Local distributed lifelong learning system architecture applied to smart buildings

Angan Mitra

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Angan MITRA

Thèse dirigée par **Denis TRYSTRAM**,Professeur

préparée au sein du **Laboratoire d'Informatique de Grenoble** dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Architecture de système de formation continue distribuée locale appliquée aux bâtiments intelligents

Local distributed lifelong learning system architecture applied to smart buildings

Thèse soutenue publiquement le **10 mars 2022**,
devant le jury composé de :

Monsieur FREDERIC DESPREZ
Directeur de recherche, INRIA CENTRE GRENOBLE-RHONE-ALPES, Président
Monsieur PHILIPPE LALANDA
Professeur des Universités , UNIVERSITE GRENOBLE ALPES, Examinateur
Monsieur THANG KIM NGUYEN
Maître de conférences HDR, UNIVERSITE EVRY VAL D'ESSONNE, Examinateur
Monsieur ALEXANDRE VAN KEMPEN
Ingénieur docteur, QARNOT COMPUTING, Examinateur
Monsieur CHRISTOPHE CERIN
Professeur des Universités, UNIVERSITE SORBONNE PARIS NORD, Rapporteur
Madame NATHALIE HERNANDEZ
Professeur des Universités, UNIVERSITE TOULOUSE 2 - JEAN JAURES, Rapporteure

Local distributed lifelong learning system architecture applied to smart buildings

Thesis report submitted in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

by

Angan Mitra

Under the supervision of

Denis Trystram, Yanik Ngoko

Acknowledgments

I express my deepest gratitude towards my *guardian angel* Professor **Denis Trystram** for the constant help and encouragement from the starting of the thesis work. I am fortunate to have worked under the supervision of **Yanik Ngoko** who has been phenomenal in striking the right balance between guidance and freedom to explore on my own. Inputs from **Qarnot Computing** has been fundamental in setting the road-map relevant for industrial usage. I have been fortunate to have my supervisors who never failed to provide me with timely reviews and suggestions wherever required. Thank you for having faith in me.

Probably it will be unfair if I undermine the immense love and support from my parents and friends, especially when the world witnessed the first and hopefully the last pandemic of this century. Notably, I am grateful to *Swarnali Roy*, a doctoral student of marketing at University of Grenoble Alpes for making me aware on the physiological aspect of technology acceptance, a key ingredient for framing this thesis work. Last but not the least, thank you Louis Closson for proof reading the manuscript and pointing out the grammatical errors to improve the readability of the document.

Table des matières

Liste des tableaux

Table des figures

Abstract

Buildings both residential and commercial together consume close to half of the world's total energy produced and is growing at a non decreasing pace. So efficient resource utilization forms the primary motivation behind integrating smartness into a brick and mortar structure. Although active from early 2000's, literature survey reveals that there are significant business gaps that bottlenecks smart building development. **Data privacy, high capital investments** and **obscure monetary benefits** are the major factors that impede the motivation to integrate smartness in a building.

This work introduces the idea of a **zero sensor intelligence** by embedding human-space interaction models on a graph based abstraction of a building. Spectral decomposition of the semantically enriched connected graph helps in ranking multiple spaces with regard to temporal importance or likely energy dissipation. Next, we extend the problem of optimal sensor placement to finding the **minimal sensing group** that can robustly approximate missing sensors to provide complete spatial coverage. Lifelong learning mechanism is used to identify robust sensor placement configurations and continually learn a metric of hardness to approximate. This culminates in a novel **pre-integration platform** to bring clarity on at-least how many sensors are to be installed and where. Once sensors are installed, the platform enforces data privacy by design, being inspired from the philosophy of **edge computing** to process data as close as possible to the generation site. In a nutshell, the work lays the blueprint of a generic smart building solution with lesser sensors, lower carbon footprint and auto-updating models with strictly localised raw data at edge.

Abstract

Les bâtiments résidentiels et commerciaux consomment ensemble près de la moitié de l'énergie totale produite dans le monde et croissent à un rythme non décroissant. L'utilisation efficace des ressources constitue donc la principale motivation derrière l'intégration de l'intelligence dans une structure de brique et de mortier. Bien qu'elle soit active depuis le début des années 2000, une étude de la littérature révèle qu'il existe des lacunes commerciales importantes qui entravent le développement de bâtiments intelligents. **Confidentialité des données, investissements en capital élevés** et **bénéfices monétaires obscurs** sont les principaux facteurs qui entravent la motivation à intégrer l'intelligence dans un bâtiment.

Ce travail introduit l'idée d'une **intelligence sans capteur** en intégrant des modèles d'interaction homme-espace sur une abstraction basée sur un graphe d'un bâtiment. La décomposition spectrale du graphe connecté enrichi sémantiquement aide à classer plusieurs espaces en fonction de l'importance temporelle ou de la dissipation d'énergie probable. Ensuite, nous étendons le problème du placement optimal des capteurs à la recherche du **groupe de détection minimal** qui peut approximer de manière robuste les capteurs manquants pour fournir une couverture spatiale complète. Le mécanisme d'apprentissage tout au long de la vie est utilisé pour identifier des configurations de placement de capteurs robustes et apprendre en permanence une métrique de dureté à approximer. Cela aboutit à une nouvelle **plate-forme de pré-intégration** pour clarifier au moins combien de capteurs doivent être installés et où. Une fois les capteurs installés, la plateforme applique la confidentialité des données dès sa conception, s'inspirant de la philosophie du **edge computing** ou traite les données au plus près du site de génération. En bref, le travail pose le modèle d'une solution générique de bâtiment intelligent avec moins de capteurs, une empreinte carbone plus faible et des modèles de mise à jour automatique avec des données brutes strictement localisées à la périphérie.

Chapitre 1

Introduction

1990 marked the year when Tim Berners-Lee established World Wide Web and 15 years later, the internet started offering online services instead of following the classical *install and run the program locally* method. This paradigm shift has been made possible largely due to hardware improvement achieving high frequency processors, faster I/O busses, and definitely technological leaps to increase the network speed.

1.1 Evolution of Edge Computing

As of 2021, Cloud computing is still the major service provider of utility computing where end users are charged in pay-as-you-go manner for executing software in a remote data center. Such cloud services are typically offered in three flavours : Software-as-a-Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). IaaS provides organizations with completely cloud-hosted servers and an associated operating system (OS) which gives the developers the ability to organize their workloads, data distribution and even place specific hardware request to the provider. The gain in using IaaS for institutions comes from avoiding the time, energy and cost due to in-premises server stacks. PaaS gives organizations a fully featured platform in which they can develop, test, and deploy their applications from the cloud without bothering about the underlying hardware or networking, and security (and potential infrastructure failures). SaaS only deals software deployment while server management and data

retention responsibilities are delegated to the data center. Typically data centers are made up of multiple servers hosting/running internet services and consume close to 5% of the world's total energy produced in the scale of Mega-Watts (MW). The classical design of cloud services possess a major bottleneck in terms of response latency, energy footprint and cooling cost.

Now let's track the evolution of low powered computing for the last 30 years by observing the market segment that are primary users of cloud services. To give an example of a perspective scale, the best computers in 1960 had maximum 20 kB of hard disk space, while some of today's best phone capacity is beyond 1 TB (1073741824 Kb). Notably in 1999, the term "Internet of Things" was first introduced in reference to automated supply-chain management. The concept of enabling a computer to sense information without human intervention was then applied to other fields such as healthcare, home technology, environmental engineering, and transportation. Success in sensing technology led to embedded cameras and sensors that act as data sources for running a plethora of applications like object detection, photo editing, indoor localization and much more. While mobile phones, tablets, laptops and desktops grew to cater to personal or office needs, the manufacturing industry too leveraged the power of computing and connectivity.

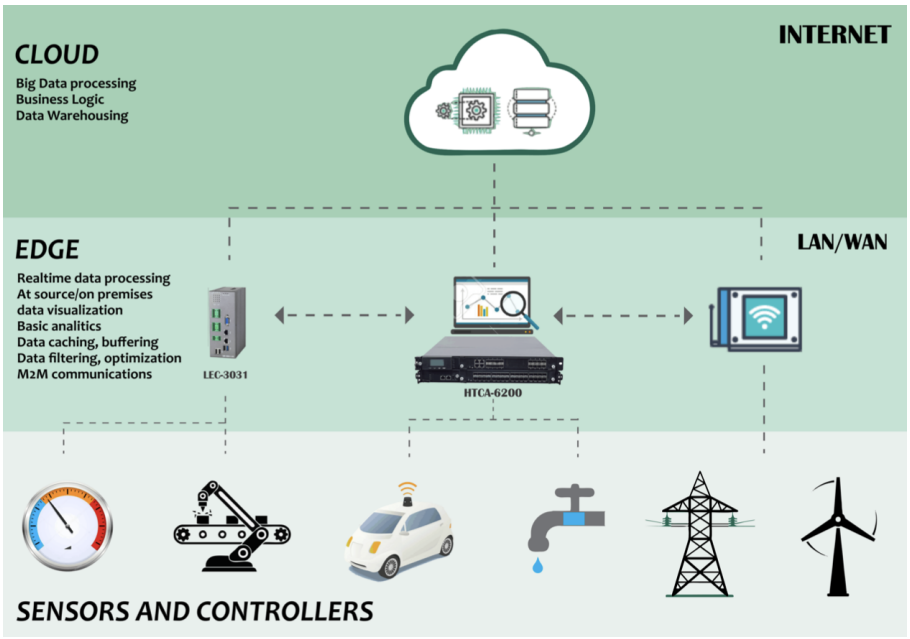


FIGURE 1.1 – Distinction between role of Cloud and Edge computing to process sensor streams and drive actuators. Reference :`www.lannerusa.com`

So far, the de-facto modus operandi for embedded applications is to upload observed data and trigger the processing in servers by calling API end points hosted in cloud. This approach comes with an alarming repercussion in terms of scalability on the context of an explosive growth and purchase of embedded systems. A ground report by CISCO [5] anticipates more than 850 ZB of data generation worldwide through IoT devices by 2022 which raises significant operating concern for cloud volumes. Around 2015, edge computing [6] was born out of the philosophy to process data as close to the generation site as possible as shown in Figure 1.1. IBM claims that the proximity to data at its source can deliver strong business benefits, including faster insights, improved response times and better bandwidth availability. In 2010, a French startup named Qarnot Computing came up with the idea of world's first computing heater that has evolved as a data center at edge. The encapsulation of computing units as a heating element further saves the explicit cost of cooling, since the thermal energy is dissipated for warming up spaces in buildings and naturally promotes a promising circular economy. Typically a smart heater is equipped with ambient sensors primarily to sense the environment and control internal parameters for a dynamic control. For instance, a heater without an indoor temperature sensor will fail to understand if it has overheated or under-heated its surroundings during operation. Popularity of sensor enabled embedded systems has spurred the growth of IoT and provides the support to develop useful data-driven applications.

1.2 Smartness for Buildings

Cheap availability of ambient sensors, smart lighting and energy meters has facilitated in an adhoc incorporation of IoT products in buildings to generate intelligence. Often the optimal solution is found by learning a data driven representation to solve business problems like occupancy detection, security alerting, predictive control of thermal units, forecasting sensor values, etc. Although integrating smartness is becoming a hot topic for reducing carbon footprint at a city scale, historically the majority of buildings were not designed to cater to ambient intelligence. Advances in Internet of Things (IoT) have led to buildings as an ensemble of interconnected systems of sensing and control. The big motivation for adding a layer of smartness is to solve the problems of energy inefficiency, maintenance cost, space utilization and resource

management in a building. Typically a building architecture constitutes of semantically partitioned space with walls embedding doors, stairs, elevators and ventilation through windows. The design and description of a building are usually saved in an ISO file format which is essentially an incremental and recursive technique to generate and store 3D co-ordinates. However, it can not be assumed that all architects will design an identical building in the exact same way, which can lead to ambiguity in parsing. It is non trivial to guess the connections from a building file for which computer aided parsing techniques are used to generate structured meta data. One can demarcate sensors distributed over spaces on the metadata for ease of querying like "Which rooms have CO_2 sensors ?"

The notion of *smartness* can be perceived through solutions of a spatio-temporal query space surrounding a building. The three key components of the Query space are spatial awareness, temporal observations and inference extraction. A query is **spatial** in nature if the resolution process only relies on geometric information about building elements like door, window, wall, stair, room, corridor, spaces, ceiling, floor, elevator etc. Spatial queries can be used to solve dynamic path formulation and maximal packing problems such as

— What is the average carpet area per floor ?

— What is the pathway to navigate to a space or exit a floor ?

— How well is a space ventilated without using external ambient controllers ?

— What part of the building is most likely to illuminated at day ?

A building is constantly interacting with environmental variables like sunlight, humidity, quality of air etc. Equipping zones with sensors help to record or stream temporal data for answering Spatio Temporal queries such as :

— What is the average power consumption during working hours in a building ?

— How does the average CO_2 level vary between weekdays and weekends ?

— What is the average temperature difference between exterior and indoor environments when the Air Conditioning unit is on ?

Although the two query types that have been introduced so far mostly retrieve domain knowledge and perform data comparison or value aggregation to generate the answer, they often lack predictive capability. Instead machine learnt techniques embed algorithms to regress future predictions or classify into discreet classes to solve for non trivial answers to questions

like

— Is there a way to detect unnecessary sensors and lower the footprint of the smart building solution itself ?
— What are the best spots in a building to place sensors, and what type they should be ?
— Is the building auto aware of the capital and operating cost to predict the break even point for implementing smartness ?
— Can there be a system design that can be data safe and process data on the fly ?

1.3 Key Contributions

Unfortunately the acceptance traction of smart building technology is facing serious bottlenecks due to installation costs, obscure returns leading to a distant break-even point and data privacy concerns. The work addresses three major pain points pertaining to smart building technology acceptance by end-users in an era of Internet of Things. **Firstly** we propose a no sensor or solely spatial intelligent framework that is built on top of a graph based abstraction for a building's architecture. One shot semantic meta data enrichment along with human-space interaction models are developed for a spectral analysis of indoor dynamics. This gives the holistic picture of a building and has the potential to model conditional interaction scenarios to understand spatio-temporal energy dissipation. **Next**, we investigate if lack of spatiotemporal awareness can lead to excessive sensors or non-optimal distribution in a building. The motivation here is to lower capital and operating cost as well as energy footprint of the smart building solution itself. The work introduces a pre-integration framework to measure the ease of approximating a sensor stream through auto-creation of relevant machine learning models. Evolutionary computing is used to solve a multi-objective optimization to come up with a minimalist sensing solution known as the Virtual Sensing Field. Experiments are carried on buildings from France and Thailand that are situated in cold and warm climatic zones respectively. The results show at-least a 60 % sensor reduction with the Virtual Sensor Field able to run occupancy detection in an unsupervised setting. **Thirdly**, we focus on executing computations as close as possible to the data generation site also known as the edge layer. This leads to in-house data retention and system maintained knowledge sharing policies between spaces. By design,

we prohibit raw data sharing amongst spaces, thereby removing the classical obscurity on data privacy concerns for a building management system. In this regard, we introduce a mechanism for federated personalization with no active mediators and a lightweight algorithm to support peer to peer online learning. In a nutshell, the work exploits the concept of "less is more" to plan methodically, bring down the initial investment and recurring expense for a smart building solution. The resulting system blueprint is spatially aware, temporally observant, reactive by nature and data safe by design.

1.3.1 Document Structure

The remaining of the work is arranged as follows :

— Chapter 2 establishes the context of the work within the recent developments on building management system and artificial intelligence. The work additionally reviews the acceptance of technology

— Chapter 3 abstracts the complexity in building information modelling into a graph-based representation. The work introduces the concept of zero-sensor intelligence by proposing graph-based human-space interaction models. Spectral decomposition of such enriched metadata is found useful to detect dynamic places of interest in real time. The experiments are carried out on the data from the office of Qarnot Computing, Paris.

— Next Chapter 4 investigates about a layer of cost and energy cutting to optimally distribute sensors over spaces. Artificially intelligent substitutes for a fraction of actual sensors for 100 % data coverage. Distributed learning with localised in house data is performed for experiments from a seven storey building data in Thailand. The work investigates effects over 25 zones with 3 ambient sensors (humidity, temperature, luminosity) and 3 energy channels (illumination, appliance, air-conditioning).

— Chapter 5 imposes stricter data privacy rules with no raw data transmission from site and explores algorithms to predict the indoor thermal profile of a building. This is of particular interest to Qarnot Computing, that offers edge computing through smart heaters that can be installed on any building with an internet and power supply. This forms the context to explore federated and decentralized learning techniques to incrementally have better in-situ models.

— Chapter 6 summarises the key components of a generic smart building management systems with a special focus on data privacy, non-intrusiveness, and utilization of edge computing. One will also find the scope of future work to further modify the system on counts of accuracy, decentralization and energy footprint.

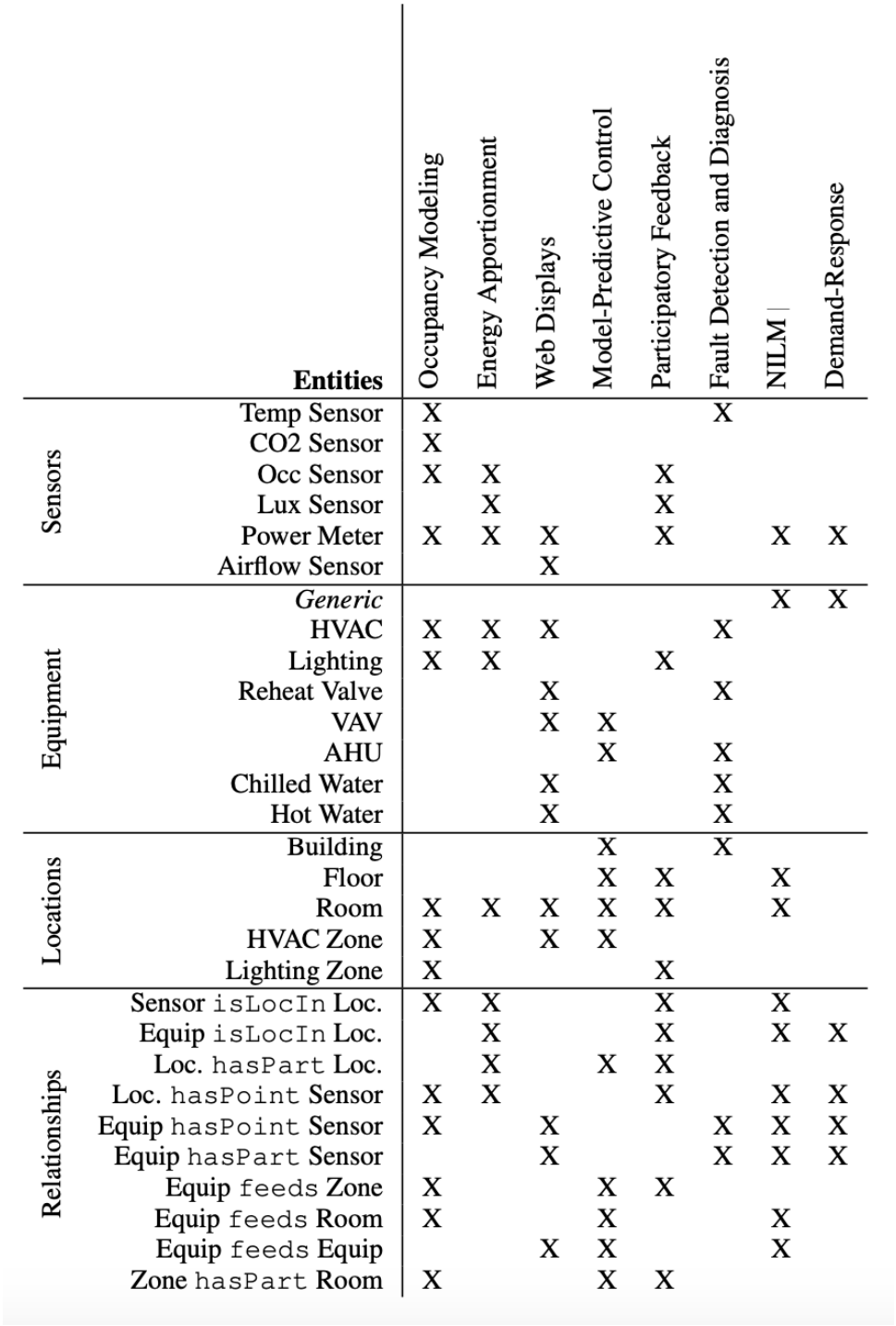
Chapitre 2

Related Works

In this chapter we provide background knowledge to the reader to understand the formulation developed in Chapters 3 - 5. This chapter contains an overview of building abstraction formats, utility of management systems as well as a brief introduction to machine learning, with emphasis on training and inference on edge. Furthermore, to increase the presentation flow and quality of this chapter, many recent and related works are also presented in a intertwined manner along the text of this chapter.

2.1 Building Management Systems

Smart applications [7] for buildings has been developed mainly for monitoring, analysis, and control of Heating Ventilation Air Conditioning (HVAC) units, illumination channels, appliance power etc. Enriched Building metadata [8] has been used to derive causal relations amongst spaces [9] and detect signature activities like finding faulty Air Handling Units [10] found in HVAC systems. Building Management System (BMS) [11] is a set of computer-controlled processes that monitor and act on building health and security. The initial interest for developing BMS was automated control with lower energy foot-printing. The authors [12] give a proof of concept on energy savings through automatic exchange of building geometry for building design processes. Building metadata is utilised to serve a variety of smart building applications with heterogeneous sensors as per Figure 2.1.



	Entities	Occupancy Modeling	Energy Apportionment	Web Displays	Model-Predictive Control	Participatory Feedback	Fault Detection and Diagnosis	NILM	Demand-Response
Sensors	Temp Sensor	X					X		
	CO2 Sensor	X							
	Occ Sensor	X	X			X			
	Lux Sensor		X			X			
	Power Meter	X	X	X		X		X	X
	Airflow Sensor			X					
Equipment	*Generic*							X	X
	HVAC	X	X	X			X		
	Lighting	X	X			X			
	Reheat Valve			X			X		
	VAV			X	X				
	AHU				X		X		
	Chilled Water			X			X		
	Hot Water			X			X		
Locations	Building				X		X		
	Floor				X	X		X	
	Room	X	X	X	X	X		X	
	HVAC Zone	X		X	X				
	Lighting Zone	X				X			
Relationships	Sensor `isLocIn` Loc.	X	X			X		X	
	Equip `isLocIn` Loc.		X			X		X	X
	Loc. `hasPart` Loc.		X		X	X			
	Loc. `hasPoint` Sensor	X	X			X		X	X
	Equip `hasPoint` Sensor	X		X			X	X	X
	Equip `hasPart` Sensor			X			X	X	X
	Equip `feeds` Zone	X			X	X			
	Equip `feeds` Room	X			X			X	
	Equip `feeds` Equip			X	X			X	
	Zone `hasPart` Room	X			X	X			

FIGURE 2.1 – Using metadata [1] to run smart building applications.

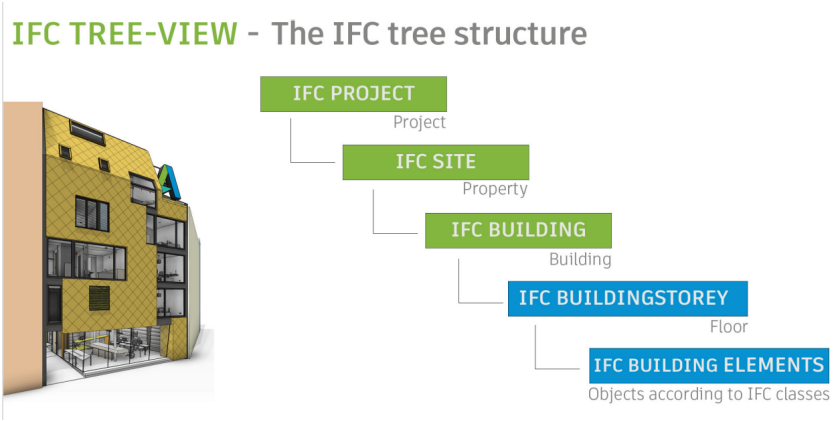


FIGURE 2.2 – Hierarchical arrangement of definitions in a IFC framework. Reference `https://technical.buildingsmart.org/standards/ifc/`

2.1.1 Spatial Representation

A building is a closed context of space time where one can observe repetitive patterns or occurrences of activities. The semantic state of a space evolves as cause-effect interaction between building elements, environment and people or human controlled/operated technology. Different configurations of the floor can influence the nature or place of activity. For instance, to have an offline meeting, a room needs have ample ambient light and thermal comfort. So we start by covering how the knowledge of a building is digitally stored and abstracted for designing a building management system.

A life cycle of a building comprises of independent stakeholders like architects, contractors, plumbers, electricians, material suppliers, etc. These people take part in either the design, construction or maintenance phase of the building life cycle. Post design phase, the primary work is in constructing the architecture physically to exist in the world. So there is a need for an agreeable format for holding the digital information, with the least amount of ambiguity. Although the know-how of buildings is as primitive as humanity it, it is only in 2013 that the first ISO format was conceived and named as Industry Foundation Classes (IFC) and is now mandatory for European construction projects. IFC is a standardized, digital description of the built asset industry to save information related to Building Information Modeling (BIM). BIM [13] is the process of designing geometric orientations and specifying functional semantics between entities. There are around 4000+ entities defined in IFC and arranged as an entity-relationship model following inheritance based hierarchy as shown in Fig. 2.2. The entry point to an IFC

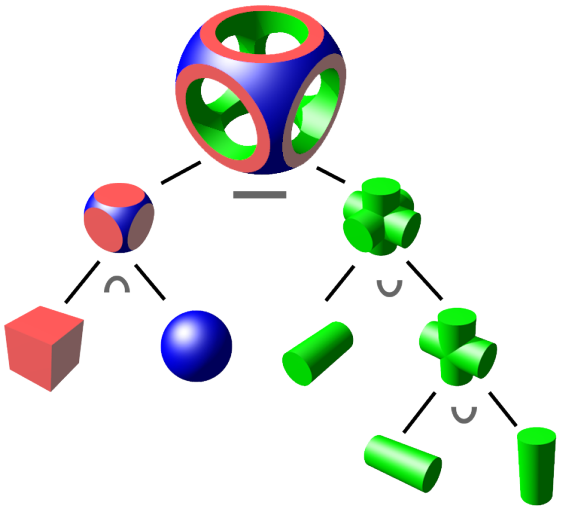


FIGURE 2.3 – Example of a Constructive Solid Geometry Model used for describing [2] physical objects to be stored in IFC

file is the entity named *IfcProject* which describes a collection of building sites. Each IFC-Site entity references to one or more building objects named *IfcBuilding*. IfcSite is spatially contained in *IfcProject* and inherits the *map transform* and *World Coordinate System* from *IfcProject*. It is possible to customize the default transformation map between IFC defined and actual physical coordinates of a building. The building is usually organized into storeys, each containing a subset of *IfcObject*s such as doors, walls, windows, elevators, roofs, ceilings, stairs, spaces, etc.

Each *IfcObject* is associated with a set of properties (*IfcObjectProperty*) that describe the physical shape and may optionally include material type. The location of a *IfcObject* is given by $IfcObjectPlacement$. Following the inheritance structure, a child object is placed inside its parent object. The geometry of the child is defined in its **Local Coordinate System** (LCS) with the origin defined in $IfcObjectPlacement$. It contains the mapping between the origin placement of child with regard to parent. Being defined in such a recursive manner, the starting point leads all the the way up to $IfcSite$. $IfcObject$ can be geometrically defined through **Implicit Geometry**, which is a geometric representation, driven by attributes or constraints on surfaces. **Bounding-Box** is the minimalist representation possible for a IFC object where a box is a 3D octahedral defined by the element's length attribute along X, Y & Z Axes in its local coordinate system. **Boundary representation** is a collection of connected surfaces

and boundaries between solid and non-solid. For example, a cube can be represented by a set of surfaces and a connectivity graph between the surfaces. Alternately, the same cube can be defined using constraints applied to the length, width, or height attributes through **Parametric** geometry that reduces complex geometry to simple or complex functional constraints between surfaces/ edges/ vertices. The IFC model also proposes a notation system which supports use of these primitives in extrusion, revolution and composition. **Constructive Solid Geometry** (CSG) is a technique used in solid modeling to create a visually complex surface or object by using Boolean operators to combine simpler objects called primitives as shown in Fig 2.3. In case implicit geometry is not adequate for shape representation, a explicit shape definition is also defined. It captures the semantic representation of a void by storing the physical parts in terms of points, curves, surfaces and solid primitives. IFC representation is extremely useful to build, visualize, exchange/port the building elements during it's construction phase. It is a complex task to look at an ".ifc" file and answer spatial queries like "Is the bathroom beside the kitchen ?", "What is the path to go from point A to B ?" or "Is the *kitchen* beside the *stair* ?" etc. It is due to the fact all the elements needed to answer the fact can be defined with regard to non-identical parents. Hence one has to compute the boundary of each object, transform to **World Coordinates** and then decide if the kitchen is beside the stair. Here in lies the motivation for spatial abstraction with a single coordinate system to support efficient computation of spatial queries.

2.1.2 Predicate Logic Frameworks

Next comes the step of integrating sensors to record or stream measurable variables related to a building. The generated data makes it possible for temporal queries where operators are used to retrieve data from a set of spaces or devices.

The sensor values are usually stored at a database or retrieved on device from the installation site. So we need to have a metadata to understand which places are generating what kind of values. BRICK [1] is state of art building metadata scheme that captures the logical relationship amongst sensors distributed across building entities. A predicate relationship *A op B* is established between two entities A and B, related through the "op" clause. The rudimentary

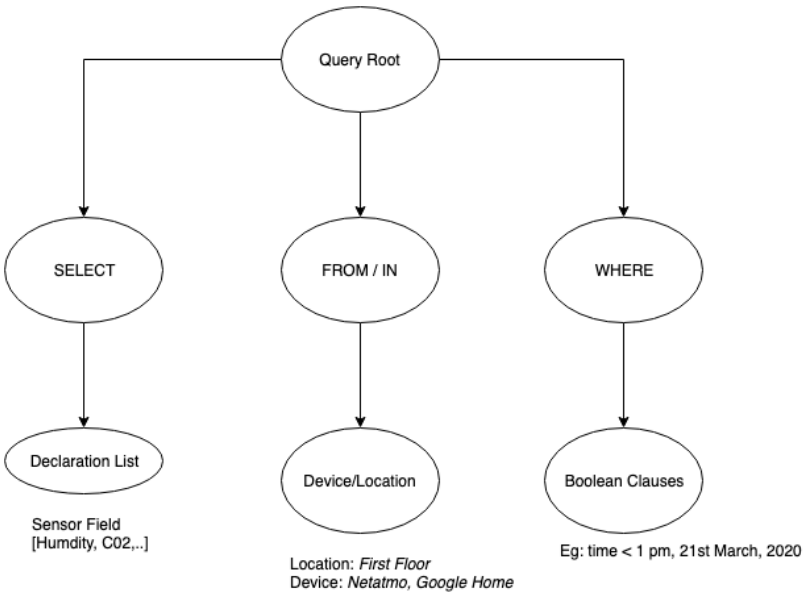


FIGURE 2.4 – Abstract Syntax Tree for a Space-time Query

operators shown in Figure 2.5, are highlighted in Italics as follows :

1. *A contains B* defines the spatial encapsulation of B by A or inversely B is *isLocatedIn* A. (Location, sensor) and (Location, Equipment) are the compatible entities.
2. *A controls B* carries the meaning that A determines or affects the internal state of B or inversely B *isControlledBy* A. Generally such relationships are developed between a functional/actuating block and an equipment.
3. To imply the notion of A has some component B, we utilize *hasPart* or its inverse *isPartOf* for the tuples of (Location, location), (Equipment, Sensors), and (Equipment, Equipment).
4. Measurements are relative in nature for which we need to define *A hasPoint B* between (A,B) from (Equipment, Sensors) or (Location, Sensors). The inverse relation given by **B isPointOf A**, describe that B can be referenced by A.
5. The notion of flow or connectivity between entities A, B is given by *feeds* or inversely *isFedBy*. Typically such relationships are observed between functional blocks and equipment or between multiple equipments.
6. *hasInput* and its inverse *isInputOf* models the fact that A serves as an input for/to B. Corollary operators for output are *hasOutput/isOutputOf* and both types relate between a functional block and a sensor.

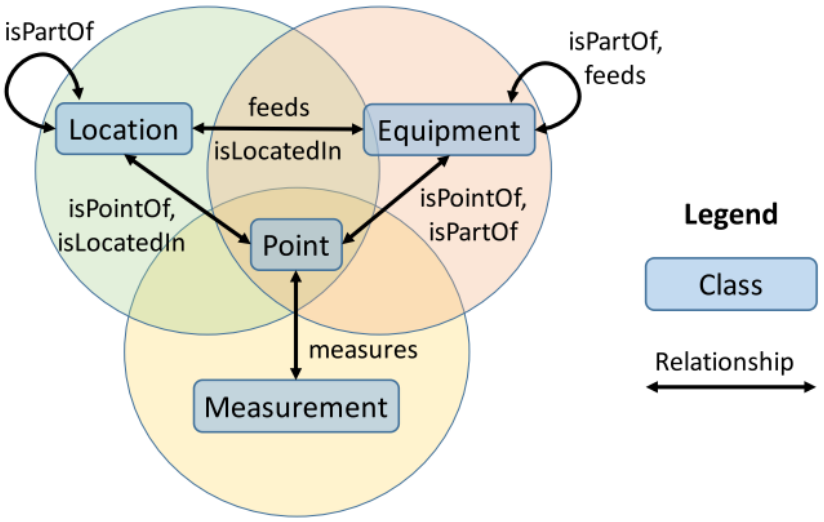


FIGURE 2.5 – Information concepts in Brick and their relationship to a data point. [1]

Such kind of frameworks represents a functional viewpoint of the building abstracting the relationship between space and sensors by a set of Boolean predicates. One can now implicitly define spatio-temporal semantics through presence of electrical appliances or non-electrical utilities like wash basin, bath-tub, toilet seats etc. Often, this layer is utilized without incorporating the spatial orientation due to high complexity in parsing the latter.

2.1.3 Sensing and Control

Sensing technology is the basis of perception for a system in order to be aware of the surroundings. Typically at deployment, sensors are coupled with a micro controller/processor to sample sensor values and store or stream data points. The first set of works in early 2000 reveals that one can extract meaningful patterns hidden/evident from sensor-generated time-series data-sets. From discreet timestamped data, the notion of semantics play the pivotal role to form *Event*s for example, taking a shower for an hour implies bathroom usage which is otherwise idle. Role of occupancy detection on energy consumption is evident from the novel work [14]. In light of this, a considerable amount of effort has been put [15], [16], [17] using semi-supervised, unsupervised learning methods. The work [18] enhances place-specific activity recognition in a closed world setting by analyzing data at different resolutions and compress into low resolution meta-data for spatio temporal querying. The challenges regarding

design of sensor networks [19] stem from velocity and volume of throughput data, resulting in bandwidth issues, loss-less compression techniques, data/signal correlation with neighbouring sensors.

The quantitative definition [20] of categorizing smartness is given by the metric Building Intelligence Quotient(BIQ) [21], equals to the ratio between number of decisive/predictable controls to observable parameters like ambient conditions, power consumption, etc. Intuitively it means that a building without sensors being able to control appliances or indoor comfort is at the highest band of intelligence. The minimum BIQ of 0 corresponds to a passive building management system where no action is taken based on recorded sensor stream. Let us present an example to compute the IQ of a building management system that controls a HVAC unit. If control outputs are fan speed and outgoing air temperature while the data inputs are the CO_2 levels and temperatures of indoor air and exterior environment, then the BIQ = $\frac{2}{3}$. Optimal sensor placements and design decisions has the potential to improve a Building IQ. For a smart building, ideally the BMS integration shall render the BIQ to be greater than 1. Although only a few papers quantify building smartness, this piece of information is fundamental to our study in order to reduce the numnber of installed sensors and derive a high BIQ.

2.1.4 Review of Technology Acceptance

Year 1990 marked the birth of world wide web and within 30 years, computing clouds or data centers has already become an integral part of internet services. Diminishing prices of sensing hardware led to a wide spread and often ad-hoc incorporation [15] of sensors on edge for monitoring, analysis [21] and control. A 2019 review [22] of the smart building industry highlights the major bottlenecks towards technological adaption. High installation costs, obscurity on data storage policies, and privacy concerns impede the acceptance [23] of Internet of Things (IoT) in buildings. We briefly describe two major problems stemming from the point of user adoption of a BMS solution.

Financing The cost of constructing [24] a smart building is usually 1.2-1.8 times a non-smart counterpart. This initial capital poses the second barrier for a stake holder [25] to overcome

Semantic Space Label	Possible Activities
Entrance	Entering, Leaving;
Kitchen	Preparing, Cooking, Washing the dishes;
Living Room	Eating, Watching TV, Computing;
Conference Room	Meeting, Teleconference
Toilet	Using the toilet
Staircase	Going up, Going down;
Walkway	Walking, Transfer
Bathroom	Using the sink, Using the toilet, Showering;
Office	Computing, Watching TV;
Bedroom	Dressing, Reading, Napping;
Common to all places	Cleaning.

TABLE 2.1 – Activity List

before system installation. Average cost is around $[10-20]$\$ for ambient sensors like humidity, temperature while smart meters can be expensive around $[50-100]$\$. We observe that the industrial or Do-It-Yourself (DIY) embedded sensor hardware are typically between $[10-25]$ Watts. Assuming 365 days a year, even a 20W sensor will consume close to $20 \times 24 \times 365 =$ 175.2 kilo Watt-hours annually. Industrial grade quality of smart sensor or power consumption meters can be more energy efficient but expensive than assembled Do-It-Yourself counterparts. Hence rather than ad-hocly placing sensors, a smart building solution needs to go through a pre-evaluation stage before finalizing the bill of materials. Such a tool is missing from the literature which can bring down the cost through a composite solution built from DIY components and industrial grade sensors.

Data Privacy Typically a smart building application thrives on real time sensor data for monitoring or actuation. The operational data accumulated throughout the life span of a building may contain sensitive patterns especially related to occupancy [14]. MavHome [9] attempts to model a home as a rational agent that perceives the environment through sensors and user interactions. Occupancy patterns can have spatial semantics attached to it. For instance in an office space, gathering of people at cafeteria around noon time is likely to be for lunch whereas

in a conference room, a group of people is most likely to have a meeting. Often the smart home inhabitant prediction problem utilizes the empirical frequency and temporal order of activities to partition an action sequence into high level tasks. Table 2.1 depicts the possible types of indoor activities [26] that can be discovered through machine learnt techniques. Although it is not unrealistic to be aware of an activity in advance from calendar events and room reservation agenda, research shows that the notion of privacy and hacking can play an adverse impact on non adoption of technology. The existing BMS solutions either barely shed light on data practises or send sensitive data to the cloud by default.

2.2 Edge Intelligence

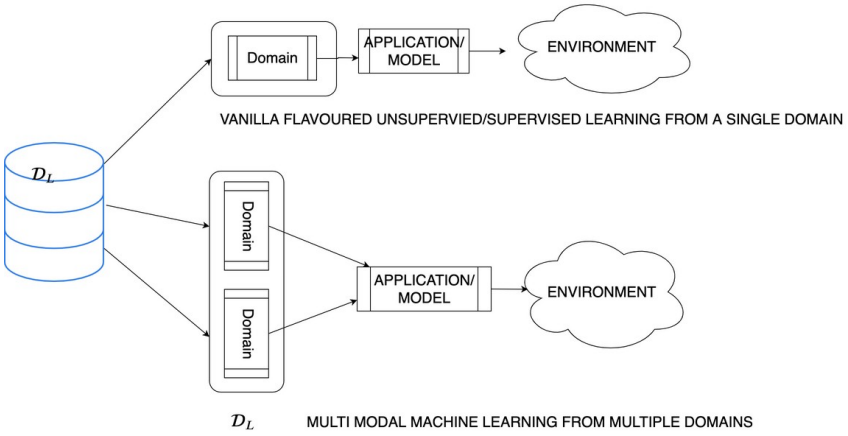


FIGURE 2.6 – Classical approach to machine learning

The motivation behind the advent of artificial intelligence was to mimic some of the data processing capabilities of our ever-fascinating human brain. Rosenblatt in 1958 proposed the Perceptron model [27] as a computational graph with auto updating edge weights. This was the first instance where a machine was munching data and approximating mathematical functions in order to mimic the neural capability of a human brain. Over the last 60 years, machine intelligence have been customized for a plethora of fields, each domain developing its signature such as computer vision and natural language processing. Four distinct elements can be roughly observed in every machine learning setup : data, domain knowledge, application model and the deployed environment. An application model leverages domain knowledge to train on the data

to affect its environment. Figure 2.6 shows the classical set up of machine learning where one or multiple domain knowledge is embedded into the model. Typically the learning mechanism for such architectures decrease the empirical loss on the train data set.

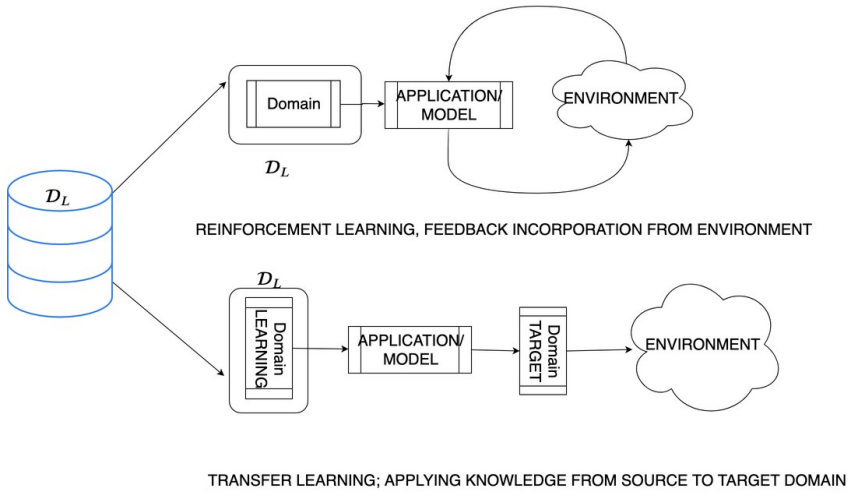


FIGURE 2.7 – Learning continually through interaction and feedback

Now once a model is trained, it can be used to answer a different problem just like the way humans cross think on several topics to come to a solution or refine the answer with time. This forms the basis for investigation in the multi domain setting, where a machine is trained on a non-identical *source* domain to acquire characteristics of the target domain. For example, a computer vision model that is trained to differentiate between cats and dogs can be extended to identify between tigers and lions ! The main implicit assumption behind this knowledge transfer is a high level of feature similarity between the domains, like all the four animals above have 4 legs and 2 eyes ! Often such techniques fails to address the challenges of a dynamic environment or especially when the characteristics of the testing sample is dissimilar from train. To efficiently solve issues related to a reactive environment, researchers proposed reinforcement learning where the application model continually learns from its environment. The classical setup requires offline availability of data while online techniques can process data on the fly. Figure 2.7 shows the schematic view of transfer and reinforcement learning. Now let us see how such techniques are relevant for smart building applications that generate or stream sensor data.

Classically, the sensor data is sent to a non-localized storage resource like structured or un-

structured databases or simple storage buckets hosted in a server or provisioned through cloud services. One of the ways to decrease volume of data flow between embedded hardware and the cloud is constraining data retention at site or transmitting useful knowledge representations instead of raw data. The spatio temporal data is usually of interest for taking decisions based on observing trends, forecasting patterns and detecting anomalies over time and space. Evolution of Artificial Intelligence over the span of 70 years has made it possible to learn compact computational models for solving such aforementioned tasks in a data-driven fashion. A centralized way of learning is to gather all the data at one central repository/machine and execute algorithms to automatically discover the representations needed for detection or classification or prediction. Such an implementation lacks the benefit of parallel computation, possesses a single point of failure and suffers from the high risk of entire data leak solely from one site.

In contrary, one can distribute the learning process over a set of computing nodes and leverage intra-node connectivity to collectively solve one or more tasks. Each machine has access to data for a different task and needs to learn a predictor, and nodes continually leverage the connectivity to optimize the relationship between tasks. While learning a task over multiple computing nodes [28], usually intermediate results are shared or aggregated by a central mediator to compute the final result. For Multi Task Learning (MTL), it is assumed all the tasks are related to a global hidden/latent space[29] which is continually optimized. Learning tasks jointly can lead to various improvements in performance when compared to solo training in capturing relationships amongst non-IID and unbalanced data [30]. For example, MTL NET [31] builds up a universal neural network, such that a hidden layer is shared between all the tasks and customisation for each task occurs through a fully connected last layer. Now if the tasks are running on embedded hardware instead on cloud, that implies computing constraint at edge with low quantity and high quality data transmission coupled with low power requirement. Edge Intelligence (EI) is defined as a confluence [32] between edge computing and Artificial Intelligence, born out of the philosophy to process sensor data in-situ. A recent area of interest in EI [33] [34] has been in porting deep learning modules to edge [35], specifically to reduce computation and transmission of millions of training parameters over heterogeneous networks.

2.2.1 Federated Learning

Federated learning was introduced in 2016 by McMahan et al [36] for solving a learning task with the help of voluntarily participating devices (also referred to as clients) and a central server coordination. The aim in federated learning is to fit a model to data, $\{X_1, ..., X_m\}$, generated by m distributed nodes. Each node, $t \in [m]$, collects data in a *non-IID* manner across the network, with data on each node being generated by a distinct distribution $X_t \sim P_t$ [37] [38] [30]. The number of data points on each node n_t may vary significantly and there can be potentially a large number of nodes m in the network. Storage, computational and communication capacities of each node may differ due to variability in hardware, network connection and power. Conceived first in 2016, over the span of the last 5 years, two distinctive approaches are found in literature of federated learning. While Federated Stochastic Gradient Descent (Fed-SGD) [39] transmits gradients to a central server, Federated Averaging (FedAvg) [40] uploads the local model. In both forms of federation, the key motivation is to obtain a high quality centralized model that can be trained through distributed site-localized data.

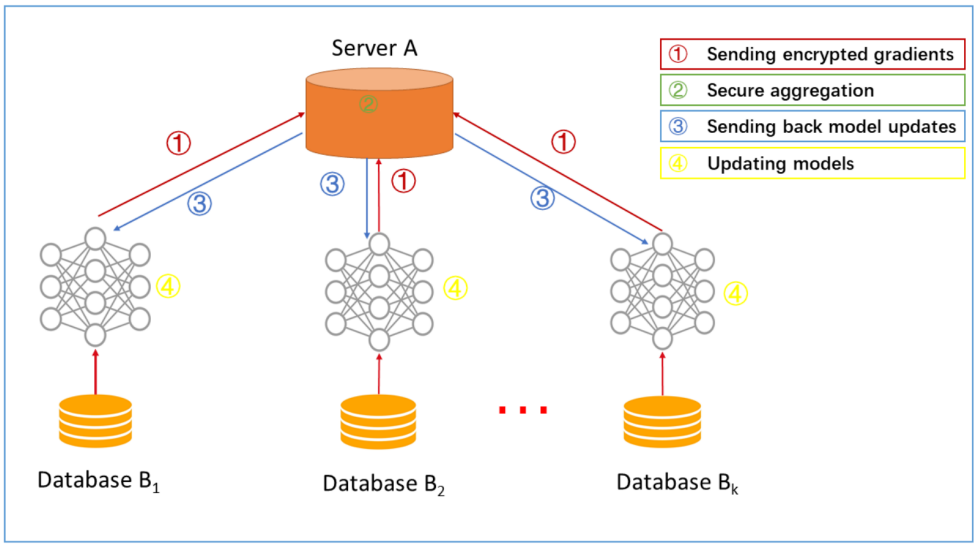


FIGURE 2.8 – Architecture of federated learning set up [3]

As per Fig 2.8, a synchronized federated learning round is described below where η, n, K, n_k denote the learning rate, total number of data samples, total clients and data sample from an individual client respectively.

1. The central server selects a subset of existing clients, each of which downloads the current global model (w_t^G) at time t.
2. Each client k performs gradient descent based on the local data and transmits either the entire or a subset of gradients $g_k = \nabla F_k(w_t)$ in the case of FedSGD. The alternate approach is to update the local model through $w_{t+1}^k \leftarrow w_t^k - \eta g_k$ and send w_{t+1}^k.
3. In FedSGD, the server aggregates the gradients (typically by a weighted averaging) to construct an improved global model by setting $w_{t+1}^G \leftarrow w_t^G - \eta \nabla f(w_t) = w_t^G - \eta \Sigma_{k=1}^K \frac{n_k}{n} g_k$. In contrast for FedAvg, the client models are averaged to yield $w_t^G \leftarrow \Sigma_{k=1}^K \frac{n_k}{n} w_{t+1}^k$.
4. A learning round is completed when every local model is updated by incorporating the new global version by iterating over local epoch.

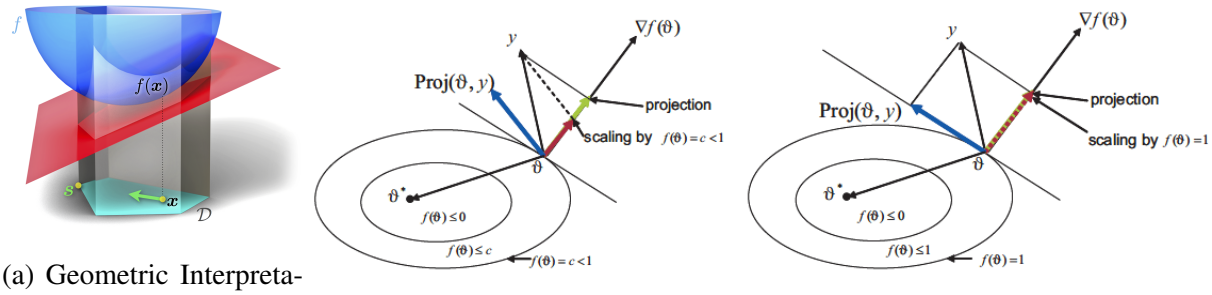
However due to data diversity [41], the learnt global model may not be the best local model possible per site. This leads the pathway for federated personalization where a local model incorporates the global knowledge for additional customization. However, one particular constraint of local computations that one needs to pay attention is the computing capacity/resource at the local level. Thus, one of the main challenges in federated learning setting is to design optimization algorithms that are light-weight and robust under uncertainty. Another drawback of this mechanism is reliance on a global mediator who has the holistic view of the problem; which also means corrupting the global can lead to adversarial attacks or performance losses.

2.2.2 Projection-free Optimization

In many online learning problems the computational bottleneck for gradient-based methods is the projection operation. For example in Figure 2.9a, one wants to minimize $f(x)$ given by the blue function subject to a plane (coloured red). Instead of solving by the classical Lagrangian through gradient descent, one can alternately investigate the projection area. Frank Wolfe algorithm was proposed in 1956 as an iterative first order approximation algorithm for convex constrained optimization that solely operates on the constrained linear space.

While gradient descent for constrained optimization requires a projection step back to the feasible set in each iteration, the Frank–Wolfe algorithm only needs the solution of a linear problem over the same set in each iteration, and automatically stays in the feasible set. The

convergence of the Frank–Wolfe algorithm is sub-linear in general : the error in the objective function to the optimum is $O(\frac{1}{k})$ after k iterations, so long as the gradient is Lipschitz continuous with respect to some norm. The same convergence rate can also be shown if the sub-problems are only solved approximately [42].



(a) Geometric Interpretation [43]

(b) Illustration of the Projection Operator [44]

FIGURE 2.9 – Projection Free Linear optimization by Frank Wolfe

Typically the online updating element in a projection free setting are linear optimization oracles. Algorithm [45] is the online version of Frank Wolfe with an implicit centralized system.

Algorithm 1 Online Frank Wolfe Algorithm [45]

Input : A convex set $\mathcal{K}$, a time horizon T, a parameter L, online linear optimization oracles $\mathcal{O}_{i,1},\ldots,\mathcal{O}_{i,L}$ for each player $1 \leq i \leq n$, step sizes $\eta_\ell \in (0,1)$ for all $1 \leq \ell \leq L$

Initialize $\mathcal{O}_{i,\ell}$ for all $1 \leq \ell \leq L$
for $t = 1$ to T **do**
  for every agent $1 \leq i \leq n$ **do**
    Initialize arbitrarily $\boldsymbol{x}_{i,1}^t \in \mathcal{K}$
    for $1 \leq \ell \leq L$ **do**
      Let $\boldsymbol{v}_{i,\ell}^t$ be the output of oracle $\mathcal{O}_{i,\ell}$ at time step t.
      Play $\boldsymbol{x}_{i,\ell}^t$ and expect $y_{i,\ell}^t \leftarrow \arg\min_{a\in\mathcal{K}} < a, \nabla F_t(\boldsymbol{x}_{i,\ell}^t) >$
      Compute $\boldsymbol{x}_{i,\ell+1}^t \leftarrow (1-\eta_\ell)\boldsymbol{y}_{i,\ell}^t + \eta_\ell \boldsymbol{v}_{i,\ell}^t$.
    end for
  end for
end for

2.2.3 Decentralized Learning

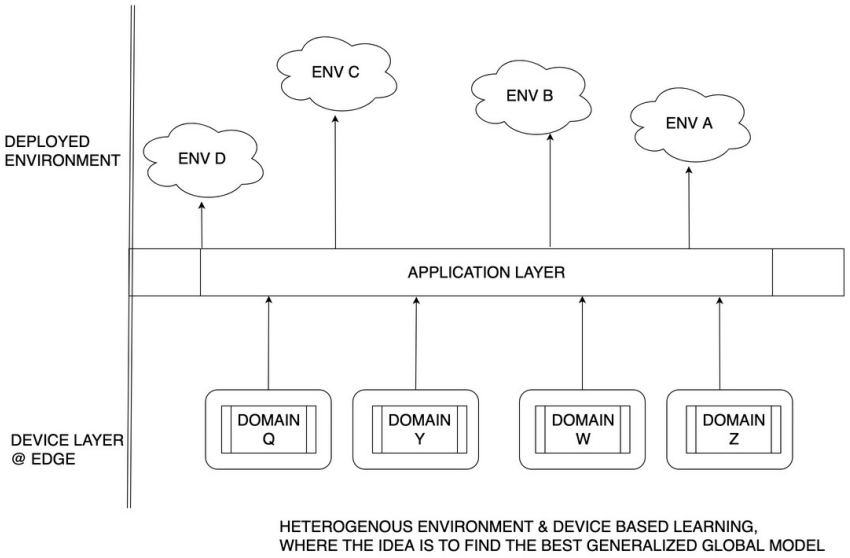


FIGURE 2.10 – Learning from heterogeneous data distributed over devices

One way to conceptualize collective learning is through a peer to peer exchange protocol, where only partial information is revealed to each participant. This paves the path for decentralized algorithms that can communicate with peers and process data in an online manner. Typically, these algorithms operate on gradient exchanges which require projecting intermediate solutions onto the feasible set. In the paradigm of edge computing, where low power and simpler hardware are deployed, the projection step is likely to be a computational bottleneck. Yan et al. [46] introduced distributed online projected subgradient descent and showed vanishing regret for convex and strongly convex functions. In contrast, Hosseini et al. [47] extended distributed dual averaging technique to online setting using a general regularized projection for both unconstrained and constrained optimization. A distributed variant of online conditional gradient [48] was designed and analyzed in [49] that requires linear minimizers and uses exact gradients. However, computing exact gradients may be prohibitively expensive for moderately sized data and intractable when a closed form does not exist. In this work, we go a step ahead in designing a distributed algorithm that uses stochastic gradient estimates and provides a better regret bound than in [49]. In a nutshell, we aim to design an online projection free algorithm that is suited for edge computing and applied to prediction problems.

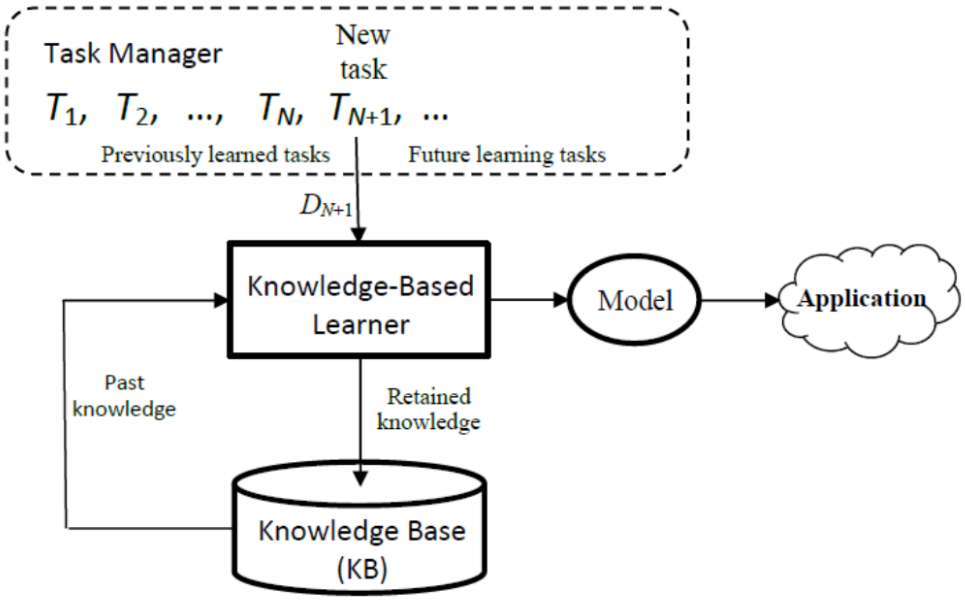


FIGURE 2.11 – Components in Lifelong Learning Schematic

2.2.4 Lifelong Learning

So far, we have seen algorithmic developments and localized data handling techniques to fine tune models where the system is focused on solving a single task perfectly on deployment. This is in stark contrast to how humans multitask, an observation rightfully discovered by Thrun and Mitchell in 1995 [50]. The authors introduce a novel system-oriented approach towards machine learning where one focuses on knowledge representation and inferring meaningful information to solve an incoming new task. The survey paper [51] defines Lifelong Learning (LL) as a continual learning process where at any point of time, the system has performed N tasks and when faced with the $(N+1)^{th}$ task, it uses knowledge gained from N tasks to solve the upcoming task. Let the previously learnt tasks $(T_1, T_2, \ldots T_N)$ with corresponding datasets $D_1, D_2, \ldots D_N$ be stored in a Knowledge Base (KB). After learning T_{N+1} using a priori knowledge, the knowledge base is updated with insights (intermediate, final results) gained from learning the new task. Ideally a life long system should be able to maintain a knowledge base over time by reusing previous learning to apply to a new task.

The four key components of a lifelong system are :

— Knowledge Base (KB) stores the information resulted from the past learning, including

the resulting models, patterns, or other forms of outcome. Additionally it may also store a meta data pertaining to (1) the original data used in each previous task, (2) intermediate results from each previous task, and (3) the final model or patterns learned from each previous task.

— Knowledge Based Learner (KBL) uses prior knowledge directly or mines features to learn a task. However, guard needs to be taken to prevent irrelevant knowledge or bad data from corrupting the performance of a LL system.

— Task Manager receives and manages the tasks that arrive in the system, and handles the task shift and presents the new learning task to the KBL to start the LML process.

— Finally for a smart application, the most suitable model is chosen to solve the problem.

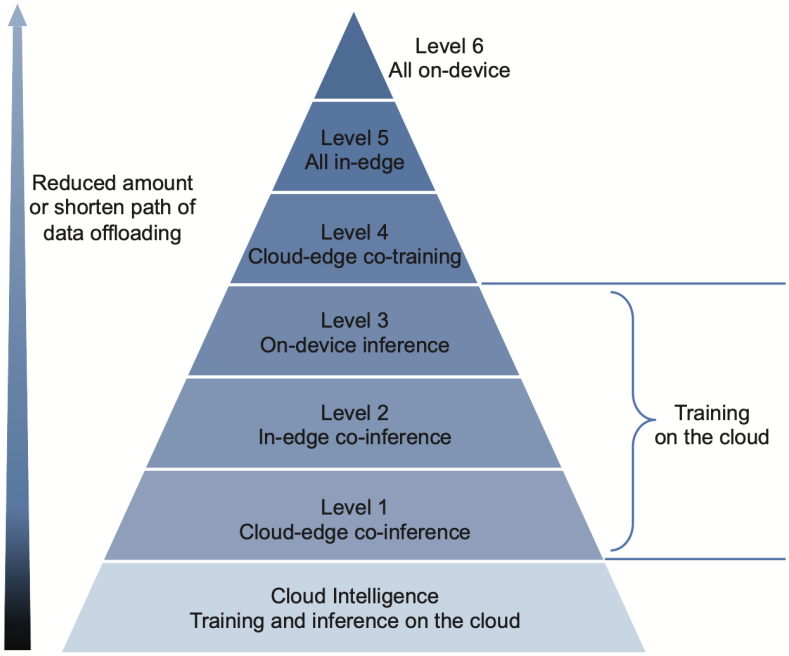


FIGURE 2.12 – A 6 level pyramid to gauge edge intelligence [4]

Let's look at the example of forecasting power usage in smart buildings as an example of how lifelong learning can be applied to a smart building problem. If data is available on a room by room, or building by building basis, we could frame prediction per room or building as a multi-task learning problem. As many rooms are likely to be in use at similar times for similar purposes, there is likely to be relationships between tasks that can be exploited by multi-task learning. If each node contains an edge computing device, this can then be seen as a distributed multi-task learning problem. Owing to non-identical usage patterns in spaces/rooms for

example, such per zone data could be described as non-IID with data from each space generated by a distinct distribution. Life long learning is yet to be applied to smart building solution but the systems approach makes it a suitable candidate to propose an auto-updating system on the edge. In case of a building, not only we need customized zonal models but also a continual update of the holistic knowledge acquired over time. Therefore we can see how basing a smart building learning system on these principles can be relevant. This inspires us to design a Building Management System to operate on a fixed data buffer, compress data on the fly, and leverage group or peer to peer exchanges to improve the quality of machine learning solutions. Our system design additionally takes into consideration the 6 level pyramid [4] shown in Fig. 2.12 to measure edge intelligence based on execution of training and inference in cloud, edge or in-situ on device. The pyramid base (Level 1) is least edge intelligent due to training and inference on-cloud while the top most layer (Level 6) corresponds to highest edge intelligence with complete execution on device. We extend the existing research in smart buildings by focusing on in-situ algorithm designs rather than a centralized on-cloud management system. The following work is organized to abstract spatial knowledge of a building (Chapter 3), optimize sensor placements in the pre-integration phase, save up on the capital and energy footprint through a Virtual sensor field with low-powered edge learning in Chapters 4 and 5 respectively.

Chapitre 3

Non Intrusive Sensing

Spatial orientation of elements within a building are utilised to generate a low complexity yet relevant meta data from an architectural file. The goal is to develop a generic building management application with a special focus on non-intrusive sensing.

3.1 Building Parser

The complexity of an IFC file grows with increasing number of building elements like spaces, doors, windows, stairs, roofs, stories etc. For querying relationships amongst building elements, IFC proposed a Resource Description Framework to represent building in Ontology Web Language, namely IFC OWL. The referential coordinates of a building product are kept unchanged in such a translation and this complicates geometrical reasoning about adjacent or overlapping building elements. Also from the software perspective, there is lack of an open source parser, that can directly generate floor-plan metadata from an IFC file.

3.1.1 Geometric Abstractions

We introduce two geometric objects : a bounding box and a 3D plane that operates on referential geometry of an IFC object and output coordinates of building elements from a single-

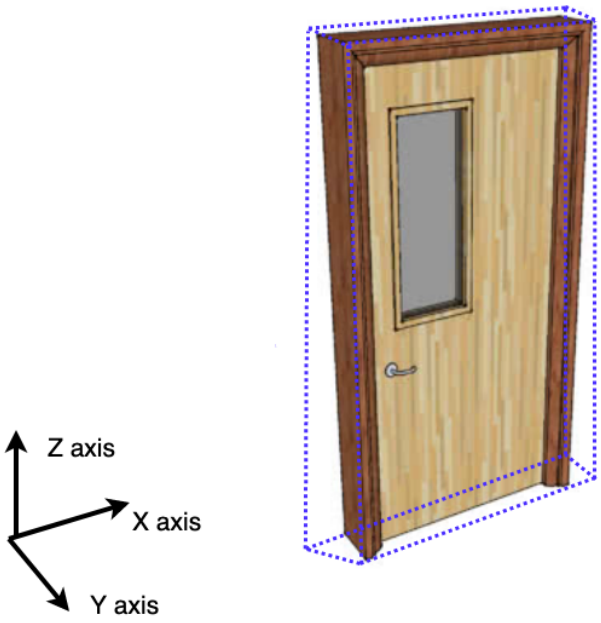


FIGURE 3.1 – Building Box illustration, edges of box highlighted in blue

origin reference frame defined on IFC-Site.

Building Box $\mathscr{B}$ is a rectangular parallelepiped enclosing a building element and represented by a pair of coordinates per axis. Each axis pair contains the maximum and minimum coordinates enclosing the box along that axis. Formally we will represent the **Building Box** Representation of a building element $\mathscr{E}$ as

$$\begin{aligned} \mathscr{B}_X(\mathscr{E}) &= \{X^{Min}(\mathscr{E}), X^{Max}(\mathscr{E})\} \\ \mathscr{B}_Y(\mathscr{E}) &= \{Y^{Min}(\mathscr{E}), Y^{Max}(\mathscr{E})\} \\ \mathscr{B}_Z(\mathscr{E}) &= \{Z^{Min}(\mathscr{E}), Z^{Max}(\mathscr{E})\} \end{aligned} \tag{3.1}$$

Cutting Plane Ξ is an imaginary plane that passes through a building and is defined by Equation 3.2 where $\hat{n}$ and $\vec{P}_0$ represents the normal vector and a fixed point on the plane. $\vec{P}_0 = (0,0,z)$, $\vec{n} = (0,0,1)$ denotes the family of horizontal planes placed at height z from the ground plane.

$$\Xi(\vec{P}_0, \hat{n}) \equiv \hat{n}(\vec{p} - \vec{P}_0) = 0 \tag{3.2}$$

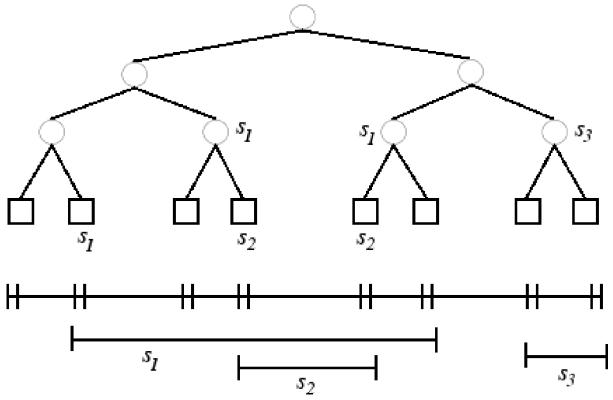


FIGURE 3.2 – Illustration of an interval tree, intervals denoted by $S_1, S_2 \& S_3$

3.1.2 Parsing Operators

We demonstrate an algorithm that gives a controllable process to generate a floor-plan from one or more cutting planes and infer a connectivity graph amongst building elements. In the process, we develop useful operators to algebraically formulate building aware intelligence use-cases like neighbourhood, path and ventilation graph.

Selection

The system generates the 3D view of all IFC Products with defined shapes and stores the Bounding Box representation in 3 interval trees, one for each axis. Each node contains the minimum and maximum points along one of the axes. p building elements or intervals can be arranged in an interval tree [52] as shown in Fig. 3.2. Such trees have an initial creation time of $O(p \log p)$ and output sensitive query time of $O(\log p + m)$ depending on number of matches m. The memory consumption is limited to $O(p)$. $\sigma : \{$**Building Element** $\} \times$ **Query Point** $\rightarrow \{$ **Building Element**$\}$ returns the set of building elements B whose bounding box encloses the point from all 3 axes.

$$\sigma(B,(p,q,r)) = \{\mathscr{E} | \mathscr{E} \in B, p \in \mathscr{B}_z(\mathscr{E}), q \in \mathscr{B}_z(\mathscr{E}), r \in \mathscr{B}_z(\mathscr{E})\} \tag{3.3}$$

Contour

Element Contour $\mathscr{C}$: {**Building Element** } $\times$ **Plane** $\rightarrow$ { **2D Points**} returns a set of intersection points that are generated by the impression of a building element on a cutting plane. The smallest convex set of points that contains the element is the contour or hull of impression. We define Element Contour of a building element $\mathscr{E}$ as an ordered set of intersecting points given by Equation 3.4. For a building object located above or below a plane $\mathscr{C}(\mathscr{E},\Xi) = \phi$ since there are no intersecting points.

$$\mathscr{C}(\mathscr{E},\Xi) = \{(u_i, u_{i+1})\} | \forall i \in [1,2,\ldots,p], u_{p+1} = u_0 \tag{3.4}$$

Hull

The next task is generating a singular impression of a building object from multiple element contours. We recall the well studied concept **Delaunay Triangulation** (**DT**) of a set of triangles constructed from a set(P) of planar points such that no point in P is inside the circumcircle of any triangle in DT. The nominal work [53] of sweep line algorithm computes Delaunay triangulation in $O(n \log n)$ expected time with $O(n)$ storage for a polygon of n points. $\mathscr{H}$: {**Element Contours** } $\rightarrow$ { **Element Contour**} (Equation 3.5) outputs a mapping for every input data point such that the corresponding coefficient ($\beta_i = 1$) if the point lies on the Delaunay hull or 0 otherwise. A sample of Delaunay triangulation for a door is shown in Fig. 3.3.

$$\mathscr{H}(\{\mathscr{C}\}) = \{\beta_i \times p_i | \beta_i \in \{0,1\} \forall i \in \{C\}\} \tag{3.5}$$

Overlap

$\Omega : \{\mathscr{C}_1, \mathscr{C}_2\} \rightarrow$ { **0,1**} takes two polygons as inputs and if they intersect or overlap the expression evaluates to True or 1. This operator is useful in inferring spatial connectivity between adjacent building elements. One can tweak the connectivity by inflating a hull to make sure the desirable overlap is reached.

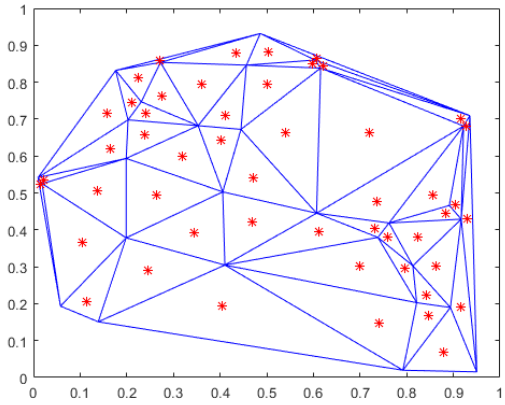


FIGURE 3.3 – Hull Generation of a door from multiple contour points (in red) by Delaunay Triangles (in blue)

For floor-plan generation of a storey, the system generates slicing planes at every $\Delta h = 0.5$ meter and processes multiple contours for every building element to output a convex hull. Multiple hulls on a 2D plane yields a floor-plan. The parsing is controlled by specifying the starting and ending height for cutting plane and fine tune the multiple contour process by tuning the vertical resolution. The parser is written in pure python, supports IFC 2x3, and 4.1 formats and is made publicly available in Docker hub under the image name "angmit/ifcparser :v2.0".

3.2 Spatial Graph

We construct a spatial graph (G^S) to capture the connection between building elements such as spaces, doors, windows, stairs, elevators, storeys and roof. Nodes (V) of the spatial graph represent elements of interest and a edge between two elements are drawn in case of spatial overlap or intersection. Formally, Equation 3.6 expresses the connectivity graph amongst a set of building elements **B** through a set of cutting planes Ξ. Graph data is stored in a XML format which supports queries to answer the spatial intelligence use-cases demonstrated below Each node additionally stores the corresponding IFC identifier and type of element.

$$G^S(V,E) = \{(u,v)|\forall u,v \in V, \Omega(\mathscr{H}(C(u,\Xi)),\mathscr{H}(C(v,\Xi)) = 1\} \tag{3.6}$$

Neighbourhood

Neighbour discovery is now made possible through geometric reasoning. Queries like "Are there stairs beside the kitchen ?" computationally retrieves the list of neighbouring elements for the query building element *kitchen*. We extend the notion of adjacency to d hops by exploring nodes in a depth first traversal up-till d links from the query node. This is often useful for estimating 1-hop connections between two spaces who share a common door or window. If $N_0(u,w)$ represents a adjacent neighbour (w) of an element (u), then one hop neighbour graph is given by Equation 3.7.

$$G^1(V,E) = \{(u,v)|\exists w, N_0(u,w), N_0(w,v)\} \tag{3.7}$$

3.2.1 Structural Path

A path in a building is defined in terms of an ordered set of building elements that can be physically visited while going from space A to B. Logically this means, the path can not pass through a wall or a window or a roof. Equation 3.8 defines a path graph (G^P) of space-space linkages that is constructed by discarding all edges from $(G^1 \cup G^S)$ whose source or destination is not a space.

$$G^P = \{(u,v)|(u,v) \in G^S \cup G^1, u.type \& v.type = space\} \tag{3.8}$$

A space link is represented by the centroid of a building element ($\vec{\mathscr{E}_i}$) and a displacement vector ($\vec{ds}_i$) connecting the next traversed element. A path ($\mathscr{P}$) starting from building element $\mathscr{E}_p$ and ending at $\mathscr{E}_q$ is an ordered sequence of L space links as per Equation 3.9 . Imposing constraints on linkages can yield a variety of paths. For example, the shortest path with net minimum displacement is specified as $\arg\min_{\vec{ds}} \Sigma_{i \in \mathscr{P}} |\vec{ds}_i|$. A path can also be derived by minimizing number of building elements traversed from by imposing the constraint $\arg\min_{\mathscr{P} \in P} |\mathscr{P}(\mathscr{E}_p, \mathscr{E}_q)|$.

$$.\mathscr{P}(\mathscr{E}_p, \mathscr{E}_q) = \{(\mathscr{E}_i, \vec{\delta}s_i)|\vec{\mathscr{E}}_{i+1} = \vec{\mathscr{E}}_t + \vec{\delta}s_i, \& E_0 = \vec{\mathscr{E}}_p, E_L = \vec{\mathscr{E}}_q\} \tag{3.9}$$

3.2.2 BIM Parsing

The IFC file used for input to the Building Information Modelling parser comes from an office building in Paris and consists nearly 120,000 IFC entries. There are 139 walls, 122 spaces, 89 doors, and 5 windows covering a net area of 3100 m^2.

The IFC parser defines 2 principal classes (QPRODUCT and Q3DPARSER) to absorb the information defined in a building modelling file. QPRODUCT has a attribute named *globalID* to store the BIM identifier, along with *type of Product*. An instance of the 3D parsing class stores multiple cross-sections of a building element on horizontal and vertical planes. Class functions *returnHullPolygon* and *getBoundingBox* return a list of 3D coordinates at a height denoting a convex hull or the bounding box of the building element. Doors, spaces, storeys, windows, walls and stairs are encapsulated as derived classes of QPRODUCT and Q3DPARSER. Finally all building elements are referenced through a class called QBUILDING which stores the topological connectivity graph and building metadata referenced by the global identifier. Figure 3.4 is a schematic UML diagram of the classes. The software is made available and can be pulled from Docker hub by imagename : *angmit/ifcparser :v2.0*. The parser allows to specify an input IFC file, store the computed results, specify hull formation or graph generation and reload an intermediate output.

Entire building is parsed by setting the direction of cutting plane along Z axis (0,0,1) and the slicing happens at every Δh = 0.5 meter on interval $I_h = [-10, 10]$ resulting in a maximum of $\frac{I_h}{\Delta h} = 40$ images per element. The IFC storeys arranged from bottom to top are underground parking slot, basement, office floor and first floor. The spatial connectivity graph G^S of the building, made of 340 nodes and 1159 links is derived by superimposing images of multiple elements per storey and inferring storey links. The partial graph G^S formed by a single slicing plane $\Xi = (0, 0, 2.5)$ as shown in Figure 3.5a corresponds to an office floor-plan. The effect of combining multiple images yields a well connected graph of 16 spaces as shown in Figure 3.5b. The space-space linkage (G^P) is made up of 16 nodes and 24 edges as shown in Figure 3.9.

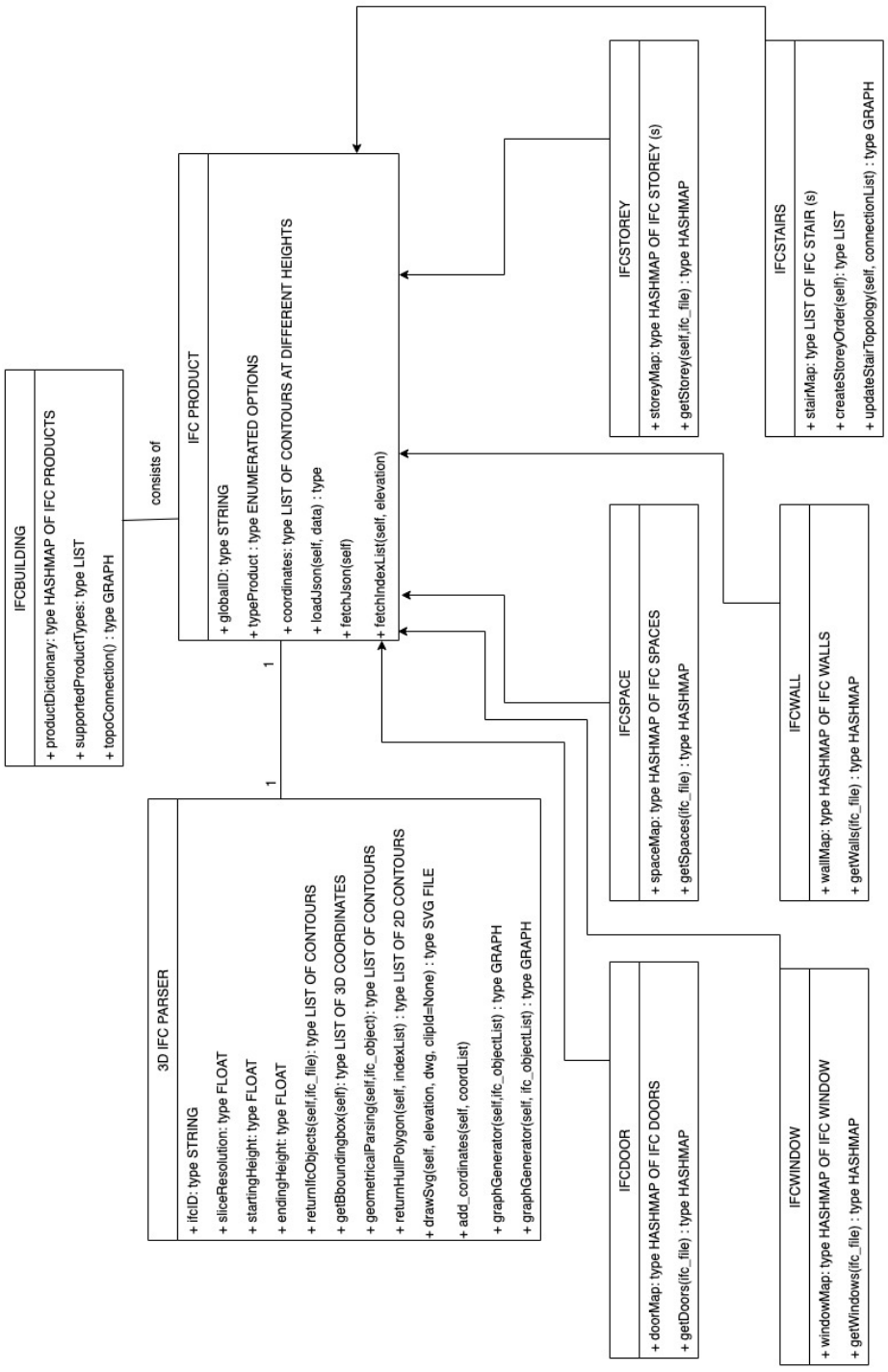
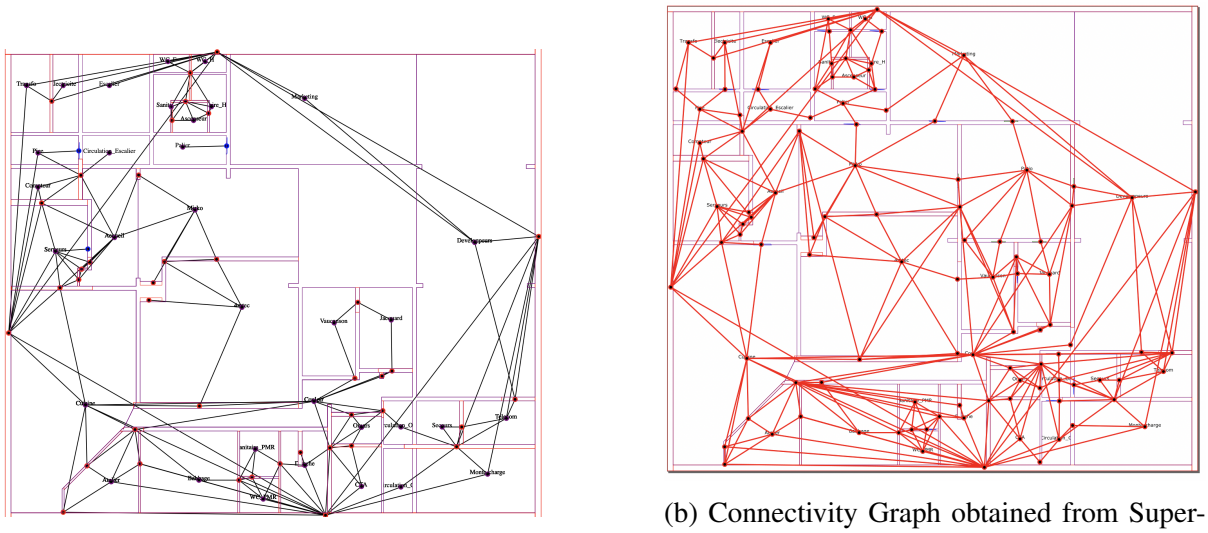


FIGURE 3.4 – Class Diagram of IFC Parser



(a) Partial Graph formed by a slicing plane

(b) Connectivity Graph obtained from Superimposing images

FIGURE 3.5 – Superimposing partial slicing places to render a floor-plan.

3.2.3 One Shot Enrichment

We enrich the building information representation by incorporating static objects that govern the semantic usage of a space. For example, presence of a coffee machine attracts people to brew a cup of coffee with an energy dissipation proportionate to the run-time of the device. Privacy concerns restrict all places to be put under video-surveillance. So to enrich a space, we create a virtual view derived from multiple images. The work aims at embedding a layer of awareness stemming from augmented reality to enhance the semantics of a space with the spatial distribution of tables, chairs, overhead tube-lights, bulbs and other electrical appliances. The intuition behind such an approach is that human activities are concentrated in regions where there are objects of interest and most likely to form the zones of energy dissipation footprints. We want to approximate the semantic utility of space from electrical appliances, lights, furniture arrangements.

Placement of furniture on a floor corresponds to a natural spatial segmentation in a space like Figure 3.7a. Counting furniture and seating capacity is useful in having some basic idea about utility of a space. Detecting the sources of illumination like over head tube-lights, desktop bulbs etc give us the illumination profile of a space as in Figure 3.7b. For the image classification task, first we perform image augmentation using the python library *Albumentation* [54] by altering the rotation, scale, hue and saturation. Next the system utilizes the state of the art deep

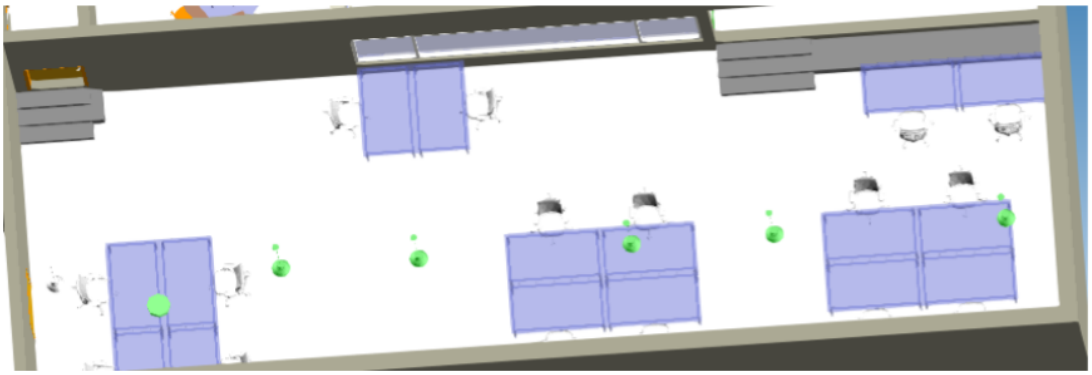
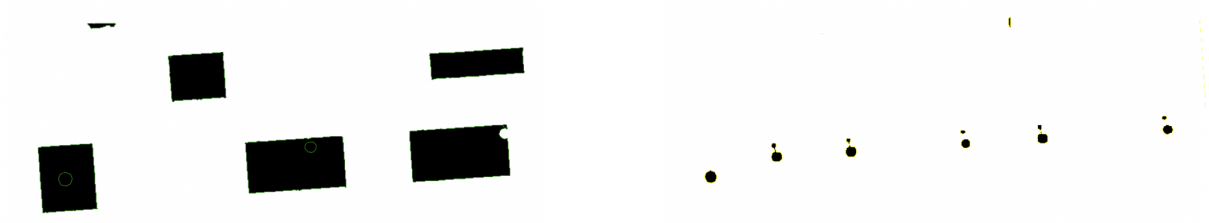


FIGURE 3.6 – Top view of an office space with 5 groups in table (purple) and overhead lights (green)



(a) Furniture Placement

(b) Illumination Coverage

FIGURE 3.7 – Extraction of Illumination Coverage and Furniture Placement from a room.

Space Name	Seating Capacity	Overhead Lights
Sanitary	1	2
Bathroom	1	2
Electricity Room	0	1
Meeting Room	4	1
Unused Room	0	1
Hardware Zone	4	3
Elevator	4	1
Private Office	4	1
Working Zone	4	2
Entrance	1	1
Kitchen	12	9
Reception	2	2
Passage	0	2
Environment	0	0
Marketing	10	6
Developers	16	8

TABLE 3.1 – Region of Interest Profiling

learning models like YOu-Look-Once (YOLO V5) to detect Objects of Interest. Each object is a contained within a semantically segmented image mask, usually a rectangular bounding box represented by two diagonally opposite vertices. We add the seating capacity and indoor lighting requirement for every space in Table 3.1 thereby enriching the structural metadata for a minimal sensing intelligence.

3.3 Non Intrusive Sensing

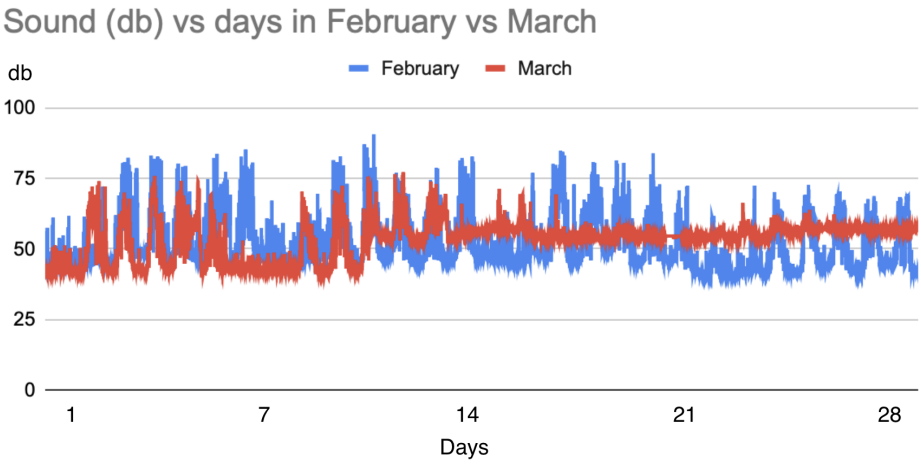
Heterogeneous non-sensory data sources like communication channels, calendars etc. can provide a semantic tag or answer the question "What kind of activity was going in a space ?" For an office setting, it can be to check the availability of meeting rooms or find out where are the most probably empty spaces during different hours of a day. Sometimes the *agenda* is so implicit in space-time that it does not need mention. For example, at lunch time, kitchen gets utilised by default and hardly any one registers a events for it, unless it is special. Also having a semantic label may not always mean the truth like, if someone forgets to cancel an agenda for a meeting that did not take place, it will generate a false positive. This observation adds to

Sensors	Observations	Reason
Light	Bias	Placed in a space with window and there is ample sunlight.
Light	Activity signal	Placed in a space without window and is switched on during an activity.
Temperature	Bias	Sensor placed close to probable heated surfaces or has strong incident light.
Sound	Activity Signal	Placed in a room where human activity generates sound.
Sound	Bias	Placed in a noisy environment.
Motion	Bias	Placed incorrectly so that it gets unwanted signals.
Motion	Activity Signal	Placed in a space to detect if the space has human intervention.

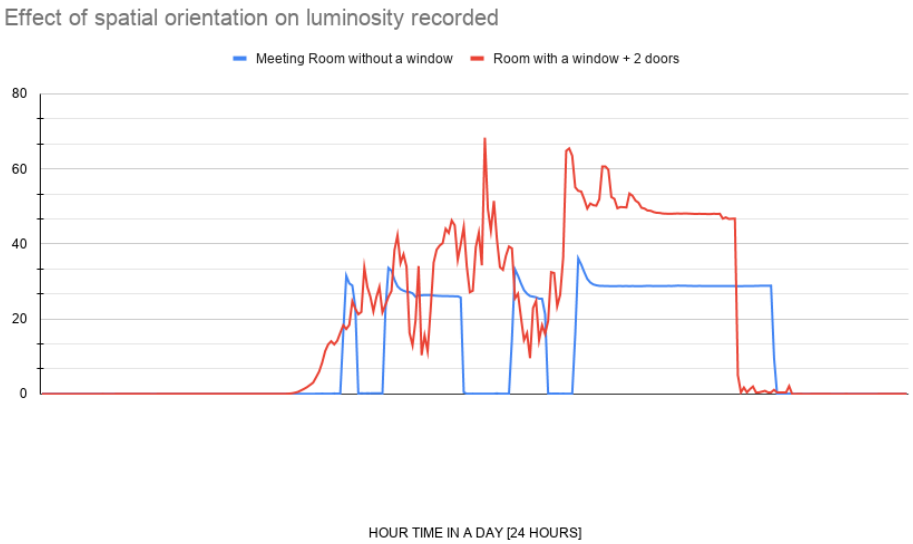
TABLE 3.2 – Possibility of sensor distribution variation due to spatial and usage constraints

the motivation for tag-less learning, where the privacy advantage is the non-incorporation of sensitive data by design. We augment the spatial knowledge abstracted from a floor-plan with interaction models to derive a layer of non-intrusive sensing.

Building data has varying spatio-temporal patterns, for example in a room with window there is a gradual increase in luminosity values from dawn till mid day before starting to decrease from the evening, although in a window-less room we see a sharp rise and fall in sensor values when lights are on and off respectively as shown in Figure 3.8b. Additionally, pre-pandemic and post pandemic data collection periods from an office building show distinct temporal changes. Table 3.2 enlists some reasons to justify disparate sensor patterns arising due to spatial difference. Figure 3.8a show the sound distributions of February and March plotted against day of month on X axis in blue and red respectively. Notice that the red line falls around week starting from 14th March, which coincides with lock-down issued in France for the first time. The distinct pattern of 5 broad and 2 small spikes corresponding to 5 working days and 2 days of a week-end in an office. We now proceed towards a system that non intrusively detects occupancy intervals without any supervision.



(a) Variation of sound intensity from an identical room in successive months.



(b) Variation of luminosity level in rooms with and without windows in red and blue respectively.

FIGURE 3.8 – Effect of spatial and temporal effects on sensor signals.

3.3.1 Tag-less Sensing

In order to automate the labelling process, the system predicts the occupancy status from non-intrusive ambient sensor values distributed across spaces. Let the data coming from i^{th} sensor be represented as D^i and total data be D = $\cup D^i$ $\forall i$. For non intrusive detection, the occupancy label O_i^t is absent and thus certain heuristics are applied to approximate the same. The task is split into guessing and exploitation stages as follows :

— **Discovery Stage** : We observe that a human activity signature is captured between time dilated maxima and minima points in a sensor stream. For every sensor channel we extract two local τ dilated optimal sets per channel, the local maxima stream $X_{max}^i = \{x_t | x_t > x_{t-\tau} \wedge x_t > x_{t+\tau} \forall x_t \in D_i\}$ and the local minima stream $X_{min}^i = \{x_t | x_t < x_{t-\tau} \wedge x_t < x_{t+\tau} \forall x_t \in D_i\}$. The percentile score of a local optimum (x_t) from sensor i is given as fraction of points lower than the current point as per Equation 3.10. For I multi-modal information sources, we extend the scoring as a weighted percentile, where w^i is the weight for i^{th} signal such that $\Sigma_{i \in I} w^i = 1$ as per Equation 3.11.

$$\eta^i(x_t, X = X_{min} + X_{max}) = \frac{|\{x | x < x_t, \forall x \in X\}|}{|X|} \tag{3.10}$$

$$\rho(\{x_t^i\}, \forall i \in I) = \frac{1}{|I|} \Sigma_{i \in I} \eta^i w^i \tag{3.11}$$

— **Learning Stage** The system selects the top and bottom m frequency counts of $\rho(\{x_t^i\}, \forall i \in I)$ and it labels the corresponding time-slices as $y_i^t = O_i^t = 1$ as "occupied" or $y_i^t = O_i^t = 0$ "idle" otherwise. The tagged data is over-sampled and is given as input data $D \equiv \{X, y\}$ to statistical machine learning algorithms, where y is the occupancy. The confidence factor per class is the average disturbance level of y_k $T(y_k) = \bar{\rho}(\{x_t^i | LABEL(x_t^i) = y_k\}, \forall i \in I)$ where h_k is the learnt hypothesis for class k. The Empirical loss $L(D,h)$ evaluated by a hypothesis or machine learnt model (h) against a data D is given as $L(D,h) = \Sigma_{(x,y) \in D} L(h(x), y)$ where x, y are the input feature vector and target respectively. The optimal local model for the i^{th} client at time t is given as $h_i^t = \arg\min_{h \in H} L^t(D_i^t, h)$. To evaluate the accuracy, the recorded spatio-temporal events in the office calendar are

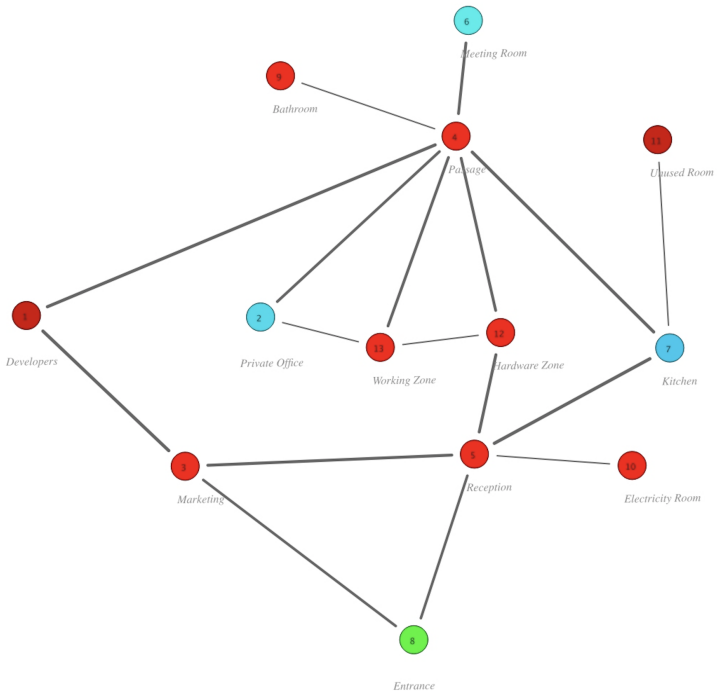


FIGURE 3.9 – Spatial Path Graph (G^P) of spaces in the floor-plan. The points in blue are spaces where sensors are placed and access to the floor-plan is highlighted in green.

considered as the ground truth.

3.3.2 Connected Space Model

In an office setting, paths that lead towards entering or exiting the building are most likely to be utilized during office beginning or ending hours respectively. Similarly at lunch, employees gather at the kitchen/cafeteria and are usually expected to desert their working spaces. We observe that after an activity at space S_0 comes to an end, people disperse from that location to traverse/occupy adjacent spaces. Here the motivation is to understand the importance of adjacent neighbours of a space with through a dynamic model of connected spaces, computed as follows :

1. We construct an adjacency matrix $\mathscr{I}^t$ where edge weights between a space and its 1 hop neighbour is modelled as the dispersion capacity of humans occupying corresponding spaces. Optimistic mixing is estimated as the maximum count of crowd amongst two spaces while the pessimistic approach is having the minimum of crowd. If c_i, O_i^t are

the seating capacity and Boolean occupancy status of space i at time interval t, then Equations 3.12, 3.13 give the edge weights for optimistic and pessimistic dispersion respectively.

2. The largest eigenvalue (λ_{max}) of the adjacency matrix (A) is computed as $AX_m = \lambda_{max}X_m$ [55] and intuitively relates to stretching A in the direction of maximum activity influence by a force vector X_m. The score of a space i is equal to the i^{th} entry of X_m. We note that the special case of no people inside the office corresponds to all edge weights equal to 0 and A reduces to a zero-determinant matrix probably indicating all unnecessary electrical appliances should be shut off.

$$\mathscr{I}^t(V_i, V_j) = \underbrace{max(c_i \times O_i^t, c_j \times O_j^t)}_{Optimistic\ Estimate} \tag{3.12}$$

$$\mathscr{I}^t(V_i, V_j) = \underbrace{min(c_i \times O_i^t, c_j \times O_j^t)}_{Pessimistic\ Estimate} \tag{3.13}$$

3.3.3 Occupancy Prediction

We demonstrate the utility of tag-less sensing on a data set consisting of 2.1 Million Sensor readings, sampled from 3 rooms : a kitchen, private space and a meeting room. The sensor channels are luminous intensity(lux), sound (decibel), temperature (celcuis), relative humidity, motion, energy meter (milliWattHour). The first step towards activity detection is automated data labelling, which is done by varying the time dilation parameter τ between 1 to 5, representing a resolution of 5 to 30 minutes. We observe the most optimal detection with $\tau = 2$ or a 10 minutes comparison window. The experimentally determined best weight table for the sensors are shown in Table 3.3. The labelled data is up-sampled to reduce class imbalance via Synthetic Minority Oversampling Technique [56]. Next comes training supervised learning models namely XgBoost, Support Vector Machine and Decision Trees on the auto-labelled data via a k-fold cross validation (k=5).

We compare the models via the performance metric of F-1 Score which is the harmonic mean of precision and recall. The supervised learners have a mean F1 score of 0.81-0.85 with

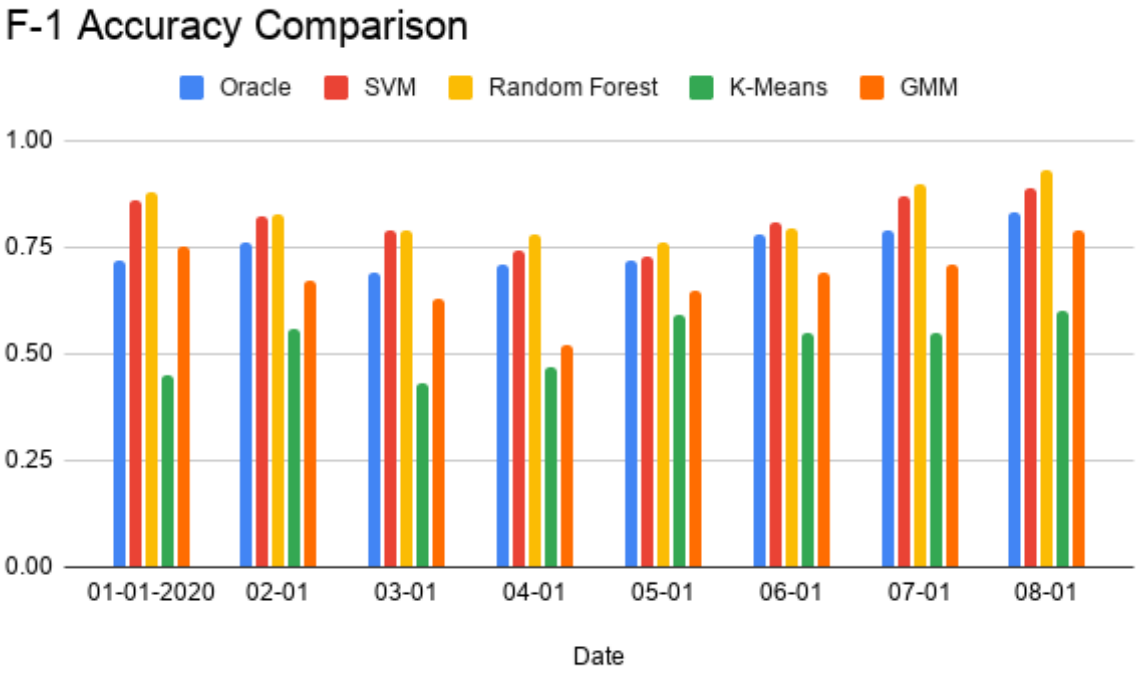


FIGURE 3.10 – Comparison between losses of oracle and trained models.

Sensor	Weight
CO_2	0.1
Light	0.2
Humidity	0.1
Temperature	0.1
Sound	0.3
Motion	0.2

TABLE 3.3 – Experimentally determined sensor weights used for activity discovery.

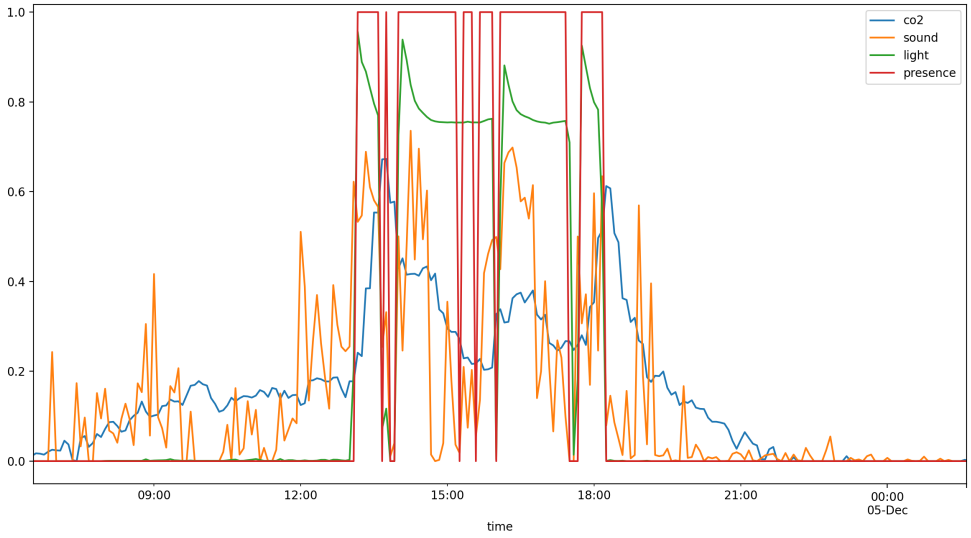


FIGURE 3.11 – Sensor values and detected occupancy on a 0-1 scale normalized view.

a variance of 0.01- 0.06. The baseline comparison in our setting are unsupervised models. For each class we averaged the input feature set into a single dimension vector to initialize the seed for K-Means and Gaussian Mixture Model (GMM). GMM performed slightly better than K-Means although the averaged F1 score varied between 0.65-0.75 with a variance of 0.07-0.1. Mean F-1 score is shown in Fig 3.10 and refers to the monthly model performance with retraining on previous month's data. We observe that efficiency of SVM and Random Forest is higher than the oracle by 9 to 16 % while unsupervised models fail to capture intuitive heuristics embedded in the oracles. We cross validate our observations by taking the cosine distance between cluster-centers, a averaged input data vector per class predicted by each classifier. The cosine distance between cluster-centers of supervised and unsupervised model groups is 0.19 while intra-group difference between K-Means and GMM is 0.27. The maximum distance of cluster-centers between SVM, XGboost and Random Forest came out as 0.07. Additionally, we cross check by filtering through sanity checks of no occupancy on weekends, holidays and at night.

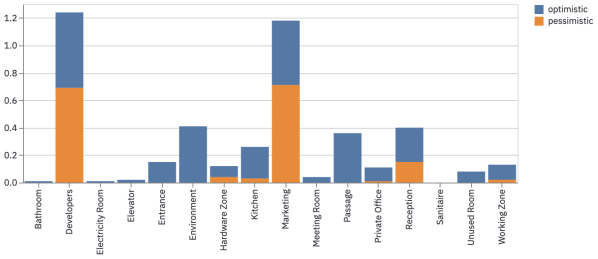


FIGURE 3.12 – Energy Score Distribution of spaces assuming fully occupancy

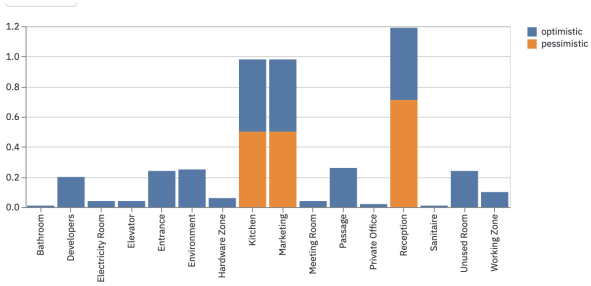


FIGURE 3.13 – Lunch time office occupancy scenario with dynamic edge weights

3.3.4 Spatial Spectral Analysis

The system leverages the semantic information about a space to generate energy insights. For example, in the case of full occupancy $O^t = [1]_{1\times I}$ represented by a vector of all ones, eigenvalues gives an intuition about spatial distribution of electrical power. Figure 3.12 shows the belief distribution against pessimistic and optimistic dispersion along edges of the fully occupied floor-plan as denoted by Equation 3.12. The 3 highest score sums correspond to the Developer's Zone (1.25), Marketing Zone (1.11) and Reception (0.4). While first 2 spaces can be explained in term's of seating capacity, the importance of reception reflects the likelihood of space-usage for entering or exiting the floor-plan. The passage has a score of 0 under minimal mixing strategy, indicating if adjacent activities are concentrated in their respective zones, then the passage is least likely to be used. But it becomes the fourth most important space (0.36) when occupants are shuffling spaces and is explainable by high degree centrality. The edge weight formulation helps to answer questions like "If there are no people at the developer's floor, hardware zone and office what are the most likely occupied places ?". Figure 3.13 helps to answer the question by showing the top 3 spaces as the reception (1.05), marketing zone (0.97) and kitchen (0.96). This scenario corresponds to the spatial distribution of people during

lunch time and the output is affirmed by the regular ground truth.

3.4 Chapter Summary

Industry Foundation Classes is a set of ISO standards to describe a building with high complexity but poor readability, although is now a mandate for European constructions. The goal is to generate a low complexity abstraction by parsing ISO formatted Building Information Modelling files. Typically recursive and referential geometry is used to describe spatial orientation of all building elements like doors, windows, walls, roof, stairs, storeys etc. In contrast, the developed geometric approach acts on an unique reference frame that helps to check if two elements share physical boundaries. Consequently, a non-zero overlap indicates connectivity between structural elements which is the key to represent building as an ensemble of nodes and edges. Furthermore the meta-data is enriched without sensors but pictures of spaces instead to estimate the seating capacity and count energy dissipation sources. The chapter introduces the concept of zero-sensor intelligence by embedding human-space interaction models on a graph-based spatial abstraction of a building, such as Qarnot's office. Spectral decomposition of the semantically enriched connected graph helps to rank multiple spaces regarding temporal importance or likely energy dissipation. A commonly occurring machine learning problem of smart buildings namely occupancy detection is studied under the constraint of zero explicit labels. Auto-learning at the sensor level helps to map the signature of human activities to the feature space through an explainable knowledge discovery step. Thus the work establishes an investigative report to exploit building intelligence in a zero sensor setting.

Chapitre 4

Minimal Sensing Solution

Sensors act as perception devices for machine executed intelligence. Let us look at the process of equipping a multi storeyed building with sensors that answers the following set of questions at least :

- Where to install what type of sensors ?
- How many sensors are needed in total ?
- What will approximately be the one-time cost ?
- How will equipping sensors help to save power consumption, whereas it itself needs power to run ?
- How will be the sensor-generated data utilised to generate intelligence ?
- Is the data localised within the building ?

This motivates us to add a layer of cost and energy savings through removing or powering off sensors from a completely equipped setting. The sensor reading is instead given by a Virtual Sensor Field which is a mix of physical and machine learnt value approximations of missing sensors. The design is inspired from the concept of a virtual machine where a parent operating system (OS) hosts sub environments each mimicking an OS. The design increases privacy levels and decreases data footprint generated at edge since sensors are missing ! The key element in designing a distributed Virtual Sensor Field is to find the optimal group of sensors that can approximate the missing ones through data driven machine learning. The system evaluates the error generated by the approximation on multiple operational objectives to yield a minimalist

set of possible solutions. Each solution is then interpreted as a set of locations to place sensors along with their types.

4.1 Virtual Sensor Field

Virtual Sensor Field is defined as a mixed basket of physical and computable sensors that creates a virtual avatar over a set of sensors distributed over multiple spaces. The system seeks to place the minimal number of real sensors that can serve as input to machine learnt approximating functions to reliably cover up for hidden/missing sensors. The system continually evaluates the penalty of approximation on multiple operational objectives to yield a set of possible solutions. Each solution is then interpreted as a set of locations to place sensors along with their types.

4.1.1 Sensor Grouping

Typically, sensors in a smart building measure ambient properties like temperature, humidity, CO_2 etc. and energy consumption from sources like illumination, heating, appliances etc. Let us assume that G disjoint logical groupings can be made from at-max S types of sensors which fill up a set of Z zones in a building. Each group $g \in G$ is further bi-parted into two sub groups namely the support (X_g) and approximated (Y_g) sets. The link between $\{X_g, Y_g\}$ is given by a hypothesis space $\mathscr{H}^g$ defined by Equation 4.1 to support a bidirectional mapping between the two subgroups.

$$\mathscr{H}^g = \begin{bmatrix} \mathscr{H}_f^g & : X_g \rightarrow Y_g \\ \mathscr{H}_b^g & : Y_g \rightarrow X_g \end{bmatrix} \tag{4.1}$$

The quality of $\mathscr{H}^g$ is evaluated through a cost function L executed over all possible pairwise interaction pairs $(u,v)\ \forall (u,v) \in \{X_g + Y_g\}^2, u \neq v$. The error in predicting channel $v \in Y_g$ using a predictor $u \in X_g$ is recorded at the $[u,v]^{th}$ cell of an error matrix M_g^E as per Equation 4.2.

$$M_g^E[u,v] = L(v, \underbrace{\mathscr{H}^g[u,v]}_{ML\ model}(u)) \tag{4.2}$$

Note that $M_g^E[u,v] \neq M_g^E[v,u]$ implies that the two losses generated by swapping the dependent and independent variable may not be equal. To estimate the value y_v of a channel v that lies in group Y_g, we first select the optimal channel (u^*) to predict by using the $[u^*,v]^{th}$ entry of hypothesis library $\mathscr{H}_f^g$ as per Equation 4.3.

$$\begin{aligned} u^* \leftarrow &\quad \arg\min_{g \in G} M_g^E[u,v] \\ y_v = &\quad \mathscr{H}_f^g[u^*,v](u^*) \end{aligned} \tag{4.3}$$

This technique bounds the maximum observable error since it is possible that another optimal mapping $\mathscr{H}^*$ can exist using more than one feature for prediction.

$$L(Y_g, \mathscr{H}^*(X_g)) \leq \max_{u \in X_g, v \in Y_g} L(v, \mathscr{H}^g[u,v](u)) \tag{4.4}$$

4.1.2 Minimal Support Group

Let S represent a set of n^S sensors that are distributed over n^Z spaces or building zones. We store the combined information of two 1D vectors of length n^s namely virtual (M^V) and affinity (M^A) masks. At position i, if a sensor has to be powered on physically, the virtual mask encoding is $m_i^V = 1$ and 0 if the digital twin suffices. The role of the affinity mask M_A is to hold the information on the logical grouping behind the sensors, so for n^K types of sensors, the tag elements will belong from $m_i^A \in \{1,2 \ldots n^K\}$, while encoding elements to group by spaces is restricted to $m_i^A \in \{1,2 \ldots n^Z\}$. For a given group $g \in M^A$, the sub-sequence consisting of $\{0,1\} \in M^V$ is interpreted as $\{X_g, Y_g\}$ denoting the real and hidden sensor respectively. The simple transformation function $M^S = 2M^A + M^V$ gives the final encoding M^S at any position in a string of n^S numbers.

The individual task for every group g is to learn the best mapping $\mathscr{H}^*$ from $X_g \xrightarrow{\mathscr{H}^*} Y_g, M^V = X_g + Y_g$ with a bi-objective optimization. The solution to such kind of problems is typically a set of 'non-dominated' solutions where any O_i can not be improved without increasing the others $O_j, j \neq i$. We define first two objectives to measure the prediction error due to

forward $\mathscr{H}_f^g$ and backward $\mathscr{H}_b^g$ hypothesis spaces by Equations 4.5 and 4.6 respectively.

$$O_1(M^V|M^A) = \Sigma_{g\in M^A}\Sigma_{v\in X_g}\Sigma_{u\in Y_g}\left\{\frac{M_g^E[\mathbf{u},\mathbf{v}]}{|X_gY_g|}\right\} \tag{4.5}$$

$$O_2(M^V|M^A) = \Sigma_{g\in M^A}\Sigma_{v\in X_g}\Sigma_{u\in Y_g}\left\{\frac{M_g^E[\mathbf{v},\mathbf{u}]}{|X_gY_g|}\right\} \tag{4.6}$$

Algorithm 2 Minimal Support Group Solver

Input : Initial chromosome pool of size n^C $\{P_t^i\}\forall i \in [1,n^C]$, N objective functions $\{O_i\}\forall i \in [1,N]$, Iteration Limit T_{max}

Initialize $t \leftarrow 0$, $Q_{t=0} = \phi$
while $t \leq T_{max}$ **do**
  $R_t \leftarrow P_t \cup Q_t$
  $\mathscr{F}$ = NON-DOMINATED-SORTING(R_t)
  while $|P_{t+1}| < M$ **do**
    crowding-distance-assessment$(\mathscr{F}_i)$
    $P_{t+1} = P_t \cup \mathscr{F}_i$
    Sort$(P_{t+1}, \leq_M)$
    $P_{t+1} = P_{t+1}[0:M]$ {▷ Select top M members every time from a population}
    Q_{t+1} = **make-new-pop**(P_{t+1})
  end while
  $t \leftarrow t+1$
end while

4.1.3 Lifelong Policy Optimiser

For a given grouping affinity mask M^A, Algorithm 2 generates a set of n^C Pareto optimal solutions or minimal groups. The system experimentally investigates the quality of these groups that can optimally power up the virtual sensor field. Over time, predictions at such blind spots may deviate in time or installing sensors can become a necessity for optimal configurations. Thus the affinity grouping M^A have the possibility to be re-calibrated with the availability of additional data, but such a process must take into account the historic performance. For any $t \in T$, reconstruction loss is the absolute difference between the actual (y_v) and predicted value

Algorithm 3 Solver Routines

procedure CROWDING-DISTANCE-ASSESSMENT(R_t, O)
  for every objective $j \in 1$ to N **do**
    Generate Fitness-Vector$_j = \{O_j(P_i)\} \forall i \in [1, M]$.
    Sort(Fitness-Vector$_j$)
    Fitness-Vector$_j$[0] = Fitness-Vector$_j$[-1] = ∞
    for i = 2 to $M-1$ **do**
      Fitness-Vector$_j$[i] += (Fitness-Vector$_j$[i-1] - Fitness-Vector$_j$[i+1])
    end for
  end for
end procedure
procedure NON-DOMINATED SORTING(R_t)
  for every chromosome $p \in R_t$ **do**
    for every chromosome $q \in R_t$ **do**
      if $p \succ q$ **then**
        $S_p \leftarrow S_p \cup \{q\}$ { ▷ Support set of a dominating chromosome p. }
      else if $q \succ p$ **then**
        $n_p = n_p + 1$ { ▷ Count how many solutions are superior in R_t to q. }
      end if
    end for
    if n_p=0 **then**
      $\mathscr{F}_1 = \mathscr{F}_1 \cup \{p\}$ { ▷ Select only non-dominating solutions as the first front. }
    end if
  end for
  i= 1
  while $\mathscr{F}_i \neq 0$ **do**
    $\mathbf{C} = \phi$ { ▷ For every front, incrementally add sensors starting from zero.}
    for each p $\in \mathscr{F}_i$ **do**
      for each q in S_p **do**
        $n_q = n_q - 1$
        if $n_q = 0$ **then**
          $\mathbf{C} = \mathbf{C} \cup \{q\}$ { ▷ Add non-dominant sensors to a placement configuration}
        end if
        $i = i + 1; \mathscr{F}_i = \mathbf{C}$
      end for
    end for
  end while
end procedure

$\hat{y_v}$ of a sensor for a virtual mask is given as per Equation 4.7.

$$O_3(M^S, T) = \frac{\Sigma_{t \in T} \Sigma_{i \in S}}{|T| n^S} |y_i(t) - \hat{y}_i(t)|$$

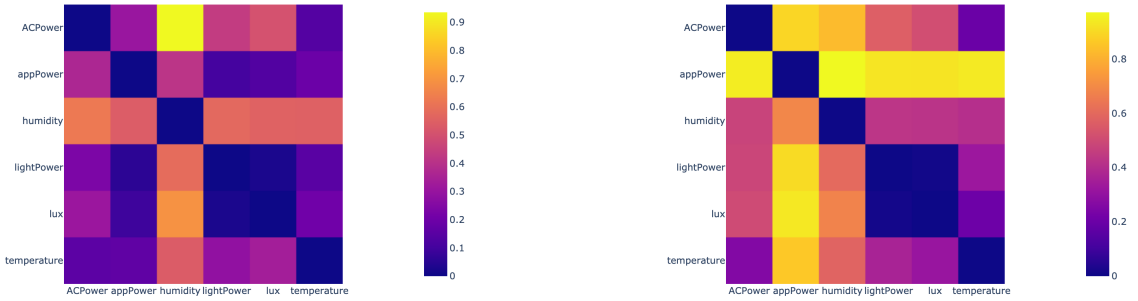
$$\hat{y}_i(t) \in \begin{cases} \mathscr{H}_f^t(X_g) & if\ m_i^V = 0 \\ \mathscr{H}_b^t\left(\mathscr{H}_f^t(X_g)\right) & if\ m_i^V = 1 \end{cases} \tag{4.7}$$

The policy maker additionally models the network topology of sensors in an effort to minimise the number of data sharing links. We define a mapping between a graph topology and M^S using the following rule : For a group g, the sub-set of nodes in X_g and Y_g form two bipartite sets where a connection between (u,v) exists if sensor channel u is the optimal choice to predict v as per Equation 4.3. Let every node i has e_i^I number of incoming edges and e_i^O outgoing connections. Equation 4.8 gives O_4 defined as the ratio between the number of edges in M^S to total edges in a complete graph.

$$O_4(M^S) = \frac{1}{n^s(n^s - 1)} \Sigma_{g \in G} \Sigma_{i \in M_g^V} (e_i^I + e_i^O) \tag{4.8}$$

In this context, lifelong learning is used to minimize the size of future uncertainty and identify a stable sensor field configuration with a minimal data sharing policy. In the re-calibration step, Algorithm 2 is re-used where objectives are represented by a $[4 \times 1]$ matrix given by Equation 4.5 - 4.8.

4.2 Virtualization Validation



(a) M^z where z = Floor 7 Zone 5

(b) M^z where z = Floor 4 Zone 1

FIGURE 4.1 – Error Matrix for 2 zones

To guess a sensor $v \in A^c(g)$, the system learns a set of supervised machine learnt predictors with a single channel data input $u \in A^c(g), u \neq v$. For every group, $O(|A^c(g)|^2)$ learners are trained to capture the intra group sensor dynamics. For a prediction task, a learner picks the least erred model from a set of classical algorithms : linear regression, lasso net, random forest, and XGBoost. Let for a group g the approximated (A_g) and ground truth be represented by vectors $\hat{y}_s^z$ and y_s^z for sensor s at location z respectively. The mean squared error is given as :

$$L_{MSE} = \frac{1}{KZ} \sum_{k \in K} \sum_{z \in Z} (\hat{y}_s^z - y_s^z)^2$$

We consider the data-set from [57] for the experiments. It comes from a 7 storey building in Thailand including 24 smart zones with 1.5 years of data collected at 1 minute resolution. The analysis highlights 3 key decomposition steps to build up a Virtual Sensor Field a) policy evidence : error matrices $(M^V, M_g^E | \forall g \in G)$ to judge the quality of virtualization accuracy b) policy : Sensor mask $(M^V | M^A)$ which is given by the encoding of sensors to generate virtual sensor field according to a specific data policy c) and life-long quality checker that optimises data sharing policy (M^A) with more data.

First we check the effect of spatial grouping on the virtual sensor field accuracy. Variations in Figure 4.1 are explained by the fact that zone 5 is from top Floor 7 and experiences a harsher environment like direct sun light, higher temperature/humidity fluctuations than zone 1 of Floor

Zone	Power			Ambience		
	AC	Light	App	Temp	RH	Lux
Floor2Z1	0.15	0.14	0.13	**0.18**	0.53	0.15
Floor2Z2	**0.08**	0.07	0.15	**0.11**	0.36	**0.06**
Floor2Z4	0.33	0.31	0.73	0.33	0.66	0.31
Floor3Z1	0.32	0.23	0.38	0.24	0.45	0.26
Floor3Z2	0.35	0.25	0.27	0.29	0.4	0.27
Floor3Z4	0.34	0.23	0.25	0.22	0.61	0.22
Floor3Z5	0.42	0.25	0.27	0.28	0.63	0.24
Floor4Z1	0.28	0.24	0.19	0.2	0.53	0.26
Floor4Z2	0.34	0.27	0.48	0.25	0.59	0.25
Floor4Z4	0.28	0.25	0.29	0.24	0.53	0.24
Floor4Z5	0.36	0.18	0.35	0.23	0.46	0.17
Floor5Z1	0.23	0.2	0.19	0.15	0.45	0.22
Floor5Z2	0.29	0.19	0.28	0.19	0.35	0.19
Floor5Z4	0.33	0.36	0.31	0.3	0.58	0.3
Floor5Z5	0.43	0.26	0.29	0.31	0.64	0.26
Floor6Z1	0.26	0.23	0.25	0.29	0.37	0.22
Floor6Z2	0.36	0.28	0.22	0.28	0.38	**0.3**
Floor6Z4	0.26	0.17	0.27	0.22	0.41	0.21
Floor6Z5	0.47	0.22	0.26	0.23	0.58	0.23
Floor7Z1	0.34	0.28	0.43	0.31	0.65	0.48
Floor7Z2	0.31	0.3	0.59	0.33	0.61	0.23
Floor7Z4	0.28	0.21	0.28	0.23	0.41	0.2
Floor7Z5	0.44	0.38	0.71	0.34	**0.61**	0.36

TABLE 4.1 – Virtual Sensor Field Accuracy (L_s) with Spatial Grouping or predicting a cell using **columns from the same row**.

4. Figure 4.2 shows the virtualization error for domain wise grouping as a heat-map covering all zones. We observe that AC Power bears a negative correlation with temperature and humidity when AC is turned on primarily for cooling. Light Power is positively related to indoor luminosity levels by observing at night the lights are off and during day, the lights are turned on for acceptable visibility levels. Power consumed due to appliances is observed to inversely vary with indoor temperature and humidity, which likely indicate working conditions in a controlled thermal environment. Temperature and humidity also has a negative correlation with AC and appliance power intuitively meaning appliances are running more when ambience is controlled due to occupancy. We see that appliance power (MSE ≈ 0.9) is the worst approximation on Floor 4 Zone 1 (F4 Z1), while it is humidity (MSE ≈ 0.8) for Floor 7 (F7 Z5).

Zone	Power			Ambience		
	AC	Light	App	Temp	RH	Lux
Floor2Z1	0.08	0.07	0.11	0.25	0.14	0.05
Floor2Z2	0.09	0.06	0.13	0.24	0.25	0.09
Floor2Z4	0.09	0.06	0.12	0.26	0.66	0.06
Floor3Z1	0.11	0.03	0.08	0.04	0.26	0.03
Floor3Z2	0.07	0.04	0.09	0.05	0.17	0.03
Floor3Z4	0.09	0.03	0.11	0.06	0.23	0.03
Floor3Z5	0.08	0.03	0.1	0.07	0.18	0.05
Floor4Z1	0.08	0.03	0.08	0.06	0.16	0.05
Floor4Z2	0.06	0.02	0.11	0.05	0.44	0.04
Floor4Z4	0.07	0.02	0.08	0.05	0.22	0.05
Floor4Z5	0.14	0.03	0.08	0.07	0.36	0.06
Floor5Z1	0.15	0.03	0.13	0.06	0.2	0.08
Floor5Z2	0.08	0.03	0.09	0.07	0.29	0.04
Floor5Z4	0.17	0.03	0.09	0.06	0.17	0.04
Floor5Z5	0.07	0.05	0.12	0.05	0.13	0.04
Floor6Z1	0.12	0.04	0.08	0.04	0.25	0.13
Floor6Z2	0.27	0.03	0.08	0.06	0.19	0.32
Floor6Z4	0.14	0.04	0.1	0.06	0.26	0.09
Floor6Z5	0.1	0.04	0.09	0.12	0.3	0.09
Floor7Z1	0.64	0.05	0.12	0.08	0.36	0.08
Floor7Z2	0.08	0.05	0.08	0.09	0.5	0.14
Floor7Z4	0.07	0.03	0.09	0.08	0.35	0.03
Floor7Z5	0.08	0.03	0.08	0.09	0.63	0.05

TABLE 4.2 – Virtual Sensor Field Accuracy (L_d) with Domain Wise Grouping or predicting a cell using **rows from the same column**.

$$\begin{aligned} L_s[z,d] = &\quad \Sigma_{v \in A^c(g=z)} \frac{M^z[v,d]}{|A^c(g=z)|} \\ L_d[z,d] = &\quad \Sigma_{v \in A^c(g=d)} \frac{M^d[v,z]}{|A^c(g=d)|} \end{aligned} \tag{4.9}$$

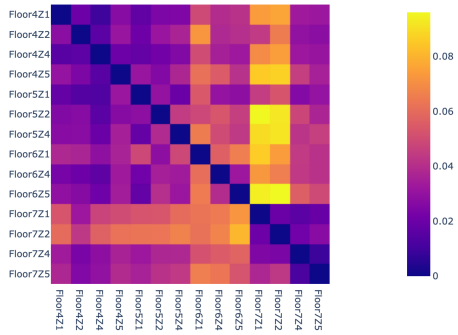
Due to lack of space, it is impossible to show all M^g tables, instead we report the spatial (L_s) and domain (L_d) wise loss to fill up Table 4.1 and 4.2 respectively. Amongst all the sensors, relative humidity is the most difficult to approximate, L_s (min=0.37, max=0.66) and L_d = (min=0.14, max=0.63) across all zones.

4.2.1 Policy Evidence

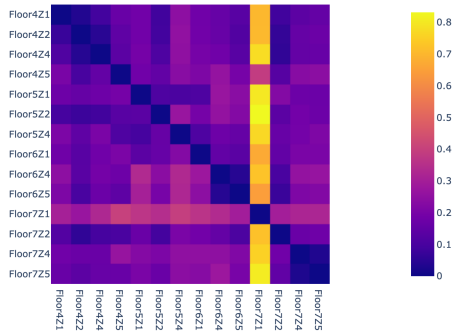
The first type of task is to train predictive algorithms patterns between any two sensor channels (u,v) for a group $g \in M^A$. Disjoint grouping helps in computing the hypothesis space $\mathscr{H}^g$ and the error matrix table over n^G computing nodes in parallel. Thus n^G logical groupings are computed with a time complexity of $O(n^G W^2)$ where $W = max_{g \in M^A} |n^g|$ and space complexity per node is $O(W)$. For every pair of sensor channels in a group, the compute center trains a set of classical non-deep algorithms and generates an error matrix for each of linear regression, random forest, and XGBoost. We study the accuracy of recorded loss $\{M_g^E\}|g \in M^A$ under 2 meaningful grouping schemes constructed to give the identical group numbers to sensors belonging to same domain type or for same zone placement. For spatial grouping (case 1) $n^G = n^Z$ and the i^{th} place encoding is given by $M_i^A \in \{1,\ldots,n^z\} \forall i \in S$ while for domain wise grouping (case 2) $n^G = n^K, M_i^A \in \{1,\ldots,n^K\}$. We analyse the error matrices M_g^E to answer, "What is the trade off in terms of accuracy between keeping a sensor powered on and alternately switched off ? The initial policy will be based on evidence till $t_0 = 10^4$ time-steps and the empirical error per type of sensor in a spatial ($L_s|g \in Z$) encoding and sensor domain wise grouping ($L_d|g \in K$) is described at time t by $L_x^t(u) =$

$$\begin{bmatrix} max_{\forall v \in g} & \underbrace{M_g^E[\mathbf{u},v]}_{Forward\ Translation} \\ min_{\forall v \in g} & \underbrace{M_g^E[v,\mathbf{u}]}_{Backward\ Translation} \end{bmatrix}$$

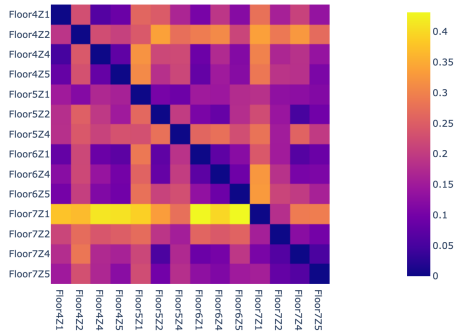
We compare our approach with a similar type of problem that eliminates non redundant ones



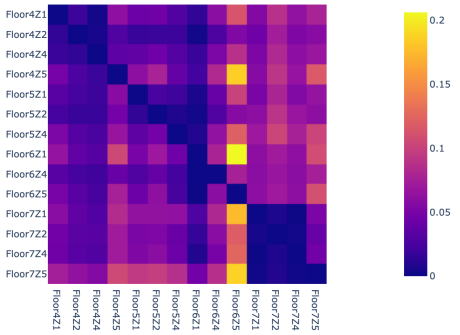
(a) Temperature



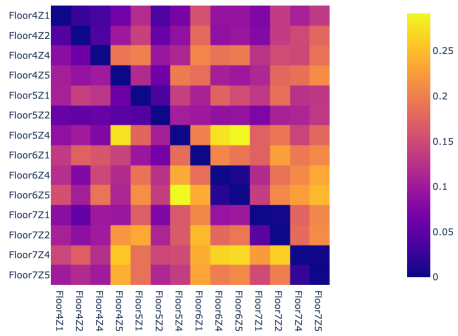
(b) Luminosity levels



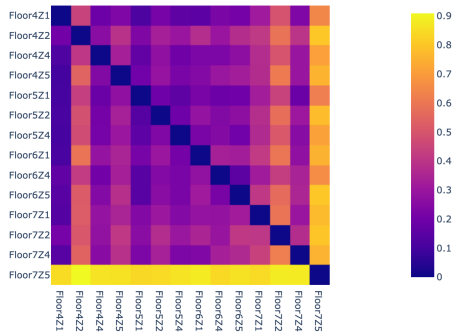
(c) Humidity



(d) AC Power

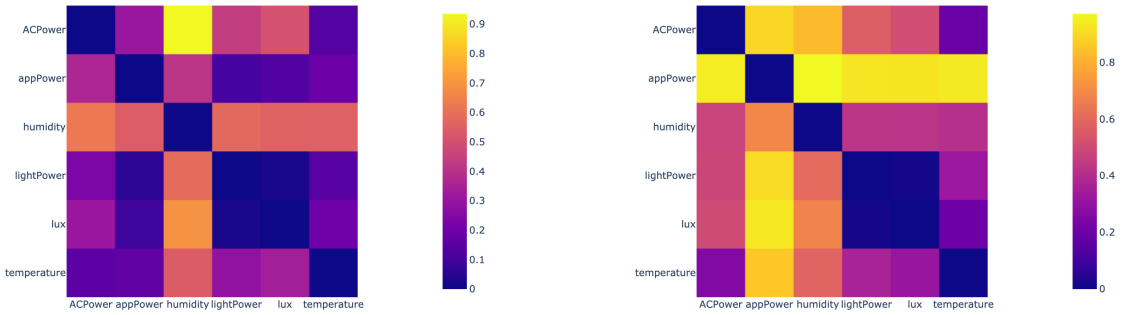


(e) Light Power



(f) Appliance Power Consumption

FIGURE 4.2 – Virtualization prediction on processing similar ambience channels as one group



(a) M^z where z = Floor 7 Zone 5

(b) M^z where z = Floor 4 Zone 1

FIGURE 4.3 – Error Matrix for 2 zones

from a sensor set. For a fair comparison, we evaluate the value of the objective set $O = [O_1, O_2]$ at every solution point. The baseline algorithm starts with a signal decomposition step where instantaneous phase estimates (IP), instantaneous frequency estimates (IF), and instantaneous amplitude estimates (IA) are extracted per sensor. Unsupervised learning is applied to the Intrinsic Mode Function space spanned by (IP,IF,IA) to generate clusters. The parameter space for K-Means [58] is varied between $k \in [30, 90]$ and for DBSCAN [59] the range for minimum clubbing distance "*eps*" $\in [0.01, 0.04]$ and minimum number of samples $\in [6, 23]$. For every cluster, we take q candidate points and encode them as $\{0, 1\}$ and compute the average objective value $\{O_1, O_2\}$ per cluster. Figure 4.5 shows the scatter plot between forward and backward translation errors for the baseline algorithms and our approach. The dual objective values from K-Means and DBSCAN converge to an error region of $(O_1, O_2 \in [0.3 - 0.5])$ while the best solution yielded by evolutionary computing has $(O_1, O_2 \in [0.20 - 0.25])$.

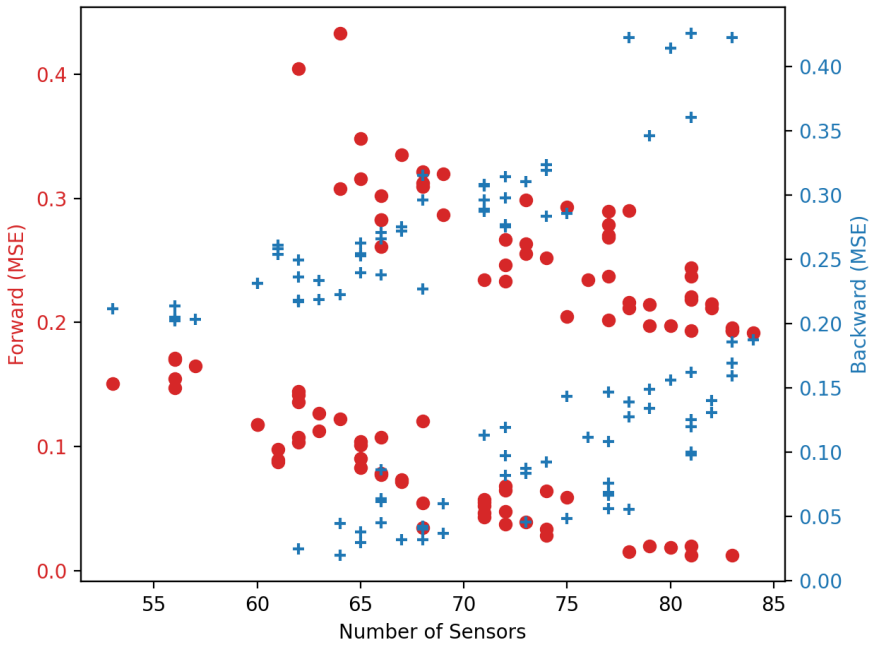


FIGURE 4.4 – Dual Objective Optimization

In contrast to unsupervised learning, our algorithm generates a Pareto front as per Figure Fig 4.5 where no objective can decrease without increasing another. From Figure 4.4, we observe that using $[60, 85]$ sensors, the forward translation MSE $\in [0.05, 0.1]$ with a backward margin between $[0.2, 0.25]$. Out of 138 sensors, the system achieves a fair trade-off between forward and backward translation MSE [0.16, 0.19] with approximately $45-50\%$ less sensors. Increasing number of observable sensors, decreases O_1 to MSE [0.05] using 80-85 sensors while the backward error O_2 increases to 0.48 since greater difficult sensor patterns have to be approximated now.

Energy Consumption Power utilization patterns are usually continuous but "non-differentiable" in nature arising due to sharp peaks and crests for fast response times. The ability to learn the patterns from inter zonal power consumption L_d = (min= 0.03, max=0.27) is more effective rather using intra-zonal data sources L_s = (min= 0.2, max=0.71) for predicting light, AC, and appliance channels.

Ambient sensing Luminosity (lux) levels has the best L_d approximation (min=0.03, max=0.14) although for Floor 2 Zone 2, we observe spatial grouping better for $L_s = 0.06 > L_d = 0.09$. For, indoor temperature prediction L_d (min=0.04, max=0.14) is lower than L_s (min=0.19, max=0.71) for all floors except in floor 2 with zone 1 $L_s = 0.18, L_d = 0.25$ and zone 2 $L_s = 0.11, L_d = 0.24$. Relative humidity is most difficult to approximate, L_s (min=0.37, max=0.66) and L_d = (min=0.14, max=0.63) across all zones. Majorly, we see that domain wise grouping performs better on average which affirms the intuitiveness of being guessed *easily* by similar peers. The error matrix as a heat-map for 6 types of sensors is shown in Figure 4.2.

4.2.2 Policy Discovery

In our data setting, a policy is given by 1D vector M^S made up of $n^S = 138$ integers, where $M^V = mod(M^S, 2)$ encodes which subset of real sensors can be switched off. Based on policy evidence, the first element to optimize is selection of $\{X_g, Y_g\}$ under the constraint of fixed group affinity (M^A). A pool of 50 candidate policies are randomly generated as input to Algorithm 2 that optimises $M^V|M^A$ with respect to objectives $[O_B, O_F]$. The execution is distributed over n^G computing nodes, one for each group (g) with a data affinity defined as per mask M^A_g. The system converges to a Pareto front as per Figure 4.5 where we observe that using $[60, 85]$ sensors, the forward translation error $O_1 \in [0.05, 0.1]$ with a backward margin between $O_2 \in [0.2, 0.25]$. Increasing number of observable sensors, decreases O_1 to MSE [0.05] using 80-85 sensors while the backward error O_2 increases to 0.48 since more number of difficult sensor patterns have to be approximated now. Figure 4.5 shows the scatter plot between forward and backward translation errors from our approach versus an unsupervised setting. The dual objective values from K-Means and DBSCAN converge to an error region of ($O_1, O_2 \in [0.3, 0.5]$) while the best solution yielded by evolutionary computing has ($O_1, O_2 \in [0.20, 0.25]$). The parameter space for K-Means [58] is varied between $k \in [30, 90]$ and for DBSCAN [59] the range for minimum clubbing distance "*eps*" $\in [0.01, 0.04]$ and minimum number of samples $\in [6, 23]$. This results presented in this section are studied with respect to grouping sensors by spatial distribution and sensor type only.

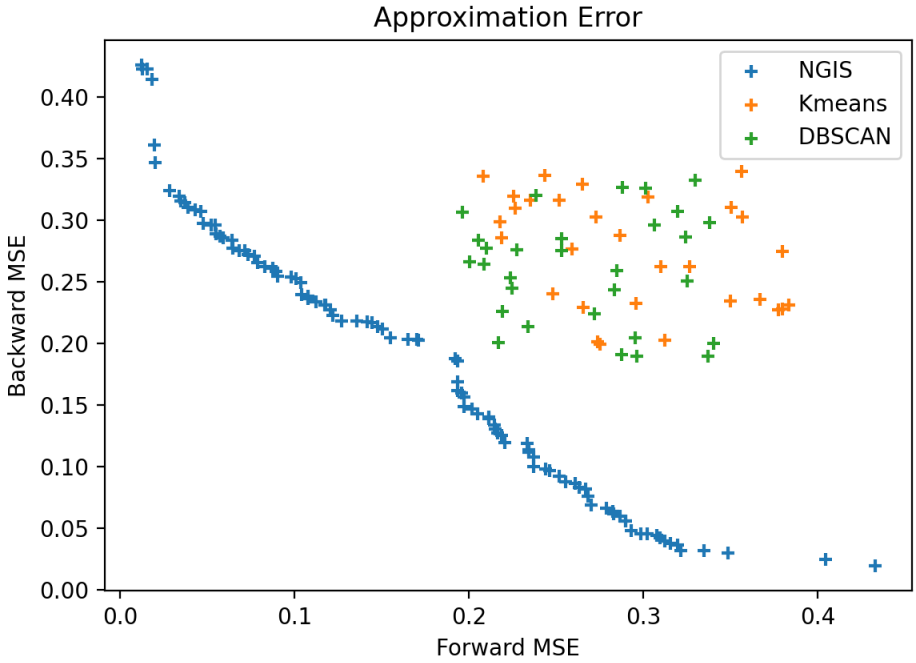
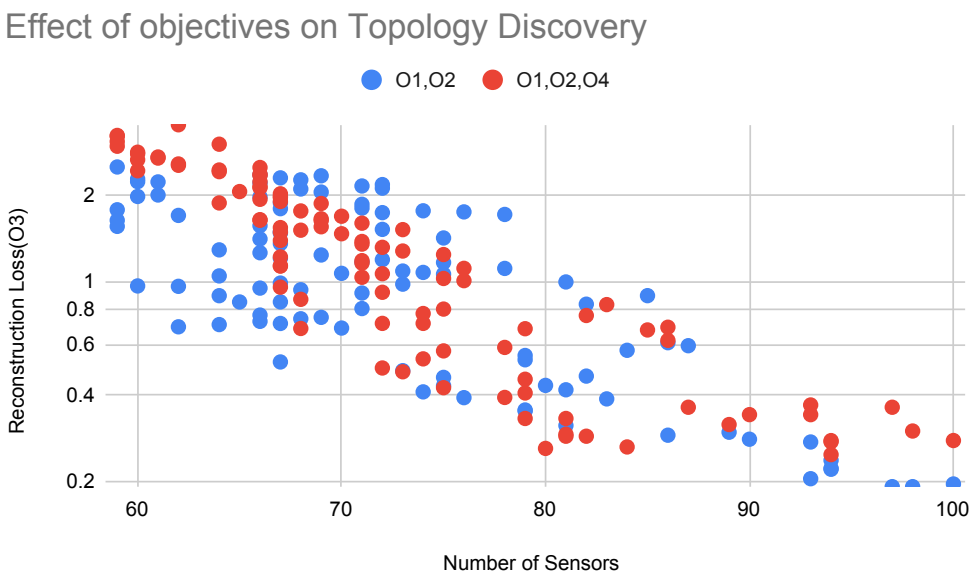


FIGURE 4.5 – Trade off between Forward O_1 versus Backward O_2 Translation

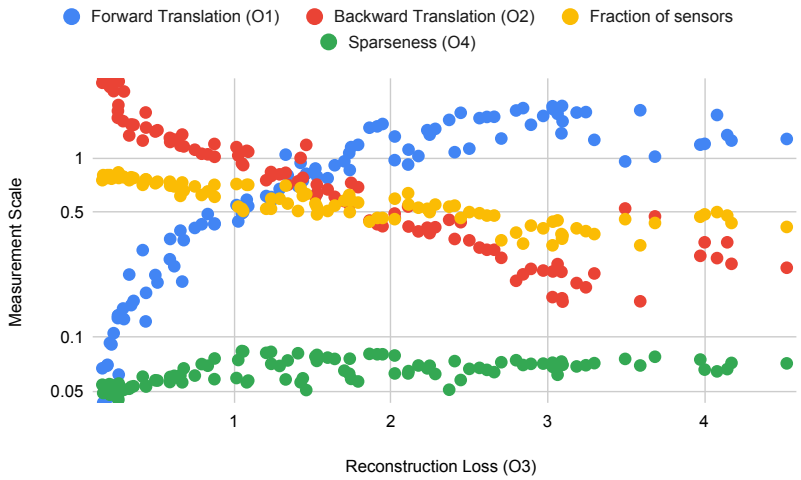
4.2.3 Policy Optimiser

Once a set of Pareto Optimal Sensor configurations are generated using the data till $t_1 << T_{max}$, the system tracks their performance over time assuming such a set of solutions are deployed. Offline data availability from $t_1 \rightarrow T$ of real sensor values make it possible to periodically retrain the Virtual Sensor Field and continually optimize towards better configurations. The data set per sensor be split into B batches, where a batch i for a sensor k placed at zone z is denoted by $D^i_{k,z} \equiv [\frac{T_{max}}{B} : \frac{T_{max}}{B} + B]$. On receiving $D^i_{k,z}$ at i^{th} time-step, the learning system evaluates 4 objectives denoted by Equations 4.5 - 4.8 to generate better learning topologies.

Figure 4.7 shows the effect of lifelong learning on temperature and power consumption channels. Ambient sensing is more prone react to unpredictable environmental changes in contrast to controlled power usage. The temperature at the top most floor of the building is susceptible to the maximum environmental fluctuations, which is expressed by the diverging nature of $L_{RC} > 10\%$ in Figure 4.7a and is least likely to benefit from additional data. In contrast as per Figure 4.7b, 75 days or 2.5 months of data collection suffices to keep the approximation

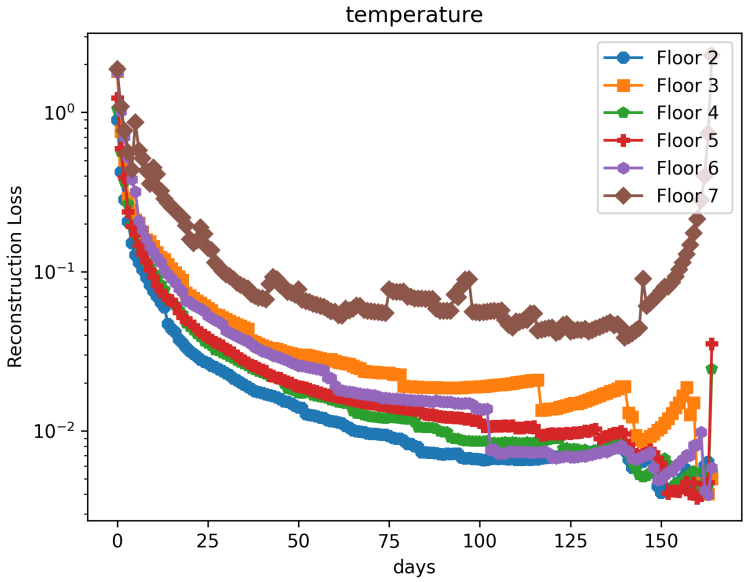


(a) Discovering Policy Space

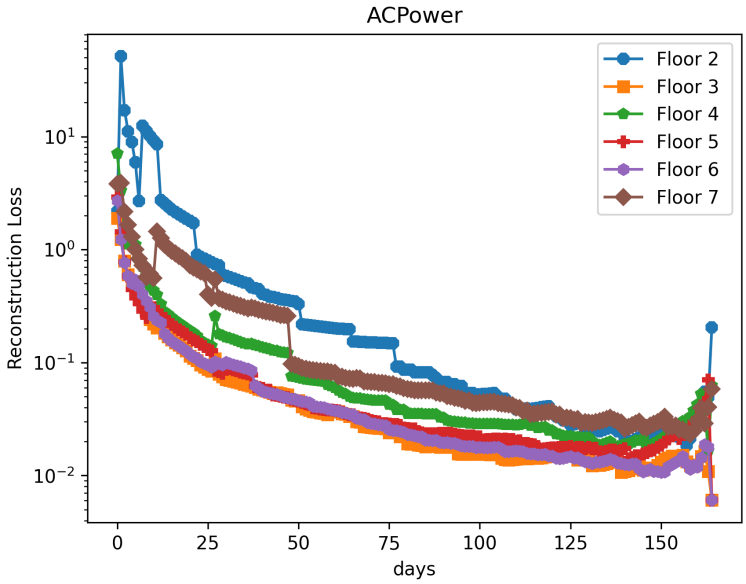


(b) Evaluating Objectives

FIGURE 4.6 – Characteristics of the Policy Space



(a) Ambience Sensing

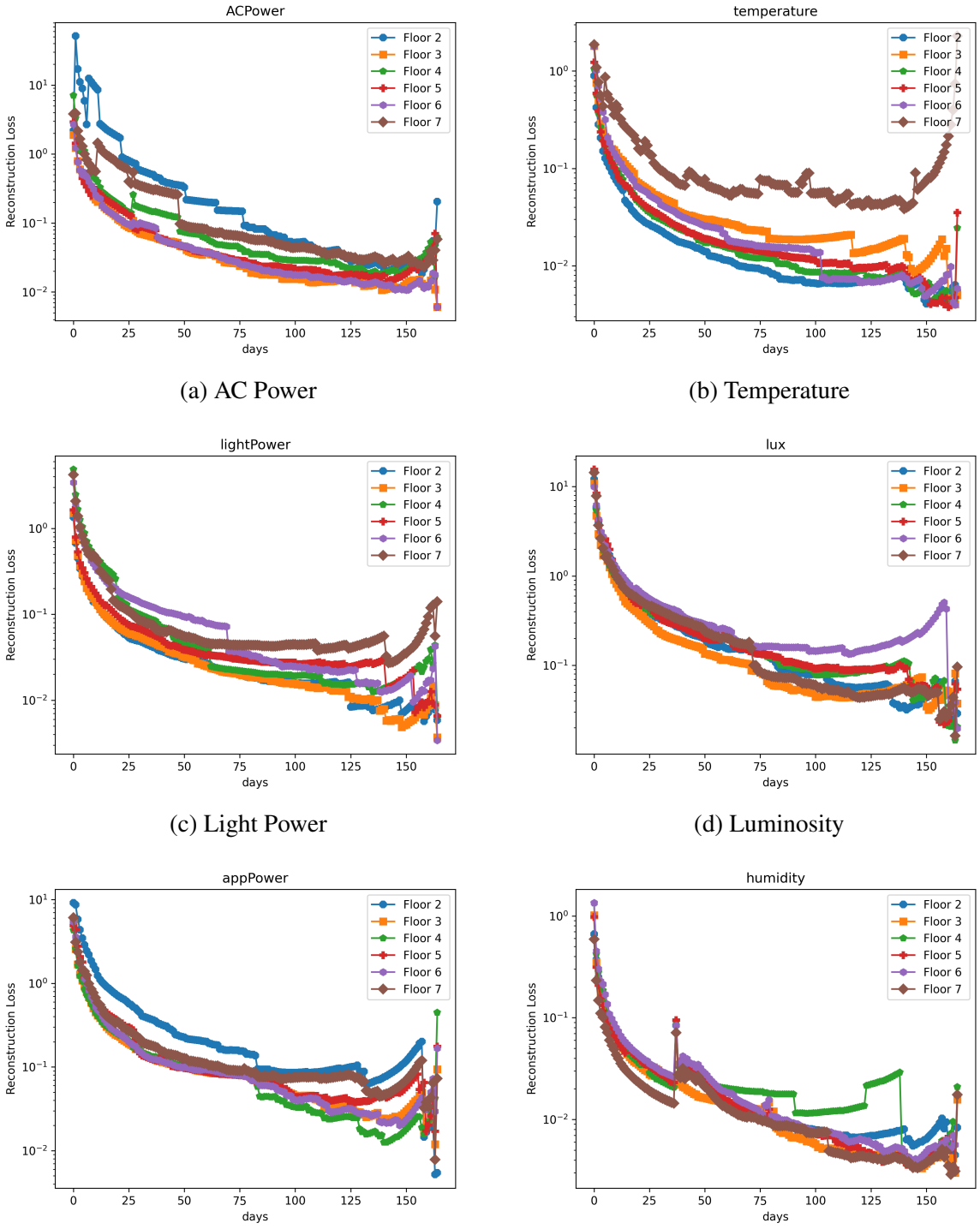


(b) Predicting Power

FIGURE 4.7 – Policy Effect on two Correlated Channels : temperature and air-conditioning power.

error below 10% for all the 6 floors ; the probable reason being controlled power consumption by an AC. Next, the approximation ability of light power and lux is close to 98 % accurate for Floor 2 in comparison 90% correct for the top two floors (6,7). Regarding O_4, from Figure 4.7b, we see that it suffices for a network topology with 5-10 % of total possible connections. In a

continual setting, the system updates the hypothesis space and auto re-calibrates to stabler sensor placement configurations with availability of more data. Table 4.3 gives the optimal sensor placement distribution that uses 45 sensors instead of 138, bringing in a 67 % sensor reduction.



(a) AC Power

(b) Temperature

(c) Light Power

(d) Luminosity

(e) Appliance Power

(f) Humidity

FIGURE 4.8 – Average Reconstruction Loss per floor for 3 ambient sensors (temperature, humidity, luminosity), and 3 power consumption channels (AC, Application, Lights)

Type	#	Save	Installation Sites	Approximated Locations
Temperature	9	0.61	'Floor4Z4', 'Floor2Z2', 'Floor4Z2', 'Floor3Z1', 'Floor7Z5', 'Floor3Z2', 'Floor5Z1', 'Floor7Z1', 'Floor3Z5'	Floor4Z5', 'Floor6Z4', 'Floor6Z5', 'Floor2Z1', 'Floor6Z1', 'Floor2Z4', 'Floor6Z2', 'Floor4Z1', 'Floor7Z4', 'Floor5Z5', 'Floor5Z4', 'Floor7Z2', 'Floor3Z4', 'Floor5Z2'
Humidity	6	0.74	'Floor4Z4', 'Floor3Z1', 'Floor7Z2', 'Floor7Z1', 'Floor3Z5', 'Floor5Z2'	Floor4Z5', 'Floor2Z2', 'Floor6Z4', 'Floor6Z5', 'Floor2Z1', 'Floor6Z1', 'Floor4Z2', 'Floor2Z4', 'Floor6Z2', 'Floor4Z1', 'Floor7Z4', 'Floor7Z5', 'Floor5Z5', 'Floor3Z2', 'Floor5Z4', 'Floor5Z1', 'Floor3Z4'
Luminosity	8	0.65	'Floor2Z2', 'Floor2Z1', 'Floor6Z1', 'Floor4Z2', 'Floor3Z1', 'Floor7Z5', 'Floor3Z2', 'Floor7Z2'	'Floor4Z5', 'Floor4Z4', 'Floor6Z4', 'Floor6Z5', 'Floor2Z4', 'Floor6Z2', 'Floor4Z1', 'Floor7Z4', 'Floor5Z5', 'Floor5Z4', 'Floor5Z1', 'Floor7Z1', 'Floor3Z5', 'Floor3Z4', 'Floor5Z2'
lightPower	7	0.7	'Floor2Z2', 'Floor6Z5', 'Floor4Z1', 'Floor3Z1', 'Floor7Z5', 'Floor7Z2', 'Floor3Z4'	'Floor4Z5', 'Floor4Z4', 'Floor6Z4', 'Floor2Z1', 'Floor6Z1', 'Floor4Z2', 'Floor2Z4', 'Floor6Z2', 'Floor7Z4', 'Floor5Z5', 'Floor3Z2', 'Floor5Z4', 'Floor5Z1', 'Floor7Z1', 'Floor3Z5', 'Floor5Z2'
ACPower	10	0.57	'Floor2Z1', 'Floor6Z1', 'Floor7Z4', 'Floor3Z1', 'Floor7Z5', 'Floor3Z2', 'Floor5Z4', 'Floor7Z2', 'Floor5Z1', 'Floor5Z2'	'Floor4Z5', 'Floor4Z4', 'Floor2Z2', 'Floor6Z4', 'Floor6Z5', 'Floor4Z2', 'Floor2Z4', 'Floor6Z2', 'Floor4Z1', 'Floor5Z5', 'Floor7Z1', 'Floor3Z5', 'Floor3Z4'
appPower	5	0.78	Floor4Z4', 'Floor2Z4', 'Floor4Z1', 'Floor5Z4', 'Floor5Z1'	'Floor4Z5', 'Floor2Z2', 'Floor6Z4', 'Floor6Z5', 'Floor2Z1', 'Floor6Z1', 'Floor4Z2', 'Floor6Z2', 'Floor7Z4', 'Floor3Z1', 'Floor7Z5', 'Floor5Z5', 'Floor3Z2', 'Floor7Z2', 'Floor7Z1', 'Floor3Z5', 'Floor3Z4', 'Floor5Z2'

TABLE 4.3 – Best configuration of the Virtual Sensor Field with least forward error for the building studied.

4.3 Chapter Summary

The aim of the study is to limit the deployment of intrusive connected objects by integrating virtual sensors on a multi storey building. This is achieved by computing which physical sensors can be replaced by virtual sensors by exploiting physical data and learned data. By comparing virtual organizations to existing deployment, the system evaluates the error generated by approximating the operational goals and provides a minimalist set of possible solutions. Experiments are carried out using data from a building in Thailand where each one of 23 working zones is equipped with 6 different types of sensors. The analysis is carried out following a relevant approach consisting in defining a baseline for the comparison, by modeling the operational objectives (number of sensors deployed, hardware cost and energy consumption). The generated configurations are re-calibrated based on data available over the considered time interval, thereby presenting an offline pre-planning tool for optimal sensor monitoring.

Chapitre 5

Co-Learning at Edge

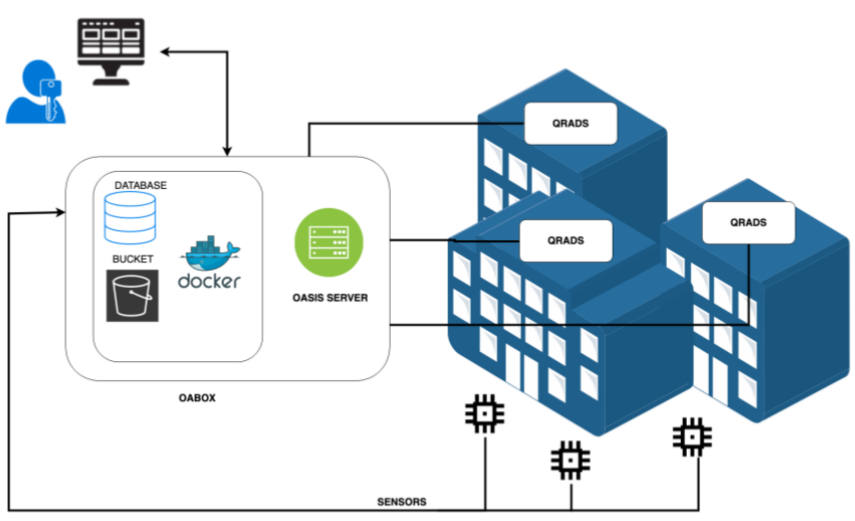


FIGURE 5.1 – Edge Computing Infrastructure

Qarnot is a computing service providing company that ecologically uses heat produced during computation to provide indoor thermal comfort or hot water facility. The company provides on-demand availability of edge resources via digital heaters (QRads) placed inside a building. Multiple smart heaters (Qrads) are logically connected to a IoT controller (Qbox) within a building or locality. The thermal profiles of such smart heaters are of particular interest since one can not run a High Performance Compute job when the indoor temperature is high for example. We model each heater act as an autonomous learner which is responsible for training and drawing regression or inference based on sensor data generated at site in real time. Each device

is equipped with 5 ambient sensors, and uploads luminosity, humidity, temperature, sound and co_2 sensors readings to a local bucket.

5.1 Energy - Ambience at Edge

The need for automatically maintaining comfort set-points of indoor temperature leads to energy demand for Heating, Ventilation and Air Conditioning (HVAC) systems in order to resist the thermal stress and provide ambient comfort [60]. Such systems rely on estimates of room temperature which is prone to activity related fluctuations. Although estimating ambient temperature with phone sensors has been tried [61], the results show high correlation with the phone internal thermal state rather than the environment. Instead, building characteristics such as spatial-orientation, wall-thermal conductivity, specific heat capacity show better predictive capabilities [62]. When thermal characteristics of the building are not available, one can rely on the intrinsic properties of the heating systems, such as the heater's power consumption and heat-sink temperature. We hypothesize that ambient temperature is an interaction between the smart heater's electrical power consumption and the thermal radiation of its heat-sink. The motivation behind this use-case stems from predicting the air temperature of environments from a smart heater's operational data-points. We follow a sliding window protocol length of 24 hours to predict the next hour and hence generate a forecast for the desired time-interval and takes the dual input of power consumption and heat-sink temperature.

5.1.1 Predictive Interaction Model

To form an explainable model, we restrict our knowledge base to sine and cosine components and optimize short term deployment temperature prediction using Prophet [63] by Facebook. Equation 5.1 decomposes a time series into non-periodic changes (trend), season recurrences (seasonality) and effect of holidays

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \tag{5.1}$$

where the observed signal $y(t)$ follows the additive property in combining $g(t)$, $s(t)$, $h(t)$ representing the trend, seasonality, holiday behaviour respectively, and ε_t represents a Gaussian white noise with 0 mean. We observe that heating does not keep on increasing ambient temperature, rather a heated space starts cooling when it is not supplied with thermal energy. The set of points where the trend function has local minima or maxima or is discontinuous, are denoted as change-points ($S = \{s_1, s_2 \dots s_n\}$) and each change-point s_i comes with a rate of adjustment (d_i). We define a binary vector $a(t) = \{a_1(t), a_2(t), \dots a_n(t)\}$ with $a_i(t) = 1$ if $t \geq s_i$ or 0 otherwise. The vector $\delta = \{d_1, d_2, \dots d_n\}$ is the collection of adjustment rates or equivalently $\delta \in R^S$. $(k + a(t)^\top \delta)t$ represents the total growth starting from $t = 0$ to t where k is the initial growth rate. We want $g(t)$ to be continuous and hence the correction factor is given by $(m + a(t)^\top \gamma)$ where $\gamma = \{\gamma_1, \gamma_2 \dots \gamma_n\}$, $\gamma_j = -s_j \delta_j$ and m is a real-numbered offset. Equation 5.2 shows the linear growth trend used for our methodology.

$$g(t) = (k + a(t)^\top \delta)t + (m + a(t)^\top \gamma) \tag{5.2}$$

Seasonal components $s(t)$ like weekly or yearly repetitive patterns are modelled with Fourier series given by Equation 5.3 where the argument of n^{th} cosine and sinusoidal functions are given by $f_n \times t$ where $f_n = \frac{2\pi n}{P}$, P = 365.25 for yearly and 7 for weekly occurrences. The variables a_n and b_n reflect the magnitude of the n^{th} harmonic and N represents the degree of approximation.

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right) \tag{5.3}$$

Weekends or holidays are special days where one can expect either a low or high degree of activity depending on an office space having no employees on a Sunday or employees staying back at home respectively. An indicator function is used to label time t as 1 if t falls on a holiday i or 0 otherwise. Let D_i represent the day of the years corresponding to a holiday i. For example Christmas happens on 25^{th} December every year, thus yielding $D = \{\dots, 25/12/2019, 25/12/2020, 25/12/2021, \dots\}$. At time t, for L holidays, we construct a binary matrix $Z(t) = [1(t \in D_1), \dots, 1(t \in D_L)]$. The effect of holidays are assumed inde-

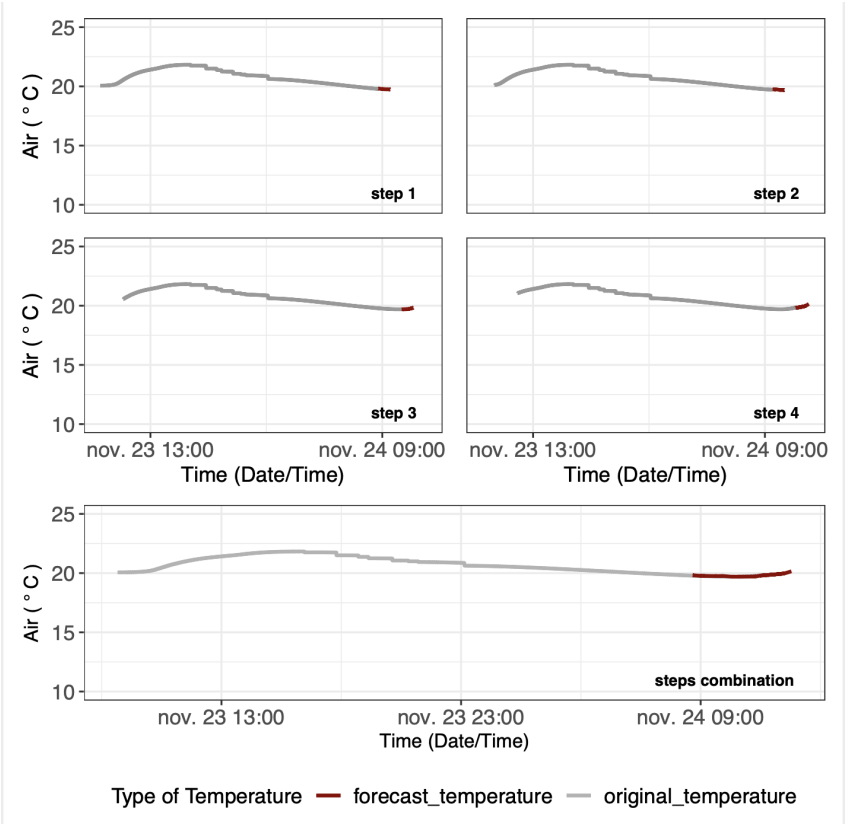


FIGURE 5.2 – Sliding window procedure for forecasting short-term air temperatures.

pendent while modeling temperature on such days and magnitude of change κ_i is drawn from $\kappa \sim Normal(0, \upsilon^2)$. Formally the effect of the holidays is given by Equation 5.4.

$$h(t) = Z(t)\kappa \tag{5.4}$$

The forecasting model is trained using Stan's implementation of the Limited-Memory Broyden-Fletcher-Goldfarb-Shanno Algorithm (L-BFGS)[64] to find a maximum *a posteriori* estimate of ambient temperature. The model accuracy is evaluated by calculating the RMSE (Root Mean Square Error) = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$ and MAPE (Mean Absolute Percentage Error) = $\frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right|$ considering n predictions where Y_i is the vector of observed values of the variable being predicted, and $\hat{Y}_i$ being the predicted values. The RMSE metric is useful in highlighting major deviations from the true sensor value and a low RMSE score ensures less signal to noise ratio for temperature prediction, while MAPE provides intuitive performance measure by averaging the absolute deviation over the set of data points.

TABLE 5.1 – Hyperparameters summary for the grid of experiments

Parameter	Set of Values
Changepoint Prior Scale	{0.01, 0.1, 1}
Seasonality Prior Scale	{0.01, 0.1, 1, 10}
Fourier Order	{1, 3, 5, 10}
Mode	{'additive', 'multiplicative'}
Size of the data	2 days

Figure 5.2 illustrates an example of the step-wise temperature forecasting with an one hour sliding window protocol. The first forecast, at *step 1*, takes the historical data of 24 hours from 23rd November 09 :00 am to 24th November 09 :00 am as input to predict the values for the 25th hour. For the next steps, we shift the window of the training data by one hour. The combined forecast of these 4 steps is shown at the bottom facet of Figure 5.2 at *steps combination*.

To achieve a model with the least generalization error, we investigate the following hyperparameters from the Prophet model : *a) Changepoint prior scale* that controls the trend functions modeling the non-periodic changes in the data, i.e $g(t)$ in Equation 5.2 ; *b) Seasonality prior scale* which models the periodic changes in the data, i.e $s(t)$ in Equation 5.3 ; *c) Fourier order* that indicates resolution of frequency decomposition and works in tandem with the seasonality prior scale ; *d) Mode* which indicates the effect of the regressors in the model.

5.1.2 Performance Evaluation

The Qarnot heater logs with 510567 samples, comprising of *time-stamp*, *ambient air temperature*, *heat-sink temperature* and *power consumed* are analyzed. Data pre-processing step involves rejection of data points with abnormal values of temperature and power. Based on product data-sheet defined operating intervals of [0, 85]°C and [0, 1000] Watts 11 % of data is omitted. Check for continuity resulted in two largest continuous periods found contain 90721 and 253320 samples, representing about 18% and 50% of the data respectively.

Figure 5.3 summarizes the hyperparameters performance on forecasting with a variation of the changepoint prior scale from the set {0.01, 0.1, 1} in the x axis, with each subplot representing the results of the seasonality prior scale from the set {0.01, 0.1, 1, 10}. The geometrical elements (dots, triangles, squares, and crosses) represent the mean RMSE of a certain Fourier

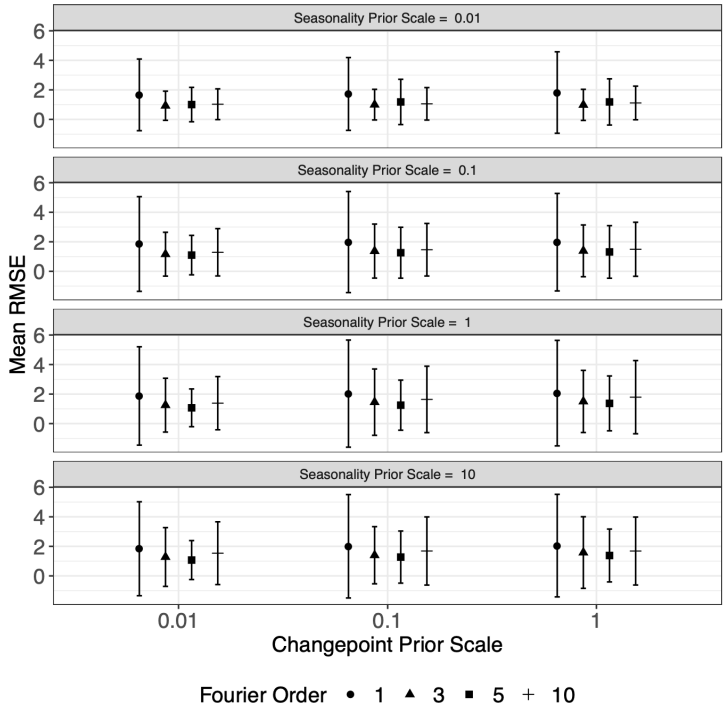


FIGURE 5.3 – Hyper-parameters Performance Evaluation with Mean RMSE and Standard Deviation

order from the set {1, 3, 5, 10}, and the vertical lines represent confidence intervals as standard deviation of RMSE values on the y axis. $2 \times 3 \times 4 \times 4 = 96$ configurations as shown in Table 5.1 are evaluated with a cross validated model. For every training task a forecast model needs to be trained on sequential partitions of data, labeled as train and test. We select the model with the least generalization error on the cross-validation test, according to the RMSE and MAPE metrics. The effect of altering *mode* from {'additive', 'multiplicative'} showed no effect on both RMSE mean or standard deviation. Hence, we only discuss the results for the additive mode hereafter, using $\frac{96}{2} = 48$ hyperparameter combinations.

Let the configuration triplet be ordered as change-point, seasonality and Fourier order. Some good performing configurations found are (1, 0.01, 3) with RMSE of $0.98°C \pm 1.05°C$, or (0.1, 0.01, 3) with a RMSE margin of $1.00°C \pm 1.04°C$. The utility of the cross validation is evident by observing bad configurations such as (1, 10, 1) with RMSE of $2.05°C \pm 3.47°C$, 45 % performing less than the best found configuration or (1, 1, 1) with error of $2.06°C \pm 3.57°C$, a 1 % increase in standard deviation error compared to the former.

In order to analyze the effect of each hyper-parameter, the results presented in Figure 5.3

Changepoint Values	Mean RMSE	Standard Deviation
0.01	1.33	2.02
0.10	1.49	2.25
1.00	1.54	2.30
Seasonality Values	**Mean RMSE**	**Standard Deviation**
0.01	1.22	1.65
0.10	1.47	2.19
1.00	1.56	2.43
10.00	1.57	2.41
Fourier Order	**Mean RMSE**	**Standard Deviation**
1	1.91	3.16
3	1.27	1.76
5	1.20	1.56
10	1.43	1.86

TABLE 5.2 – Effect of Hyper-parameters Measured by RMSE and Standard Deviation

are re-grouped in Table 5.2. The change-point prior scale variation captures the non-periodic abrupt changes in the time series. The longer the interval of consideration, more likely are the chances of abnormal patterns. Thus, we lower the scope of error by considering only a day ahead of training while forecasting for the next day. From historical data, we observe that the major of pre-seen ambient temperature pattern stays in an interval larger than two days and hence our methodology is reactive towards capturing a new pattern and adjusting the forecast accordingly. Nevertheless, it is possible to see that as far as we increase the value of this hyper-parameter the errors increase as well, such as from $1.33°C \pm 2.02°C$ to $1.54°C \pm 2.30°C$. Low values of seasonality prior scale exhibit lesser generalization error. Higher values reflect greater temperature fluctuations thereby increasing the error margin, such as from $1.22°C \pm 1.65°C$ to $1.57°C \pm 2.41°C$. Notably as per our approach, alternate day training-forecasting reduces the possibility to register a high number of periodic changes within a span of 48 hours.

The Fourier order value of 1 resulted in average RMSE of $1.91°C \pm 3.16°C$ where as setting the value to 3 leads to $1.27°C \pm 1.76°C$. The values of 5 and 10 yields mean RMSE of $1.20°C \pm 1.56°C$ and $1.43°C \pm 1.86°C$ respectively. The optimal value of 3 has been stated by the authors of Prophet [63] to capture weekly periodic changes and matches the expectation of our office space setting, where one can expect repetition of similar activities every 7 days. Finally, comparing the results of the groups of Table 5.2, we can see that the Fourier order is the hyperparameter which affects more the models RMSE accuracy varying from the worst to the

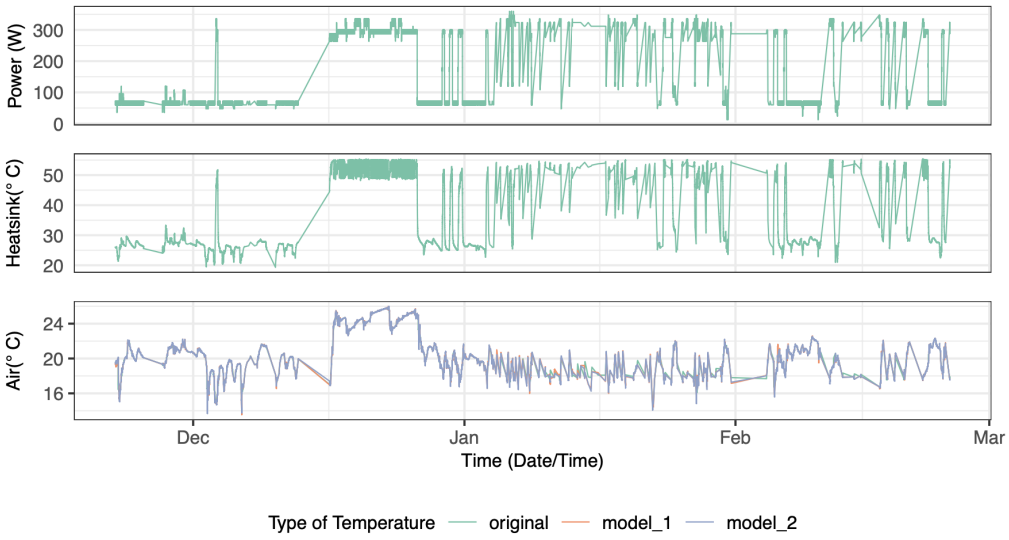


FIGURE 5.4 – Comparison of temperature forecasts for the two models. The first and second facets represent the power and heat-sink inputs to the forecasting models. The results of such models are illustrated in the third facet in addition to the original air temperature data.

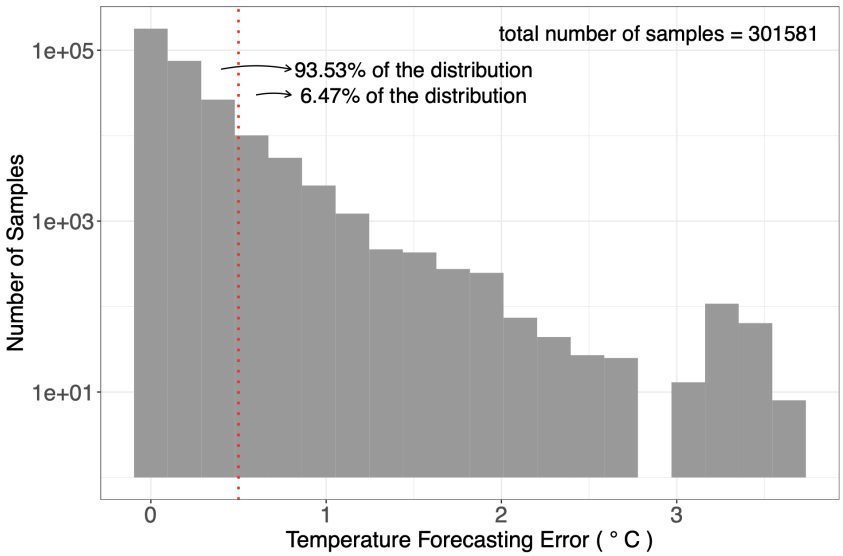


FIGURE 5.5 – Histogram distribution (log scale on y axis) of the absolute difference between the original temperature data and the temperature forecasts performed by model 2. Samples at the left of the vertical dotted line account for 93.53% of the total number of samples (forecasts), and the samples at the right of the same line account for the remaining 6.47%.

best results in $0.71°C$, followed by the seasonality prior scale with $0.35°C$ and the changepoint prior scale with $0.21°C$. The third facet of Figure 5.4 shows the comparative forecast of two models : *power* versus *power and heat-sink* combination in forecasting ambient air temperature for 4 months (x axis) and the pre-processed device-intrinsic values of power (Watts), heat-sink ($°C$) and ambient air temperature ($°C$) on the y axis. For both, power and heat-sink, there is only the original values and no forecasts. The optimal uni-channel *model 1* recorded a MAPE of $2.66\% \pm 2.52\%$ and RMSE of $0.92°C \pm 1.00°C$ versus *model 2*'s MAPE of $2.75\% \pm 2.55\%$ and RMSE of $0.92°C \pm 1.03°C$. In addition, we analyse the distribution of the absolute error $|Y_t - \hat{Y}_t|$ between the forecast $\hat{Y}_t$ and the original value Y_t as shown in Figure 5.5 for *model 2*. We see that 93.53% of the net predictions are lower or equal to $0.5°C$ with maximum error difference at $3.64°C$. This is most likely attributed to unpredictable environmental usages like opening of a door or window.

5.2 Offline Federated Personalization

Let us suppose there are k rooms monitored by sensors where the system collaboratively trains localized deep learning models at edge. A typical assumption is that the participants are honest whereas the server is honest-but-curious, therefore no leakage of information from any participants to the server is allowed. The investigation is to reveal if a local model has incentives to take part in a federated round of learning with a group of identical peers.

Split learning was first introduced by Guptar and Raskar [65]. As described in [66], in the simplest of configurations, each client trains a partial deep network up to a specific layer known as the cut layer. The outputs at the cut layer are sent to another client which completes the rest of the training. Once passed through the chain, at the completion of the round the gradients at the cut layer are sent back to a central server which completes another round of back propagation. This forms the base for Federated Personalization [67] where a deep learning network is divided into two logical parts namely for federated training and non-shared personal customization. Typically a federated setting involves a server who listens and co-ordinates the upload and download of model information with clients at edge. For the initial round, the global node creates the initial weights for learning, pushes the weights to the clients, and as the rounds

progress, it receives, aggregates and stores the updated parameters.

5.2.1 Scheduled Shared Storage Learning

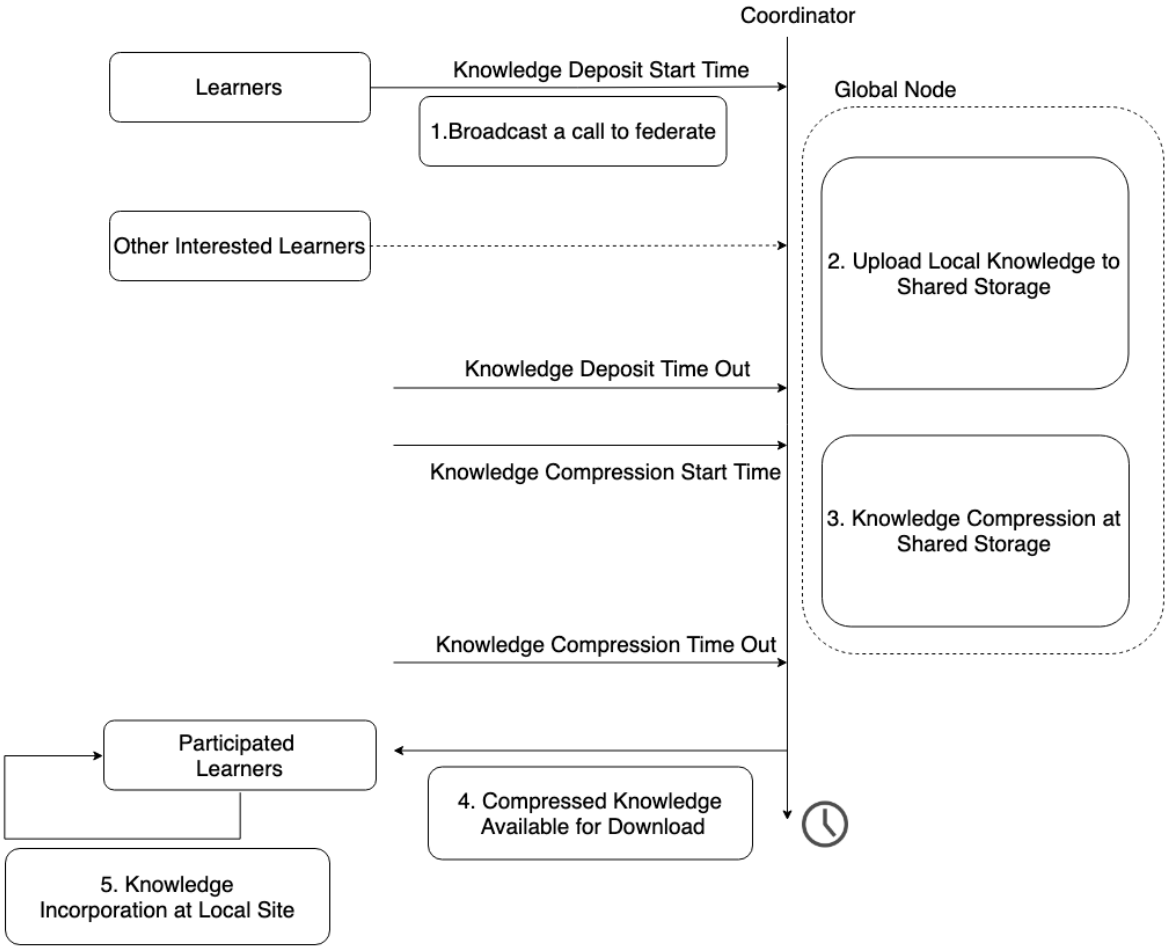


FIGURE 5.6 – Chronologically executed steps for synchronous federated personalization.

The dependency of an external mediator is relaxed through a synchronized learning protocol. The role of the coordinator is masked through a trigger function that operates on an edge hosted storage bucket from Qarnot. Alternately one use any S3 storage provisioned by all major cloud players like Amazon Web Services, Google Cloud, Azure etc.

We give autonomy to a microprocessor at edge to either take part or ignore a federation round by accepting or rejecting a learning agreement. For a federated learning scheme, the computational complexity is represented by (E,R) where E denotes local epochs or the frequency of data point utilization during a training update and y indicates the total rounds of

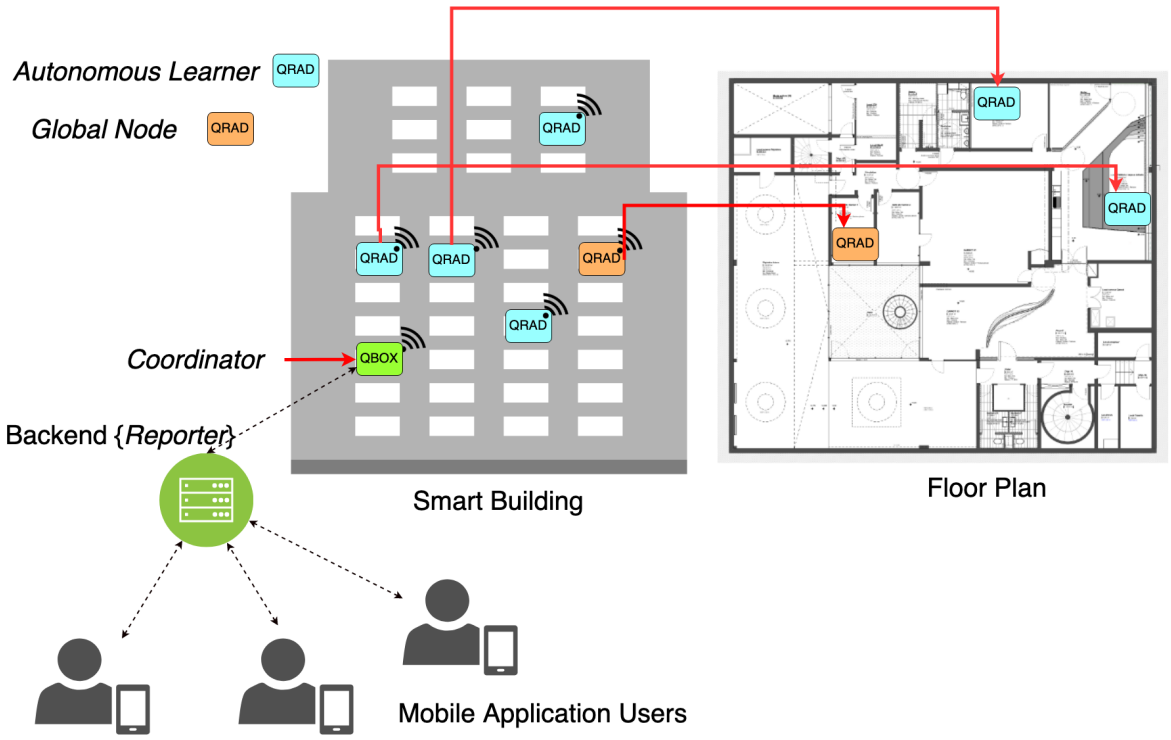


FIGURE 5.7 – Framework Components running on the Qarnot Infrastructure

communication. For example, a model restricted to train in isolation for 20 epochs is given by (20E, 0R). To initiate a learning round, at-least one learner amongst a set of K clients publishes a federation contract. Our synchronized federated contract comes with two time intervals :

1. Interested learners announce their willingness to federate by uploading the transferable layer during the Knowledge Deposition phase.
2. Knowledge Compression Interval is the time when a server-less computing task performs FedAvg [40] to compress multiple transfer layers into a single entity, to be downloaded by clients post timeout.

5.2.2 Performance of Federated Forecasting

The data generating sources (Qrad) as shown in Fig. 5.7 is equipped to measure temperature (in Celcius), relative humidity, noise levels (in decibels), power (in Watts), luminous intensity (in lux) and indoor air quality (in ppm). The ambient sensor data-set consists of 1.4 Million Qrad sensor data readings, sampled at every 5 minutes leading to 288 samples a day, totalling

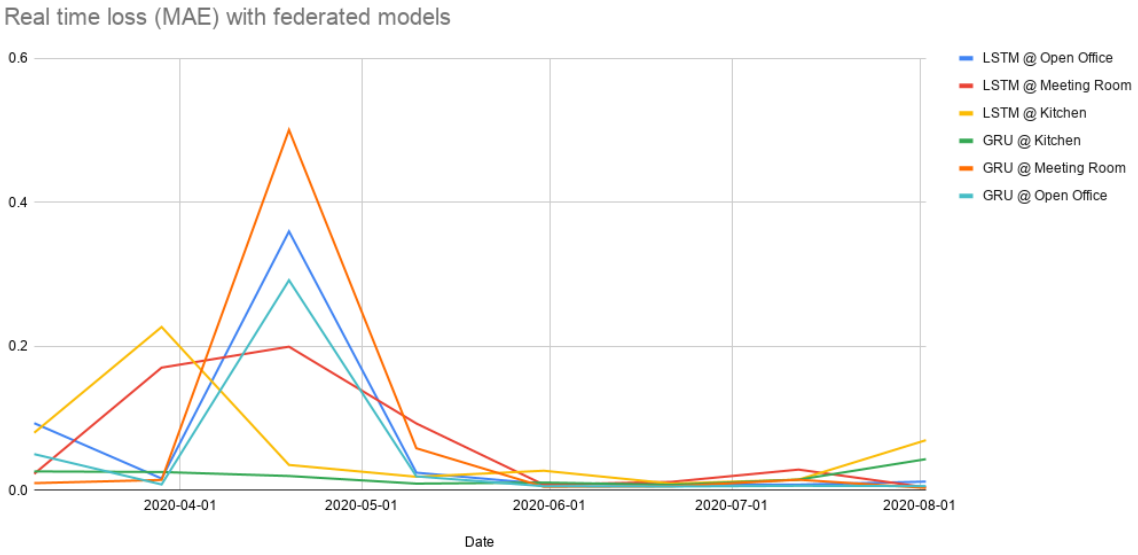


FIGURE 5.8 – Mean Average Error of 6 auto-updating models distributed over 3 spaces with 2 models per room over 8 months.

to 208 k samples in 8 months spread across 3 rooms namely kitchen, meeting room and open office.

Firstly, we observe the impact of training on sequence to sequence forecast of sensor values. For the forecasting task, we implement a deep learning model that has a look back of 36 timestamps or 3 hours, feature size of 6, 1 hidden layer, with a fully connected last layer of $12 \times 6 = 72$ outputs or 1 hour prediction. One variant of the model has 50 Long Short Term Memory (LSTM) units with 11200 hyper-parameters (40-45 kB). The other variant has 32 Gated Rectified Units (GRU) with 4000 hyper-parameters ($<$10 kB). Adam optimizer and mean absolute loss functions are used to train and evaluate both of the deep-learning models. Table 5.3 illustrate the generalization loss for a federation policy of fortnightly knowledge exchange. In Fig. 5.8 we observe loss peaks at March-April which coincides with the distinct spatio-temporal pattern shift of reduced office activity during the lockdown in France. We split 8 months of data to perform a 4 fold cross-validation, with average residual error between 7-12 % higher than retrained forecasting models.

We validate the impact of the federation by comparing against a setting without knowledge transfer. Hence the baseline model follows zero round (0R) strategy and is illustrated with 2 federation configurations out of infinite combinations of (x epochs, y rounds). In Fig. 5.9 we plot

Update Date	LSTM @ Office	GRU @ Office	LSTM @ Kitchen	GRU @ Kitchen
2020-03-08	0.093	0.050	0.080	0.026
2020-03-29	0.016	0.008	0.227	0.025
2020-04-19	0.359	0.292	0.035	0.019
2020-05-10	0.024	0.019	0.018	0.009
2020-05-31	0.008	0.005	0.027	0.010
2020-06-21	0.008	0.005	0.009	0.007
2020-07-12	0.007	0.006	0.014	0.014
2020-08-02	0.012	0.005	0.069	0.043

TABLE 5.3 – Generalization error of LSTM and GRU models executing in a kitchen and open office area.

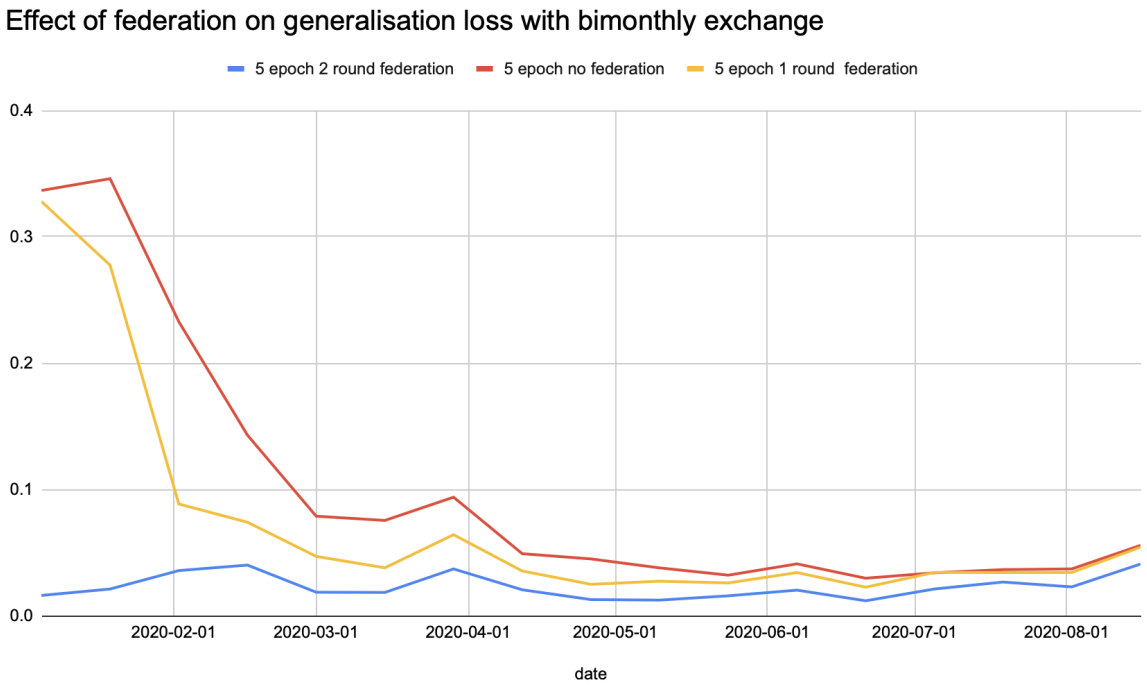


FIGURE 5.9 – Performance gain due to federation

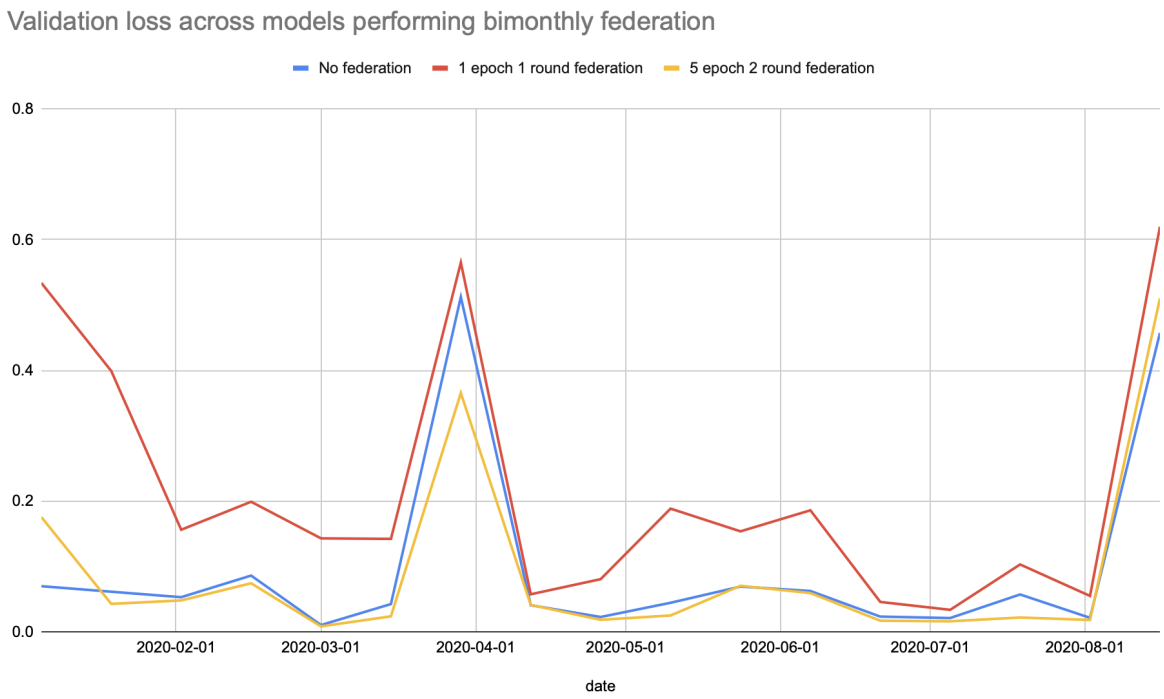


FIGURE 5.10 – Federated model performance against isolated baseline model.

the average generalization error of three collaborating learners after every bimonthly update across 8 months. We observe a (5E,2R) federation configuration experiences lower generalization loss over a non federated (5E,0R) and less federated (5E,1R) solution. It is evident there is no one fit for all but certain combinations can lead to an under-fit model (5E,1R) performing worse than a non-federated local learning strategy (5E,0R) as shown in Fig. 5.10.

5.2.3 Impact of Federation Affinity

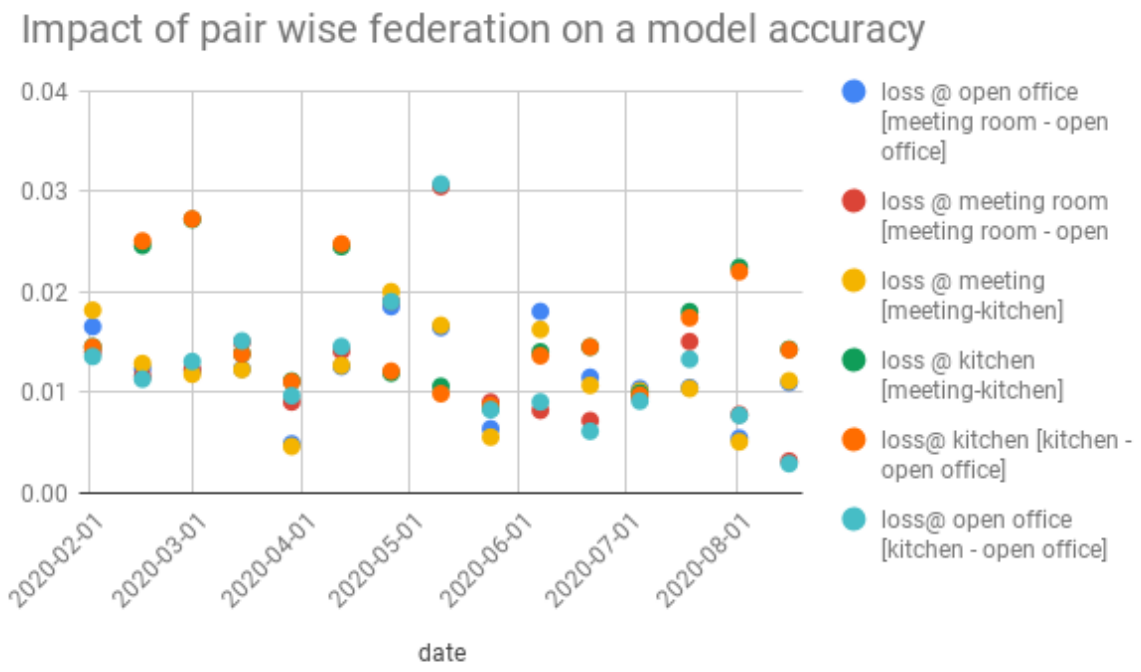


FIGURE 5.11 – Pairwise Federated Loss curves of unit layered LSTM models placed at 3 rooms.

Now we illustrate the impact of selective or pairwise federation on loss values of individual forecasting models placed at an open office, a meeting room and a kitchen as shown in Fig. 5.11. We observe that the meeting room model records 7.6 % lower generalization error post federating with a kitchen (in yellow) than a distant open office (in red). Compared to a non-federated version, the federated open office model learns better from the kitchen (12 % improvement) rather than the meeting room (2.5 % improvement). The intuitive validation lies in people gathering naturally at both the areas during breaks or an informal meeting.

The in-house data sharing policy is encoded through a federation configuration which is simply the number of communication (*comm*) rounds, number of epochs per *comm* round. The small-scale experiments validate that for a smart building, an in-situ model may have an incentive to share their knowledge representation and benefit from a global feedback to improve on a local task. On a storage constraint hardware, such an implementation will still require historical data pile up to train the model over time. As the number of sensors increase, the scalability of a synchronous system becomes questionable. Compounding to distributed behaviour is the role

of pair-wise affinity which forms the basis to investigate in direct peer to peer exchanges for optimizing deep neural network models in-situ.

5.3 Projection Free Online Decentralized Learning

When to train and when to apply the learning forms a major part in the design decision of any deploy-able machine learning solution. Offline setting typically means that the learning procedure distinguishes between at-least a training and testing data partition. Such a setting is really useful when more-or-less the patterns are similar across the two partitions, but it prohibits a model to pick up features (if useful) while testing. We highlight the key bottlenecks of offline training for a loosely coupled distributed system :

— Increasing number of devices exacerbates the problem of **scalability** when dealing with sharing virtualized memory spaces hosted on either the edge or the cloud.

— Time based triggering of a trusted function is susceptible to uncertainties of networking specially while uploading and downloading models. In short, the notion of centrality for a loosely coupled system bottlenecks **robustness**.

— Averaging model weights to customize a local model suffers from the uncertainty due to lack of **theoretical guarantee** for understanding the performance behaviour of learners.

Online learning enables a model to learn constantly upon arrival of new data samples; thereby relaxing the demarcation between train and test data. We want to design a learning policy for a group of online learners each having its own local data. The design choice of zero trusted mediators follows the idea of decentralization and promotes peer to peer exchanges. Let n denote the number of agents (vertices) organized in a graph $G = (V, E)$ such that $n = |V|$ and number of direct one to one connections is $|E|$. Each agent optimizes and shares a structurally identical deep learning network. At time t, a batch of data is revealed exclusively to agent i and a non-convex function $f_i^t : \mathscr{K} \to \mathbb{R}$ is learnt locally. Each agent $i \in V$ updates its own local neural network which yields a prediction $\boldsymbol{x}_i^t \in \mathscr{K}$ where the convex set $\mathscr{K} \subseteq \mathbb{R}^d$. Although each agent i observes only function f_i^t, the collective learning performance per agent i is $F^t(\boldsymbol{x}_i^t)$ where $F^t(\cdot) := \frac{1}{n}\sum_{j=1}^{n} f_j^t(\cdot)$. Intuitively one can understand F^t as the loss accumulated over all predictive models, somewhat analogical to a global model loss in case of Federated Learning.

The objective is to minimize the overall loss via local communication with adjacent agents in G. When the loss/cost functions are convex, a typical measure is the *regret* notion. An online algorithm is $R(T)$-*regret* if for every agent $1 \leq i \leq n$, Equation 5.5 holds true.

$$\frac{1}{T}\left(\sum_{t=1}^{T} F^t(\boldsymbol{x}_i^t) - \min_{\boldsymbol{o}\in\mathscr{K}} \sum_{t=1}^{T} F^t(\boldsymbol{o})\right) \leq R(T) \tag{5.5}$$

Now we describe informally the main ideas of the decentralized algorithm. In the algorithm, at every time t, each agent i executes L steps of the Frank-Wolfe algorithm where every update vector (for iterations $1 \leq \ell \leq L$) is constructed by combining the outputs of linear optimization oracles $\mathscr{O}_{j,\ell}$ and the current vectors of its neighbors $j \in N(i)$. The solution $\boldsymbol{x}_i^t$ for each agent/node $1 \leq i \leq n$ is chosen uniformly random among $\{\boldsymbol{x}_{i,\ell}^t : 1 \leq \ell \leq L\}$. Subsequently, after aggregating the information related to functions f_j^t for $j \in N(i)$, the algorithm subtly computes a vector $\boldsymbol{d}_{i,\ell}^t$ and feedbacks $\langle \boldsymbol{d}_{i,\ell}^t, \cdot \rangle$ as the reward function at time t to the oracle $\mathscr{O}_{i,\ell}$ for $1 \leq \ell \leq L$. The formal description is given in Algorithm 4.

Analysis. In the analysis, denote $\bar{\boldsymbol{x}}_\ell^t := \frac{1}{n}\sum_{j=1}^n \boldsymbol{x}_{j,\ell}^t$, we make use of the following lemma.

Lemma 1 ([68], Lemmas 1 and 2)**.** *Assume that functions f_j^t's are β-smooth, G-Lipschitz that is, $\|\nabla f_j^t\| \leq G$ for every $1 \leq t \leq T$ and every $1 \leq j \leq n$ and the diameter of $\mathscr{K}$ is D. Then, there exists a constant ℓ_0 such that for every $1 \leq \ell \leq L+1$,*

$$\Delta p_\ell := \max_{t=1}^{T} \max_{i=1}^{n} \|\boldsymbol{y}_{i,\ell}^t - \bar{\boldsymbol{x}}_\ell^t\| \leq \frac{C_p}{\ell}$$

$$\Delta d_\ell := \max_{t=1}^{T} \max_{i=1}^{n} \|\boldsymbol{d}_{i,\ell}^t - \frac{1}{n}\sum_{j=1}^{n} \nabla f_j^t(\boldsymbol{y}_{j,\ell}^t)\| \leq \frac{C_d}{\ell}$$

where $C_p = \ell_0\sqrt{n}D$ and $C_d = \sqrt{n}\cdot\max\left\{2\left(\frac{\ell_0\sqrt{n}D}{\ell}+D\right)\beta; |\lambda_2(W)|\ell_0\left(\frac{\beta D}{1-|\lambda_2(W)|}+G\right)\right\}$ where $\lambda_2(W)$ is the second largest eigenvalue of W.

The Frank-Wolfe gap In order to bound the Frank-Wolfe gap for each individual node, we consider the *average* Frank-Wolfe gap, which is defined at every time $1 \leq t \leq T$ and for every

Algorithm 4 Decentralized online algorithm

Input : A convex set $\mathscr{K}$, a time horizon T, a parameter L, online linear optimization oracles $\mathscr{O}_{i,1},\ldots,\mathscr{O}_{i,L}$ for each player $1 \le i \le n$, step sizes $\eta_\ell \in (0,1)$ for all $1 \le \ell \le L$

Initialize linear optimizing oracle $\mathscr{O}_{i,\ell}$ for all $1 \le \ell \le L$
for $t = 1$ to T **do**
  for every agent $1 \le i \le n$ **do**
    Initialize arbitrarily $\boldsymbol{x}_{i,1}^t \in \mathscr{K}$
    for $1 \le \ell \le L$ **do**
      Let $\boldsymbol{v}_{i,\ell}^t$ be the output of oracle $\mathscr{O}_{i,\ell}$ at time step t.
      Send $\boldsymbol{x}_{i,\ell}^t$ to all neighbours $N(i)$
      Once receiving $\boldsymbol{x}_{j,\ell}^t$ from all neighbours $j \in N(i)$, set $\boldsymbol{y}_{i,\ell}^t \leftarrow \sum_j W_{ij}\boldsymbol{x}_{j,\ell}^t$.
      Compute $\boldsymbol{x}_{i,\ell+1}^t \leftarrow (1-\eta_\ell)\boldsymbol{y}_{i,\ell}^t + \eta_\ell \boldsymbol{v}_{i,\ell}^t$.
    end for
    Choose $\boldsymbol{x}_i^t \leftarrow \boldsymbol{x}_{i,\ell}^t$ for $1 \le \ell \le L$ with probability $\frac{1}{L}$ and play $\boldsymbol{x}_i^t$
    Receive function f_i^t
    Set $\boldsymbol{g}_{i,1}^t \leftarrow \nabla f_i^t(\boldsymbol{x}_{i,1}^t)$
    for $1 \le \ell \le L$ **do**
      Send $\boldsymbol{g}_{i,\ell}^t$ to all neighbours $N(i)$.
      After receiving $\boldsymbol{g}_{j,\ell}^t$ from all neighbours $j \in N(i)$, compute $\boldsymbol{d}_{i,\ell}^t \leftarrow \sum_{j\in N(i)} W_{ij}\boldsymbol{g}_{j,\ell}^t$ and $\boldsymbol{g}_{i,\ell+1}^t \leftarrow \left(\nabla f_i^t(\boldsymbol{x}_{i,\ell+1}^t) - \nabla f_i^t(\boldsymbol{x}_{i,\ell}^t)\right) + \boldsymbol{d}_{i,\ell}^t$.
      Feedback function $\langle \boldsymbol{d}_{i,\ell}^t, \cdot \rangle$ to oracles $\mathscr{O}_{i,\ell}$. (The cost of the oracle $\mathscr{O}_{i,\ell}$ at time t is $\langle \boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t \rangle$.)
    end for
  end for
end for

iteration $1 \le \ell \le L$ as

$$\mathscr{G}_\ell^t := \max_{\boldsymbol{o}\in\mathscr{K}} \langle \nabla F(\overline{\boldsymbol{x}}_\ell^t), \overline{\boldsymbol{x}}_\ell^t - \boldsymbol{o} \rangle = \langle \nabla F(\overline{\boldsymbol{x}}_\ell^t), \overline{\boldsymbol{x}}_\ell^t - \boldsymbol{o}_\ell^t \rangle \tag{5.6}$$

where $\boldsymbol{o}_\ell^t = \arg\min_{\boldsymbol{o}\in\mathscr{K}} \langle \nabla F(\overline{\boldsymbol{x}}_\ell^t), \boldsymbol{o} \rangle$.

Lemma 2. *For all $1 \le i \le n$, for every iteration $1 \le \ell \le L$, it holds that*

$$\begin{aligned} &\max_{\boldsymbol{o}\in\mathscr{K}} \langle \nabla F_t(\boldsymbol{x}_{i,\ell}^t), \boldsymbol{x}_{i,\ell}^t - \boldsymbol{o} \rangle \\ &\le \max_{\boldsymbol{o}\in\mathscr{K}} \langle \nabla F_t(\overline{\boldsymbol{x}}_\ell^t), \overline{\boldsymbol{x}}_\ell^t - \boldsymbol{o} \rangle + (\beta D + G) C_p \frac{\log L}{L}. \end{aligned}$$

Lemma 3. *For every $1 \leq t \leq T$ and $1 \leq \ell \leq L$, it holds that*

$$\overline{\boldsymbol{x}}_{\ell+1}^t - \overline{\boldsymbol{x}}_\ell^t = \eta_\ell \left(\frac{1}{n} \sum_{i=1}^n \boldsymbol{v}_{i,\ell}^t - \overline{\boldsymbol{x}}_\ell^t \right) \tag{5.7}$$

Theorem 1. *Let $\mathscr{K}$ be a convex set with diameter D. Assume that functions F^t (possibly non convex) are β-smooth and G-Lipschitz for every t. With the choice of step size $\eta_\ell = \min\left(1, \frac{A}{\ell^\alpha}\right)$ where $A \geq 0$ and $\alpha \in (0,1)$. Then, Algorithm 1 guarantees that for all $1 \leq i \leq n$:*

$$\begin{aligned}
&\max_{\boldsymbol{o} \in \mathscr{K}} \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{\boldsymbol{x}_i^t} \left[\langle \nabla F^t(\boldsymbol{x}_i^t), \boldsymbol{x}_i^t - \boldsymbol{o} \rangle \right] \\
&\qquad \leq \frac{GDA^{-1}}{L^{1-\alpha}} + \frac{AD^2\beta/2}{L^\alpha(1-\alpha)} + O\left(\mathscr{R}^T\right) \\
&\qquad\quad + \left((\beta C_p + C_d) D + (\beta D + G) C_p \right) \frac{\log L}{L}
\end{aligned}$$

where $\mathscr{R}^T$ is the regret of online linear minimization oracles. Choosing $L = T$, $\alpha = 1/2$ and oracles as gradient descent or follow-the-perturbed-leader with regret $\mathscr{R}^T = O\left(T^{-1/2}\right)$, we obtain the gap convergence rate of $O\left(T^{-1/2}\right)$.

Démonstration. By β-smoothness, $\forall \ell \in \{1, \cdots, L\}$:

$$\begin{aligned}
&F^t\left(\overline{\boldsymbol{x}}_{\ell+1}^t\right) - F^t\left(\overline{\boldsymbol{x}}_\ell^t\right) \\
&\leq \left\langle \nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right), \overline{\boldsymbol{x}}_{\ell+1}^t - \overline{\boldsymbol{x}}_\ell^t \right\rangle + \frac{\beta}{2} \left\| \overline{\boldsymbol{x}}_{\ell+1}^t - \overline{\boldsymbol{x}}_\ell^t \right\|^2
\end{aligned} \tag{5.8}$$

Using Lemma 3, the inner product in (5.8) can be re-written as :

$$\begin{aligned}
&\left\langle \nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right), \overline{\boldsymbol{x}}_{\ell+1}^t - \overline{\boldsymbol{x}}_\ell^t \right\rangle \\
&= \eta_\ell \left\langle \nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right), \frac{1}{n} \sum_{i=1}^n \boldsymbol{v}_{i,\ell}^t - \overline{\boldsymbol{x}}_\ell^t \right\rangle \\
&= \eta_\ell \left\langle \nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right), \frac{1}{n} \left(\sum_{i=1}^n \boldsymbol{v}_{i,\ell}^t - n \cdot \overline{\boldsymbol{x}}_\ell^t \right) \right\rangle \\
&= \frac{\eta_\ell}{n} \sum_{i=1}^n \left\langle \nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right), \boldsymbol{v}_{i,\ell}^t - \overline{\boldsymbol{x}}_\ell^t \right\rangle
\end{aligned} \tag{5.9}$$

Let $\boldsymbol{o}_\ell^t$ be such that $\boldsymbol{o}_\ell^t \in \arg\min_{\boldsymbol{o}\in\mathscr{K}} \langle \nabla F(\overline{\boldsymbol{x}}_\ell^t), \boldsymbol{o}\rangle$. Hence,

$$\mathscr{G}_\ell^t = \max_{\boldsymbol{o}\in\mathscr{K}} \langle \nabla F(\overline{\boldsymbol{x}}_\ell^t), \overline{\boldsymbol{x}}_\ell^t - \boldsymbol{o}\rangle = \langle \nabla F(\overline{\boldsymbol{x}}_\ell^t), \overline{\boldsymbol{x}}_\ell^t - \boldsymbol{o}_\ell^t\rangle$$

We have :

$$\begin{aligned}
&\left\langle \nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right), \boldsymbol{v}_{i,\ell}^t - \overline{\boldsymbol{x}}_\ell^t \right\rangle \\
&= \langle \nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right) - \boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t - \boldsymbol{o}_\ell^t\rangle \\
&\qquad + \langle \boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t - \boldsymbol{o}_\ell^t\rangle \\
&\qquad + \langle \nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right), \boldsymbol{o}_\ell^t - \overline{\boldsymbol{x}}_\ell^t\rangle \\
&\leq \|\nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right) - \boldsymbol{d}_{i,\ell}^t\| \|\boldsymbol{v}_{i,\ell}^t - \boldsymbol{o}_\ell^t\| \\
&\qquad + \langle \boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t - \boldsymbol{o}_\ell^t\rangle \\
&\qquad + \langle \nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right), \boldsymbol{o}_\ell^t - \overline{\boldsymbol{x}}_\ell^t\rangle \\
&\leq \|\nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right) - \boldsymbol{d}_{i,\ell}^t\| D + \langle \boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t - \boldsymbol{o}_\ell^t\rangle \\
&\qquad + \langle \nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right), \boldsymbol{o}_\ell^t - \overline{\boldsymbol{x}}_\ell^t\rangle.
\end{aligned}$$

where we use Cauchy-Schwarz in the first inequality.

Using Lemma 1 and β-smoothness of F^t,

$$
\begin{aligned}
&\left\|\nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right)-\boldsymbol{d}_{i,\ell}^t\right\| \\
&\leq\left\|\nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right)-\frac{1}{n}\sum_{i=1}^n \nabla f_i^t\left(\boldsymbol{y}_{i,\ell}^t\right)\right\| \\
&\quad+\left\|\frac{1}{n}\sum_{i=1}^n \nabla f_i^t\left(\boldsymbol{y}_{i,\ell}^t\right)-\boldsymbol{d}_{i,\ell}^t\right\| \\
&\leq\left\|\frac{1}{n}\sum_{i=1}^n \nabla f_i^t\left(\overline{\boldsymbol{x}}_\ell^t\right)-\frac{1}{n}\sum_{i=1}^n \nabla f_i^t\left(\boldsymbol{y}_{i,\ell}^t\right)\right\| \\
&\quad+\left\|\frac{1}{n}\sum_{i=1}^n \nabla f_i^t\left(\boldsymbol{y}_{i,\ell}^t\right)-\boldsymbol{d}_{i,\ell}^t\right\| \\
&\leq\frac{1}{n}\sum_{i=1}^n\left\|\nabla f_i^t\left(\overline{\boldsymbol{x}}_\ell^t\right)-\nabla f_i^t\left(\boldsymbol{y}_{i,\ell}^t\right)\right\| \\
&\quad+\left\|\frac{1}{n}\sum_{i=1}^n \nabla f_i^t\left(\boldsymbol{y}_{i,\ell}^t\right)-\boldsymbol{d}_{i,\ell}^t\right\| \\
&\leq\frac{\beta}{n}\sum_{i=1}^n\left\|\overline{\boldsymbol{x}}_\ell^t-\boldsymbol{y}_{i,\ell}^t\right\| && \text{(by } \beta \text{ smoothness)} \\
&\quad+\left\|\frac{1}{n}\sum_{i=1}^n \nabla f_i^t\left(\boldsymbol{y}_{i,\ell}^t\right)-\boldsymbol{d}_{i,\ell}^t\right\| \\
&\leq\frac{\beta C_p+C_d}{\ell} && \text{(by Lemma 1)}
\end{aligned}
$$

Thus,

$$
\begin{aligned}
&\left\langle\nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right), \boldsymbol{v}_{i,\ell}^t-\overline{\boldsymbol{x}}_\ell^t\right\rangle \\
&\leq\left(\frac{\beta C_p+C_d}{\ell}\right) D+\left\langle\boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t-\boldsymbol{o}_\ell^t\right\rangle-\mathscr{G}_\ell^t
\end{aligned}
$$

Upper bound the right hand side of (5.9) by the above inequality, we have :

$$
\begin{aligned}
&\left\langle\nabla F^t\left(\overline{\boldsymbol{x}}_\ell^t\right), \overline{\boldsymbol{x}}_{\ell+1}^t-\overline{\boldsymbol{x}}_\ell^t\right\rangle \\
&\leq\eta_\ell \frac{\left(\beta C_p+C_d\right) D}{\ell}+\frac{\eta_\ell}{n}\sum_{i=1}^n\left\langle\boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t-\boldsymbol{o}_\ell^t\right\rangle-\eta_\ell \mathscr{G}_\ell^t
\end{aligned} \tag{5.10}
$$

Combining (5.8) with (5.10) and re-arrange the terms, as $\eta_\ell = \frac{A}{\ell^\alpha}$, we have :

$$\begin{aligned}\eta_\ell \mathscr{G}_\ell^t \leq F^t\left(\bar{\boldsymbol{x}}_\ell^t\right) - F^t\left(\bar{\boldsymbol{x}}_{\ell+1}^t\right) + \eta_\ell \frac{(\beta C_p + C_d)D}{\ell} \\ + \frac{\eta_\ell}{n}\sum_{i=1}^n \langle \boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t - \boldsymbol{a}_\ell^t\rangle + \eta_\ell^2 D^2 \frac{\beta}{2}\end{aligned} \tag{5.11}$$

Dividing by η_ℓ yields :

$$\begin{aligned}\mathscr{G}_\ell^t \leq \frac{L^\alpha}{A}\left(F^t\left(\bar{\boldsymbol{x}}_\ell^t\right) - F^t\left(\bar{\boldsymbol{x}}_{\ell+1}^t\right)\right) + \frac{(\beta C_p + C_d)D}{\ell} \\ + \frac{1}{n}\sum_{i=1}^n \langle \boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t - \boldsymbol{o}_\ell^t\rangle + \eta_\ell D^2 \frac{\beta}{2}\end{aligned} \tag{5.12}$$

Let $\mathscr{G}^t$ be a random variable such that $\mathscr{G}^t = \mathscr{G}_\ell^t$ with probability $\frac{1}{L}$. We are now bounding $\mathbb{E}_{\bar{\boldsymbol{x}}^t}[\mathscr{G}^t]$. By Inequality (5.12), using the definition of $\eta_\ell = \frac{A}{\ell^\alpha}$, G-Lipschitz property of F, we have (detail shown in [69])

$$\begin{aligned}\mathbb{E}_{\bar{\boldsymbol{x}}^t}\left[\mathscr{G}^t\right] \leq \frac{GDA^{-1}}{L^{1-\alpha}} + (\beta C_p + C_d) D \frac{\log L}{L} \\ + \frac{1}{nL}\sum_{\ell=1}^L \sum_{i=1}^n \langle \boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t - \boldsymbol{o}_\ell^t\rangle \\ + \frac{AD^2\beta/2}{L^\alpha(1-\alpha)}\end{aligned} \tag{5.13}$$

Summing the above inequality for $1 \leq t \leq T$ and note that $\frac{1}{T}\sum_{t=1}^T \langle \boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t - \boldsymbol{o}_\ell^t\rangle$ is the regret of the oracle $\mathscr{O}_i$, we get

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^T \mathbb{E}_{\bar{\boldsymbol{x}}^t}\left[\mathscr{G}^t\right] \leq \frac{GDA^{-1}}{L^{1-\alpha}} + O\left(\mathscr{R}^T\right) \\ + (\beta C_p + C_d) D \frac{\log L}{L} \\ + \frac{AD^2\beta/2}{L^\alpha(1-\alpha)}\end{aligned} \tag{5.14}$$

By uniformly random choice of $\boldsymbol{x}_i^t$ (over all $\boldsymbol{x}_{i,\ell}^t$ for $1 \leq \ell \leq L$) in the algorithm, we have (detail

shown in [69])

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\boldsymbol{x}_i^t}\big[\max_{\boldsymbol{o}\in\mathscr{K}}\langle\nabla F^t(\boldsymbol{x}_i^t),\boldsymbol{x}_i^t-\boldsymbol{o}\rangle\big] \leq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\overline{\boldsymbol{x}}^t}\left[\mathscr{G}^t\right]+(\beta D+G)C_p\frac{\log L}{L} \tag{5.15}$$

where we have used Lemma 2 and the fact that

$$\mathbb{E}_{\overline{\boldsymbol{x}}^t}\left[\mathscr{G}^t\right]=\mathbb{E}_{\overline{\boldsymbol{x}}^t}\left[\max_{\boldsymbol{o}\in\mathscr{K}}\langle\nabla F_t(\overline{\boldsymbol{x}}_\ell^t),\overline{\boldsymbol{x}}_\ell^t-\boldsymbol{o}\rangle\right]$$

By Jensen's inequality, we have :

$$\max_{\boldsymbol{o}\in\mathscr{K}}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\boldsymbol{x}_i^t}\big[\langle\nabla F^t(\boldsymbol{x}_i^t),\boldsymbol{x}_i^t-\boldsymbol{o}\rangle\big] \leq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\boldsymbol{x}_i^t}\big[\max_{\boldsymbol{o}\in\mathscr{K}}\langle\nabla F^t(\boldsymbol{x}_i^t),\boldsymbol{x}_i^t-\boldsymbol{o}\rangle\big] \tag{5.16}$$

The theorem follows (5.16), (5.15) and (5.14) and setting $L=T$, gives a $O(\sqrt{T})$ regret. □

5.3.1 Prediction Performance

We demonstrate the utility of our algorithm on an online temperature prediction problem in a building. The data were taken from a building dataset contains ambient time-series captured on seven floors; each floor has four sensor-equipped zones. We set a zone-wise knowledge exchange via an undirected graph of n nodes/zones participating in the learning process. For every round t, each node i receives a batch $\mathscr{B}_i^t$ of 32 time-series sequences corresponding to a look-back period 13 timestep to predict the temperature of the next timestep. At a resolution of 5 minutes, this corresponds to using 1 hour past data to predict the next 5 minutes. We extract the data from March 7^{th} to April 20^{th} for training, making the total iteration number equal to 360. We set L equal to the iteration number, $\alpha=0.95$ and $A=1$, a smaller value of L is possible to reduce the training time. A min-max scaler is used to normalize the data and we apply a rolling window with stride 1 on the original time series.

We embed each node with a model built from a two-layers long-short-time-memory (LSTM) network followed by a fully connected layer. Denote the output of the model i for a data sequence b at time t by $\hat{y}^t_{i,b}$ and its ground truth by $y^t_{i,b}$. Consider the ℓ_1 loss as the objective function :

$$\mathscr{L}(\hat{y}^t_{i,b}, y^t_{i,b}) = \begin{cases} \dfrac{(\hat{y}^t_{i,b} - y^t_{i,b})^2}{2} & \text{if } |\hat{y}^t_{i,b} - y^t_{i,b}| \\ & \leq 1 \\ |\hat{y}^t_{i,b} - y^t_{i,b}| - \frac{1}{2} & \text{otherwise} \end{cases}$$

Consider the constraint set $\mathscr{K} = \{\boldsymbol{x} \in \mathbb{R}^d, \|\boldsymbol{x}\|_1 \leq r\}$, where $\boldsymbol{x}$ is the model's weight, d its dimension and $r = 1$. The (normalized) loss incurred by the data of agent i is :

$$\frac{1}{|\mathscr{B}^t_i|} \sum_{b \in \mathscr{B}^t_i} \mathscr{L}(\hat{y}^t_{i,b}, y^t_{i,b}).$$

The global loss function incurred by the overall data is

$$F^t(\boldsymbol{x}) = \frac{1}{|\cup_{i=1}^n \mathscr{B}^t_i|} \sum_{b \in \cup_{i=1}^n \mathscr{B}^t_i} \mathscr{L}(\hat{y}^t_{i,b}, y^t_{i,b}),$$

that can be written as $F^t(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^n f^t_i(\boldsymbol{x})$ where

$$f^t_i(\boldsymbol{x}) = \frac{1}{|\mathscr{B}^t_i|} \sum_{b \in \mathscr{B}^t_i} \mathscr{L}(\hat{y}^t_{i,b}, y^t_{i,b}).$$

Note that the non-convexity here is due to the non-convexity of $\hat{y}^t_{i,b}$ as a function of $\boldsymbol{x}^t_i$.

In the following section, if not specify otherwise, we call *loss* the temporal average of the global loss function F^t defined as $\frac{1}{T}\sum_{t=1}^T F^t$.

Figures 5.12 and 5.13 shows the loss and gap values for different network size. We make a selection of zones and floors to include in our network. We select zone 1, zone 2, zone 4, and 5 from the sixth floor for a 4 zone configuration. The 7 zone configuration combines zonal data on the sixth and seventh floor, where we drop zone 1 from the seventh floor. In the 10 zone configuration, we combine the data from the fourth, fifth, and sixth floor where we drop

Nodes	Floors	MSE	MAE
13	4567	0.85	1.26
10	456	0.71	1.01
7	67	**0.65**	**0.78**
4	6	1.5	0.99

TABLE 5.4 – Performance comparison between 4 network configurations

zone 1 and 4 from the fifth floor. Finally, the 13 zone configuration combines all four floors with the same zonal selection. The implementation justifies our theoretical results about the convergence of the gap. Besides, we also observe the convergence of loss value, an expected implication of the gap convergence.

Moreover, by the analysis below, we also justify the particular thermal variation of the top floor compared to other ones. Table 5.4 reports the mean absolute error (MAE) and mean square error (MSE) between the prediction and the ground truth of the 4 presented configurations for three days. We set M the number of prediction points between the 21^{st} and 24^{th} of April and n the number of zones within one configuration. We call $\hat{y}_{i,m}$ and $y_{i,m}$, the predicted and the ground truth of model i for point m. The two metrics are computed as follows :
Mean absolute error :

$$\frac{1}{nM}\sum_{i=1}^{n}\sum_{m=1}^{M}|\hat{y}_{i,m}-y_{i,m}|$$

Mean squared error :

$$\frac{1}{nM}\sum_{i=1}^{n}\sum_{m=1}^{M}(\hat{y}_{i,m}-y_{i,m})^2$$

We observe that increasing nodes in a network does not always lead to better online performance. In-fact, a 7 node configuration achieves the lowest MSE (0.65) and MAE (0.78) for floors 6 and 7. We see a 40 % drop in MSE and 20 % reduction in MAE for Floor 6 zonal models when 3 extra peers from floor 7 joined the group. We observe 19 % and 25 % increase in MSE and MAE values by adding zonal nodes from floor 7 to a 10 node group. This can be best argued by the fact that the top floor of a building has a non identical thermal variation with the rest of the storeys. A supporting observation is the zones of the top 2 floors of the building collectively generalize the best compared to any other configuration.

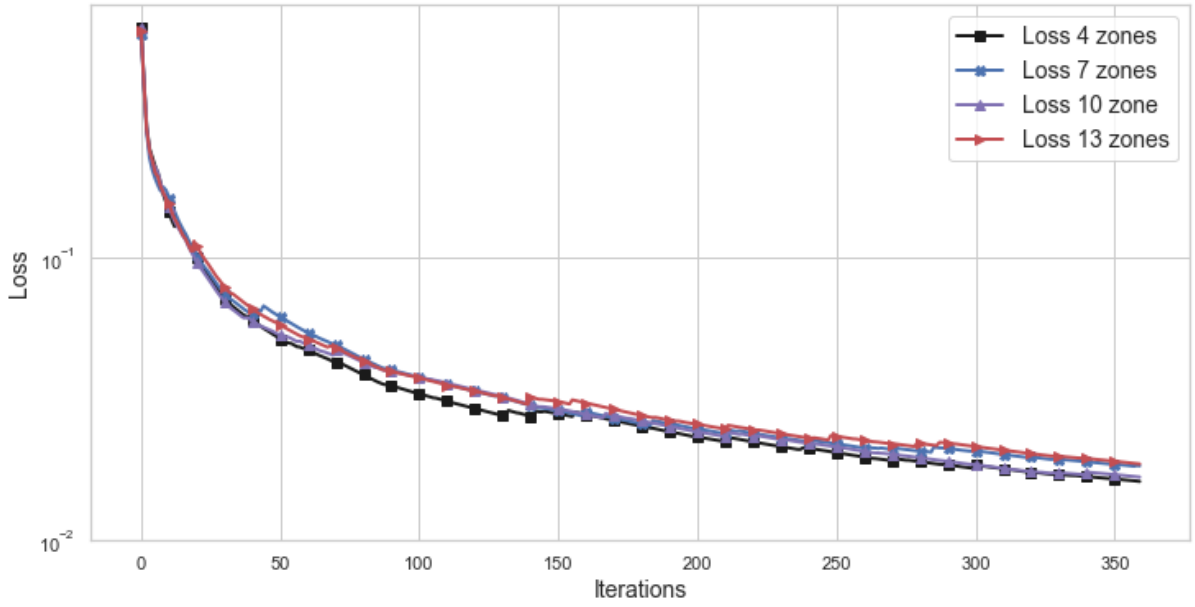


FIGURE 5.12 – Loss values of different network size on complete topology. *(Plot on log-scale)*

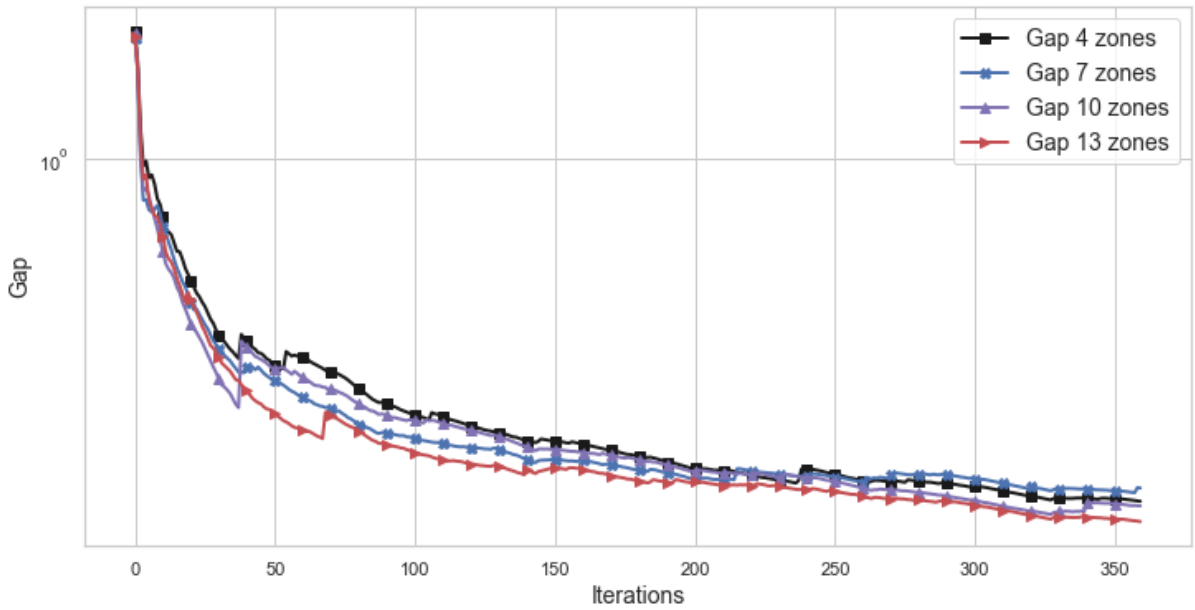


FIGURE 5.13 – Gap values of different network size on complete topology. *(Plot on log-scale)*

Topology	Metric	Mean	Variance	Max	Min
cycle	MAE	1.099	0.485	1.808	0.56
cycle	MSE	0.788	0.21	1.094	0.529
complete	MAE	**0.778**	0.381	1.478	0.27
complete	MSE	**0.646**	0.202	1.047	0.39
line	MAE	0.813	0.532	1.953	0.243
line	MSE	0.667	0.288	1.266	0.344

TABLE 5.5 – Impact of Networking on 7 learners configuration.

5.3.2 Effect of Network Topology on Learning

We study the effect of topology in learning by increasing networks in terms of links or nodes. We study a 7 node configuration with a complete, cycle and line graph containing 28, 7 and 6 edges respectively and with 13 nodes having 78,13 and 12 edges respectively. For both 7 (Table 5.5) and 13 (Table 5.6) node configurations, we observe that the complete graph yields the least amount of prediction error, mean absolute error $\in [0.66, 1.3]^{\circ}C$. However we note the peculiarity that the line graph can perform better than a cycle graph and has roughly a 10 % error margin compared to the complete configuration.

Topology	Metric	Mean	Variance	Max	Min
cycle	MAE	1.511	1.456	6.159	0.361
cycle	MSE	0.938	0.384	1.897	0.483
complete	MAE	**1.257**	**0.82**	3.64	**0.32**
complete	MSE	**0.852**	**0.272**	**1.505**	0.417
line	MAE	1.385	0.915	3.169	0.5
line	MSE	0.905	0.352	**1.664**	0.492

TABLE 5.6 – Impact of Topology on Temperature Forecasting Performance with 13 learners.

5.3.3 Effect of Decentralization

We are interested to understand the role of decentralization in terms of accuracy of zonal learners. Let $L_{MFW}(t)$ be the loss from Meta Frank Wolfe (MFW) at time t. The approximation ratio

$$A(t) = \frac{L_{DMFW}(t)}{L_{MFW}(t)}$$

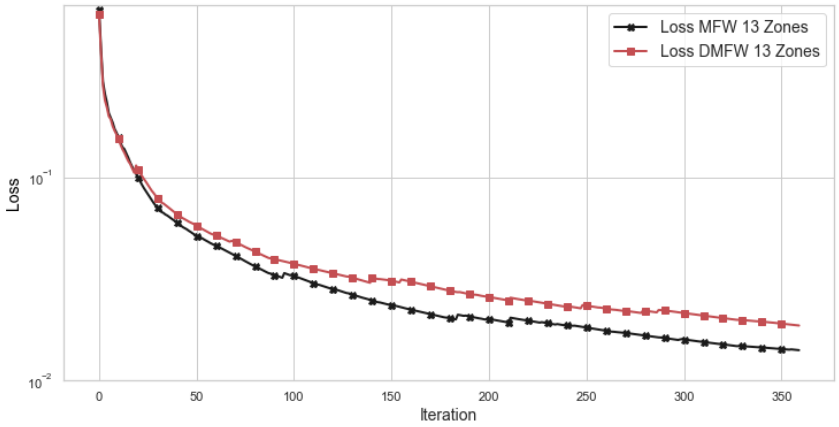


FIGURE 5.14 – Loss values of decentralized and centralized Meta Frank-Wolfe *(Plot on log-scale)*. We use data from 13 zones connected over a complete topology on decentralized setting *(red curve)* to compare with its centralized counterpart *(black curve)*

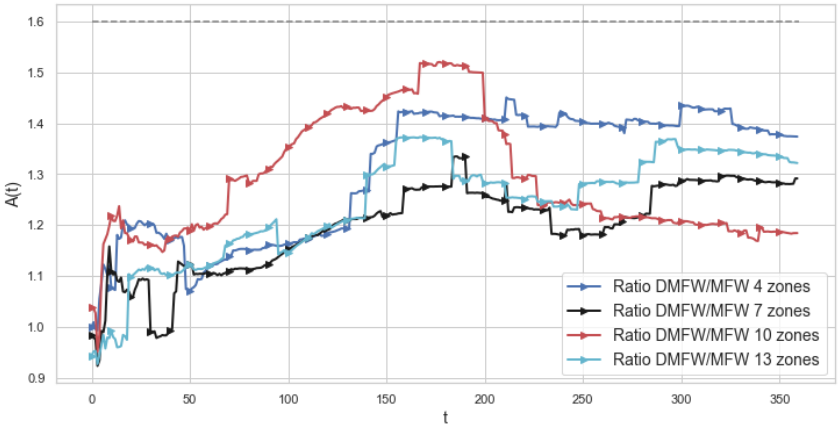


FIGURE 5.15 – Loss ratio of decentralized and centralized Meta Frank-Wolfe on different network size.

at time t is a heuristic to define how worse is our decentralized version compared to a centralized optimization. $A(t) \leq B_{max}$ will mean our algorithm performs no worse than B_{max} times of the MFW. Figure 5.14 shows that our algorithm performs slightly worse than MFW. On figure 5.15, we plot the ratio $A(t)$ for a 13 node network and show that $A(t) \leq 1.4$. The 7 node network has the closest approximation bounded by 1.35 which can be explained by earlier insights on performance accuracy. We notice that the 10 node network performs worse till $t = 200$, after which from $t \geq 250$ or 21 hours, the approximation ratio becomes close to centralised version with less than 20 % error.

5.4 Software Implementation

To ensure software code portability, an open-source application packager named Docker is used. This helps in abstracting heterogeneity in hardware through virtual containers that can run on Linux supported embedded hardware or classically execution in a public/private cloud/ data centers.

5.4.1 Architecture

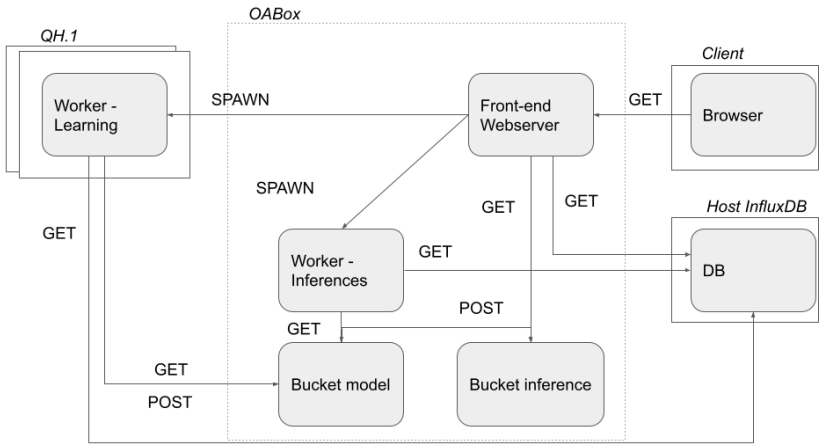


FIGURE 5.16 – Lifelong learning Architecture

A Qbox comes with a processor and persistent memory to host a time series like *InfluxDB* or S3 buckets while QRads offer an in-memory computation with no persistent data storage. Sensor-equipped space configurations are specified in a YAML format. The config file specifies an object where multiple spaces can be enlisted by their name inside each space, devices are registered with their type and access identifier. The ambient information comprises of humidity, temperature, sound, light, and carbon dioxide (CO_2) coming from sensors distributed across rooms. Below is a sample of a meeting room, with a Qrad and private office having a Netatmo sensor recording the above-mentioned channels to the Qinflux-db at a **resolution** of 5 minutes.

Electrical power is measured across a supply line catering to multiple sockets or appliances or illumination needs. An energy-meter (*Schneider eco-compteur*) records the power consumption required for the cuisine, heating services, sockets and lightnings. The minimum and maxi-

```yaml
---
version: '3.3' # CHECK

services:
  Qapi-service:
    build: ./backend
    image : angmit/oasisbackend:v2.3
    volumes:
      - ./backend/app:/app
    ports:
      - 5001:80 # host:container
      - 4200:4200 # to receive data from Qarnot API
    command: "conda run -n myenv python app/main.py"

  Qfrontend:
    build: ./frontend
    image : angmit/oasisfrontend:v2.2
    volumes:
      - ./frontend/src:/opt
    ports:
      - 8501:80 # to receive data from client
    depends_on:
      - api-service
    command: "streamlit run launch.py --server.port 80"

  Qinfluxdb:
    build: ./influxdb
    image: angmit/oasisdb:v0.1
    ports:
      - 8086:8086
    environment:
      INFLUXDB_DB: "sensorDB"
      INFLUXDB_ADMIN_USER: "oasis"
      INFLUXDB_ADMIN_PASSWORD: "oasis123"
      INFLUXDB_HTTP_AUTH_ENABLED: "true"
    volumes:
      - ./influxdb/data:/var/lib/influxdb

  Qlearning-service:
    build: ./backend
    image : angmit/oasisbackend:v2.3
    volumes:
      - ./backend/app:/app
    links:
      - influxdb
    command: "conda\ run -n myenv python app/rwdata.py"

---
```

(a) YAML specification (part 1)

(b) YAML specification (part 2)

FIGURE 5.17 – Example of YAML configuration file to deploy the 4 containers underlying the Building Management Software.

```yaml
    floorid: FLOOR-1

spaces:
    meeting-room:
        devices:
            qrad:
                - QRAD-xxxx
    private-office:
        devices:
            netatmo:
                - QNETATMO-xxx

data:
    temperature:
        attributes:
            - data.temperature.air
        time: data.temperature.time
        max: 50
        min:  1

    humidity:
        attributes:
            - data.humidity.value
        time: data.humidity.time
        max: 100
        min:  1

    co2:
        attributes:
            - data.co2.value
        time:  data.co2.time
        max: 1500
        min:  1

    sound:
        attributes:
            - data.sound.max
            - data.sound.mean
        time: data.sound.time
        max: 100
        min:  1

    luminosity:
        attributes:
            - data.luminosity.light
        time: data.luminosity.time
        max: 100
        min:  1


dataaggregators:
    - MEAN


resolution : 5m
```

(a) Placing ambient sensors in room (part 1)

(b) Ambient Sensor Data specification (part 2)

FIGURE 5.18 – Example of YAML configuration to specify the sensors configuration

```
  floorid:  FLOOR-xxx
utility: energy
spaces:
    kitchen:
        devices:
            schneider:
                - SCHNEIDER-iEM1541-Process 3
    heating:
        devices:
            schneider:
                - SCHNEIDER-iEM1541-Process 2

    sockets:
        devices:
            schneider:
                - SCHNEIDER-iEM1541-Process 1

    illumination:
        devices:
            schneider:
                - SCHNEIDER-iEM1541-Lights
```

(a) Space and electrical channel integration (part 1)

```
--
data:
    powerAll:
        attributes:
            - data.power.powerAll
        time: data.power.time
        max : 150,000
        min : 0


dataaggregators:
    - MEAN


resolution : 5m
```

(b) Electrical power processing specification (part 2)

FIGURE 5.19 – Example of YAML configuration to specify the sensors configuration

mum power cap reading is from 0.001 to 150 Kilo-watt.

The Building Management Software is implemented via 4 dockers containers providing the following services :

— Qfrontend is pythonic frontend built using Streamlit Framework and made available under the Docker-image name "angmit/oasisfrontend :v2.2". The front-end sends http requests to a back-end server known as the Qapi-service.

— Qapi-service is a REST server whose API (Application Programming Interface) routes are enlisted in Table 5.7. The flask application listens on port 5001 which is mapped to http port 80 for replying to incoming web-requests.

— Qlearning-service is invoked periodically and executes a set of routines that analyze incoming data, predict semantic activities, save the insights on to a database for the building.

— Qinfluxdb is a time-series database which stores ambient sensor measurements, electrical power readings and derived insights. The machine learning models are stored on S3 bucket in the Qarnot Compute platform.

The following code-snippet is a yaml file specifying the services and can be run by executing the command "docker-compose up". The REST-api and learning-services of the docker application is executed on Qrads. The computed results are sent back to QBox and is pushed either

Route	Argument
/v1/get-raw-data	config-file, start-time, end-time
/v1/get-oracle-detection	config-file, start-time, end-time
/v1/guess-appliances	config-file, start-time, end-time
/v1/running-appliances	config-file, start-time, end-time
/v1/train-forecast-channel	config-file, start-time, end-time, modelSavePath, column, sensor
/v1/predict-forecast-channel	modelSavePath, start-time, lookAheadIntervals
/v1/get-room-status	config-file, start-time, end-time
/v1/get-air-quality	config-file, start-time, end-time
/v1/get-energy-metadata	config-file, start-time, end-time
/v1/fetch-calendar-events	config-file, start-time, end-time
/v1/get-waste-energy	config-file, start-time, end-time
/v1/spatial-navigation	sourceID, targetID
/v1/get-storeys	
/v1/building-metadata	
/v1/show-floorplan	storeyID
/v1/get-live-weather	
/v1/get-daily-weatherdata	day
/v1/predict-learner	start-time, end-time, model-name, config-file, outputPath
/v1/train-multi-column-forecast	config-file, scalerSavePath, modelSavePath, trainOutputPath
/v1/predict-multi-column-forecast	config-file, scalerSavePath, modelSavePath, testOutputPath
/v1/train-learner	config-file, scalerSavePath, modelSavePath, trainOutputPath, inferencePath
/v1/federated-learning	config-file, scalerSavePath, modelSavePath, trainOutputPath, inferencePath
/v1/ventilation-suggestions	config-file, start-time, end-time

TABLE 5.7 – REST API list

to the bucket or database. This ensures in-house privacy by design, since neither data nor computation leaves the premises as shown in Figure 5.1. The backend image is available from the Docker hub under *angmit/oasisbackend :v2.3* where the code is organised into 7 pythonic class modules. The API list given by Table 5.7 is responsible for storing the building data, utilising spatial orientation for navigation, ventilation, fetching sensor plus weather data, aggregating calendar events, generate automatic insights about room occupancy, illumination status, CO_2 dissipation suggestions and energy savings.

Route	Task Caller	Average Run-time (seconds)
/v1/get-raw-data	Qlearning-service, Qapi-service	87 ± 23
/v1/get-oracle-detection	Qlearning-service	10 ± 6
/v1/guess-appliances	Qlearning-service	360 ± 10
/v1/running-appliances	Qlearning-service	500 ± 49
/v1/train-forecast-channel	Qlearning-service	400 ± 34
/v1/predict-forecast-channel	Qapi-service	9 ± 2
/v1/get-room-status	Qapi-service	<1 second
/v1/get-air-quality	Qapi-service	<1 second
/v1/get-energy-metadata	Qapi-service	<1 second
/v1/fetch-calendar-events	Qlearning-service	1-2 seconds
/v1/get-waste-energy	Qapi-service	<1 second
/v1/spatial-navigation	Qapi-service	<1 second
/v1/get-storeys	Qapi-service	<1 second
/v1/building-metadata	Qapi-service	<1 second
/v1/show-floorplan	Qapi-service	<1 second
/v1/get-live-weather	Qapi-service	<2 seconds
/v1/get-daily-weatherdata	Qlearning-service	7 ± 3
/v1/predict-learner	Qapi-service	<10 seconds
/v1/train-multi-column-forecast	Qapi-service	600-800 seconds
/v1/predict-multi-column-forecast	Qapi-service	<10 seconds
/v1/train-learner	Qlearning-service	670 ± 8
/v1/federated-learning	Qapi-service	550 ± 34
/v1/ventilation-suggestions	Qlearning-service	130 ± 3

TABLE 5.8 – Average API runtimes

5.4.2 Run-times

We observe a range of run-times from a few seconds to upto 15 minutes for tasks catering to data fetching, inference generation and learning. In Table 5.8, we present the average execution time taken by API routines over 10 random runs and also highlight the calling function (*Task Caller*). The volume of sensor data fetched can range from 5 minutes to 7 days and is proportional to the look-back interval. The maximum run time reported is for $24 \times 7 \times 288 = 48384$ data samples, with each data sample having at most 5 features for ambience plus 4 for energy reading. We observe that the inference tasks generally take less than 10 seconds while the training tasks can last from 7-15 minutes. During an inference task, the software downloads the latest version of a machine-learnt model from Qarnot's S3 bucket, readies the input data, returns the inference and deletes the model from local memory.

5.5 Chapter Summary

This chapter aims at collaboratively forecasting the indoor thermal profile of a building following a strict on site data retention policy. Indoor air temperature is susceptible to random fluctuations in environments like opening of a door or window and hence the work investigates for a possible mapping between intrinsic properties of a smart heater (Qarnot) and spatially ambient air temperature. Utilising heater's power and heat-sink temperature allows predictive modelling between energy footprint and ambient comfort. Since the model was only trained from logs of Qarnot computing heaters, one may question "What if there are heterogeneous hardware from multiple vendors ?". One way to resolve this bottleneck, is to embrace a learning technique that prohibits raw-data transmission between multiple agents but rather compressed knowledge representation. Applying federated orchestration, we saw a deep neural network model can benefit from sharing knowledge and locally customize a prediction task. Experiments from Qarnot data set confirm the intuition that a space tends to collaborate better with a similar usage peer rather than one with a different pattern. This naturally leads to the context : "Is it possible that devices at edge can communicate freely with peers and learn better with time ?" Decentralization is often a key design component in robust learning architectures

and has a close alignment with edge paradigm. Keeping in mind that eventually such deployments will be on low powered devices, the work addresses two key bottlenecks : piling up of time-series data and intense computing requirements. Correspondingly the proposed algorithm optimises the knowledge representation is an online manner with a low compute *projection-free* nature. Experiments on data from a 7 storey high-rise in Thailand show the effect of network topology on distributed learning while providing less than 20 % error in compared to a centralised version. In a nutshell, the chapter addresses how sensor equipped computing units at edge can form a distributed learning system for smart buildings. Higher levels of data privacy can be ensured with a **no raw data sharing** policy with representation learning to update knowledge collaboratively.

Chapitre 6

Conclusion

The highlights of the work related to smart building are extracted along with the direction and motivation for future work.

6.1 Take-away Highlights

The work gives the blueprint of a generic building management system with a novel minimalist sensing solution for non-intrusive spatio-temporal coverage. The system incrementally investigates ambience intelligence by detecting *good* sites to place sensors and re-calibrate placement configurations over time. Furthermore, by design raw data is localized at the generation site, thus enabling federated and peer to peer learning algorithms to run at edge.

6.1.1 Sensor Less Intelligence

Firstly, a full stack smart building solution will need to incorporate structural knowledge and spatial orientation about the building. Open sourcing the geometric parser fills in the gap of a tool that robustly parses IFC 2x3,4 file format to abstract the high level complexity of spatial connectivity. The IFC parsing technique has been pivotal in conceptualising buildings as a dynamic graph structure. The layer of sensor-less intelligence uses spectral decomposition

of the enriched spatial graph to derive occupancy insights. Human-space interaction models are used additionally to highlight the instantaneous temporal importance over a group of spaces.

6.1.2 Virtual Sensor Field

We broaden the scope of the Optimal Sensor Location Problem to find the minimal support sensors that can predict the redundant sensor values. The aim here is to generate a virtual sensor field for a building that is spatially optimised to have the least generalization error in prediction. Generating a distributed virtual field contributes to the solution of monitoring spaces non intrusively. The virtual sensor field can be utilized to approximate the original data in case of sensor fault or turn off unnecessary permissible sensors. We observe that the missing sensor approximation can be kept competitively accurate with bidirectional power-ambience converters with explainable insights. For example, the behaviour of a group of temperature sensors situated across multiple zones probably can be learnt by an optimal fraction of embedded devices. Noticeably, the energy footprint in powering up all sensors is higher than a fraction of the same. In a non-intrusive setting, the semantic label for an activity is unknown for a real deployment. Furthermore, data heterogeneity is evident when it comes to deploying non-identical sensors across spaces. We investigate the impact of activities on ambient sensing channels and find that selecting temporally optimal values can serve as an auto labelling heuristic. Training low-complexity classical machine learning models with cross validation shows incremental improvement in occupancy detection over time.

6.1.3 Energy Ambience Dynamics

Next, the interaction between ambience and energy through a bi-directional mapping between two semantic features sets is investigated by studying real life logs from an edge conglomeration of smart computing heaters. We developed an ambient temperature forecasting method based on short-term time series prediction, which considers the historical data of the ambient temperature and the smart heater's properties, notably its power consumption and heatsink temperature. Our experimental results show that a relatively simple trained time-series pre-

diction model is capable of accurately predicting the ambient temperature of an environment heated by a smart heater. Our simplest evaluated model – which uses the ambient temperature and the heater's power consumption historical data – performed ambient temperature predictions with a Mean Absolute Percentage Error of $2.66\% \pm 2.52\%$ and Root Mean Squared Error $0.92°C \pm 1.00°C$, when compared with the real temperature data.

6.1.4 Co-operative Edge Intelligence

By design, the proposed smart building solution retains data at site which helps in auto-updating deep learning models. According to the literature, to use a technique such as federated averaging one has to rely on an external mediator, typically another server connected to clients. We add to the security by replacing the mediator with a cloud storage and automated trigger functions. The simple shared storage learning algorithm allows to experiment controllable federated techniques to learn from connected objects at edge without a mediator. The framework is used for deep federated personalization at edge to better generalize forecasting models and understand the preferential relationships.

Lastly, an edge friendly online algorithm is developed to minimize non-convex loss functions aggregated from local data distributed over a network. We show the convergence of the Frank-Wolfe gap, a standard stationary measure related to non-convex functions, in both exact and stochastic gradient settings. Besides, we utilize our algorithm to train a sequence to sequence deep learning model to forecast indoor temperature per zone. Experimental results from a real-life smart building data-set make our contribution suitable for a distributed setting.

Lastly, we summarise the working of the developed lifelong learning system with the four components as follows :

1. Data driven **models** are learnt primarily for two kinds of task : short term forecasting, and imputation of missing sensor values.
2. Upon generation of new data, the task manager updates the stored machine learning models without leaking data from site, either through isolated learning, federated personalization or decentralised peer to peer exchange to be discussed in details in Chapter

5.

3. **Knowledge Base** for a building composes of spatial property graphs and sensor metadata to capture the spatio-temporal semantics and be aware of specific locations respectively. For powering up the Virtual Sensor Field, if Z, M, N be the number of zones, maximum available and missing sensors per zone then the system creates $O(ZMN)$ models. The predictive field is powered by $O(ZM)$ models in the federated setting and additionally $O(ZM)$ oracles for a projection free setting.
4. **Knowledge Base Learner** is responsible in tracking the performance of Pareto Optimal configurations and discover robust configurations. It utilises the previously learnt knowledge to auto-analyse on the test period through the metric L_{RC}. Additionally for each configuration, the system takes into account the approximation hardness during learning and *presumably* post deployment phase to generate robuster configurations when more data is available.

In a nutshell, the work contributes towards a generic smart building management system which showcases the possibility of business intelligence with an ad-hoc non intrusive IoT integration to lower the capital cost and energy footprint.

6.2 Future Work

There is ample scope of investigation in the trade-off between exploration and exploitation of the configuration and the hypothesis space. The approximate value of a hidden sensor channel is computed using the best single domain to domain mapping possible. Herein lies the possibility of further improvement using multiple features to predict a missing value. It can be interesting to compare the Lifelong learning setting against a reinforcement strategy to incrementally re-configure the virtual sensing field.

So far the work relies on a top down approach in which sensors are removed from a complete information set. The drawback to such an approach is the availability of complete data and also the cost of procuring all types of sensors for all the zones at-least for data collection. The alternate way of solving the same can be a bottom up approach, where a system can reach

desired configurations by incremental learning starting from a sensor.

One of the most promising directions to investigate is the design of *safe* networks at edge that can react to adversarial changes. The assumption for collective learning in this work only takes co-operative learners into consideration. There needs to be real-time strategies to dynamically re-organize the federation network for minimizing the spread of corrupt bad data. Furthermore, the algorithm for decentralized learning does not take into account realistic network delays and possible drop outs which is a likely edge behaviour.

Another possible update for the Virtual Sensing Field can be to integrate control inputs for devices like HVAC, illumination, appliances etc in a controlled environment. Adapting to renewable energy sources plays an important role in lowering down the carbon footprint. There needs to be a data driven optimization to determine when to shift between conventional grid and greener energy. With a non-constant/ variable energy tariff, the importance of a smart building management system is felt more to schedule consumption patterns.

The work carried out only answers to questions after a building has been made and not during the construction phase of the same, which leaves a scope for further investigation. It can be beneficial to incorporate sensors during build time to obtain the complete picture of a digital twin that can not only to save time, but also provide use-cases on safety of on-site workers.

6.3 Work Dissemination

The following set of work have been peer reviewed and accepted in conferences, they are directly related to the contributions presented in this thesis :

— Angan Mitra, Yanik Ngoko, Denis Trystram *Smart Oracle Based Building Management System*, 7^{th} IEEE International Conference on Smart Computing (SMARTCOMP), 61-68, 2021

— Angan Mitra, Yanik Ngoko, Denis Trystram, *Impact of Federated Learning On Smart Buildings*, IEEE International Conference on Artificial Intelligence and Smart Systems, 2021

— Angan Mitra, Y Ngoko, D Carastan-Santos, AA Da Silva, D Trystram, A Goldman *Short-Term Ambient Temperature Forecasting for Smart Heaters*, 26^{th} IEEE Symposium on Computers and Communications, 2021

The following set of work are under review :

— Angan Mitra, Denis Trystram *Next Generation Intelligent Spapces*, IEEE Pervasive Computing (Percom), 2022.

— Angan Mitra, Yanik Ngoko, Denis Trystram, *Decentralized Meta Frank Wolfe for online neural network optimization*, Journal on Parallel and Distributed Computing, 2022.

Bibliographie

[1] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, *et al.*, "Brick : Metadata schema for portable smart building applications," *Applied energy*, vol. 226, pp. 1273–1292, 2018.

[2] Wikipedia, "Constructive solid geometry," 2004. [Online ; accessed 22-July-2013].

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning : Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[4] 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, pp. 1738–1762, 2019.

[5] G. M. D. T. Forecast, "Cisco visual networking index : global mobile data traffic forecast update, 2017–2022," *Update*, vol. 2017, p. 2022, 2019.

[6] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[7] J. K. Wong, H. Li, and S. Wang, "Intelligent building research : a review," *Automation in construction*, vol. 14, no. 1, pp. 143–159, 2005.

[8] S. Roth, "Open green building xml schema : A building information modeling solution for our green world," *URL http ://gbxml. org/–Überprüfungsdatum*, pp. 08–25, 2014.

[9] D. J. Cook, M. Youngblood, E. O. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "Mavhome : An agent-based smart home," in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003.(PerCom 2003).*, pp. 521–524, IEEE, 2003.

[10] T. G. Stavropoulos, D. Vrakas, D. Vlachava, and N. Bassiliades, "Bonsai : a smart building ontology for ambient intelligence," in *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, pp. 1–12, 2012.

[11] L. Liu, B. Li, S. Zlatanova, and P. van Oosterom, "Indoor navigation supported by the industry foundation classes (ifc) : A survey," *Automation in Construction*, vol. 121, p. 103436, 2021.

[12] V. Bazjanac and D. B. Crawley, "Industry foundation classes and interoperable commercial software in support of design of energy-efficient buildings," in *Proceedings of Building Simulation'99*, vol. 2, pp. 661–667, 1999.

[13] G. Iulian and S. A. Oae, "Evolution of smart buildings," in *Proceedings of the 2013 International Conference on Environment, Energy, Ecosystems and Development*, 2013.

[14] V. Garg and N. K. Bansal, "Smart occupancy sensors to reduce energy consumption," *Energy and Buildings*, vol. 32, no. 1, pp. 81–87, 2000.

[15] B. Dong and B. Andrews, "Sensor-based occupancy behavioral pattern recognition for energy and comfort management in intelligent buildings," in *Proceedings of building simulation*, pp. 1444–1451, 2009.

[16] B. Ai, Z. Fan, and R. X. Gao, "Occupancy estimation for smart buildings by an auto-regressive hidden markov model," in *2014 American Control Conference*, pp. 2234–2239, IEEE, 2014.

[17] L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models," *Energy and Buildings*, vol. 112, pp. 28–39, 2016.

[18] J. Cumin, G. Lefebvre, F. Ramparany, and J. L. Crowley, "Human activity recognition using place-based decision fusion in smart homes," in *International and Interdisciplinary Conference on Modeling and Using Context*, pp. 137–150, Springer, 2017.

[19] D. Ganesan, D. Estrin, and J. Heidemann, "Dimensions : Why do we need a new data handling architecture for sensor networks ?," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 143–148, 2003.

[20] A. A. Volkov and E. I. Batov, "Simulation of building operations for calculating building intelligence quotient," *Procedia Engineering*, vol. 111, pp. 845–848, 2015.

[21] E. I. Batov, "The distinctive features of "smart" buildings," *Procedia Engineering*, vol. 111, pp. 103–107, 2015.

[22] M. Jia, A. Komeily, Y. Wang, and R. S. Srinivasan, "Adopting internet of things for the development of smart buildings : A review of enabling technologies and applications," *Automation in Construction*, vol. 101, pp. 111–126, 2019.

[23] S. N. Hojjati and M. Khodakarami, "Evaluation of factors affecting the adoption of smart buildings using the technology acceptance model," *International Journal of Advanced Networking and Applications*, vol. 7, no. 6, p. 2936, 2016.

[24] Z. Ma, J. D. Billanes, and B. N. Jørgensen, "A business ecosystem driven market analysis : The bright green building market potential," in *2017 IEEE Technology & Engineering Management Conference (TEMSCON)*, pp. 79–85, IEEE, 2017.

[25] Y. Xu, P. Ahokangas, M. Turunen, M. Mäntymäki, and J. Heikkilä, "Platform-based business models : Insights from an emerging ai-enabled smart building ecosystem," *Electronics*, vol. 8, no. 10, p. 1150, 2019.

[26] J. Cumin, G. Lefebvre, F. Ramparany, and J. L. Crowley, "A dataset of routine daily activities in an instrumented home," in *International Conference on Ubiquitous Computing and Ambient Intelligence*, pp. 413–425, Springer, 2017.

[27] F. Rosenblatt, "The perceptron : a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[28] J. Wang, M. Kolar, and N. Srebro, "Distributed multi-task learning," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016*, pp. 751–760, PMLR, oct 2016.

[29] Y. Zhang and Q. Yang, "A Survey on Multi-Task Learning," tech. rep.

[30] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated Multi-Task Learning," may 2017.

[31] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[32] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence : the confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, 2020.

[33] H. Peng, J. Wu, S. Chen, and J. Huang, "Collaborative channel pruning for deep networks," in *International Conference on Machine Learning*, pp. 5113–5122, 2019.

[34] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 875–886, 2018.

[35] X. Qi and C. Liu, "Enabling deep learning on iot edge : Approaches and evaluation," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 367–372, IEEE, 2018.

[36] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, vol. 54, 2017.

[37] J. Konečný, B. McMahan, and D. Ramage, "Federated Optimization :Distributed Optimization Beyond the Datacenter," nov 2015.

[38] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated Learning : Strategies for Improving Communication Efficiency," 2016.

[39] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving data privacy via federated learning : Challenges and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 8–16, 2020.

[40] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, pp. 1273–1282, PMLR, 2017.

[41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[42] J. C. Dunn and S. Harshbarger, "Conditional gradient algorithms with open loop step size rules," *Journal of Mathematical Analysis and Applications*, vol. 62, no. 2, pp. 432–444, 1978.

[43] M. Jaggi, "Revisiting frank-wolfe : Projection-free sparse convex optimization," in *International Conference on Machine Learning*, pp. 427–435, PMLR, 2013.

[44] N. Hovakimyan, "Robust adaptive control course notes," *University of Illinois at Urbana Champaign (UIUC)*, 2009.

[45] J. Lafond, H.-T. Wai, and E. Moulines, "On the online frank-wolfe algorithms for convex and non-convex optimizations," *arXiv preprint arXiv :1510.01171*, 2015.

[46] F. Yan, S. Sundaram, S. V. N. Vishwanathan, and Y. Qi, "Distributed autonomous online learning : Regrets and intrinsic privacy-preserving properties," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2483–2493, 2013.

[47] S. Hosseini, A. Chapman, and M. Mesbahi, "Online distributed optimization via dual averaging," in *52nd IEEE Conference on Decision and Control*, pp. 1484–1489, 2013.

[48] E. Hazan, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[49] W. Zhang, P. Zhao, W. Zhu, S. Hoi, and T. Zhang, "Projection-free distributed online learning in networks," in *Proceedings of the 34th International Conference on Machine Learning*, pp. 4054–4062, 2017.

[50] S. Thrun, "Lifelong learning : A case study.," tech. rep., Carnegie-Mellon Univ Pittsburgh pa Dept of Computer Science, 1995.

[51] Z. Chen and B. Liu, "Lifelong machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 10, no. 3, pp. 1–145, 2016.

[52] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf, "Computational geometry," in *Computational geometry*, pp. 1–17, Springer, 1997.

[53] M. I. Shamos and D. Hoey, "Geometric intersection problems," in *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, pp. 208–215, IEEE, 1976.

[54] Buslaev, "Albumentations : fast and flexible image augmentations," *ArXiv e-prints*, 2018.

[55] D. Cvetković and P. Rowlinson, "The largest eigenvalue of a graph : A survey," *Linear and multilinear algebra*, vol. 28, no. 1-2, pp. 3–33, 1990.

[56] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote : synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[57] M. Pipattanasomporn, G. Chitalia, J. Songsiri, C. Aswakul, W. Pora, S. Suwankawin, K. Audomvongseree, and N. Hoonchareon, "Cu-bems, smart building electricity consumption and indoor environmental sensor datasets," *Scientific Data*, vol. 7, no. 1, pp. 1–14, 2020.

[58] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[59] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited : why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[60] L. Klein, J.-y. Kwak, G. Kavulya, F. Jazizadeh, B. Becerik-Gerber, P. Varakantham, and M. Tambe, "Coordinating occupant behavior for building energy and comfort management using multi-agent systems," *Automation in construction*, vol. 22, pp. 525–536, 2012.

[61] P. Wijukkana, A. Julsereewong, and T. Thepmanee, "Temperature variation modeling for improving building hvac control using built-in temperature sensors in smartphones," in *2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pp. 790–793, IEEE, 2017.

[62] T. Oreszczyn, S. H. Hong, I. Ridley, P. Wilkinson, W. F. S. Group, *et al.*, "Determinants of winter indoor temperatures in low income households in england," *Energy and Buildings*, vol. 38, no. 3, pp. 245–252, 2006.

[63] S. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, 09 2017.

[64] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," 1989.

[65] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, aug 2018.

[66] M. G.Poirot, P. Vepakomma, K. Chang, J. Kalpathy-Cramer, R. Gupta, and R. Raskar, "Split Learning : Distributed and collaborative learning." `https://www.media.mit.edu/projects/distributed-learning-and-collaborative-learning-1/overview/`.

[67] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv :1912.00818*, 2019.

[68] H. Wai, J. Lafond, A. Scaglione, and E. Moulines, "Decentralized frank–wolfe algorithm for convex and nonconvex problems," *IEEE Transactions on Automatic Control*, vol. 62, no. 11, pp. 5522–5537, 2017.

[69] *Decentralized Meta Frank Wolfe for Online Neural Network Optimization*, Zenodo, 2021.