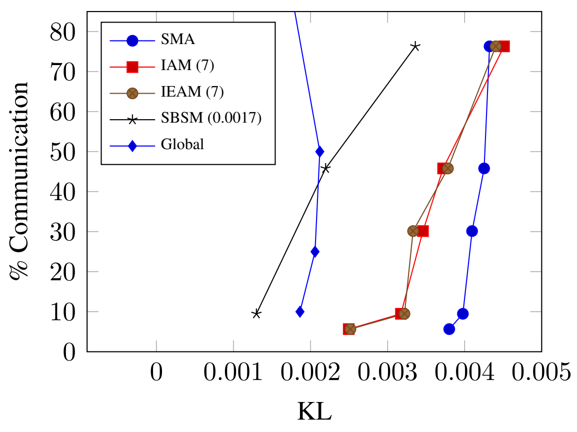
In Figure 4.10 (b) the trade-off between communication reduction and information loss using the KL divergence metric is highlighted. Noticeable is that the global model generates less information loss than any model selection method introduced in this chapter, except the SBSM. The SBSM method is developing a lower information loss when the communication falls below 40%. SMA is causing the most loss of information, followed by IAM and IEAM. The global model performs well in the KL metric as it transmits raw-data and can therefore capture relevant details of the data.

The trade-off with respect to the MAE for all model selection methodologies is shown in Figure 4.10 (c). Observable from this figure is when the communication is reduced below 60%, all model selection strategies proposed in this chapter perform better than the global generated model with raw data. The IEAM model generates less error with the same communication than the SMA. The IAM, however, only performs better than SMA when the communication is reduced below 40%. SBSM greatly outperforms all other model selection strategies. This method reduces with the same communication the MAE nearly four times compared to the other selection methods.

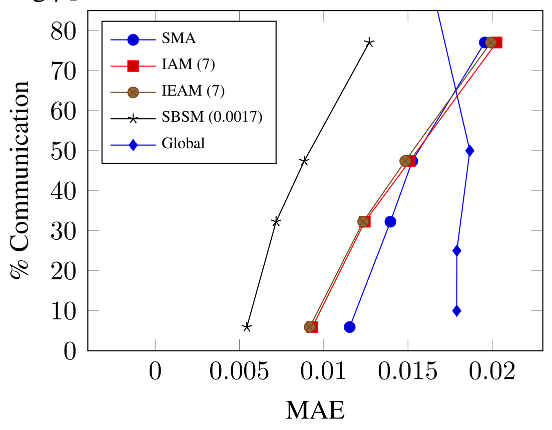
In the last Figure 4.10 (d), the trade-off for RMSE over all methods is compared. In this figure, the SMA can be defined as the worse model selection method. The global model outperforms all other strategies above a remaining communication of 50%. This turning point shows the improvement of SBSM, IAM and IEAM in comparison to the global model. All these proposed model selection strategies generate less RMSE with the same communication level than the global model. The ranking of the best trade-off is lead by the SBSM,



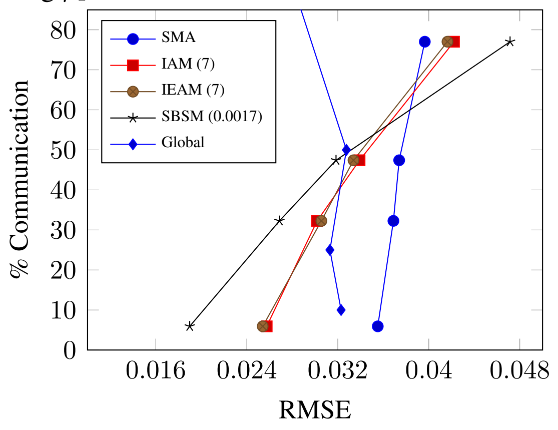
(a) Communication-Accuracy trade-off for SMAPE over all model selection methods using $\rho_I = 0.1$.



(b) Communication-Information loss trade-off for KL divergence over all model selection methods using $\rho_I = 0.1$.



(c) Communication-Accuracy trade-off for MAE over all model selection methods using $\rho_I = 0.05$.



(d) Communication-Accuracy trade-off for RMSE over all model selection methods using $\rho_I = 0.05$.

Figure 4.10: Evaluation of all qualitative model selection methodologies at the EG assessing the trade-off between communication and accuracy using the model-forwarding strategy of prediction-error.

followed by the IAM and IEAM. Both of these models generate similar trade-off results. SMA produces the highest RMSE with the same communication as all other methods.

Generally, in all of these four illustrations of Figure 4.10, the inaccuracy increases with more communication between ED and EG. The possible explanation of this is by assuming that with more communication, more variety of model updates is presented in the EG and therefore, outliers and fault measurements can place more importance on them. However, the global model with raw-data transfer stays constant while reducing the frequency of data forwarding.

4.8 Chapter Summary

In this chapter, the focus has been moved from simple data-forwarding strategies developed in Chapter 3 towards edge-centric analytics with intelligent model retraining and forwarding decision making in Edge Devices. Different strategies of model-forwarding methods based on prediction, parameter divergence and the fitting discrepancy have been evaluated in Section 4.7.1, showing the possibility to maintain accuracy at the EG while reducing the retraining and sending of model parameters and therefore communication overhead. Moreover, this chapter contributed to investigating the ability of EG to select from individual models collected the best subset to provide a qualitative ensemble prediction for a given query $\mathbf{q}$. Four different methods have been introduced and evaluated in Section 4.7.2. Given the performance assessment of this chapter, SBSM clearly outperforms all other models in terms of accuracy and information loss. IEAM only slightly improves the IAM strategy by the accuracy efficiency trade-off. SMA in contrast to the other three methods, only performs better for a communication level above 50%. The global model mainly generates higher errors with the same communication than all different model selection strategies. Therefore, it can be concluded that it is possible to enable local learning with model selection strategies at the central coordinator to intelligently adapt to concept changes and provide qualitative analytics, while reducing the communication and bandwidth by only transmitting model parameters and meta-data statistics.

Chapter 5

Privacy-Efficient Learning at the Edge

5.1 Chapter Overview

In the previous Chapter 4, edge-centric analytics at Edge Devices (EDs) is considered under the assumption that data can be shared between devices and Central Location (CL), e.g. the Edge Gateway (EG) or Cloud. Performing local analytics was adopted to overcome the issue of latency in continuously changing environments by performing real-time analytics at the ED. This enables the ED to adapt to concept drifts without relying on the central coordinator to notify or update the device. However, in the previous chapter, the input/error space prototypes representing the data is shared. With the assumption of data sharing, nowadays, applications are restricted by regulatory bodies. This prevents sharing data with other devices or forwarding data to a CL. Therefore, privacy enabling analytics involving EDs local machine learning ability is the fundamental motivation of this section. The previous section focused on real-time query-analytics from a user towards the CL, without considering the privacy of the data coming from the EDs. In this chapter, the focus is on keeping the data at the ED to preserve privacy by not sharing metadata, summaries about the data, or raw data over the network. Generally, this thesis aims on performing qualitative analytics and predictive modelling in edge networks while being efficient in terms of energy, storage, and bandwidth. Therefore in the continued chapter, a privacy-enhancing methodology in EDs is proposed that ensures qualitative and efficient local learning. Continue the research from the previous chapter on continuous evolving data coming from EDs, the following chapter enhances current privacy-preserving edge learning techniques (in concrete Federated Learning (FL)). Focus is using the advantages of personalisation discussed in Section 4.3 and the importance of generalisation in concept drift occurrences to introduce a weighting and selecting of a dual model implementation inside each ED. The following sections do not explicitly focus on the latency, but the described latency-efficiency from the previous chapter is also given with the introduced dual model implementation including the privacy aspect. It should be noted that the work presented in this chapter has been submitted as journal [3]. This chapter aims to answer the third hypothesis formulated in Section 1.2:

> *Hypothesis 3:* Generalised models over privacy-preserved data by only transferring analytical model parameters over the network will not provide qualitative results in constantly changing and heterogeneous environments. Using the prospect of locally learning models with an intelligent model selection and weighting mechanism for personalisation and generalisation in Edge Devices enables data privacy and qualitative prediction results.

5.2 State-of-the-Art on Privacy-Efficient local learning at the Edge

Privacy-preserving of user data has recently gained interest due to laws and regulations emerging, most popular are General Data Protection Regulation (GDPR) [164] and California Consumer Privacy Act (CCPA) [165]. Data security and privacy over Edge Computing environments have been summarised with open issues and topics by Zhang et al. [257]. The problem of transmitting raw data over the network has been engaged from a security perspective by applying different edge deployable encryption algorithms to the system. The aspect of efficient privacy-preserving techniques for resource-constrained devices has been highlighted as an essential issue for further research. The following chapter does not contribute to encryption or other security-related technical topics but contributes to efficient privacy-preserving machine learning and analytics techniques in edge networks. The focus relies on enabling edge device learning and training to preserve users' sensitive data from transmitting over the network to be centrally analysed. The aim is to keep the data at the location where it has been generated. This local learning and predictive analytics inside Edge Devices (EDs) can be seen as a possible solution towards efficient privacy-preserving analytics for sensible user data in edge environments.

Presented in Section 2.4, the methodology of Federated Learning (FL) has evolved as a possible strategy to engage EDs to learn a global machine learning function without revealing any data towards other devices or the Central Location (CL). For IoT applications, FL has been studied extensively using supervised learning and neural networks. FL aims to learn a global function located at a CL by pushing the training towards the device itself. The CL aggregates all locally trained models' gradients together into a new global function using FedAvg [189, 190]. This form of distributed learning provides the possibility to use the computational power of devices and their locally generated data, creating beneficial aspects, such as privacy, real-time actuation, and robustness. FL can be performed on massively distributed, non-i.i.d., and unbalanced data with fast convergence of the global trained model. The privacy aspect is implemented in FL by using the data of the ED only on the device itself and train the global model by sharing only the updated model weights over the network. The basis FL deployment does, however, allow possible security and privacy-related attacks. Not only is it possible to insert through a malicious user some data to poison the trained algorithm [155], but also to identify user data from the model updates sent over the network [156, 157]. Research on advancing the basic FL with security and privacy-enhancing techniques is an ongoing field [158, 159, 160]. Important FL can only provide complete security and privacy over any attacks when used with additional security, e.g. the encryption as mentioned earlier.

In many applications, generating a global model that is generalising the data is of utter importance. However, in many IoT systems, the generated data is non-independent and identically distributed (i.i.d.). This non-i.i.d. provokes personalised and local models to outperform the generalised global model mainly. This effect and the importance of local model deployment has been shown in the previous chapter (Chapter 4). Additionally, the authors in [188] argue that one of the open research questions is when to choose the global model, providing generalisation over the local, providing individuality, and vice versa. The following use case should emphasise the importance of having a global and an individualised model at the ED. A possible use case can be the function inside a vehicle to identify a speech command of a driver using audio data. The generalised model to identify the command needs constant improvement to new driving situations, languages and dialects. However, as each person is pronouncing and using the order of words differently, a local and personalised model will provide more accurate predictions than the generalised model. However, considering the event in which the driver is rapidly changing (car-sharing) or the voice has changed due to sickness of the driver. The in these cases, the generalised model will provide more accurate predictions than the local personalised model.

A focus on personalisation using the methodology of FL has only recently found attention in the research community, which the authors in [258] are summarising. The authors in [259, 260, 261, 262] show the impact of personalisation in FL environments and its significant improvement towards qualitative prediction results. The research in [259, 260, 261, 262] design a fully decentralised architecture in which no global server coordinates the communication or has knowledge about the model gradients or the overall generated global model. The architecture is arranged so that EDs collaborate with each other to share information and to improve their local model through other devices. This idea of decentralised personalised FL assumes complete connectivity between the different devices, which can not be assumed or realised in most IoT applications. Other methods proposed for personalised quality and privacy-efficient learning using FL rely on a hierarchical system order with a central coordinator performing the model merging. One of these related work is presented by Liang et al. [263]. The authors use representation learning over the data features at each individual learner (ED). Their work provides individuality and personalisation through having neural networks with a lower-level global model locally enriched by a personalised representation layer. Meta-learning for FL as a possibility to overcome the issue of heterogeneous data has been introduced by the authors in [264, 265]. They show the improvement in accuracy using personalised models in non-i.i.d. environments. The authors Hanzely et al. [266] proposed a FL optimisation that merges locally train models to a global model providing a specialised gradient update procedure. Moreover, the authors Deng et al. [267] present a combination of a globally learned model and a locally fine-tuned model. Deng et al. [267] introduced an adaptive parameter in controlling the relationship between global FL and local fine-tuned model. In the authors' proposed strategy, each device stores three models, the global, the local, and the personalised model. All of them have to be updated in each round of communication. Their adaptive parameter is calculated by the difference between the local and global model with respect to their gradient divergence.

The presented work of personalised and privacy-efficient local FL only considers stationary environments in which the model converges towards its optimum is true and reached over a short training period. Highlighted in Section 2.3.3.1, concept drifts and non-stationary environments, which are common in IoT deployments, require continuous learning and re-training of the model. Performing FL on developing data is still an open research question, and little to no research has been presented. The closest related work is from the authors in [268], proposing an online asynchronous version of FL. Their research suggests a convergence strategy of locally updating the FL model using feature representation learning at the CL and balancing with coefficients the relationship of previous gradients and current. Nevertheless, this work does not consider efficiency and resource-constrained environments. This aspect is considered in the work of Wang et al. [269]. They show the importance of local distributed learning in resource-constrained edge networks. The implementation of FL in a resource-constrained environment is presented in the work of Sattler et al. [184]. They contribute to efficient FL by providing a communication efficient compression technique. The most recent work of Li et al. [270] considers the essential requirement of using efficient communication methods in resource-constrained environments and the importance of personalisation in a combined implementation. Yet again, the previous two mentioned methods assume the global convergence of a model without concept drifts in their work. The later work of Li et al. [270] only improves communication by compression, sending fewer layers.

5.3 Rationale & Problem Fundamentals

Federated Learning (FL) can be seen as a distributed machine learning optimisation problem in a network of Edge Devices (EDs). The aim is to locally optimise an objection function $\mathcal{J}$ as defined in Equation (2.3) over distributed datasets as highlighted in Section 2.3.4. A network of $k \in \{1, \dots, k, \dots, K\}$ EDs is designed with each holding a local subset of data D_k so that $D = \{D_1, \dots, D_k, \dots, D_K\}$. This dataset D_k is generated through Sensing and Actuating Nodes (SANs) sensing continuously contextual data in form of a d-dimensional vector $\mathbf{x}_t$ with $\mathbf{x} \in \mathbb{R}^d$ and $t \in \mathbb{T} = \{1, \dots, T\}$ with $T \in \mathbb{T}$. Each time t a new contextual vector $\mathbf{x}_t$ is received at the ED. In centralised learning this data is forwarded to a CL that minimises the objective function $\mathcal{J}$ over the entire dataset D. The optimisation of the loss function $\mathcal{L}(\mathbf{x}, y; \mathbf{w})$ can be approximated by exploring the Empirical Risk Minimisation (ERM) instead of the expected risk. Therefore, an approximation of $\hat{\mathcal{J}}$ over a training set D of size N using a set of models $f \in \mathcal{F}$ with represented parameters $\mathbf{w} \in \omega$. ω describes the parameter space, can find the optimal solution for the ERM. This has been highlighted in Equation (2.3) and is summarised in the following equation:

$$\arg\min_{\mathbf{w} \in \mathcal{F}} \mathcal{J}(\mathbf{w}) \approx \hat{\mathcal{J}}_N(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}) \tag{5.1}$$

In FL the optimisation of the function $\mathcal{J}$ is performed by locally optimising a objective function $\mathcal{J}_k$ in each ED k over the local dataset D_k with length n_k so that $\sum_{k=1}^{K} n_K = N$. The CL is aggregating the local generated objective functions $\mathcal{J}_k$ as shown in Equation (2.10) so that:

$$\mathcal{J}(\mathbf{w}) = \sum_{k=1}^{K} \frac{n_k}{N} \mathcal{J}_k(\mathbf{w}) = \sum_{k=1}^{K} \frac{n_k}{N} \frac{1}{n_k} \sum_{i \in D_k} \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}) \tag{5.2}$$

The loss function $\mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w})$ or local optimisation function $\mathcal{J}_i(\mathbf{w})$ can be also noted as $j_i(\mathbf{w})$. One algorithm to solve the problem of Equation (5.2) is using the previous introduced Stochastic Gradient Descent (SGD). Each iteration t the ED k aims to converge towards the minimum loss function $\mathcal{L}$ over its local data D_k at time t given a new datapoint $\mathbf{x}_t$. This is by adapting the model parameters $\mathbf{w}_k^t$ with some factor η, called learning rate, and the gradient to the previous model parameters $\nabla j_k(\mathbf{w}_k)$. Each ED k is performing the following equation at time t:

$$\mathbf{w}_k^{t+1} \leftarrow \mathbf{w}_k^t - \eta \nabla j_k(\mathbf{w}_k^t) \tag{5.3}$$

After multiple iterations, the updated model parameters of each ED k are sent to the CL. The introduced FedAVG [189] algorithm aggregates the received model parameters inside the central coordinator towards the new generalised model. This averaging is summarised in

Equation 5.4.

$$\mathbf{w}^{t+1} = \sum_{k=1}^{K} \frac{n_k}{N} \mathbf{w}_k^{t+1} \tag{5.4}$$

After merging the local gradients of each ED k at the CL, the final federated model is distributed to the EDs and used locally for inference. Each ED k is selected at a random time epoch s with $s = \{1, \ldots, s, \ldots, S\}$ to update the distributed federated model f_{FL} with its local data stored in a Sliding Window (SLW) W_k^t and send the updated model parameters $\mathbf{w}_k$ back to the CL.

Highlighted in the State-of-the-Art section, the basic implementation of FL introduces multiple problems. The first issue is the adaption of the generalised model to constantly changing environments, in which the assumption of converging to a global minimisation of the objective function $\mathcal{J}$ does not hold. The second problem arises as the ED always overwrites the locally adapted model $\hat{f}_{FL}$ with a newly received generalised model f_{FL}. The issues of generalised models of non-i.i.d. data have been highlighted in Section 4.3, showing the importance of personalisation for qualitative analytics at CLs. Therefore, this chapter introduces the dual model deployment inside the ED k. Each ED k implements an evolving personalised model f_k in parallel to the generalised federated model f_{FL} locally. An overview of the constructed network architecture can be seen in Figure 5.1.

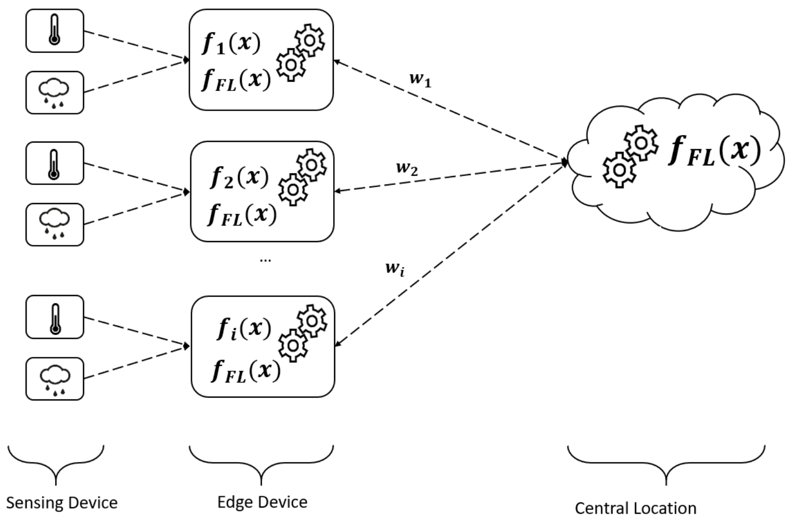


Figure 5.1: Architecture of the dual model deployment inside the ED k with a local personalised model and a generalised federated model.

The major challenge inside each ED k with the deployed dual model methodology is choosing the correct model for the prediction $\hat{y}$ with either $\hat{y} = f_{FL}(\mathrm{x})$ or $\hat{y} = f_k(\mathrm{x})$. The ED is limited with their resources, which does not allow complex algorithms to be implemented for decision making locally. Therefore, in the following sections, lightweight methods for balancing the personalised and generalised federated model inside each ED for qualitative analytical results are presented. Moreover, current deployed and tested FL methodologies consider a supervised learning environment. In many applications, e.g. autonomous cars or smart homes, the label to check the performance of the supervised learning algorithm is not provided. Therefore, systems and algorithms using continuous data prediction or forecasting with a FL deployment are of high interest in industrial implementation.

5.4 Personalised Privacy-Efficient Learning at the Edge

Highlighted in the State-of-the-Art section above, personalising Federated Learning (FL) models inside the Edge Device (ED) has gained tremendous interest in the last years to overcome the problems of heterogeneous non-independent and identically distributed (i.i.d.) data. The open issues of continuously changing data and concept drift adaption have been tried to overcome by using a local evolving personalised model inside the ED. In Chapter 4, the importance of local models inside EDs with selecting from an ensemble of models at the Central Location (CL) has been proven to generate qualitative and efficient analytics for query-driven predictions. Using local personalised models inside the ED and performing ensemble pruning strategies, as highlighted in Chapter 4, is due to restrictions of sharing user-specific models or meta-data to the CL impossible. The introduced general method of FL presented in Section 2.4 and Section 5.3 shows the process of generating a model on the CL by training on distributed datasets without accessing the raw data on each ED, so having the privacy of the data by design. However, the drawback of generalised models implemented in the FL setup can be overcome by local build models. This results in the problem that by only having a local personal model at the ED, the adaptation towards unseen data induced by concept shift will generate inaccurate and wrong predictions. Therefore the following work focuses on using the combined power of generalisation using the FL model and the advantage of the personalisation model inside each ED. This should provide qualitative prediction over changing environments in edge networks while being resource-efficient and reducing the communication and complexity. The basic implementation of FL has a complexity inside the EDs with $O(n_k(d+1))$ and at the CL with $O(k+d+1)$.

Having a parallel model deployment of the federated model and a personalised model inside the EDs requires first defining how and when to update both models and second which model to choose from for qualitative predictions at the ED. As presented in Section 5.3 at epoch s, the ED k is selected to receive the current generalised federated model f_{FL} from the CL. Then, the ED performs over its locally stored data in a SLW $\mathcal{W}_k$ the model gradient updates using Equation (5.3). These updates are forwarded to the CL and used into the new model f_{FL} combining all selected k EDs model parameter updates with FedAVG, as shown in Equation (5.4). All EDs receive the updated federated model after the merging and can use this as a model to perform the predictive analytics locally.

The personalised model f_k is set to be at $t = 0$ the centrally received federated model f_{FL}. The choice of using a pre-build model as starting point lies in the theories of transfer learning, [271, 272] which highlights that using a baseline model for retraining is generating an earlier convergence and higher accuracy than starting from scratch. In the majority of IoT or edge environments, SANs continuously measures the surrounding. The ED receives this data from the SANs at each time t with $t \in \mathbb{T} = \{1, \ldots, t, \ldots, T\}$ and $T \in \mathbb{T}$. Each time t a d-dimensional data vector $\mathbf{x}_t$ is collected and stored in the local window storage $\mathcal{W}_k$ of size M. Only at the selected epoch s this data is used to update the newly received model f_{FL}. If the ED needs to perform a prediction, it is using the previous cached model. The local model f_k is in compare updated and retrained each time t at the ED using the local SGD process presented in Section 2.3.3.2. The retraining methodology of Chapter 4 is additionally possible to use for energy-efficient retraining inside the ED but is not further considered in this section.

The aim is to present privacy-efficient federated learning methodologies improving the current work by deploying personalised aspects and balancing strategies inside the continuous changing environment of EDs. Therefore, in the following, the five strategies are illustrated contributing to personalised adaptive privacy-efficient learning in edge networks.

Evolving Federated Model (EFM):

The first method of incorporating personalisation into a privacy-preserving edge environment is advancing on the basic implementation of FL. The EFM introduces a communication-efficient strategy that discards the updating communication from the CL to the ED after generating the new generalised model f_{FL}. The ED k received at the selected epoch s the model f_{FL} and updates the model parameters using Equation (5.3), then it communicates these updated model parameters to the CL. Instead of receiving the new merged model f_{FL}, the ED keeps the updated model as f_k and continuously evolves the model each time t a new measurement $\mathbf{x}_t$ is collected. If the ED k is selected at another epoch s, the model f_k is replaced with the new f_{FL}. In this strategy, the communication is reduced, and the ED can adapt to the evolving data and concept drifts by continuously learning and retraining the generalised model to its personalised environment.

Local Federated Model (LFM):

The second introduced privacy-efficient FL based methodology is limiting the communication further between the ED and the CL. The concept is to initialise at time $t = s = 0$, the local model f_k with the received federated model, so that $f_k = f_{FL}$ inside each ED k. The ED is then continuously updating the model using SGD, see Equation (5.3). Instead of the CL requesting each epoch s the update of the distributed f_{FL}, the ED is regularly sending the local, model f_k to the CL. Inside the CL the model parameters of each f_k are merged into a global model f_{FL}. This can be distributed if the divergence is too large or if new devices join the network as the initial model. This type of personalised local FL has been partly introduced in the related work of [263]. The authors train local representatives of higher network layers and merging them centrally with the original deeper layers. No distribution of merged models is required anymore.

Adaptive Smoothing Model (ASM):

The following method of privacy-efficient personalised local learning introduces the combination of keeping two models in parallel inside the ED. As LFM and EFM only store one model and continuously retraining it, the ASM methodology supports the generalised model f_{FL} and the local model f_k inside each ED k. The importance of keeping both models inside the ED is mainly to overcome the issue of losing generalisation and the adaption to unseen data by only deploying personal models. Moreover, the previously unknown relationship between the devices in evolving systems can be changing from an i.i.d. to a non-i.i.d. data relationship. This cannot identify at the start of the application deployment and can cause better accuracy to the generalised model or personalised model depending on the connection. Using a balancing mechanism of the generalised and personalised model inside each ED can provide the benefits of both data relations towards qualitative prediction results. The first introduced this strategy are the author in [267]. However, their balancing of both models relies purely on the gradient difference between these two models. Not including previous predictions or any considering of adapting to changing environments. As this adaptation is of high importance to most time series analytics in IoT and edge environments, the ASM contributes to the research gap of using a reward system to weigh the local and federated model into a combined prediction $\hat{y}$. The final prediction $\hat{y}$ is calculated locally in each ED k by using the local personalised function f_k and the FL-function f_{FL} with a adaptive balancing weight α.

The adaptive weight is calculated through multiple steps. First, whenever the ED receives from the SANs a new contextual vector $\mathbf{x}_t$ at time t, the prediction error ϵ_k for the local, personalised model f_k with respect to the actual prediction y_t and the prediction error ϵ_{FL} for the federated model f_{FL} is calculated as shown in Equation (5.5) and (5.6) respectively.

$$\epsilon_L = |y_t - f_k(\mathbf{x_t})| \tag{5.5}$$

$$\epsilon_{FL} = |y_t - f_{FL}(\mathbf{x_t})| \tag{5.6}$$

Given these two errors of Equation (5.5) and (5.6) inside each ED k, it is possible to generate a reward value θ that is used as the source for balancing the two models f_k and f_{FL}. So in each ED k at each time t based on ϵ_{FL} and ϵ_L the reward θ is set to 0 if the local model is performing better than the generalised and $\theta = 1$ if the generalised model f_{FL} performs better predictions. This reward setting can be found in the following equation:

$$\theta_t = \begin{cases} 0, & \epsilon_{FL} > \epsilon_L \\ 1, & \epsilon_{FL} \leq \epsilon_L \end{cases} \tag{5.7}$$

As shown in Equation (5.7), a positive reward is given to the federated model f_{FL} if the absolute error ϵ_{FL} is smaller than the absolute error of the local model f_k ϵ_L. As the reward system deployed inside the ED should not only incorporate the current performance of f_k and f_{FL} but also the historical performance, the reward-values θ at each time t are stored in a Sliding Window (SLW) $\mathcal{O}_k$ with size U for the last $t-U$ times. The SLW is defined in Equation (5.8) and contains just zeros and ones.

$$\mathcal{O}_k^t = \{\theta_{t-U}, \ldots, \theta_t\} \tag{5.8}$$

The deployment of a SLW is chosen based on the literature and State-of-the-Art towards lightweight methods for handling continuous evolving data and adaptation to changing environments able to act immediately on concept drifts. This has been highlighted in Section 2.3.3.3. The SLW $\mathcal{O}_k$ consists of the most recent U rewards of $\theta \in \{0,1\}$. At each time t, the SLW of rewards is used as a reference to generate a ratio that represents the performance of both models over the time horizon $t-U$. This ratio is defining the adaptive weighting value α. The computation of α can be seen in Equation (5.9) and represents the historical performance of both models inside the ED.

$$\alpha = \frac{1}{U}\sum_{i=1}^{U}\theta_i \tag{5.9}$$

Each time the ED is performing a prediction, the two model predictions of f_k and f_{FL} are weighted to the final prediction $\hat{y}$. The adaptive value α is used to combine the federated model f_{FL} and the local personalised model f_k towards $\hat{y}$ using the theory of exponential smoothing [273]. The balancing of these two models is defined as shown in Equation (5.10):

$$f_{ASM} = \alpha f_{FL} + (1-\alpha) f_k \tag{5.10}$$

The value range of α is between 1 and 0. Setting $\alpha \longrightarrow 1$ more influence towards the FL model predictions coming from f_{FL}, whereas a value of $\alpha \longrightarrow 0$ places more importance on the locally generated model f_k predictions. The proposed adaptive model selection implementation is highlighted in the Algorithm 4, showing the steps of calculating the reward θ and ratio α, which is used to combine the generalised model f_{FL} and local personalised model f_k towards a weighted and more accurate analytical model inside each ED.

Algorithm 4 Adaptive Smoothing Model.

```
Central Location:
 1: initialise $\mathbf{w}_0$
 2: for each round $s = 1, 2, ...$ do
 3:     $i$ = random subset of $K$
 4:     for each ED $i$ in parallel do
 5:         $\mathbf{w}_i^{s+1} \leftarrow ED(i, \mathbf{w}_s)$
 6:         $\mathbf{w}_i^{s+1} \leftarrow \sum_{i=1}^{K} \frac{n_k}{N} \mathbf{w}_i^{s+1}$
 7:     end for
 8: end for

Edge Device: //Run on each Edge Device $k$
 9: $f_{FL} \leftarrow$ (received from CL at $t = 0$)
10: for each round $t = 1, 2, ...$ do
11:     contextual vector $\mathbf{x}_t$ is received
12:     $f_k \leftarrow$ updated via Equation (5.3)
13:     $\epsilon_L \leftarrow |y_t - f_k^t(\mathbf{x}_t)|$
14:     $\epsilon_{FL} \leftarrow |y_t - f_{FL}^t(\mathbf{x}_t)|$
15:     $\theta \leftarrow$ calculated as in Equation (5.7)
16:     $\mathcal{O}_t \leftarrow \theta$
17:     $\alpha \leftarrow$ calculated as in Equation (5.9)
18:     if $\hat{y}$ is needed then
19:         $f_{ASM} \leftarrow$ calculated as in Equation (5.10)
20:     end if
21:     if $s = t$ and ED $i = k$ then
22:         $f_{FL} \leftarrow$ updated as in Equation (5.3) or use $\nabla f_k$
23:         return updated $\mathbf{w}$ to CL
24:     end if
25: end for
```

Smoothing Model (SM):

Relying on the methodology of ASM, the next proposed privacy-efficient strategy inside the ED is SM. SM performs the balancing between local model f_k and general model f_{FL} using a fixed value of α. This value is not adapting based on historical predictions and does not change over the system's run time. It is, however, possible to define different α values for each ED. Based on the related work of exponential smoothing, the value is set to $\alpha = 0.3$ for more importance to the personal or $\alpha = 0.7$ with more significance to the generalised model.

Time-Optimised Switching Model (TOSM):

The method of ASM already introduced the idea of rewarding historical performance values towards balancing the local model f_k and generalised federated model f_{FL} by weighting the two predictions towards the final prediction $\hat{y}$. Relying on the concept of incorporating historical decisions with respect to the performance, the TOSM introduces the concept of Optimal Stopping Theory (OST). The design of TOSM is to overcome the problem of selecting between the two models f_k and f_{FL} inside the ED. Instead of weighting the predictions, just one model is chosen for the prediction $\hat{y}$ by identifying the optimal model at this time t.

The main concept of TOSM is to calculate inside each ED k at time t for each local model f_k and federated model f_{FL} the prediction error ϵ_L and ϵ_{FL} as defined in Equation (5.5) and (5.6) respectively. At time $t = s = 0$ the chosen model inside the ED is the f_{FL}. Given the two prediction errors ϵ_L and ϵ_{FL}, the idea is to decide when the best time t^* is to switch from the federated model f_{FL} to the local model f_L inside the ED. In Section 3.5.2, the theory of OST has been used through the reconstruction error difference for finding the optimal time to forward the raw data from the SAN to the ED. For TOSM the fundamental idea of Section 3.5.2 is used and performed on the prediction error ϵ_{FL} and ϵ_L as performance value. Each time t the two prediction error of f_k and f_{FL} are compared and rewarded as binary value Z. The binary variable Z is introduced as highlighted in Equation (5.11).

$$Z_t = \begin{cases} 0 & \text{if } \epsilon_L > \epsilon_{FL}, \\ 1 & \text{if } \epsilon_L \leq \epsilon_{FL}. \end{cases} \tag{5.11}$$

Z_t is defined as 0 if the local model f_k is generating a higher prediction error than the federated model. If the local model is, however, generating a lower prediction error ϵ_L in comparison to the federated model f_{FL}, the value of Z_t is set to be 1. Suppose the ED is deciding instantly based on the given comparison between ϵ_{FL} and ϵ_L at time t to switch the model. In that case, no historical behaviour is incorporated towards this decision. As previous behaviour and prediction quality are of importance to most applications, the values of the cumulative sum of comparison, including the history divergence of these two models, are used as switching decision. The instruct history of rewarded prediction performance comparison is then calculated by the cumulative sum of Z_t defined as R_t and introduced in the following equation:

$$R_t = \sum_{i=0}^{t} Z_i \tag{5.12}$$

In Section 3.5.2, proof has been given that by defining the reward function Y_t, it is possible to construct the optimal delay-tolerant level between SANs and EDs to forward data. In the previous chapter, the accumulated sum of reconstruction errors has been used as the reward function Y_t. Adopting this concept, the accumulated sum of Z_t defined as R_t is used to find the optimal time to decide when to switch from the federated model f_{FL} to the local model f_k. Therefore the reward function for switching f_{FL} to f_k is defined in Equation (5.13) using the quantity of R_t. The value of β indicates the delay tolerance level with $\beta \in (0, 1)$. If $\beta \to 1$ the delay is increased.

$$Y_t = \beta^t R_t = \beta^t \sum_{i=0}^{t} Z_i \tag{5.13}$$

Finding the optimal time t^* the reward function Y_t as in Equation (5.13) is used by maximising the expectation of Y_t with $\mathbb{E}[Y_t]$ having a fixed delay tolerance of β. Formally, this can be described as finding the supremum of the expectation of Y_t as, given the following equation:

$$\sup_{t \geq 0} \mathbb{E}[Y_t]. \tag{5.14}$$

Proof that an optimal time t^* exists has been given in Section 3.5.2 with the Equations (3.15) to (3.19). Based on this proof, the optimal time t^* for switching from the federated model f_{FL} to the local model f_k can be defined as:

$$t^* = \inf\{t \geq 1 | \sum_{i=1}^{t} Z_i \geq \frac{\beta}{1-\beta} \mathbb{E}[Z]\}. \tag{5.15}$$

The expectation $\mathbb{E}[Z]$ can be calculated with the formulation of Z_t in Equation (5.11) to the summation of the expectation $\mathbb{E}[Z|\epsilon_L > \epsilon_{FL}]$ and $\mathbb{E}[Z|\epsilon_L \leq \epsilon_{FL}]$ presented in the following equation:

$$\mathbb{E}[Z] = \mathbb{E}[Z|\epsilon_L > \epsilon_{FL}]P(\epsilon_L > \epsilon_{FL}) + \mathbb{E}[Z|\epsilon_L \leq \epsilon_{FL}]P(\epsilon_L \leq \epsilon_{FL}) \tag{5.16}$$

As the expectation of Z is given Equation (5.11) set to $\mathbb{E}[Z|\epsilon_L > \epsilon_{FL}] = 0$ and $\mathbb{E}[Z|\epsilon_L \leq \epsilon_{FL}] = 1$, so the overall expectation of $\mathbb{E}[Z]$ can be defined as:

$$\begin{aligned} \mathbb{E}[Z] &= P(\epsilon_L \leq \epsilon_{FL}) = P(\epsilon_{FL} \geq \epsilon_L) \\ &= 1 - P(\epsilon_L < \epsilon_{FL}) = 1 - F_{\epsilon_L}(\epsilon_{FL}) \end{aligned} \tag{5.17}$$

During a training period for the application the Probability Density Function (PDF) of ϵ_{FL} and ϵ_L can be obtained, which leads to generate the optimal time t^* for switching the federated model f_{FL} to the local model f_k to:

$$t^* = \inf\{t \geq 1 | \sum_{i=1}^{t} Z_i \geq \frac{\beta}{1-\beta}(1 - F_{\epsilon_L}(\epsilon_{FL})\}. \tag{5.18}$$

The introduced methodology of TOSM is further considering the reverse switching from the local model f_k to the generalised model f_{FL}. Therefore, once the model is switched from the federated model to the local model, the time value of R is set to 0. Then the concept of finding the optimal time t^* to switch back to the federated model with using the same concept as previous is starting inside the ED. The optimal time to switch from the local model f_k to the generalised model f_{FL} is made through the variable Q in which the prediction errors ϵ for both models are monitored. Similar to the variable Z_t, Q_t is calculated as:

$$Q = \begin{cases} 0 & \text{if } \epsilon_{FL} > \epsilon_L, \\ 1 & \text{if } \epsilon_{FL} \leq \epsilon_L. \end{cases} \tag{5.19}$$

From Equation (5.19), the expectation of Q using the ideas of Equation (5.17) and (5.18) from lead to:

$$\begin{aligned} \mathbb{E}[Q] &= \mathbb{E}[Q|\epsilon_{FL} > \epsilon_L]P(\epsilon_{FL} > \epsilon_L) + \mathbb{E}[Q|\epsilon_{FL} \leq \epsilon_L]P(\epsilon_{FL} \leq \epsilon_L) \\ &= P(\epsilon_{FL} \leq \epsilon_L) = P(\epsilon_L \geq \epsilon_{FL}) \\ &= 1 - P(\epsilon_{FL} < \epsilon_L) = 1 - F_{\epsilon_{FL}}(\epsilon_L) \end{aligned} \tag{5.20}$$

From the given expectation of Q in Equation (5.21), it is possible to derive the optimal time t^* to switch from the local f_k back to the generalised f_{FL}. Only the Probability Density Function (PDF) of the error values ϵ of the models f_k and f_{FL} is needed and can be generated through a training period of the application inside each ED. In Equation (5.21), the optimal time t^* for the switch is defined. Once the model is switched back to the generalised f_{FL}, the summation of values of Q is reset, and the Z accumulation is starting. This method is performed until a new federated model f_{FL} is sent from the CL, replacing the old one. This provokes a reset and a starting of using Z with the current chosen model f_{FL}.

$$t^* = \inf\{t \geq 1 | \sum_{i=1}^{t} Q_i \geq \frac{\beta}{1-\beta}(1 - F_{\epsilon_{FL}}(\epsilon_L))\}. \tag{5.21}$$

5.5 Performance Evaluation

5.5.1 Experimental Setup

The experiments for this section are based on the already introduced dataset (DS) in Section 3.6.1 and Section 4.6.1. Overall 415 weather stations are selected over a time horizon of around 87 days (December 2017 till March 2018). Each weather station represents a Edge Device (ED) k with $k \in K = \{1, \ldots, k \ldots, K\}$ so that $K = 415$. Each time t with $t \in \mathbb{T} = \{1, 2, \ldots\}$ the ED received a d-dimensional input data vector $\mathbf{x}_t$. The DS provides a 9-dimensional data in form of $(\mathbf{x}, y)$ each time t including temperature, dew point, humidity, wind-speed, wind-gust, wind direction, pressure, windchill, and precipitation. The value for y is set with the DS's measurement of temperature, while the remaining measurements are used for $\mathbf{x}$. The data collection frequency is every 5 minutes over the time horizon of 100 days, resulting in a dataset size of $N = 9,044,683$, assembling roughly 250 values measured per ED per day. All data is normalised and scaled, i.e., each parameter $x \in \mathbb{R}$ is mapped to $\frac{x-\mu}{\sigma}$ with mean value μ and variance σ and scaled in [0,1], thus vector $\mathbf{x} \in [0, 1]^d$.

Similar to the edge analytics task in Chapter 4, the setup for the federated learning is the introduced multivariate Linear Regression Model (LM) for $f(\mathbf{x})$. A training period of 1 month is considered before testing the proposed methods of this section. This splitting of DS of size $N = N_T + N_M$ results in $N_T = 1,500,000$ data points for the training period and leaves around $N_M = 7,500,000$ data points for performing the proposed privacy-efficient approaches. Converting this into a percentage, only 15.8 % of the collected values are used during the training period to overcome the cold-start problem using transfer learning, and over 84% is used for testing the different approaches. The starting date is located on the 1st of January 2018 and presents the time $t = s = 0$. During the training period, each ED k generates a local model f_k. At time $t = 0$ each ED sends its local model f_k towards the CL. Inside the CL, these models are merged using FedAVG as defined in Equation (5.4) towards the first generalised central model f_{FL}. At time $t = s = 0$ this generated f_{FL} is distributed to the EDs as first federated model f_{FL}.

5.5.2 Performance Metrics

The performance for the presented DS is evaluated with the performance metrics introduced in detail throughout Section 3.6.2. More specifically, the metric Root-Mean-Squared-Error (RMSE), Mean absolute error (MAE) and Symmetric mean absolute percentage error (SMAPE) are used for accuracy performance evaluation. For analysing the information loss between each of the proposed methodologies, the metric of Kullback-Leibler (KL) divergence is used in performance evaluation. In comparison to the previously used communication metric, the communication is constantly independent of the parameter settings and is given with the introduced methodology in this chapter.

5.5.3 Comparative Assessment

A baseline for comparison is needed to evaluate the performance of the privacy-efficient analytics methodologies proposed in this chapter. Therefore, during the assessment, the performance of the **Global Model (G)** with transmitting raw data to the CL from each ED at each time t is used as a comparison model towards the others. Concentrating on the privacy aspect of local machine learning and analytics, further methods are the basic deployment of Federated Learning represented during the assessment with **Federated Model (FM)**. The contrary is the other deployment of using just the local model **Local Model (L)** without any usage of the central coordinator or generalised model. The local model is built from stretch without any previous model coming from the CL.

Given these three baseline methods, they are compared against the in this chapter introduced methodologies of personalised privacy-efficient learning:

1. The **Evolving Federated Model (EFM)** uses as initial local model f_k the at time $t = 0$ received generalised model f_{FL}. At each selected epoch s, the CL requests the local model and merges the generalised f_{FL} over them. Only the model f_k is stored inside the ED.

2. The **Local Federated Model (LFM)** is extending the EFM by updating the generalised model f_{FL} locally at each time t until the new update from the CL is sent at epoch s.

3. The **Adaptive Smoothing Model (ASM)** introduces a dual parallel model implementation inside each ED. A local model f_k is set to $f_k = f_{FL}$ at $s = t = 0$ and continuously updated each time t. The generalised model f_{FL} is received and updated as the FM. The final prediction is generated through a reward function with respect to the historical performance of each model and balancing the f_k and f_{FL} model through the parameter α.

4. The **Smoothing Model (SM)** is based on the same concept as the ASM but with a fixed value for the balancing weight α.

5. The **Time-Optimised Switching Model (TOSM)** provides a selecting mechanism of the optimal model for this time. The fundamental concept is based on finding the optimal time t^* to switch between the two models f_k and f_{FL} and vice versa using Optimal Stopping Theory (OST).

5.5.4 Parameter Configuration

Assessing the accuracy, multiple parameters for the different introduced methods and ED storage capacities have to be set. After the starting point at $t = 0$ each time t, the models of

EFM, LFM, L are updated using SGD. The values for θ are inserted into the SLW $\mathcal{O}$ each time t for the proposed ASM strategy. The adaptive weighting α is generated through the ratio over this SLW $\mathcal{O}$ of size U. In the assessment U is set to $U = \{50, 100, 250, 500\}$. Moreover, for the TOSM method, the Z_t and Q_t values are defined each time t and the respective current optimal model flagged inside each ED k. The delay tolerance level for the TOSM strategy is defined with β and set to be $\beta = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. If the CL requests at epoch s an update of the FM, the values inside the SLW $\mathcal{W}$ are used for the SGD. The assessment parameters for the SLW $\mathcal{W}$ of size M is set to be the previous day and the time since the last update was requested, so $M = \{250, 500, 1000\}$. The request epoch s is set for the assessment to be every day, every other day and every fourth day so that $S = \{74, 37, 18\}$ and at the times $t = \{s \cdot 250, s \cdot 500, s \cdot 1000\}$ respectively.

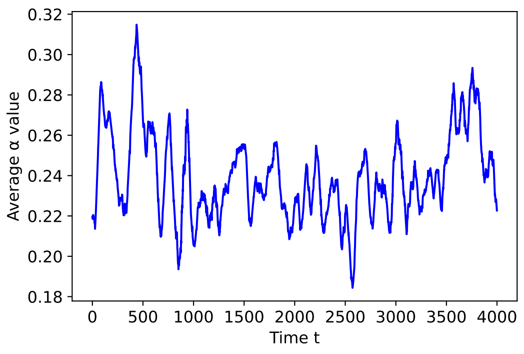
Assessing the overall performance of each proposed method, a type of cross-validation has been deployed to guarantee independent validation of the results. Similar to the validation method of Section 4.6.4, the application is stopped at 24 random time points t. At the selected time t the methodology implemented is stopped and the next 250 values (representing one day) of each ED k is used as prediction input to analyse the performance of the different privacy-efficient analytics strategies. In detail, the following time points have been chosen: 2018-01-02, 2018-01-05, 2018-01-12, 2018-01-22, 2018-01-25, 2018-01-29, 2018-02-01,2018-02-05, 2018-02-07, 2018-02-08, 2018-02-09, 2018-02-11, 2018-02-14, 2018-02-16,2018-02-17, 2018-02-18, 2018-02-19, 2018-02-20, 2018-02-21, 2018-02-22, 2018-02-23,2018-02-24, 2018-02-25, 2018-02-26, 2018-03-03, 2018-03-07.

In the previous sections, the importance of deploying a parallel model setup in EDs having the generalised model f_{FL} and a personalised model f_k both running towards quality-aware predictive modelling has been argued based on the possible concept drifts and evolving nature of IoT environments. To assess this behaviour and show the effectiveness of having both models and certain techniques to weigh or select the optimal model is made in the performance evaluation through using the above mentioned time points and chosen at random data from another weather station (ED) to perform the predictions using the different approaches introduced. This artificial concept drift can provide insights into each model's behaviour towards unseen and unfamiliar data coming from another weather station.

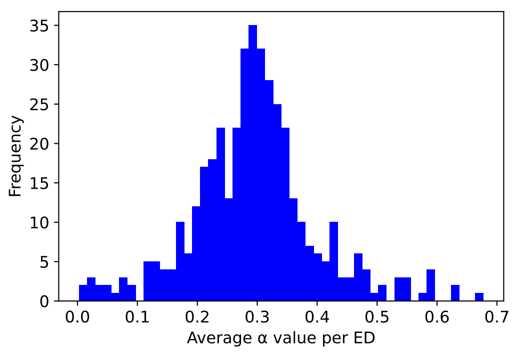
5.6 Performance Assessment

5.6.1 Personalised Privacy-Efficient Learning

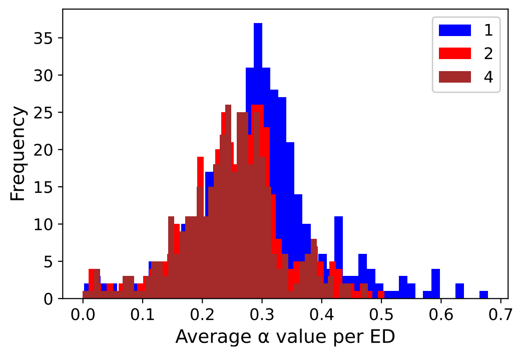
The assessment of the previously introduced methods is highly dependent on the parameter settings explained in Section 5.5.4. This section starts with analysing the behaviour of α indicating the weighting between local model prediction using f_k and the generalised federated model f_{FL} for the weighted prediction $\hat{y}$ using the proposed method of ASM.The influence of s (epoch of CL request and updates the f_{FL} model) and U (SLW $\mathcal{O}$ size of considered rewards θ) to the model weighting parameter α using the introduced performance metrics are illustrated in Figure 5.2.



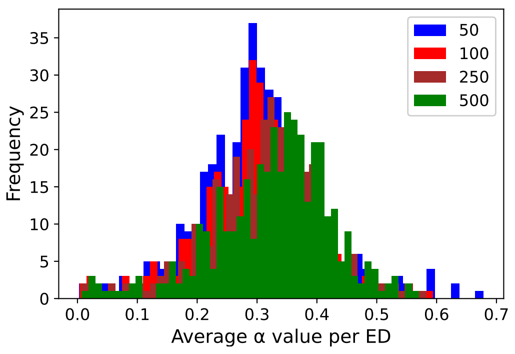
(a) Average value of α over time t.



(b) Histogram of average α in EDs.



(c) Histogram of average α in EDs depending on the setting of the federated update frequency s.



(d) Histogram of average α in EDs depending on the setting of SLW $\mathcal{O}$ of different length U.

Figure 5.2: Parameter setting influence on α for the Adaptive Smoothing Model (ASM).

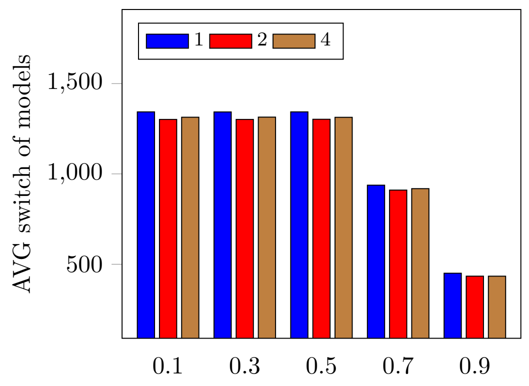
In Figure 5.2 (a) the average value of α over all EDs K until $t = 4000$ is presented. It can be seen that the value of α is greatly changing over time but lies in the range of $\alpha \geq 0.18$ and $\alpha \leq 0.32$. This figure has been using the setting of $U = 50$, $M = 1000$, and s to be every day with $S = 74$. The same settings are used for the frequency analysis and α distribution over each ED $k \in K$ illustrated in Figure 5.2 (b). This figure shows the distribution of average α values for each ED and can be identified as a normal distribution with the mean around $\alpha = 0.3$. This correlates with the findings of Figure 5.2 (a) for the average α-values per time instance t over the entire EDs K lying in the highlighted range. To identify the influence of s and U towards the weighting parameter α, Figure 5.2 (c) and Figure 5.2 (d) highlight this respectively. In Figure 5.2 (c), the setting of $s = \{1, 2, 4\}$ indicating the update frequency of the federated model f_{FL} to be each day, every second day, and every fourth day. From this figure, the influence of the update frequency s towards the model weighting α indicated that increasing the frequency is decreasing the mean of α and increases the variance. In opposite to the influence of U to α, illustrated in Figure 5.2 (d). In this figure, an increase of the mean is presented by increasing the SLW size of rewards to be considered. The rise of U is only influencing the mean but not the variance for the value α in each ED k.

The next proposed model in which the parameters have to be first analysed is the TOSM. Introducing the value of β for the delay tolerance of switching the models f_k and f_{FL} to use the optimal model at the prediction time. This value of β is analysed in Figure 5.3 with the number of model switches that occur during the runtime. In Figure 5.3 (a), the influence of switching the models dependent on β with increasing the update frequency of s is illustrated. In this figure, no difference between the variation of s can be seen influencing the model switching of the TOSM strategy. However, a decrease of average switches can be seen when the value of $\beta \geq 0.7$. Similar results are highlighted in Figure 5.3 (b). This illustration shows the influence of increasing the SLW $\mathcal{W}$ size M with respect to β and the number of times the model is switched inside the ED. Figure 5.3 (c) shows the distribution of $\beta = 0.1$ and the number of switches inside each ED and the distribution of $\beta = 0.9$ with delaying the switching. In $\beta = 0.9$, the variation of average switches is higher than with $\beta = 0.1$ in which the density is around the mean of 500. In Figure 5.3 (d), the distributions of $\beta = 0.1$ using different values of M are presented. Already shown in Figure 5.3 (b), no change of the frequency by differing the size of M can be seen.

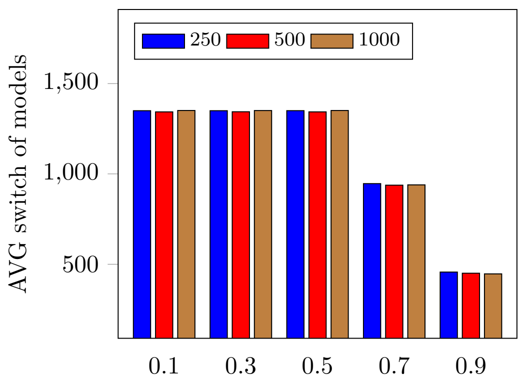
In Figure 5.4, the performance for RMSE, MAE, SMAPE and KL divergence over the different possible β values is analysed. Highlighted in the previous figure, increasing the delay of switching between the models f_k and f_{FL} inside the ED indicated through the value of β shows that $\beta \to 1$ increases the tolerance and less often the model is switched.

In Figure 5.4 (a), the analysis of the performance of RMSE with respect to the update frequency s is illustrated. It can be seen that the update frequency of s is only slightly increasing the error represented by the metric RMSE. Similar results have been generated using MAE and SMAPE, but these results are removed to space limitations. Moreover, in this figure, the behaviour of s towards the accuracy is illustrated. Using $s = 1$ indicating an update frequency of the f_{FL} model every day and results in the lowest accuracy error. However, assuming that increasing the frequency is rising the error does not hold as the performance of $s = 4$ (update frequency every fourth day) generates lower prediction errors than $s = 2$. The information loss metric KL divergence highlighted in Figure 5.4 (b) indicate a clear decrease of information loss by increasing the update frequency s. Additionally, the influence of β towards the information gain shows that using more frequent switching of models results in higher entropy, presented by the KL metric, inside the ED.

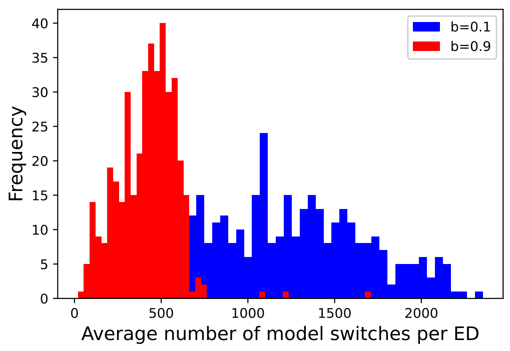
Figure 5.4 (c) and Figure 5.4 (d) investigate the behaviour of increasing the SLW $\mathcal{W}$ of size M representing the data used for performing SGD on at time $t = s$ for updating the f_{FL} to



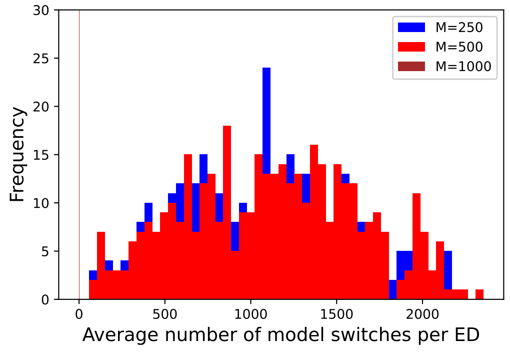
(a) Average switching time for different β and federated update frequency s.



(b) Average switching time for different β and SLW sizes for M.



(c) Histogram of average switching time for $\beta = \{0.1, 0.9\}$ using $M = 250$.

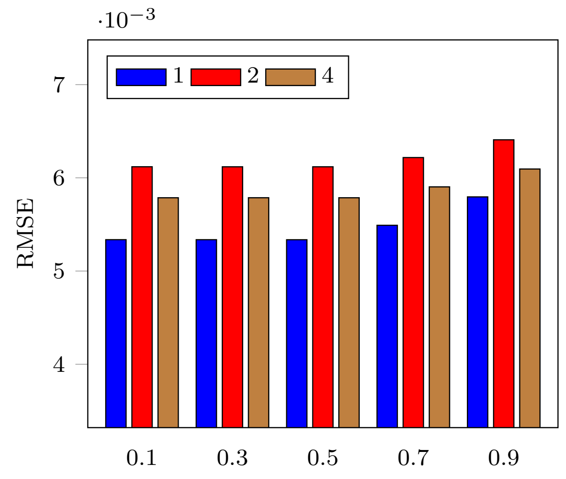


(d) Histogram of average switching time for $\beta = \{0.5\}$ using $M = \{250, 500, 1000\}$

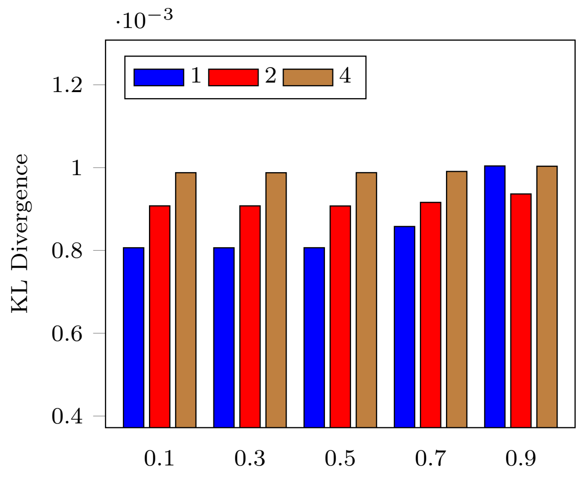
Figure 5.3: Parameter setting influence on β for the Time-Optimised Switching Model (TOSM).

the CL. In Figure 5.4 (c), the influence is presented by the metric MAE. Increasing the size M to $M = 1000$ is decreasing the error. The dependency of β with respect to the accuracy over different SLW sizes M does show a slight increase of the MAE when increasing the delay tolerance of $\beta \rightarrow 1$. In Figure 5.4 (d), similar behaviour and dependency are illustrated. Raising $\beta \rightarrow 1$ is increasing the SMAPE and increasing the SLW $\mathcal{W}$ of size M to $M = 1000$ is decreasing the prediction error.

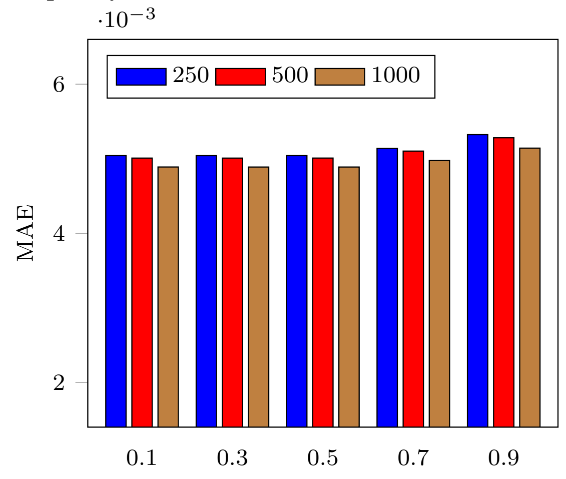
In Figure 5.5, the performance of all models introduced in Section 5.5.3 and the influence of the parameters s and M is illustrated for assessment. In Figure 5.5 (a), the metric of RMSE for $M = 250$ and $U = 250$ over all epoch selection variables s is highlighted. In this figure the influence of s towards the models FM, ASM, SM and TOSM is illustrated. The models G, L, EFM, and LFM do not change their performance by changing the the parameters of M or s as their learning and adaptation to the data input is continuously and independent of the FM. Increasing the frequency of s is increasing the prediction error for the method SM and ASM. Whereas the accuracy is increasing for the FM method. The TOSM approach is as indicated in Figure 5.4 only slightly increasing and performing worst for $s = 2$, representing



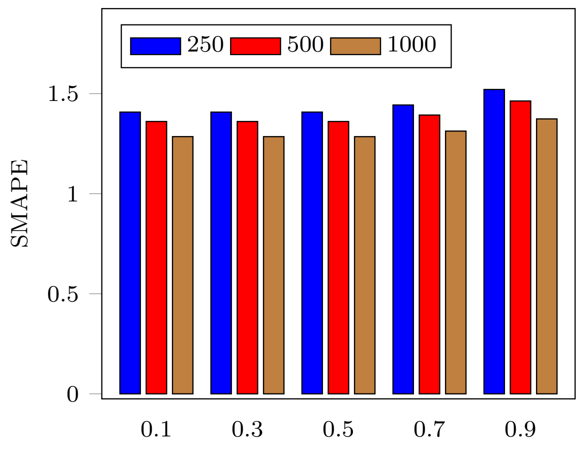
(a) Average RMSE for different β and update frequency s.



(b) Average KL for different β and update frequency s.



(c) Average MAE for different β and SLW sizes for M.

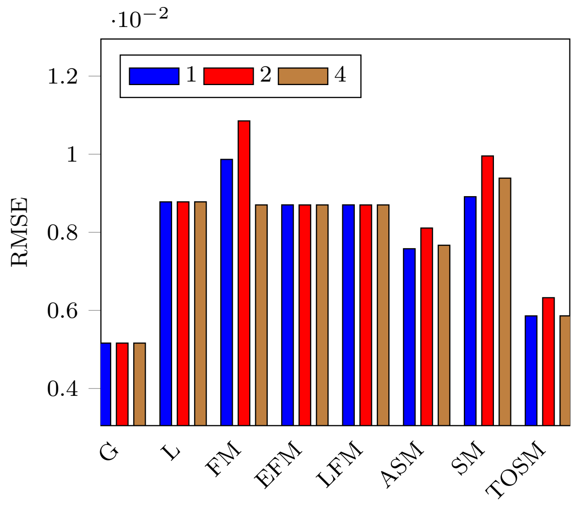


(d) Average SMAPE for different β and SLW sizes for M.

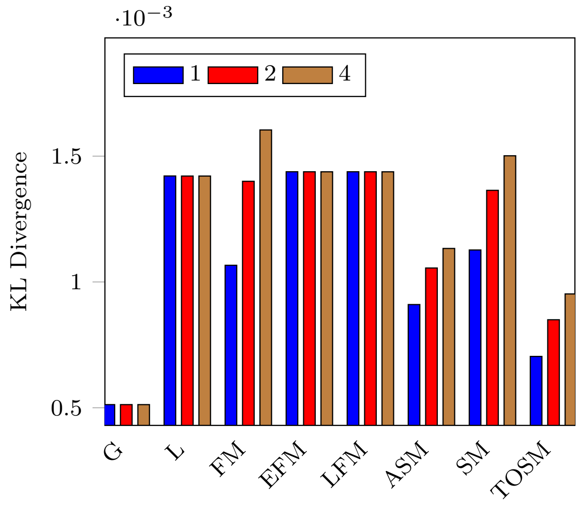
Figure 5.4: Comparison of influence of delay tolerance β for TOSM using different s and M towards the performance metrics KL, RMSE, MAE, and SMAPE.

updating every other day. Further, this figure shows that the G generates the best accuracy, but the TOSM strategy provides accuracy closest to the G. In Figure 5.5 (b) the information loss over changing update frequencies s over all models is highlighted. Using the setting of $M = 1000$ and $U = 50$ with the performance metric KL divergence, the results show that for the models depending on these parameters (FM, ASM, SM and TOSM) the KL is increasing with the increase of the update frequency s. A greater information loss can be seen for all four affected methods comparing $s = 1$ and $s = 4$ with each other. Moreover, the information loss is smallest when transmitting raw data and generating a centralised model (G).

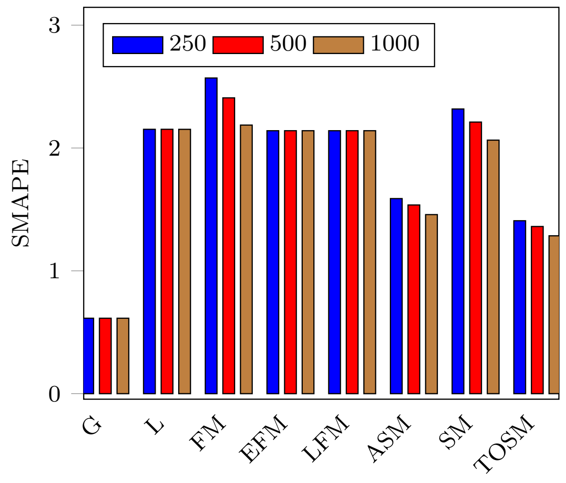
Figure 5.5 (c) and Figure 5.5 (d) highlight the performance towards changing the SLW $\mathcal{W}$ size M and the corresponding behaviour of each model. Setting the parameters for Figure 5.5 (c) with $s = 1$ and the SLW $\mathcal{O}$ size $U = 50$ using the metric SMAPE shows the following



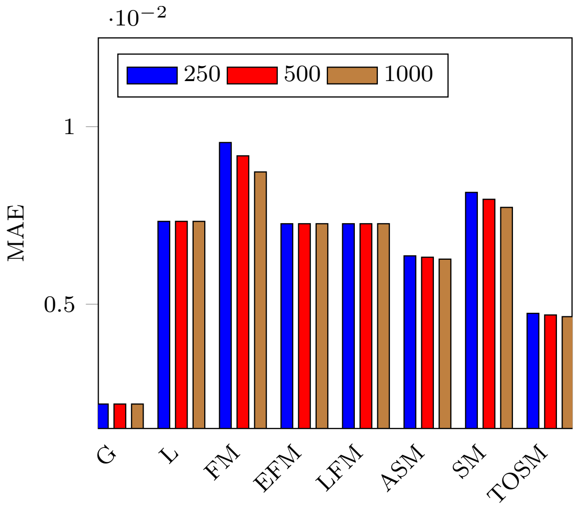
(a) Performance for RMSE divergence using $U = 250$ and $M = 250$.



(b) Performance for KL divergence using $U = 50$ and $M = 1000$.



(c) Performance for SMAPE using $U = 50$ and $s = 1$.



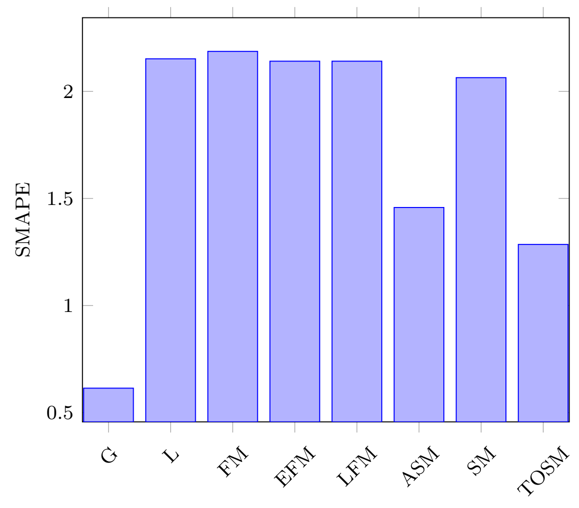
(d) Performance for MAE using $U = 250$ and $s = 4$.

Figure 5.5: Performance evaluation over all proposed models using different settings for s and M using the performance metrics KL, RMSE, MAE, and SMAPE.

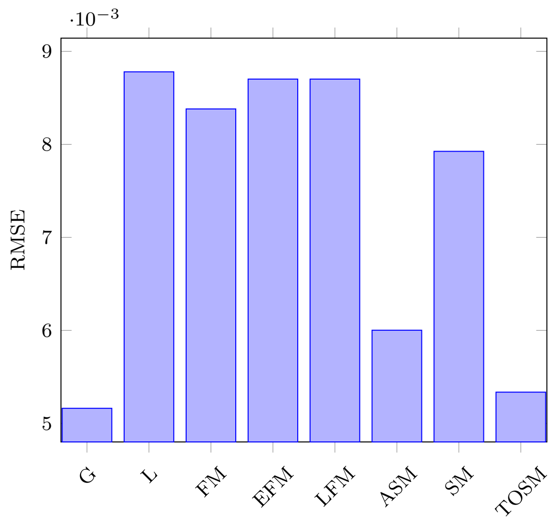
results. Over all dependent models, the increase of the training dataset size in the SLW $\mathcal{W}$ inside each ED k for SGD presented by M is decreasing the accuracy error. In Figure 5.5 (d), similar behaviour is illustrated with the update frequency of $s = 4$ and $U = 250$ using the MAE performance metric.

After identifying the influence of the SLW $\mathcal{W}$ of size M, the SLW $\mathcal{O}$ of length U for the ASM approach, and the central defined update frequency s, towards the models proposed in this chapter (see Section 5.5.3) with respect to their performance the following best settings have been used over the four performance metrics KL divergence, RMSE, MAE, and SMAPE and illustrated in Figure 5.6. Figure 5.6 uses the setting of $M = 1000$ as shown in Figure 5.5 performing lowest of prediction error over all models, $U = 50$ as highlighted in Figure 5.2

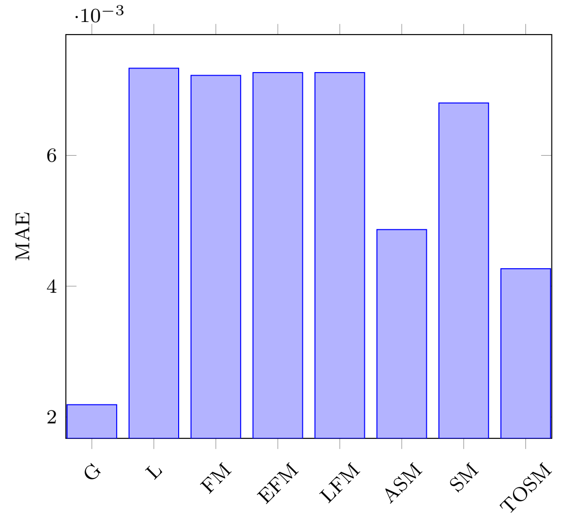
for minor variance in the average α value, $s = 1$ indicating an update of the FM of each day with $S = 74$, and $\beta = 0.3$ analysed in Figure 5.4 and Figure 5.3 to be the most accurate value with respect to the used metrics.



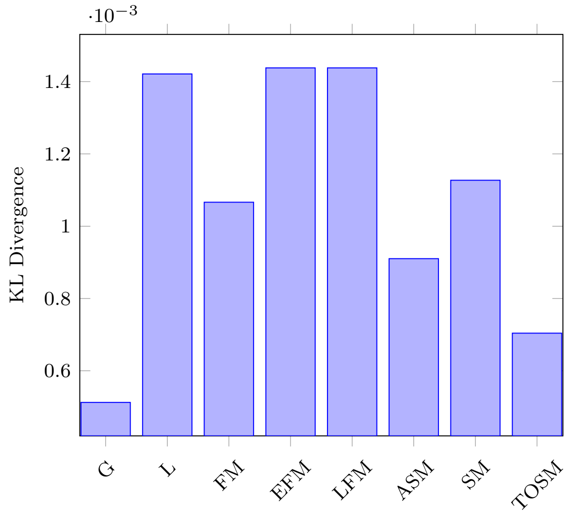
(a) Performance with best parameter setting on SMAPE.



(b) Performance with best parameter setting on RMSE.



(c) Performance with best parameter setting on MAE.



(d) Performance with best parameter setting on KL divergence.

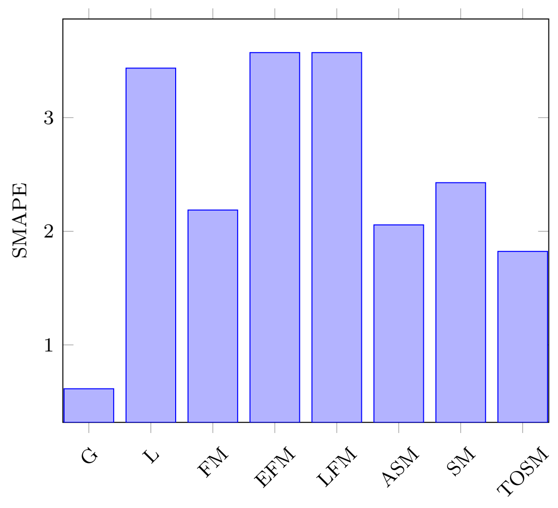
Figure 5.6: Comparison of using the best parameter settings towards the performance metrics KL, RMSE, MAE, and SMAPE.

In Figure 5.6 (a), the metric of SMAPE over all assessed methods is compared using the best settings. From this figure, the aim is to identify what approach is closest to the G with respect to their performance. Even though the G is in privacy-preserving IoT applications not feasible, the aim is to have quality-aware and efficient models inside the ED performing as accurate as of the G. Having this aspect in mind, the strategy of TOSM using the Optimal Stopping Theory (OST) to identify the optimal time to switch between the local generated model f_k (L) and the generalised model f_{FL} (FM) shows the best performance. However, the introduced method of TOSM still increases the error by 1%. The second best with regards to the performance metric SMAPE is the approach of ASM. The other methods do not show a great difference. Figure 5.6 (b) does the analysis using the metric of RMSE. In this figure, the proposed approaches differ more from each other. Highlighted in this figure is the great performance of TOSM with RMSE error very close to the G. Moreover, the prediction accuracy performance of the FM is better than that of the L. ASM provides next to TOSM the best accuracy for predictive analytics in privacy-preserving environments. Figure 5.6 (c) and Figure 5.6(d) illustrate these findings for the performance metrics MAE and KL divergence . Especially, for the presented KL divergence metric showing the information loss inside the ED by deploying the proposed mechanism, offer a great improvement to the FM by deploying a dual model strategy inside the ED and using either adaptive weighting with ASM or optimal selecting using TOSM.

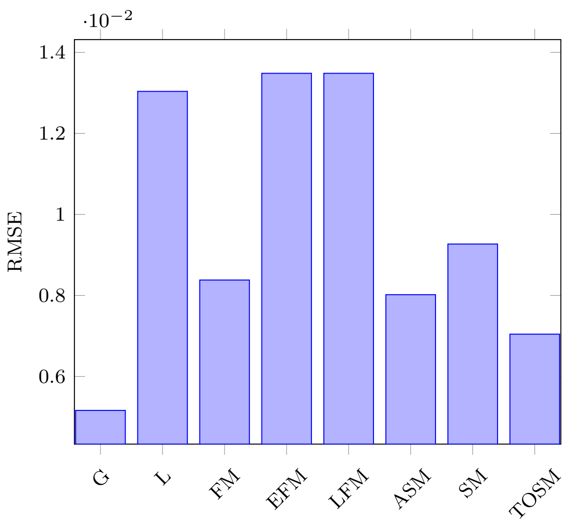
5.6.2 Behaviour on Concept Drifts

This chapter argues the importance of deploying adaptive and evolving learning inside EDs to support qualitative predictive analytics and modelling. Assessing this hypothesis, the following section is introducing an artificial constructed concept drift to test the ability of each proposed method towards changing environments and quality-aware predictions. In Figure 5.7, the performance over all four metrics KL divergence, RMSE, MAE, and SMAPE is illustrated using the identified best settings of Figure 5.6. The values are compared against each other, showing the change of accuracy and information loss with concept drift appearance towards regular predictive tasks.

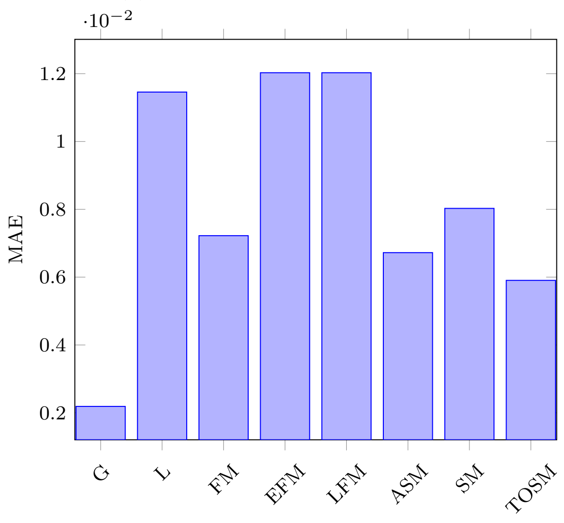
Figure 5.7 (a) illustrates the performance with respect to the metric SMAPE. The illustration shows a clear improvement of the FM performance when concept drift occurs. However, the adaptive and parallel model adaption of ASM and TOSM perform equal or better than the FM. This indicates the ability to adapt to changing environments when using the generalised model and incorporating the local individualised model towards a prediction. Moreover, it should be noted that the models of L, EFM, and LFM highly increase their error. In Figure 5.7 (c), the same behaviour towards the approaches is illustrated using the MAE performance assessment metric. Figure 5.7 (b) shows the RMSE during the concept drift appearance and highlights the good adaptation of ASM with similar performance than the presented values in Figure 5.6 (b) for familiar data inputs inside the ED k. The FM and ASM generate similar prediction results that indicate that the adaptive parameter α places more importance on the FM as the L can not adapt that fast to concept drifts. In Figure 5.7 (d) provides insights into the information loss by presenting the performance metric KL divergence. The value of the KL divergence of the FM approach is improving through the



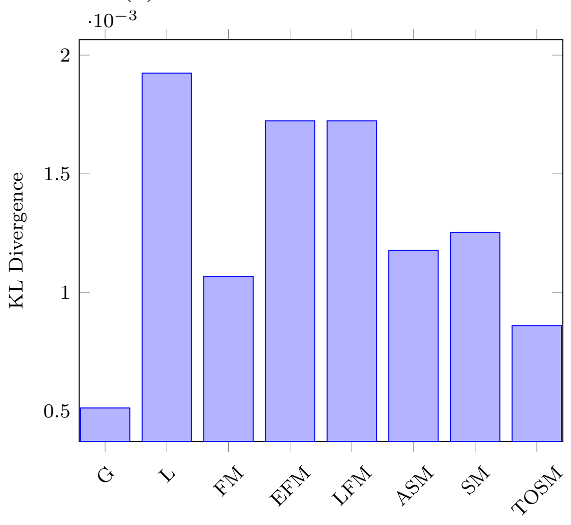
(a) Performance on SMAPE.



(b) Performance on RMSE.



(c) Performance on MAE.



(d) Performance on KL.

Figure 5.7: Comparison of using the best parameter settings and artificially construct a concept drift towards the performance metrics KL, RMSE, MAE, and SMAPE.

concept drift, showing the importance of generalisation inside EDs. Some improvement of the information loss value is given for the ASM method also, as it highly depends on the accuracy of the FM or L (depending on the weighting factor α). However, the method of TOSM provides constantly low information loss independent of the occurrence of the concept drift or not.

5.7 Chapter Summary

In this chapter, the focus on privacy-efficient analytics in resource constraint environments has been investigated. It has been shown through related work on privacy-preserving local edge learning, that Federated Learning (FL) introduced local learning and privacy of data by design. The newly developing research community leaves open research questions towards quality and efficiency of FL. In this chapter two strategies have been proposed that enable the ability to centrally learn a predictive model and enhancing the quality of local inferred and predictive results. Quality of analytical results through enabling the local individuality of heterogeneous devices provided the fundamentals of these approaches. The first model, Adaptive Smoothing Model (ASM), uses the local model and generalised model to provide a new prediction outcome by weighting these two models based on historical rewards. The second strategy introduces the optimisation to find the best time to switch between local model and generalised federated model by using Optimal Stopping Theory (OST). Through real data evaluation the effectiveness of both methods have been shown. Further it was possible to provide evidence, that a mixture or switching strategy between models inside EDs enables qualitative privacy-preserving predictive analytics for continuous changing and evolving environments (including concept drifts).

Chapter 6

Conclusion

6.1 Chapter Overview

The thesis introduced in Section 1.2 the following statement:

> Performing analytical tasks and machine learning over data from edge devices such as the Internet of Things experience constraints in latency, privacy, and bandwidth due to communication overhead and data processing at a central location. This thesis provides a communication, latency, and privacy-efficient methodology through enabling analytics at the source of the data - the edge. Intelligent decision-making mechanisms combined with collaborative intelligence between the edge and the central location are presented to reduce the communication and empower local learning. This thesis further concentrates on the quality of analytics in privacy-preserving environments with real-time local learning and inference.

The presented chapters of this thesis supported the statement above and contributed towards the defined hypotheses. Chapter 3 provided methods towards communication and bandwidth-efficient data forwarding in edge systems. Chapter 4 introduced edge-centric analytics with intelligent model retraining and forwarding mechanism to overcome latency issues in real-time IoT applications. Chapter 5 focused on the aspect of qualitative and privacy-efficient learning in energy constraint environments for continuously evolving data. This chapter will provide a revisit of the defined hypotheses in Section 1.2, summarise the contributions made to the research community by this thesis and presents limitations and future work. Finally, some concluding remarks to quality-aware predictive modelling and inferential analytics at the network edge is given.

6.2 Revisit of Hypotheses

Hypothesis 1:

> Pushing computational intelligence of advanced decision-making in data forwarding to the edge of the network will overcome energy and bandwidth constraints due to the deployment of efficient communication methodologies. Combining this with intelligent reconstruction at a collection point leads to highly accurate analytical tasks and reconstruction of the imputed values.

This hypothesis is investigated throughout Chapter 3. It has been shown that combining the individual computational capacity of Sensing and Actuating Nodes (SANs) and Edge Devices (EDs) into intelligent data forwarding decision making and reconstruction methods leads to highly accurate reconstruction functionalities. While deploying a light-weighted algorithm inside the SANs to reduce the data transmission, the ED can use more complex but still efficient reconstruction models for data imputation. This enables the ED to perform quality-aware analytics for aggregation and predictive modelling while reducing the required resources of SANs with respect to energy and battery lifetime. Further, the reduced data transfer induces a bandwidth reduction through the deployed low complex communication reduction mechanism inside the SANs. Therefore, this hypothesis can be accepted and stated as true with respect to the findings of Chapter 3.

Hypothesis 2:

> Enabling machine learning and predictive analytics locally at edge devices will empower real-time applications that can adapt intelligently to concept drifts and changes of the continuous data arriving. These locally learned (trained) models can be selected through qualitative model selection methodologies at central coordinators, e.g., Cloud.

In Chapter 4, this hypothesis has been studied. Enabling the ED to perform edge-centric local analytics and learning require resource-efficient retraining strategies. In this thesis, a light weighted novelty identification mechanism has been proposed to allow retraining only for unfamiliar data measurements. Based on the prediction performance, model fitting, or model parameter divergence, the decision on forwarding the locally trained model to a Central Location (CL) has been made. This method has proven to be adaptive to changing environments and reduce communication while maintaining the accuracy performance of query-driven predictions. Answering the application requested prediction queries, multiple model-selection methodologies inside the Edge Gateway (EG) have been proposed to solve the distributed learning problem. These model-selection strategies considerably improved related work with respect to accuracy and communication efficiency by only transmitting model parameters and statistics.

Hypothesis 3:

> Generalised models over privacy-preserved data by only transferring analytical model parameters over the network will not provide qualitative results in constantly changing and heterogeneous environments. Using the prospect of locally learning models with an intelligent model selection and weighting mechanism for personalisation and generalisation in edge devices enables data privacy and qualitative prediction results.

The third hypothesis is investigated in Chapter 5. This chapter focuses on the privacy of data generated by EDs and the importance of embracing local individuality towards model training. Using the fundamentals of Federated Learning (FL) in which no sharing of the local data is required, the work of this thesis proposes personalisation and mechanism to balance or swap between the generalised federated model and the individualised model. It has been proven that combining or selecting the optimal model provides qualitative predictive results in heterogeneous and continuously changing environments. The importance of relying on individualised and personalised models is highlighted through the significance of adapting to the evolving data in which generalisation is improving the quality of results compared to the fine-tuned local model. Therefore, the hypothesis has been accepted that balancing or intelligently deciding which model is optimal for inference and prediction can provide quality-aware and privacy-efficient performance for analytics in edge networks.

6.3 Contributions

A detailed list of the contributions concerning each chapter has been presented in Section 1.3. Summarised, the contributions of this thesis should provide theoretical and experimental improvement towards quality-aware edge networks for the aspects bandwidth, latency and privacy.

Explicitly, Chapter 3 contributes with bandwidth and energy-efficient data forwarding mechanism by exploring the computational capacity of resource constraint devices (e.g. SANs and EDs). A significant improvement has been shown by extending current energy-efficient routing strategies, which missed quality-aware reconstruction and analytics functionalities, towards a qualitative-efficient prediction design based on time series algorithms inside SANs and EDs. Exploring the game-theoretical method of Optimal Stopping Theory (OST) inside SANs enabled these resource constraint devices to perform intelligence by incorporating historical decisions towards a quality-aware forwarding of measurements. The assessment of the strategies has considered the reconstruction ability of the imputed values and the performance evaluation of the predictive and analytical aggregation tasks. In an extensive experimental evaluation, the effectiveness and improvement of the proposed methodologies have been presented with a great reduction of communication in light of low-complex and quality-aware intelligence algorithms in SANs and EDs.

In Chapter 4, a novel methodology has been proposed towards latency-efficient model retraining and forwarding in resource-constrained devices by enabling local edge-centric analytics. The deployment of familiarity-based input data classification by using quantisation of the input space through cluster connected with the associated predictive error has been shown to improve greatly qualitative retraining of the locally generated models. Adapting efficient retraining and model forwarding in resource constraint devices provides sustainable capacity usage by reducing the bandwidth and communication (only model parameters and statistics are forwarded). Moreover, with local predictive modelling implemented in changing and continuously evolving environments, devices can react in real-time without any communication delay to the Central Location (CL). A possible solution towards the issue of selecting from the ensemble of models at the CL transmitted from each individual learner (EDs) has been presented in Chapter 4. It has been shown, through comprehensive experiments, that similarity-based clustering is notably improving the accuracy in comparison to a simple averaging of model predictions. The forwarded input data statistics can be used as valuable information for query-driven analytics. Introducing this quality-efficient low complex local edge-centric methodology improves the EDs ability to perform real-time analytics and the CL to perform qualitative query-driven analytics without communication overhead (e.g. requesting model updates over the network).

Finally, Chapter 5 provides a privacy-focused analytical methodology by enabling the local computational capacity of each ED to perform personalised Federated Learning (FL). In some Internet of Things (IoT) application environments, the collected data requires privacy-preserving techniques while providing qualitative and resource-efficient analytics. The deployment of FL introduces privacy by design and enables local real-time privacy-preserving data analytics in resource constraint devices. However, the new research area still requires optimisation towards more accurate and qualitative focused analytics over non-independent and identically distributed (i.i.d.) data and changing environments. Therefore, the thesis contributes to providing a novel personalisation FL methodology in identifying how to best balance a local individualised model and a generalised federated model inside each individual learner. The two proposed methods greatly improve the current deployed related work. The first approach is to incorporate both models towards a balanced weighted new prediction. The second technique is through identifying the best time to switch between the two given models. Both strategies explore the local data space of the ED to build prediction models locally using a low-complex computational process for providing accurate forecasts while being efficient concerning the given resource constraints.

6.4 Limitations and Future Work

The presented work in this thesis is limited to multivariate linear regression models for Edge Computing environments using contextual data. This leads to future work in advancing the proposed methodologies to advanced analytic functions, including neural networks. However, using more complex algorithms in resource constraint devices enables the need to prove the required resources for transmitting data compared to performing local training and analytics. In the presented work, the used resources are efficiently balanced in which the local learning enables sustainable energy usage, as low computational complexity algorithms are deployed and used. This work did not provide a one-size-fits-all model for a specific use case but instead explores strategies towards efficient and quality-aware local analytics to already introduced analytical use cases. This leads to further extending the methodology to use cases towards smart homes or autonomous cars deploying video and image data alongside the sensor data and contextual data.

The topic of privacy-efficient analytics in edge networks is currently one most required areas in the research and industrial community. The governmental requirements and user-requested privacy of the collected data lead to future work in performing analytics over highly distributed, unbalanced and heterogeneous data sets. The deployment of Federated Learning (FL) to overcome law regulations is one possible solution, as highlighted. However, combining other privacy mechanisms and comparing the accuracy towards, e.g. differential privacy, can be seen as a future research direction. In some future work, we already presented the use of IoT with Blockchain [274]. This can be further explored by enabling analytics in this environment or in combination using FL [275, 276]. The presented work of FL is using usually supervised learning methods for their applied use cases [277, 278, 279]. Many applications are semi-supervised or unsupervised in industrial environments. Labelling the data manually is for most use cases infeasible. Therefore, the usage of FL in unsupervised machine learning environments is an excellent direction for further research [280].

Interestingly, there is also the aspect of fully decentralised FL [188, 281] in combination with a centralised coordinator but allowing the exchange and learning from the surrounding devices. This leads to the most promising area of providing a cluster-based FL methodology that identifies similar devices, maybe using the proposed approach of Section 4.5.2. Given these clusters, it is possible to perform the federated model aggregation on the group to have different hierarchies of federated models. This, in combination with the proposed personalisation strategy in Chapter 5, can lead to more quality-aware analytical results.

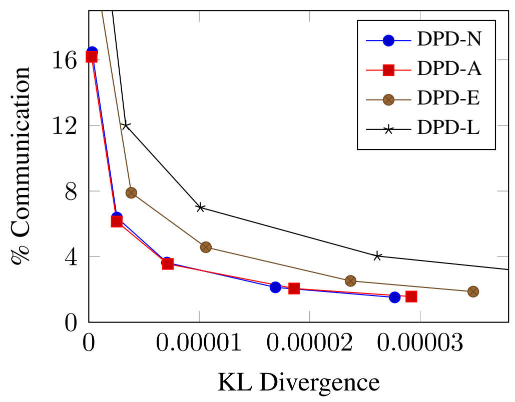
6.5 Concluding Remarks

The increase of deployed Internet of Things (IoT) devices and their computational capacity enables the use of local edge-centric intelligence and analytics. The importance of efficiency with extending the battery lifetime or reducing the bandwidth and communication is the primary aspect in these applications. Only low complex and sustainable algorithms or strategies can be deployed in these devices. The performed analytical tasks and calculated predictions result in decisions that can sometimes lead to critical consequences to the user, the environment or the application. Therefore providing qualitative predictive and inferential results while being sustainable with the given resources in edge networks has motivated this thesis.

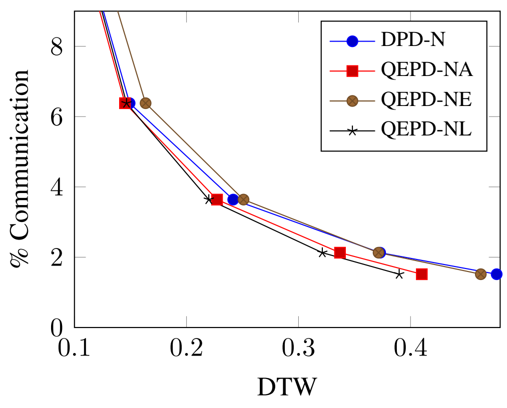
This thesis has focused on providing a communication-efficient data forwarding strategy for applications with enormous and frequent data collection applications, especially for low computational capacity devices. In environments providing more computational power of devices and having real-time requirements over evolving data, the presented work in this thesis enabled local efficient retraining and forwarding of only model parameters. Moreover, qualitative model-selection strategies for central query-driven analytics combined with reduced communication and transmission of the local individualised models have shown to provide quality-aware prediction performance in latency constraint networks. To fulfil the necessity of quality-aware analytics in resource-constrained and constantly changing environments, data privacy under this aspect has been highlighted in this thesis. It has been shown that enabling personalisation combined with generalised federated models can lead to qualitative prediction results while keeping the data secure and unshared in EDs. The aspects of bandwidth, latency, and privacy regarding efficient resource usage and quality-aware analytical results have been presented with the proposed strategies in this thesis to advance the current State-of-the-Art knowledge within the research community.

Appendix A

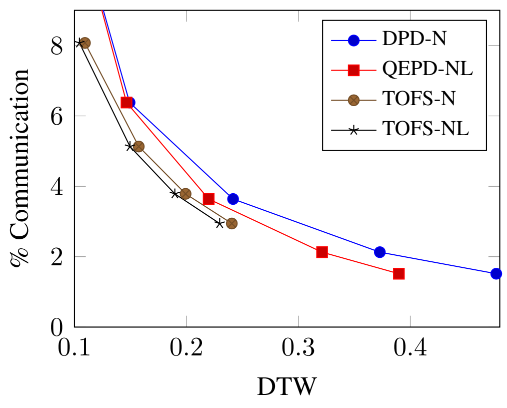
Quality-Efficient Data Forwarding Evaluation Additional Dataset



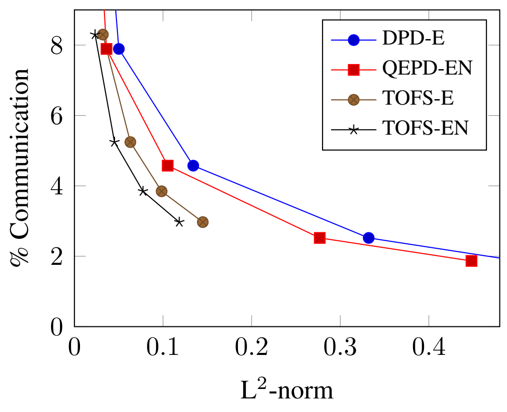
(a) Trade-off between KL divergence and communication over all DPD methods.



(b) Comparison of the trade-off using the DTW metric and remaining communication using the NAIVE function inside the SAN.

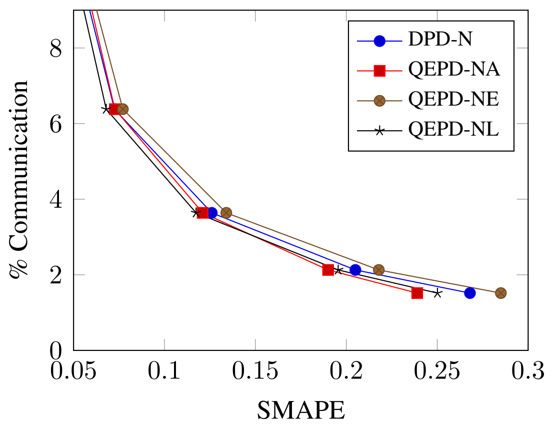


(c) Comparison of the trade-off using the DTW metric and remaining communication using the NAIVE function inside the SAN.

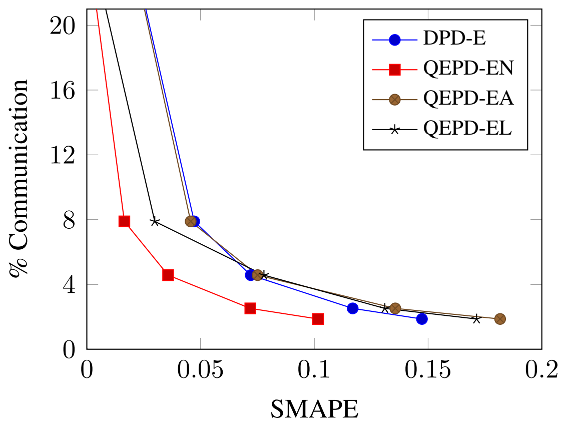


(d) Comparison of the trade-off using the L^2-norm metric and remaining communication using the EWMA function inside the SAN.

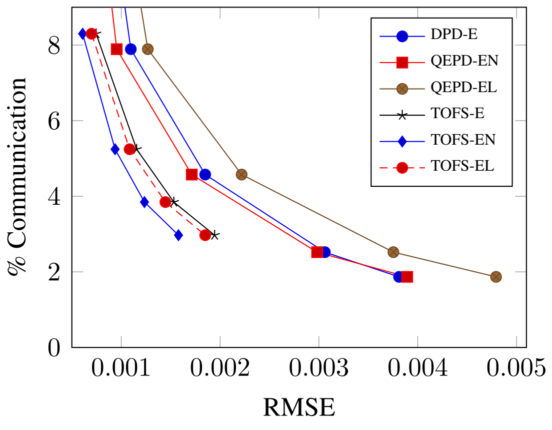
Figure A.1: Evaluation of DPD trade-off Communication and KL, DTW for DS2.



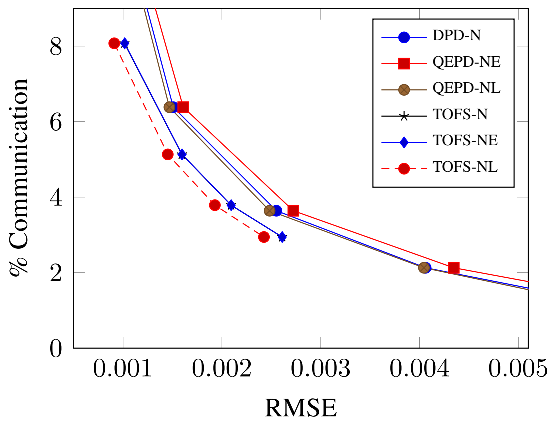
(a) Trade-off using the SMAPE and remaining communication using the NAIVE function inside the SAN over the aggregation function `MAX`.



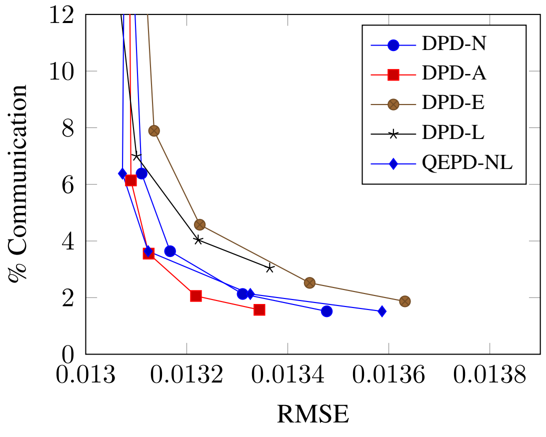
(b) Trade-off using the SMAPE and remaining communication using the EWMA function inside the SAN over the aggregation function `AVG`.



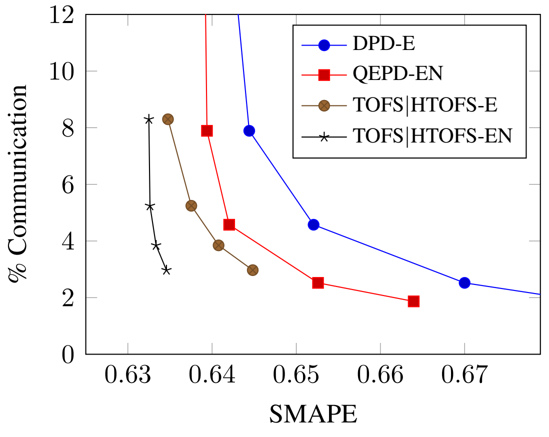
(c) Trade-off using the RMSE and remaining communication over the aggregation function `MEDIAN`.



(d) Trade-off using the RMSE and remaining communication over the aggregation function `MAX`.



(e) Trade-off using the RMSE and remaining communication over the predictive function for all DPD and QEPD for the best NAIVE.



(f) Trade-off using the SMAPE metric and remaining communication over the predictive function for all methods with EWMA function inside the SAN.

Figure A.2: Evaluation of trade-off between communication and accuracy metrics (SMAPE, RMSE) over aggregation functions for QEPD,TOFS and HTOFS methods for DS2.

Appendix B

Latency-Efficient Edge-Centric Analytics Dataset Information

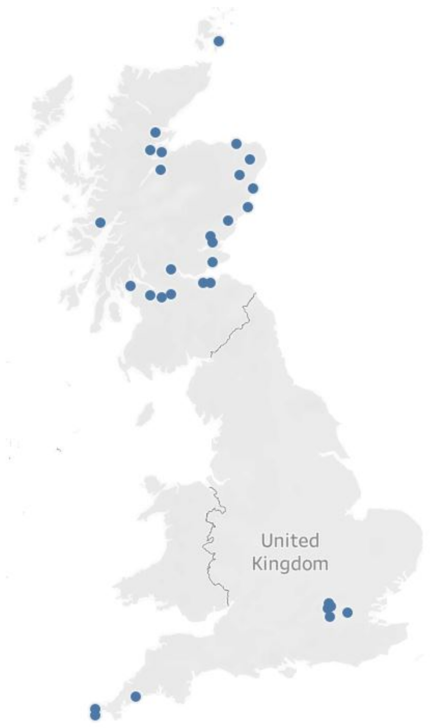


Figure B.1: Map of 30 randomly selected stations representing each an ED i for model retraining and forwarding strategies highlighted in Chapter 4.

Station	ID	Lat	Lon
IABERDEE40	29	57.29	-2.4
IABERDEE58	8	56.9	-2.21
IACHARAC3	1	56.7	-5.57
IANGUSDU2	26	56.53	-3.07
IANSBREC2	2	56.73	-2.67
IBLACKIS2	25	57.57	-4.17
IDUNDEE10	12	56.46	-3.03
IEDINBUR82	9	55.95	-3.23
IENGLAND499	5	51.53	0.03
IENGLAND570	23	51.66	-0.39
IFIFE16	27	56.21	-3.03
IGLASGOW1	18	55.78	-4.42
IGREATER38	7	51.61	-0.34
IHIGHLAN4	15	57.36	-4.2
IHIGHLAN8	14	57.59	-4.43
IINVERCL8	22	55.9	-4.87
ILONDON658	24	51.58	-0.41
IMACDUFF2	6	57.67	-2.49
IMIDDLES2	4	51.46	-0.37
ISCOTLAN68	13	57.12	-2.1
ISCOTLAN175	3	57.48	-2.17
ISCOTLAN189	20	58.9	-2.89
ISCOTLAN367	0	55.94	-3.07
ISOUTHLA5	16	55.76	-4.18
ISTAUSTE2	10	50.33	-4.75
ISTIRLIN5	21	56.12	-3.95
ISTJUSTI3	17	50.15	-5.67
IUNITEDK50	11	50.06	-5.67
IUNITEDK397	19	57.81	-4.32
IWISHAW4	28	55.8	-3.96

Table B.1: Mapping of name, ID and geographical information of stations.

Bibliography

[1] Gartner, "Gartner Says 5.8 Billion Enterprise and Automotive IoT Endpoints Will Be in Use in 2020," 2019. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io [Cited at Page: 1]

[2] G. Peskir and A. Shiryaev, "Optimal stopping and free-boundary problems," in *Optimal Stopping and Free-Boundary Problems*. Basel: Springer, 2006, pp. 123–142. [Online]. Available: https://doi.org/10.1007/978-3-7643-7390-0_3 [Cited at Page: 4, 31, and 52]

[3] N. Harth, H.-J. Voegel, K. Kolomvatsos, and C. Anagnostopoulos, "Local Learning at the Network Edge for Efficient & Secure Real-Time Predictive Analytics," *arXiv preprint arXiv:2109.12375*, 2021. [Online]. Available: https://arxiv.org/abs/2109.12375 [Cited at Page: 6 and 112]

[4] N. Harth and C. Anagnostopoulos, "Edge-centric efficient regression analytics," in *IEEE International Conference on Edge Computing, EDGE 2018 - Part of the 2018 IEEE World Congress on Services*. IEEE, 7 2018, pp. 93–100. [Online]. Available: https://ieeexplore.ieee.org/document/8473382/ [Cited at Page: 6, 79, 80, and 97]

[5] ——, "Quality-aware aggregation & predictive analytics at the edge," in *IEEE International Conference on Big Data, Big Data 2017*, vol. 2018-Jan, 2017, pp. 17–26. [Cited at Page: 6 and 50]

[6] N. Harth, C. Anagnostopoulos, and D. Pezaros, "Predictive intelligence to the edge: impact on edge analytics," *Evolving Systems*, vol. 9, no. 2, pp. 95–118, 2018. [Cited at Page: 6 and 45]

[7] N. Harth, K. Delakouridis, and C. Anagnostopoulos, "Convey intelligence to edge aggregation analytics," in *Studies in Computational Intelligence*. Springer Verlag, 2018, vol. 715, pp. 25–44. [Cited at Page: 6 and 45]

[8] A. S. Tanenbaum and M. Van Steen, *Distributed Systems: Principles and Paradigms*. Prentice-Hall, 2007. [Cited at Page: 9]

[9] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*. Pearson Education, 2012, vol. 4. [Cited at Page: 9 and 10]

[10] S. Ghosh, *Distributed Systems: An Algorithmic Approach*, 1st ed. Chapman and Hall/CRC, 2006. [Cited at Page: 9]

[11] M. Newman, *Networks: An Introduction*. Oxford University Press, 2018. [Cited at Page: 10]

[12] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach*. Morgan Kaufmann, 2007. [Cited at Page: 10]

[13] P. R. Monge and N. Contractor, *Theories of communication networks*. Oxford University Press, 3 2003. [Cited at Page: 10]

[14] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed. Pearson, 2011, vol. 52, no. 169. [Cited at Page: 10]

[15] A. D. Kshemkalyani and M. Singhal, *Distributed Computing: Principles, Algorithms, and Systems*. Cambridge University Press, 2011. [Cited at Page: 10 and 14]

[16] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. [Cited at Page: 10]

[17] H. Attiya and J. Welch, *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. John Wiley & Sons Ltd, 2004, vol. 19. [Cited at Page: 10]

[18] K. Hwang, J. Dongarra, and G. C. Fox, *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Morgan Kaufmann, 2013. [Cited at Page: 10]

[19] L. D. Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014. [Cited at Page: 11]

[20] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015. [Cited at Page: 11]

[21] Ultimatearm, Becris, and Freepik et al., "Flaticon." [Online]. Available: https://www.flaticon.com/ [Cited at Page: 11, 20, and 33]

[22] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*, 2nd ed. Morgan Kaufmann, 2003. [Cited at Page: 12 and 18]

[23] L. D. Wittie, "Communication Structures for Large Networks of Microcomputers," *IEEE Transactions on Computers*, no. 4, pp. 264–273, 1981. [Cited at Page: 12 and 18]

[24] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004. [Cited at Page: 12 and 18]

[25] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 414–454, 2014. [Cited at Page: 12]

[26] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research," *Open Dartmouth: Faculty Open Access Articles*, no. 3212, pp. 1–16, 2000. [Cited at Page: 12]

[27] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons Ltd, 2007. [Cited at Page: 12 and 15]

[28] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008. [Cited at Page: 12]

[29] K. Römer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54–61, 12 2004. [Cited at Page: 12 and 15]

[30] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, 2010. [Cited at Page: 12 and 16]

[31] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, and M. Eisenhauer, "Internet of things strategic research roadmap," *Internet of things-global technological and societal trends*, vol. 1, no. 2011, pp. 9–52, 2011. [Cited at Page: 12]

[32] B. L. Risteska Stojkoska and K. V. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.jclepro.2016.10.006 [Cited at Page: 13]

[33] H. Farhangi, "The path of the smart grid," *IEEE Power and Energy Magazine*, vol. 8, no. 1, pp. 18–28, 2010. [Cited at Page: 13]

[34] D. J. Cook, M. Youngblood, E. O. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "MavHome : An Agent-Based Smart Home," in *IEEE International Conference on Pervasive Computing and Communications*, 2003, pp. 3–6. [Cited at Page: 13]

[35] P. J. Soh, G. A. E. Vandenbosch, M. Mercuri, and D. M.-P. Schreurs, "Wearable wireless health monitoring: Current developments, challenges, and future trends," *IEEE Microwave Magazine*, vol. 16, no. 4, pp. 55–70, 2015. [Cited at Page: 13]

[36] B. G. Celler, T. Hesketh, W. Earnshaw, and E. Ilsar, "An instrumentation system for the remote monitoring of changes in functional health status of the elderly at home," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 2, 1994, pp. 908–909. [Cited at Page: 13]

[37] A. D. Wood, J. A. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru, "Context-aware wireless sensor networks for assisted living and residential monitoring," *IEEE Network*, vol. 22, no. 4, pp. 26–33, 2008. [Cited at Page: 13]

[38] U. Varshney, "Pervasive healthcare and wireless health monitoring," *Mobile Networks and Applications*, vol. 12, no. 2-3, pp. 113–127, 2007. [Cited at Page: 13]

[39] S. M. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. S. Kwak, "The internet of things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015. [Cited at Page: 13]

[40] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014. [Cited at Page: 13]

[41] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things," *Transactions on emerging telecommunications technologies*, vol. 25, no. 1, pp. 81–93, 2014. [Cited at Page: 13]

[42] M. Li and Y. Liu, "Underground structure monitoring with wireless sensor networks," in *International Symposium on Information Processing in Sensor Networks (IPSN)*, 2007, pp. 69–78. [Cited at Page: 13]

[43] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *International Symposium on Information Processing in Sensor Networks (IPSN)*, 2007, pp. 254–263. [Cited at Page: 13]

[44] K. Martinez, J. Hart, and R. Ong, "Environmental sensor networks," *Computer*, vol. 37, no. 8, pp. 50–56, 8 2004. [Online]. Available: http://ieeexplore.ieee.org/document/1323016/ [Cited at Page: 13]

[45] K. K. Khedo, R. Perseedoss, and A. Mungur, "A Wireless Sensor Network Air Pollution Monitoring System," *International Journal of Wireless & Mobile Networks*, vol. 2, no. 2, pp. 31–45, 5 2010. [Online]. Available: http://www.airccse.org/journal/jwmn/0510ijwmn03.pdf [Cited at Page: 13]

[46] H. Bai, M. Atiquzzaman, and D. Lilja, "Wireless sensor network for aircraft health monitoring," in *International Conference on Broadband Networks*, 2004, pp. 748–750. [Cited at Page: 13]

[47] V. W. S. Tang, Y. Zheng, and J. Cao, "An intelligent car park management system based on wireless sensor networks," in *International Symposium on Pervasive Computing and Applications*, 2006, pp. 65–70. [Cited at Page: 13]

[48] a. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *ACM International Workshop on Wireless Sensor Networks and Applications*, 2002, pp. 88–97. [Cited at Page: 13]

[49] L. Yu, N. Wang, and X. Meng, "Real-time forest fire detection with wireless sensor networks," *International Conference on Wireless Communications, Networking and Mobile Computing (WCNM)*, vol. 2, pp. 1214–1217, 2005. [Cited at Page: 13]

[50] V. Romanov, I. Galelyuka, and Y. Sarakhan, "Wireless sensor networks in agriculture," in *IEEE International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2016, pp. 77–80. [Cited at Page: 13]

[51] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard Computing: Sensor Networks in Agricultural Production," *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 38–45, 2004. [Cited at Page: 13]

[52] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, no. 2, pp. 18–25, 2006. [Cited at Page: 13]

[53] I. F. Akyildiz, D. Pompili, and T. Melodia, "Challenges for efficient communication in underwater acoustic sensor networks," *ACM Sigbed Review*, vol. 1, no. 2, pp. 3–8, 2004. [Cited at Page: 13]

[54] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *Journal of Industrial Information Integration*, vol. 6, pp. 1–10, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.jii.2017.04.005 [Cited at Page: 13]

[55] S. Wang, J. Wan, D. Li, and C. Zhang, “Implementing smart factory of industrie 4.0: an outlook,” *International Journal of Distributed Sensor Networks*, vol. 12, no. 1, p. 3159805, 2016. [Cited at Page: 13]

[56] P. Fraga-Lamas, T. M. Fernández-Caramés, M. Suárez-Albela, L. Castedo, and M. González-López, “A review on internet of things for defense and public safety,” *Sensors*, vol. 16, no. 10, p. 1644, 2016. [Cited at Page: 13]

[57] M. Satyanarayanan, “Pervasive computing: Vision and challenges,” *IEEE Personal Communications*, vol. 8, no. 4, pp. 10–17, 2001. [Cited at Page: 14]

[58] J. Krumm, *Ubiquitous computing fundamentals.* Chapman and Hall/CRC, 2018. [Cited at Page: 14]

[59] M. Weiser, “ The computer for the 21 st century ,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 3, no. 3, pp. 3–11, 1999. [Cited at Page: 14]

[60] S. Poslad, *Ubiquitous computing: smart devices, environments and interactions.* John Wiley & Sons Ltd, 2011. [Cited at Page: 14]

[61] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, “Instrumenting the world with wireless sensor networks,” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, pp. 2033–2036, 2001. [Cited at Page: 14]

[62] C. S. Raghavendra, K. M. Sivalingam, and T. Znati, *Wireless sensor networks.* Springer, 2006. [Cited at Page: 14]

[63] K. Sohraby, D. Minoli, and T. Znati, *Wireless sensor networks: technology, protocols, and applications.* John Wiley & Sons Ltd, 2007. [Cited at Page: 14]

[64] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: A survey,” *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002. [Cited at Page: 14, 15, and 39]

[65] K. Akkaya and M. Younis, “A survey on routing protocols for wireless sensor networks,” *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, 2005. [Cited at Page: 15 and 40]

[66] W. Dargie and C. Poellabauer, *Fundamentals of Wireless Sensor Networks: Theory and Practice.* John Wiley and Sons Ltd, 1 2011. [Cited at Page: 15]

[67] E. H. Callaway Jr, *Wireless sensor networks: architectures and protocols.* Chapman and Hall/CRC, 2003. [Cited at Page: 15]

[68] K. Ashton and others, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009. [Cited at Page: 16]

[69] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011. [Cited at Page: 16 and 20]

[70] A. Botta, W. De Donato, V. Persico, and A. Pescapé, "Integration of Cloud computing and Internet of Things: A survey," *Future Generation Computer Systems*, vol. 56, pp. 684–700, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.future.2015.09.021 [Cited at Page: 16]

[71] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.future.2013.01.010 [Cited at Page: 16]

[72] H. Kopetz, *Real-time systems: design principles for distributed embedded applications*. Springer, 2011. [Cited at Page: 16]

[73] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2012.02.016 [Cited at Page: 16]

[74] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "IOT gateway: Bridging wireless sensor networks into Internet of Things," in *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC)*, 2010, pp. 347–352. [Cited at Page: 17]

[75] L. Mainetti, L. Patrono, and A. Vilei, "Evolution of wireless sensor networks towards the Internet of Things: A survey," in *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2011, pp. 16–21. [Cited at Page: 17]

[76] A. Čolaković and M. Hadžialić, "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues," *Computer Networks*, vol. 144, pp. 17–39, 2018. [Cited at Page: 17 and 33]

[77] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," Tech. Rep. [Cited at Page: 19]

[78] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and others, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010. [Cited at Page: 19]

[79] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010. [Cited at Page: 19]

[80] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree compared," in *Grid Computing Environments Workshop (GCE)*, 2008. [Online]. Available: http://www.us-vo.org [Cited at Page: 19]

[81] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," in *ACM SIGCOMM*, 2008. [Cited at Page: 19]

[82] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.future.2008.12.001 [Cited at Page: 19]

[83] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and others, "Above the clouds: A berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, no. 13, p. 2009, 2009. [Cited at Page: 19]

[84] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *SIGCOMM*, 2008, p. 63. [Cited at Page: 20]

[85] R. Jain and S. Tata, "Cloud to Edge: Distributed Deployment of Process-Aware IoT Applications," in *IEEE International Conference on Edge Computing (EDGE)*, 2017. [Cited at Page: 20]

[86] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, 2016. [Cited at Page: 20]

[87] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017. [Cited at Page: 20]

[88] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009. [Cited at Page: 20]

[89] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *ACM Mobile Cloud Computing Workshop (MCC)*, 2012, pp. 13–15. [Cited at Page: 20]

[90] A. V. Dastjerdi and R. Buyya, “Fog computing: Helping the Internet of Things realize its potential,” *Computer*, vol. 49, no. 8, pp. 112–116, 2016. [Cited at Page: 20]

[91] M. Chiang and T. Zhang, “Fog and IoT: An overview of research opportunities,” *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016. [Cited at Page: 20]

[92] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases,” *AI magazine*, vol. 17, no. 3, p. 37, 1996. [Cited at Page: 22]

[93] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Morgan Kaufmann, 2012. [Cited at Page: 22 and 23]

[94] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Morgan Kaufmann, 2016. [Cited at Page: 22]

[95] E. Alpaydin, *Introduction to machine learning*, 3rd ed. MIT Press Cambridge, 2014. [Cited at Page: 22 and 28]

[96] K. P. Murphy, *Machine learning: a probabilistic perspective*. Cambridge, MA: MIT Press, 2012. [Cited at Page: 22, 23, 24, and 96]

[97] T. H. Davenport, L. Prusak, and others, *Working knowledge: How organizations manage what they know*. Harvard Business Press, 1998. [Cited at Page: 22]

[98] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*, 3rd ed. Pearson Education, 2010. [Cited at Page: 22, 23, and 25]

[99] A. P. Engelbrecht, *Computational intelligence: an introduction*. John Wiley & Sons Ltd, 2007. [Cited at Page: 22]

[100] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006. [Cited at Page: 24]

[101] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009. [Cited at Page: 24]

[102] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014. [Cited at Page: 24 and 28]

[103] L. Bottou, “Online Learning and Stochastic Approximations,” in *On-line learning in neural networks*. Cambridge University Press, 1971, pp. 9–42. [Cited at Page: 25]

[104] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1. [Cited at Page: 25]

[105] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015, vol. 734. [Cited at Page: 26]

[106] J. D. Hamilton, *Time series analysis*. Princeton university press, 2020. [Cited at Page: 26]

[107] P. J. Brockwell, R. A. Davis, and S. E. Fienberg, *Time series: theory and methods: theory and methods*. Springer Science & Business Media, 1991. [Cited at Page: 26]

[108] P. Esling and C. Agon, “Time-series data mining,” *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, pp. 1–34, 2012. [Cited at Page: 27]

[109] T. C. Fu, “A review on time series data mining,” *Engineering Applications of Artificial Intelligence*, 2011. [Cited at Page: 27]

[110] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, “Mining data streams: a review,” in *ACM SIGMOD International Conference on Management of Data*, vol. 34, no. 2. ACM New York, NY, USA, 2005, pp. 18–26. [Cited at Page: 27]

[111] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, “Models and issues in data stream systems,” in *ACM PODS*, 2002, p. 1. [Cited at Page: 27 and 29]

[112] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of massive data sets*. Cambridge University Press, 2020. [Cited at Page: 27]

[113] G. Cauwenberghs and T. Poggio, “Incremental and decremental support vector machine learning,” in *Advances in Neural Information Processing Systems*, 2001, pp. 409–415. [Cited at Page: 27]

[114] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, “Incremental learning for robust visual tracking,” *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 125–141, 2008. [Cited at Page: 27]

[115] B. R. Prasad and S. Agarwal, “Stream data mining: Platforms, algorithms, performance evaluators and research trends,” *International Journal of Database Theory and Application*, vol. 9, no. 9, pp. 201–218, 2016. [Cited at Page: 27]

[116] P. Domingos and G. Hulten, “Mining high-speed data streams,” in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 71–80. [Cited at Page: 27]

[117] S. Shalev-Shwartz and Y. Singer, “Online learning: Theory, algorithms, and applications,” Ph.D. dissertation, The Hebrew University of Jerusalem, 2007. [Cited at Page: 27]

[118] E. Hazan, *Introduction to Online Convex Optimization*. Now Foundations and Trends, 2016. [Online]. Available: https://arxiv.org/abs/1909.05207 [Cited at Page: 27]

[119] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge University Press, 2006. [Cited at Page: 27]

[120] L. Bottou and Y. L. Cun, “Large scale online learning,” in *Advances in Neural Information Processing Systems*, 2004, pp. 217–224. [Cited at Page: 28]

[121] L. Bottou, “Stochastic gradient descent tricks,” in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436. [Cited at Page: 28]

[122] H. Robbins and S. Monro, “A stochastic approximation method,” *The annals of mathematical statistics*, pp. 400–407, 1951. [Cited at Page: 28]

[123] M. Zinkevich, “Online convex programming and generalized infinitesimal gradient ascent,” in *International Conference on Machine Learning (ICML)*, 2003, pp. 928–936. [Cited at Page: 28]

[124] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *International Conference for Learning Representations*, 2015. [Cited at Page: 28]

[125] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016. [Online]. Available: https://arxiv.org/abs/1609.04747 [Cited at Page: 28]

[126] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated Learning with Non-IID Data,” *arXiv preprint arXiv:1806.00582*, 2018. [Online]. Available: http://arxiv.org/abs/1806.00582 [Cited at Page: 28 and 37]

[127] B. Babcock, M. Datar, and R. Motwani, “Sampling from a moving window over streaming data,” in *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002)*. Stanford InfoLab, 2001. [Cited at Page: 28]

[128] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," *SIAM International Conference on Data Mining*, pp. 443–448, 2007. [Cited at Page: 28]

[129] P. B. Gibbons and S. Tirthapura, "Distributed streams algorithms for sliding windows," in *ACM symposium on Parallel algorithms and architectures*, 2002, pp. 63–72. [Cited at Page: 28]

[130] T. G. Dietterich, "Machine learning for sequential data: A review," in *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR).* Springer, 2002, pp. 15–30. [Cited at Page: 28]

[131] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining stream statistics over sliding windows," *SIAM journal on computing*, vol. 31, no. 6, pp. 1794–1813, 2002. [Cited at Page: 29]

[132] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014. [Cited at Page: 29 and 29]

[133] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 97–106. [Cited at Page: 29]

[134] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011. [Cited at Page: 29]

[135] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 226–235. [Cited at Page: 29]

[136] I. Žliobaite, "Learning under Concept Drift: an Overview," *arXiv preprint arXiv:1010.4784*, 2010. [Online]. Available: https://arxiv.org/abs/1010.4784 [Cited at Page: 29]

[137] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches.* Cambridge University Press, 2011. [Cited at Page: 29]

[138] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *OSDI USENIX Symposium on Operating Systems Design and Implementation*, 2014, pp. 583–598. [Cited at Page: 30]

[139] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *arXiv preprint arXiv:1705.09056*, 2017. [Cited at Page: 30]

[140] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2010, pp. 2595–2603. [Cited at Page: 30]

[141] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent," *v*, vol. 24, pp. 693–701, 2011. [Cited at Page: 30]

[142] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *IEEE Symposium on mass storage systems and technologies (MSST)*, 2010, pp. 1–10. [Cited at Page: 30]

[143] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, and others, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv preprint arXiv:1603.04467*, 2016. [Online]. Available: https://arxiv.org/abs/1603.04467 [Cited at Page: 30]

[144] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine learning*, vol. 36, no. 1-2, pp. 105–139, 1999. [Cited at Page: 31]

[145] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996. [Cited at Page: 31]

[146] Y. Freund, R. E. Schapire, and others, "Experiments with a new boosting algorithm," in *International Conference on Machine Learning (ICML)*, vol. 96. Citeseer, 1996, pp. 148–156. [Cited at Page: 31]

[147] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997. [Cited at Page: 31]

[148] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992. [Cited at Page: 31]

[149] P. R. Freeman, "The secretary problem and its extensions: A review," *International Statistical Review/Revue Internationale de Statistique*, pp. 189–206, 1983. [Cited at Page: 31]

[150] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg, “Online auctions and generalized secretary problems,” *ACM SIGecom Exchanges*, vol. 7, no. 2, pp. 1–11, 2008. [Cited at Page: 31]

[151] M. Babaioff, M. Dinitz, A. Gupta, N. Immorlica, and K. Talwar, “Secretary problems: weights and discounts,” in *Proceedings of the twentieth annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2009, pp. 1245–1254. [Cited at Page: 31]

[152] D. V. Lindley, “Dynamic programming and decision theory,” *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 10, no. 1, pp. 39–51, 1961. [Cited at Page: 32]

[153] J. P. Gilbert and F. Mosteller, “Recognizing the maximum of a sequence,” in *Selected Papers of Frederick Mosteller*. Springer, 2006, pp. 355–398. [Cited at Page: 32]

[154] W. T. Rasmussen and S. R. Pliska, “Choosing the maximum from a sequence with a discount function,” *Applied Mathematics and Optimization*, vol. 2, no. 3, pp. 279–289, 1975. [Cited at Page: 32]

[155] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, “How To Backdoor Federated Learning,” *arXiv preprint arXiv:1807.00459*, 7 2018. [Online]. Available: http://arxiv.org/abs/1807.00459 [Cited at Page: 32 and 113]

[156] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, “A survey on security and privacy of federated learning,” *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021. [Cited at Page: 32 and 113]

[157] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, 2019. [Cited at Page: 32, 37, and 113]

[158] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical Secure Aggregation for Federated Learning on User-Held Data,” in *Conference on Neural Information Processing Systems (NIPS)*, 2016. [Cited at Page: 32, 37, and 113]

[159] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, “Verifynet: Secure and verifiable federated learning,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019. [Cited at Page: 32 and 113]

[160] C. Ma, J. Li, M. Ding, H. H. Yang, F. Shu, T. Q. S. Quek, and H. V. Poor, “On safeguarding privacy and security in the framework of federated learning,” *IEEE network*, vol. 34, no. 4, pp. 242–248, 2020. [Cited at Page: 32 and 113]

[161] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, “Machine learning in wireless sensor networks: Algorithms, strategies, and applications,” *IEEE Communications Surveys and Tutorials*, vol. 16, no. 4, pp. 1996–2018, 2014. [Cited at Page: 32 and 44]

[162] C.-W. Tsai, C.-F. Lai, M.-C. Chiang, and L. T. Yang, “Data mining for internet of things: A survey,” *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 77–97, 2013. [Cited at Page: 33 and 34]

[163] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, “A survey on the edge computing for the Internet of Things,” *IEEE Access*, vol. 6, pp. 6900–6919, 2017. [Cited at Page: 33]

[164] “REGULATIONS REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation),” Tech. Rep. [Cited at Page: 34 and 112]

[165] “California Consumer Privacy Act (CCPA), AB-375,” Tech. Rep. [Cited at Page: 34 and 112]

[166] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, “A taxonomy and survey of energy-efficient data centers and cloud computing systems,” in *Advances in computers*. Elsevier, 2011, vol. 82, pp. 47–111. [Cited at Page: 34]

[167] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” *Hawaii International Conference on System Sciences*, vol. 00, no. c, p. 223, 2000. [Cited at Page: 34]

[168] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, “Energy conservation in wireless sensor networks: A survey,” *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2008.06.003 [Cited at Page: 34]

[169] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, “Distributed regression: an efficient framework for modeling sensor network data,” in *International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004, pp. 1–10. [Cited at Page: 34]

[170] A. A. Abbasi and M. Younis, “A survey on clustering algorithms for wireless sensor networks,” *Computer Communications*, vol. 30, no. 14-15, pp. 2826–2841, 2007. [Cited at Page: 34]

[171] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," *IEEE INFOCOM*, vol. 3, no. C, pp. 1713–1723, 2003. [Cited at Page: 34]

[172] J. Zhang and K. B. Letaief, "Mobile Edge Intelligence and Computing for the Internet of Vehicles," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 246–261, 2020. [Cited at Page: 35]

[173] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless Network Intelligence at the Edge," *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2204–2239, 2019. [Cited at Page: 35]

[174] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 1, pp. 416–464, 2018. [Cited at Page: 35]

[175] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018. [Cited at Page: 36]

[176] C. Hardy, E. Le Merrer, and B. Sericola, "Distributed deep learning on edge-devices: Feasibility via adaptive compression," in *IEEE International Symposium on Network Computing and Applications (NCA)*, 2017. [Cited at Page: 36]

[177] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017. [Cited at Page: 36]

[178] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017. [Cited at Page: 36]

[179] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2, pp. 869–904, 2020. [Cited at Page: 36]

[180] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," *Proceedings of the IEEE*, vol. 107, no. 8, 2019. [Cited at Page: 36 and 78]

[181] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018. [Cited at Page: 36]

[182] J. B. Predd, S. B. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56–69, 2006. [Cited at Page: 36]

[183] K. Kopparapu and E. Lin, "FedFMC: Sequential Efficient Federated Learning on Non-iid Data," *arXiv preprint arXiv:2006.10937*, 2020. [Online]. Available: https://arxiv.org/abs/2006.10937 [Cited at Page: 37]

[184] F. Sattler, S. Wiedemann, K. R. Muller, and W. Samek, "Robust and Communication-Efficient Federated Learning from Non-i.i.d. Data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2020. [Cited at Page: 37 and 114]

[185] J. Konečný, B. McMahan, and D. Ramage, "Federated Optimization:Distributed Optimization Beyond the Datacenter," *arXiv preprint arXiv:1511.03575*, 11 2015. [Online]. Available: http://arxiv.org/abs/1511.03575 [Cited at Page: 37 and 79]

[186] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated Learning in Mobile Edge Networks: A Comprehensive Survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020. [Cited at Page: 37]

[187] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020. [Online]. Available: http://arxiv.org/abs/1908.07873 [Cited at Page: 37]

[188] E. b. P. Kairouz and H. B. McMahan, "Advances and Open Problems in Federated Learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1, 12 2021. [Online]. Available: http://arxiv.org/abs/1912.04977http://www.nowpublishers.com/article/Details/MAL-083 [Cited at Page: 37, 113, and 141]

[189] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Federated Learning of Deep Networks using Model Averaging," *arXiv preprint arXiv:1602.05629*, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629 [Cited at Page: 37, 112, and 115]

[190] ——, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *arXiv preprint arXiv:*, 2017. [Online]. Available: http://arxiv.org/abs/1602.05629v3 [Cited at Page: 37 and 112]

[191] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication

Efficiency," *arXiv preprint arXiv:1610.05492*, 2016. [Online]. Available: http://arxiv.org/abs/1610.05492 [Cited at Page: 37]

[192] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, no. December, pp. 6–28, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1368893 [Cited at Page: 40]

[193] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: A survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 2, pp. 551–591, 2013. [Cited at Page: 40]

[194] S. K. Singh, M. P. Singh, D. K. Singh, and others, "Routing protocols in wireless sensor networks–A survey," *International Journal of Computer Science & Engineering Survey (IJCSES)*, vol. 1, no. 2, pp. 63–83, 2010. [Cited at Page: 40]

[195] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, 1999, pp. 174–185. [Cited at Page: 41]

[196] S. Lindsey and C. S. Raghavendra, "PEGASIS: Power-efficient gathering in sensor information systems," in *IEEE Aerospace Conference*, vol. 3, 2002, p. 3. [Cited at Page: 41]

[197] O. Younis and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004. [Cited at Page: 41]

[198] A. Manjeshwar and D. P. Agrawal, "TEEN : A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks ." in *International Parallel and Distributed Processing Symposium. (IPDPS)*, 2001, pp. 2009–2015. [Cited at Page: 41]

[199] ——, "APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks," in *International Parallel and Distributed Processing Symposium. (IPDPS)*, 2002, p. 0195b. [Cited at Page: 41]

[200] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *International Conference on Information Technology: Coding and Computing (ITCC)*, vol. 2, 2005, pp. 8–13. [Cited at Page: 41]

[201] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki, "Practical data compression in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 37–59, 2012. [Cited at Page: 41]

[202] G. M. Dias, B. Bellalta, and S. Oechsner, “A survey about prediction-based data reduction in wireless sensor networks,” *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, 2016. [Cited at Page: 41, 43, and 44]

[203] H. Jiang, S. Jin, and C. Wang, “Prediction or Not? An Energy-Efficient Framework for Clustering-based Data Collection in Wireless Sensor Networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 1064–1071, 2011. [Cited at Page: 41]

[204] R. Rajagopalan and P. K. Varshney, “Data aggregation techniques in sensor networks: A survey,” *Electrical Engineering and Computer Science*, no. 22, 2006. [Cited at Page: 41]

[205] D. Tulone and S. Madden, “PAQ: Time series forecasting for approximate query answering in sensor networks,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3868 LNCS, pp. 21–37, 2006. [Cited at Page: 44]

[206] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, “Evaluating Probabilistic Queries over Imprecise Data,” in *ACM SIGMOD International Conference on Management of Data*, 2003, pp. 551–562. [Cited at Page: 44]

[207] D. J. McCorrie, E. Gaura, K. Burnham, N. Poole, and R. Hazelden, “Predictive data reduction in wireless sensor networks using selective filtering for engine monitoring,” in *Wireless Sensor and Mobile Ad-Hoc Networks*. Springer, 2015, pp. 129–148. [Cited at Page: 44]

[208] G. Li and Y. Wang, “Automatic ARIMA modeling-based data aggregation scheme in wireless sensor networks,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, pp. 1–13, 2013. [Cited at Page: 44]

[209] F. A. Aderohunmu, G. Paci, D. Brunelli, J. D. Deng, and L. Benini, “Prolonging the lifetime of wireless sensor networks using light-weight forecasting algorithms,” in *IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2013, pp. 461–466. [Cited at Page: 44]

[210] S. Santini and K. Römer, “An adaptive strategy for quality-based data reduction in wireless sensor networks,” *International Conference on Networked Sensing Systems (INSS)*, no. December 2014, p. 29–36, 2006. [Cited at Page: 49 and 60]

[211] S. Haykin and B. Widrow, *Least-mean-square adaptive filters*. John Wiley & Sons Ltd, 2003, vol. 31. [Cited at Page: 50]

[212] B. Farhang-Boroujeny, *Adaptive filters: theory and applications*. John Wiley & Sons Ltd, 2013. [Cited at Page: 50]

[213] H. Robbins, D. Sigmund, and Y. Chow, "Great expectations: the theory of optimal stopping," *Houghton-Nifflin*, vol. 7, pp. 631–640, 1971. [Cited at Page: 53]

[214] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 1986, vol. 26. [Cited at Page: 55]

[215] S. Madden, "Intel Berkeley research lab data," 2003. [Online]. Available: http://db.csail.mit.edu/labdata/labdata.html [Cited at Page: 56]

[216] "Weather Underground." [Online]. Available: https://www.wunderground.com/api/ [Cited at Page: 57 and 97]

[217] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, USA:, 1994, pp. 359–370. [Cited at Page: 59 and 93]

[218] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 3 2005. [Cited at Page: 59, 93, and 93]

[219] C. Tofallis, "A better measure of relative prediction accuracy for model selection and model estimation," *Journal of the Operational Research Society*, vol. 66, no. 8, pp. 1352–1362, 2015. [Cited at Page: 59]

[220] Y.-A. Le Borgne, S. Santini, and G. Bontempi, "Adaptive model selection for time series prediction in wireless sensor networks," *Signal Processing*, vol. 87, no. 12, pp. 3010–3020, 2007. [Cited at Page: 60]

[221] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh, "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals," *Data mining and knowledge discovery*, vol. 1, no. 1, pp. 29–53, 1997. [Cited at Page: 63]

[222] A. Arasu, S. Babu, and J. Widom, "The CQL continuous query language: semantic foundations and query execution," *The VLDB Journal*, vol. 15, no. 2, pp. 121–142, 2006. [Cited at Page: 63]

[223] S. Babu and J. Widom, "Continuous queries over data streams," in *ACM SIGMOD International Conference on Management of Data*, vol. 30, no. 3, 2001, pp. 109–120. [Cited at Page: 63]

[224] M. Kuhn and K. Johnson, *Applied Predictive Modeling.* New York, NY: Springer New York, 2013, vol. 26. [Online]. Available: http://link.springer.com/10.1007/978-1-4614-6849-3 [Cited at Page: 64]

[225] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, “All one needs to know about fog computing and related edge computing paradigms: A complete survey,” *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019. [Cited at Page: 78]

[226] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, K. Ha, K. Elgazzar, P. Pillai, R. Klatzky, and others, “An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance,” in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, 2017, pp. 1–14. [Cited at Page: 78]

[227] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, “Edge-centric Computing,” *ACM SIGCOMM Computer Communication Review*, 2015. [Cited at Page: 79]

[228] G. Kamath, P. Agnihotri, M. Valero, K. Sarker, and W. Z. Song, “Pushing analytics to the edge,” in *IEEE Global Communications Conference (GLOBECOM)*, 2016. [Cited at Page: 79]

[229] S. Teerapittayanon, B. McDanel, and H.-T. Kung, “Distributed deep neural networks over the cloud, the edge and end devices,” in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 328–339. [Cited at Page: 79]

[230] H. Wang and C. Li, “Distributed Quantile Regression over Sensor Networks,” *IEEE Transactions on Signal and Information Processing over Networks*, 2018. [Cited at Page: 79]

[231] M. Gabel, D. Keren, and A. Schuster, “Monitoring Least Squares Models of Distributed Streams,” in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2015. [Cited at Page: 79]

[232] L. Bottou and O. Bousquet Google Zürich, “The Tradeoffs of Large Scale Learning,” Tech. Rep. [Cited at Page: 79]

[233] C. Anagnostopoulos, “Edge-centric inferential modeling and analytics,” *Journal of Network and Computer Applications*, vol. 164, no. September 2019, p. 102696, 2020. [Cited at Page: 79 and 97]

[234] E. Aleksandrova, C. Anagnostopoulos, and K. Kolomvatsos, "Machine Learning Model Updates in Edge Computing: An Optimal Stopping Theory Approach," in *International Symposium on Parallel and Distributed Computing (ISPDC)*, 2019, pp. 1–8. [Cited at Page: 79]

[235] A. Boulis, S. Ganeriwal, and M. B. Srivastava, "Aggregation in sensor networks: An energy-accuracy trade-off," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 317–331, 2003. [Cited at Page: 80]

[236] G. Claeskens, N. L. Hjort, and others, *Model selection and model averaging*. Cambridge University Press, 2008. [Cited at Page: 80]

[237] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa, "Ensemble approaches for regression," *ACM Computing Surveys*, 2012. [Cited at Page: 80]

[238] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Ensemble diversity measures and their application to thinning," *Information Fusion*, vol. 6, no. 1, pp. 49–62, 2005. [Cited at Page: 80]

[239] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002. [Cited at Page: 80]

[240] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *International Conference on Machine Learning (ICML)*, vol. 97, 1997, pp. 211–218. [Cited at Page: 80]

[241] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012. [Cited at Page: 80]

[242] D. E. Goldberg, "Genetic algorithms in search," *Optimization, and MachineLearning*, 1989. [Cited at Page: 80]

[243] F. Shen and O. Hasegawa, "An adaptive incremental LBG for vector quantization," *Neural Networks*, 2006. [Cited at Page: 85]

[244] I. K. Fodor, "A survey of dimension reduction techniques," Lawrence Livermore National Lab., CA (US), Tech. Rep., 2002. [Cited at Page: 87]

[245] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction," *Journal of Applied Science and Technology Trends*, vol. 1, no. 2, pp. 56–70, 2020. [Cited at Page: 87]

[246] N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," *Neural computation*, vol. 9, no. 7, pp. 1493–1516, 1997. [Cited at Page: 87]

[247] H. R. Choi and T. Y. Kim, "Directional dynamic time warping for gesture recognition," in *ACM International Conference Proceeding Series*, vol. 300, 2017, pp. 22–25. [Cited at Page: 94]

[248] M. H. Ko, G. West, S. Venkatesh, and M. Kumar, "Using dynamic time warping for online temporal fusion in multisensor systems," *Information Fusion*, vol. 9, no. 3, pp. 370–388, 2008. [Cited at Page: 94]

[249] M. Wöllmer, M. Al-Hames, F. Eyben, B. Schuller, and G. Rigoll, "A multidimensional dynamic time warping algorithm for efficient multimodal fusion of asynchronous data streams," *Neurocomputing*, vol. 73, no. 1-3, pp. 366–380, 2009. [Cited at Page: 94]

[250] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measures," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 216–225. [Cited at Page: 94]

[251] D. M. Gavrila and L. S. Davis, "3-D model-based tracking of human upper body movement: a multi-view approach," in *IEEE International Conference on Computer Vision*, 1995, pp. 253–258. [Cited at Page: 94]

[252] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 262–270. [Cited at Page: 94]

[253] Y. Sakurai, C. Faloutsos, and M. Yamamuro, "Stream monitoring under the time warping distance," in *International Conference on Data Engineering*, 2007, pp. 1046–1055. [Cited at Page: 94]

[254] E. J. Keogh and M. J. Pazzani, "Scaling up dynamic time warping for datamining applications," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 285–289. [Cited at Page: 94]

[255] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco, "Practical Data Prediction for Real-World Wireless Sensor Networks," *IEEE Transactions on Knowledge and Data Engineering*, 2015. [Cited at Page: 95]

[256] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, “Time-series clustering–a decade review,” *Information Systems*, vol. 53, pp. 16–38, 2015. [Cited at Page: 95]

[257] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, “Data security and privacy-preserving in edge computing paradigm: Survey and open issues,” *IEEE Access*, vol. 6, pp. 18 209–18 237, 2018. [Cited at Page: 112]

[258] V. Kulkarni, M. Kulkarni, and A. Pant, “Survey of Personalization Techniques for Federated Learning,” *arXiv preprint arXiv:2003.08673*, 2020. [Online]. Available: http://arxiv.org/abs/2003.08673 [Cited at Page: 113]

[259] V. Zantedeschi, A. Bellet, and M. Tommasi, “Fully Decentralized Joint Learning of Personalized Models and Collaboration Graphs,” in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 108, 2020, pp. 864–874. [Online]. Available: http://arxiv.org/abs/1901.08460 [Cited at Page: 113]

[260] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, “Decentralized collaborative learning of personalized models over networks,” in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017. [Cited at Page: 113]

[261] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, “Personalized and private peer-to-peer machine learning,” *International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 84, pp. 473–481, 2018. [Cited at Page: 113]

[262] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage, “Federated Evaluation of On-device Personalization,” *arXiv preprint arXiv:1910.10252*, 2019. [Online]. Available: https://arxiv.org/abs/1910.10252 [Cited at Page: 113]

[263] P. P. Liang, T. Liu, L. Ziyin, R. Salakhutdinov, and L.-P. Morency, “Think Locally , Act Globally : Federated Learning with Local and Global Representations,” in *Workshop on Federated Learning (NeurIPS)*, 2019, pp. 1–17. [Online]. Available: https://arxiv.org/abs/2001.01523 [Cited at Page: 114 and 119]

[264] A. Fallah, A. Mokhtari, and A. Ozdaglar, “Personalized federated learning: A meta-learning approach,” in *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. [Cited at Page: 114]

[265] Y. Jiang, J. Konečný, K. Rush, and S. Kannan, “Improving Federated Learning Personalization via Model Agnostic Meta Learning,” *arXiv preprint arXiv:1909.12488*, 2019. [Online]. Available: https://arxiv.org/abs/1909.12488 [Cited at Page: 114]

[266] F. Hanzely and P. Richtárik, "Federated Learning of a Mixture of Global and Local Models," *arXiv preprint arXiv:2002.05516*, 2020. [Online]. Available: http://arxiv.org/abs/2002.05516 [Cited at Page: 114]

[267] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive Personalized Federated Learning," *arXiv preprint arXiv:2003.13461*, 2020. [Online]. Available: http://arxiv.org/abs/2003.13461 [Cited at Page: 114 and 119]

[268] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," *Proceedings of the IEEE*, vol. 107, no. 8, 2019. [Cited at Page: 114]

[269] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205 – 1221, 2019. [Online]. Available: http://arxiv.org/abs/1804.05271 [Cited at Page: 114]

[270] A. Li, J. Sun, B. Wang, L. Duan, S. Li, Y. Chen, and H. Li, "Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets," *arXiv preprint arXiv:2008.03371*, 2020. [Online]. Available: https://arxiv.org/abs/2008.03371 [Cited at Page: 114]

[271] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264. [Cited at Page: 118]

[272] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009. [Cited at Page: 118]

[273] J. Durbin, "A simple and efficient simulation smoother for state space time series analysis," *Biometrika*, vol. 89, no. 3, pp. 603–616, 8 2002. [Online]. Available: https://academic.oup.com/biomet/article-lookup/doi/10.1093/biomet/89.3.603 [Cited at Page: 120]

[274] J. Lockl, V. Schlatt, A. Schweizer, N. Urbach, and N. Harth, "Toward Trust in Internet of Things Ecosystems: Design Principles for Blockchain-Based IoT Applications," *IEEE Transactions on Engineering Management*, 2020. [Cited at Page: 141]

[275] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2019. [Cited at Page: 141]

[276] J. Passerat-Palmbach, T. Farnan, R. Miller, M. S. Gross, H. L. Flannery, and B. Gleim, “A blockchain-orchestrated federated learning architecture for healthcare consortia,” *arXiv preprint arXiv:1910.12603*, 2019. [Cited at Page: 141]

[277] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, “Federated Learning for Ultra-Reliable Low-Latency V2V Communications,” in *IEEE Global Communications Conference (GLOBECOM)*, 2018. [Cited at Page: 141]

[278] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, “Federated learning of predictive models from federated Electronic Health Records,” *International Journal of Medical Informatics*, vol. 112, 2018. [Cited at Page: 141]

[279] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, and A. Feng, “Privacy-Preserving Federated Brain Tumour Segmentation,” in *Machine Learning in Medical Imaging*, vol. 11861 LNCS, 2019, pp. 133–141. [Online]. Available: http://arxiv.org/abs/1910.00962 [Cited at Page: 141]

[280] W. Jeong, J. Yoon, E. Yang, and S. J. Hwang, “Federated Semi-Supervised Learning with Inter-Client Consistency,” in *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: http://arxiv.org/abs/2006.12097 [Cited at Page: 141]

[281] J. Jiang and L. Hu, “Decentralised federated learning with adaptive partial gradient aggregation,” *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 230–236, 2020. [Cited at Page: 141]