PROJECT "LOCUS": LOCalization and analytics on-demand embedded in the 5G ecosystem, for Ubiquitous vertical applicationS

Grant Agreement Number: 871249
(https://www.locus-project.eu/)

DELIVERABLE D5.2

"Design and implementation of virtualization technologies and pattern recognition mechanisms for physical analytics"- final version

Deliverable Type:	R
Dissemination Level:	Public
Contractual Date of Delivery to the EU:	31/12/2021
Actual Date of Delivery to the EU:	04/02/2022
WP contributing to the Deliverable:	WP5 – Localization & Analytics for New Services
Editor(s):	**IBM**, Faisal Ghaffar
Author(s):	**NXW**, Michael De Angelis, Giacomo Bernini **NEC**, Gurkan Solmaz **VIAVI**, Takai Eddine Kennouche **CNIT**, Andrea Giani, Flavio Morselli, Andrea Conti **SAMSUNG**, Tomasz Mach **Incelligent**, Yannis Filippas **Ericsson**, Stefano Stracca
Internal Reviewer(s):	**Orange**, Sana Ben Jemaa **NEC**, Gurkan Solmaz **IBM**, Joseph Antony
Short Abstract:	The goal of this deliverable is to describe the design and implementation of the virtualization techniques as well as the pattern recognition and machine learning algorithms for physical

	analytics involved in the deployment of the proposed new services.
Keyword List:	Spatial-Temporal Data Analytics, Machine Learning Pipelines, Virtual Network Functions, Deployment Functions

Executive Summary

One of the goals of the LOCUS project is to provide support for location-based analytics and vertical applications. To achieve this, LOCUS aims at exposing a set of functionalities and services on top of a flexible and scalable architecture. It relies on industry-proven API, virtualization and data management technologies, and integration with state-of-the-art 3GPP and non-3GPP localization technologies, and ML-based techniques.

This deliverable extends the data-driven approaches described in D5.1 with respect to the proposed Location & Analytics as a Service Solution. The first version provided an analysis of the use cases studied in WP5 in terms of functionalities and proposed solutions as a set of decoupled microservices. This decomposition allows the definition of a catalogue of services LOCUS can provide to support a variety of applications and target a wide range of scenarios. In addition to that, details on service orchestration, virtualization and ML optimization were provided. In this deliverable, ML models are extended to improve upon forecasting accuracy, easy employability, and faster inference. Finally, this deliverable provides details on the advancements in virtualisation of analytics functions for efficient and scalable data management service interaction with data, in addition to integration with external data sources, based on Kafka and modern database technologies.

VERSION CONTROL TABLE			
VERSION N.	**PURPOSE/CHANGES**	**AUTHOR (S)**	**DATE**
0.1	Initial ToC	Faisal Ghaffar	01/10/2021
0.2	Finalised ToC	Faisal Ghaffar	19/11/2021
0.3	Partially added section 2.1	Faisal Ghaffar	10/12/2021
0.4	Added section 2.5	Tomasz Mach	07/12/2021
0.5	Added section 2.8	Yannis Filippas	10/12/2021
0.5	Added section 2.3	Gurkan Solmaz	15/12/2021
0.6	Added chapter 3	Giacomo Bernini	22/12/2021
0.7	Added section 2.2	Takai Eddine Kennouche	10/01/2022
0.8	Added section 2.7	Andrea Conti	10/01/2022
0.9	Added section 2.6 to address mid-term comments	Stefano Stracca	12/01/2022
1.0	Added Chapters 1,4. Consolidated all inputs into main document	Faisal Ghaffar	27/01/2022
1.1	Internal Review	Sana Ben Jemaa Gurkam Solmaz	02/02/2022
1.2	Reviews correction	Faisal Ghaffar	02/02/2022
1.3	Final Version	Nicola Blefari Melazzi	03/02/22

INDEX

EXECUTIVE SUMMARY **3**

LIST OF ABBREVIATIONS **6**

TABLE INDEX **8**

FIGURE INDEX **9**

1 INTRODUCTION **11**

1.1 UPDATES FROM PREVIOUS VERSION 12

2 ANALYTICS SERVICES IMPLEMENTATION **16**

2.1 INDIVIDUAL MOBILITY PATTERN DETECTION AND PREDICTION 16

2.1.1 Modelling mobility behaviour 16

2.1.2 Proposed Interaction Module 17

2.1.3 Overall Model Architecture 20

2.1.4 Datasets and Experiments 24

2.1.5 Results and Baseline Comparison 25

2.2 CONTEXTUALIZATION OF LARGE URBAN MOBILITY DATA WITH OPEN STREET MAP ENTITIES 26

2.2.1 OpenStreetMap elements dataset 26

2.2.2 Solution Design and Evaluation 28

2.2.3 Conclusion 31

2.3 GROUP MOBILITY DETECTION AND PREDICTION 31

2.3.1 Dataset Description 32

2.3.2 Implemented Methods 33

2.3.3 Results and Evaluation 34

2.3.4 Application in UC & PoCs 41

2.4 MULTI-MODAL CROWD MOBILITY 41

2.4.1 Dataset Description 42

2.4.2 Implemented Methods 44

2.4.3 Results and Evaluation 45

2.4.4 Application in UC2 46

2.5 ANALYTICS SERVICE FOR VULNERABLE ROAD USER 47

2.5.1 Dataset Description 47

2.5.2 Implemented Methods 48

2.5.3 Results and Evaluation 50

2.6 LOGISTICS IN A SEAPORT TERMINAL USING AGVS 52

2.6.1 Remotely controlled AMR/AGV for logistics 52

2.6.2 Autonomous Mobile Robot 61

2.7 POSITIONING AND FLOW MONITORING FOR CONTROLLING COVID-19 63

2.8 TRANSPORTATION OPTIMIZATION BASED ON THE IDENTIFICATION OF TRAFFIC PROFILES 66
2.8.1 Dataset Description 66
2.8.2 Implemented Methods 67
2.8.3 Results and Evaluation 70
2.8.4 Application in PoCs 73
3 VIRTUALIZATION OF ANALYTICS SERVICES AND PIPELINES 74
3.1 KUBEFLOW PIPELINE: MODEL STORING 77
3.2 KUBEFLOW PIPELINE: SELDON DEPLOYMENT 79
3.3 CONTEXT-AWARE VIRTUALIZED MACHINE LEARNING PIPELINE 82
3.4 NEXT STEPS 83
4 CONCLUSIONS AND FUTURE WORK 84
5 REFERENCES 86

List of Abbreviations

ABBREVIATION	FULL NAME
5G	Fifth generation technology standard for cellular networks
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
API	Application Programming Interface
CC-BY-SA	The Creative Commons Attribution–Share Alike License
CNF	Containerized Network Function
COVID-19	Coronavirus disease 2019
CRUD	Create, Read, Update, Delete
ETSI	European Telecommunications Standards Institute
FR	Functional Requirement
GAN	Generative Adversarial Network
GAT	Graph Attention Network
GCN	Graph Convolutions Network
GRU	Gated Recurrent Unit
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IBA	Intent-Based API
JSON	JavaScript Object Notation
KFN	Kubernetes Network Function
KPI	Key Performance Indicator
LSTM	Long Short-Term Memory
ML	Machine Learning
NFV	Network Function Virtualization
NSD	Network Service Descriptor
NSE	New Services
MANO	Management and Orchestration
MSE	Mean Squared Error
OSM	Open Street Map
PCA	Principal Component Analysis
POI	Point of Interest
RBF	Gaussian Radial Basis Function
RAT	Radio Access Technology

REST	REpresentational State Transfer
SBA	Service Based Architecture
SBI	Service Based Interface
SDK	Software Development Kit
SoTA	State-of-the-art
UC	Use Case
VDU	Virtual Deployment Unit
VNF	Virtual Network Function
VM	Virtual Machine
VRU	Vulnerable Road User
WiFi	Wireless Fidelity
WP	Work Package
WP5	Work Package 5

Table 1: Abbreviation List

Table Index

Table 1: Abbreviation List 7
Table 2 List of datasets used for analytics functions 12
Table 3 ADE and FDE metrics for several models that are similar to ours 25
Table 4 OSM Node fields 27
Table 5 A view on the wireless dataset pre-processed for enabling the application of weak supervision. The labels at the end column represents if person_x and person_y belong to same group. 34
Table 6 Summary of the algorithms used in the experimental evaluation (from [16]). 36
Table 7 The neural network architecture with the best performance for crowd estimation [21]. 45
Table 8 The performance of CountMeIn, along with the raw Wi-Fi PP count performance and PFA [22] performance. 45
Table 9 Vehicle Navigation Combination 54
Table 10 Sensors characteristics 61
Table 11 Examples of data bandwidth of provided by common sensors for robotic localization and navigation. Data can change according to several factors, such as frequency, resolution, etc 62
Table 12 Results in ETH/UCY dataset 71
Table 13 Results in Incelligent simulated dataset 71
Table 14 Computational Comparison of the three models 72
Table 15 Classification results for trajectory profiling 73

Figure Index

Figure 1 Interaction Module 17
Figure 2 Overall Model Architecture 20
Figure 3: A dilated causal convolution with dilation factors d = 1, 2, 4 and filter size k = 3. (source: [10]). 23
Figure 4: Tokyo OSM data from Geofabrik[14] 27
Figure 5: Trajectory Context Information in OSM 28
Figure 6: Pipeline with OSM context creation 29
Figure 7 ADE 30
Figure 8 FDE 30
Figure 9 . Experimental setup and basic setup in the office environment 32
Figure 10 Group-In multi-phased pipeline approach with pre-processing, centralized/decentralized computing, and clustering. 33
Figure 11 Result of the pre-processing phase: Top: Before pre-processing, bottom: after pre-processing 35
Figure 12 Group monitoring performance using different trajectory matching algorithms (from [16]). 35
Figure 13 . Experimental results for controlled and real-world office scenario setups. Using the centralized computing approach 37
Figure 14 Group monitoring performance of Group-In based on group inter-distances 38
Figure 15 Group monitoring performance of Group-In for the “random-walk” scenario. Top: centralized algorithms, bottom: decentralized algorithms. 39
Figure 16 People counts and number of peoples observed every 10 minutes based on wireless activities in a city [18]. 40
Figure 17 Group-In Web dashboard showcasing the crowd mobility analytics results 41
Figure 18 CountMeIn approach of learning correlations between Wi-Fi and auxiliary sensor and applying calibration in the long-term crowd estimation period. [21] 42
Figure 19 The analytics method of correlation of Wi-Fi and auxiliary sensors for learning and applying in the long-term in smart cities [21]. 44
Figure 20 Comparison of error values (Red = RMSE, Blue = MAD) with raw Wi-Fi PP count, the two state-of-the-art approaches CrowdEstimator [23], PFA [22], and CountMeIn [21]. 46
Figure 21 An Overview of the Shibuya Scramble Crossing (Google Earth) 48
Figure 22 : VRUs at crossing in urban intersection (adopted from ETSI TS 103 300-1 [27]) 49
Figure 23 VAM messaging traffic in VRU clustering 50
Figure 24 VRU clustering statistics for varying number of users 51
Figure 25 Vehicle (A=vehicle reference point, B=symmetry axis). 54
Figure 26 (Left) Infectious evolution for Lombardy - (Right) infectious evolution for Emilia Romagna 65
Figure 27 (Left) Hospitalized evolution for Lombardy - (Right) Hospitalized evolution for Emilia-Romagna 65
Figure 28 Snapshot from the Incelligent simulator with the mobility areas (roads, parking lots, shops) 67
Figure 29 Social GAN architecture 68
Figure 30 Trajectory Transformer Architecture 69
Figure 31 GRU with Attention Architecture 70
Figure 32 Results in ETH/UCY dataset 71
Figure 33 Results in Incelligent simulated dataset 72
Figure 34 Solution integration flow diagram for PoC3 73
Figure 35 Reference Machine Learning pipeline 74
Figure 36 Microk8s - Kubeflow environment 75
Figure 37 Fully virtualized machine learning pipeline 76
Figure 38 : Create a reusable Kubeflow pipeline component 77
Figure 39 Seldon Graph deployment file 78
Figure 40 Non-reusable Model Servers 78
Figure 41 Reusable Model Servers 79
Figure 42 Initialization function for reusable Model Server 79
Figure 43 Seldon graph deployment code 80
Figure 44 UC1 Functionality-1 Kubeflow Pipeline execution 81
Figure 45 Seldon inference graph deployed in Kubernetes cluster 81
Figure 46 Seldon Graph prediction 81

Figure 47 Context-Aware virtualized machine learning pipeline 82
Figure 48 Continuous Models Integration architecture..... 83

1 Introduction

This report is the final update of the two parts report for Task5.1 and provides the updates for the analytic functions and services developed for LOCUS use-cases (UCs). This deliverable was due in December 2021 but was delayed due impact of the COVID on involved persons.

The first deliverable (D5.1) listed analytics functions per use-case, summarised state-of-the-art in the relevant domain for each use-case, and provided an initial inventory of possible data sources to be used to implement analytic functions. The first deliverable provided the initial plan for experimentation of data-drive techniques to detect mobility patterns and predict future steps from the data that captures spatial and temporal aspects of objects under investigation. The key aspect of D5.1 was that adopted approach targeted the development of models that can detect mobility patterns from spatial-temporal data of any size while satisfying the requirements from multiple UCs. For example, one of the analytic functions implemented is human trajectory prediction which can provide insights as input to other functions across multiple UCs such as for flow monitoring in urban environment (NSE-UC1), crowd-mobility (NSE-UC2), and transportation optimisation (NSE-UC5), etc. This approach of unifying functions across multiple UCs where output of one ML model can serve multiple scenarios, has been continued in current version. However, the emphasis has been given to more technical details in the current report instead of describing the UCs, the analytics functions, and the relevant stat-of-the-art. To read more on this, reader is referred to D5.1.

The focus of the document is on technical details of the ML techniques/functions and extending them for generalisation, better performance in terms of inference speed, and the improved accuracy. In particular, human trajectory prediction models for NSE-UC1 and NSE-UC5 have been extended from simple RNN based implementation (in D5.1) to more advanced models such as generative adversarial networks, transformers, graph convolution networks, and the graph attention networks. A key aspect in the extended models for human trajectory predictions is the addition of contextual information (e.g., shops, roads, streets) as well as surrounding of pedestrians (e.g., other pedestrians in urban environment) while forecasting future trajectories. Graph based methods enable capturing of interactions among pedestrians as these interactions influence the trajectories as human walk in a dense urban environment. This all is described in detail in section 2.1.2 and in section 2.1.3.1.

In addition to interactions mong pedestrians, contextual information is infused into encoder-decoder architecture for trajectory prediction. This contextual information is acquired from Open Streep Map (OSM). The experiments show the improvement in predicting trajectories with such contextual information (see section 2.2 for details).

From crowd mobility prediction perspective, in addition to “GroupIn” solution described in D5.1, in this deliverable, a new solution called “CountMeIn” is developed. The CountMeIn is a wireless scanning system with auxiliary sensors (i.e., stereoscopic people counting cameras) to detect static or mobile people groups in indoor or outdoor environment (see section 2.4 for more details).

The section 2.5 of this report describes the methods developed for evaluating the performance of grouping pedestrians at an urban intersection and its impact on reducing the communication overhead for doing the VRU awareness message exchanges.

Finally, chapter 3 of the report provides advancements in virtualisation of ML pipelines for the NSE UCs. This work has a separate strand as per task 5.2 and 5.3; analytics functions and ML operations used to develop those functions form the basis of virtualisation work. The chapter summarises analytics functionality as ML pipeline, generates it in Kubeflow, and deploys on a server using various deployment technologies scubas Kubernetes along with Seldon for ML models management. More details on virtualisation platform can be found in D5.3 where final deployment of virtualised analytics services will be reported in the final deliverable (D5.4) from WP5 in July of 2022.

1.1 Updates from previous version

This report provides the following additions and deviations from its previous version (D5.1), addressing reviewers' comments on D5.1 from the mid-term review:

- Added state-of-the-art for the UC4 in section 2.6 as per reviewers' comments during the mid-term review.
- UCs names are revised to address the discrepancies in names between deliverable D5.1 and D2.1. The revised UCs names are:
    1. NSE-UC1 - Flow monitoring and management in large venues and dense urban environment
    2. NSE-UC2 - Crowd mobility analytics using mobile sensing and auxiliary sensors
    3. NSE-UC3- Vulnerable Road user
    4. NSE-UC4: Logistics in a seaport terminal using Automated Guided Vehicle
    5. NSE-UC5 - Transportation optimization based on the identification of traffic profiles
    6. NSE-UC6: Positioning and Flow Monitoring for Controlling COVID-19
- A table with list of datasets used to develop ML models is provided as per comment during the mid-term review

Table 2 List of datasets used for analytics functions

Name	Description	Used for Analytics service
ETH[1] [1]	Public trajectories of humans extracted from image frames	(1) Trajectory Prediction (NSE-UC1) (2) Trajectory Collision for VRU UC (NSE-UC3)

[1] ETH, https://data.vision.ee.ethz.ch/cvl/aess/dataset/

		(3) Trajectory profiling (NSE-UC5) (4) Transportation optimization based on identification of traffic profiles (NSE-UC5)
UCY[2] [2]	The UCY dataset consist of real pedestrian trajectories with rich multi-human interaction scenarios	(1) Trajectory Prediction (NSE-UC1) (2) Trajectory Collision for VRU UC (NSE-UC3) (3) Trajectory profiling (NSE-UC5) (4) Transportation optimization based on identification of traffic profiles (NSE-UC5)
OpenPFLOW[3]	The PFLOW dataset is an aggregation of person trip surveys from 25 metropolitan areas in Japan stored on a minute-by-minute basis in a spatio-temporal database with tracking id, latitude, longitude, time stamps, transportation mode, and magnification factor. This is a huge dataset that contains more than 660 million data records	(1) Trajectory Prediction (NSE-UC1) (2) Trajectory Collision for VRU UC (NSE-UC3) (3) Trajectory profiling (NSE-UC5) (4) Transportation optimization based on identification of traffic profiles (NSE-UC5
WiFi crowd-sourced Fingerprinting[4] [3]	This dataset covers three buildings of Universitat Jaume I with 4 or more floors and almost 110.000m2. It can be used for classification, e.g. actual building and floor identification, or regression, e.g. actual longitude and latitude estimation. It was created in 2013 by means of more than 20 different	(1) Trajectory Prediction (NSE-UC1) (2) Trajectory collision

[2] UCY https://graphics.cs.ucy.ac.cy/research/downloads/crowd-data
[3] OpenPFLOW, https://githubplus.com/sekilab/OpenPFLOW
[4] https://archive.ics.uci.edu/ml/datasets/ujiindoorloc

	users and 25 Android devices. The database consists of 19937 training/reference records (trainingData.csv file) and 1111 validation/test records (validationData.csv file). The 529 attributes contain the WiFi fingerprint, the coordinates where it was taken, and other useful information.	
NEC simulated WiFi Fingerprinting	This dataset consists of wireless traces from the Bluetooth-enabled mobile devices (see section 2.3.1) and people counting data from WiFi scanners and stereoscopic cameras (section 2.4.1)	(1) GroupIn - Crowd mobility pattern detection (NSE-UC2) (2) CountMeIn - Crowd mobility pattern detection (NSE-UC2) (3)
Population Flow in Italy [4]	This is a human mobility dataset consisting of triples having the following structure: source province, destination province, and a time series reporting the percentage of population moving between source and destination provinces at each day	(1) COVID-19 spread monitoring (NSE-UC6)
Mobile operator data	This is an aggregated contagion data per cell of approximately 5000 users. This contains total number of detected cases up to March 23, 2020. The time span covers cases that most likely originated before the confinement (March 13, 2020), as a direct consequence of the mobility of previous days	(1) COVID-19 spread monitoring (NSE-UC6)
Incelligent simulated outdoor location data (see section 2.8.1)	containing both pedestrian and car trajectories, in a realistic mixed indoor-outdoor mobility scenario taking place in McArthurGlen shopping center in Athens. The simulation produces realistic mobility patterns by utilizing a probabilistic model for the connection of the	(1) Trajectory profiling (NSE-UC5) (2) Transportation optimization based on identification of traffic profiles (NSE-UC5)

	different areas of the scene and generates trajectories of stochastic velocity and movement.	
ETSI ITS VRU standard based simulation by Samsung	Dataset simulates the real dataset generated from the Shibuya Pedestrian crossing in Tokyo, Japan, with a square of size 70 m	(1) Pedestrian clustering for VRU (NSE-UC3)

2 Analytics Services Implementation

This section elaborates on the analytics functions that are developed as part of WP5 task5.1 and form the basis of virtualised analytics functions for the localization.

2.1 Individual mobility pattern detection and prediction

Location based analytics blends data from different sources with geo-spatial locations (geographic data) to provide context and derive valuable insights, and in turn helps at better decision making for businesses. Location analytics has been one of the fastest growing fields in recent times. Many businesses have successfully embraced location analytics to create ever more sophisticated and pragmatic scenarios. Some of the benefits of using location analytics in business scenarios include optimizing the business use cases based on location intelligence, effective customer targeting, increased marketing relevancy, long-term strategic and cost-effective planning, better marketing territory management, and even to create contingency plans to cope with the arrival of other business competitors.

LOCUS aims to offer localization, together with analytics, and their combined provision "as a service". The various functionalities developed in WP5 (detailed in the following sections) along with their localization analytics will be exposed to 3rd party verticals as analytic services. These LOCUS services can be leveraged to suit several business applications. For instance, the location intelligence derived from 'the crowd flow monitoring service' combined with real-time analytics can be leveraged in a retail application to understand visitor journeys i.e. to visualize how customers interact with a venue at a given time using real-time heat maps and dashboards, to serve customer targeted advertisements, to analyse customer loyalty and engagement to boost customer retention and conversion. If the locations are geo-fenced this service can be used to monitor the crowd movement, operations, and resources for surveillance purposes.

2.1.1 Modelling mobility behaviour

The forecasting of accurate trajectories of humans and moving objects is a crucial component for a variety of applications. Autonomous vehicles such as self-driving cars, and robotic delivery vehicles need to understand human movement to avoid collisions. Smart cities deployed intelligent tracking systems to understand the crowd behaviour for city planning and management of infrastructure. For all these systems, trajectory prediction is a crucial, at the same a challenging task due to human behaviour. Human trajectories in dense urban areas, are affected by their interactions with other humans and physical objects around them. There are several ways in which human interact while on move, from taking actions, avoiding collisions, to walking in groups. From the surroundings perspective, not only physical objects such as buildings, lampposts, etc. direct human movement, but visual scene such as sidewalks or grass may enable or restrict human movement. It is the human -to-human social interactions aspect that is focused on towards accurate forecasting of human trajectories as well as trajectories of crowds. Human trajectory forecasting is particularly challenging due to the social behaviour factors such as walking in parallel with others, within groups, avoiding collisions, and merging

from various directions. Understanding and factoring in these behaviours can improve the trajectory forecasting for not only individuals, but also for crows and can further be leveraged in emergency situations such as trajectory collision detection.

2.1.2 Proposed Interaction Module

Humans are well capable of navigating through complex and crowded urban environments by following unspoken social rules which results in social interactions. These interactions can be captured with data-driven encoders. The key aspects in designing interaction module are:

1. Relative spatial positions, velocities, and historical trajectory states of neighbours with respect to a primary pedestrian whose trajectory is being predicted
2. An embedding technique such as LSTM to capture and represent as vectors, the evolution of trajectories or hidden states over time.
3. An aggregation strategy to aggregate information of all the neighbouring trajectories. For example, Max-pooling, Attention, or Concatenation. A baseline aggregation method can simply concatenate the neighbourhood embeddings Recent studies have leveraged attention mechanisms [5] to determine the weights of different neighbours in predicting the future trajectories.
4. An embedding (e.g., MLP or LSTM) for the aggregated neighbour vector

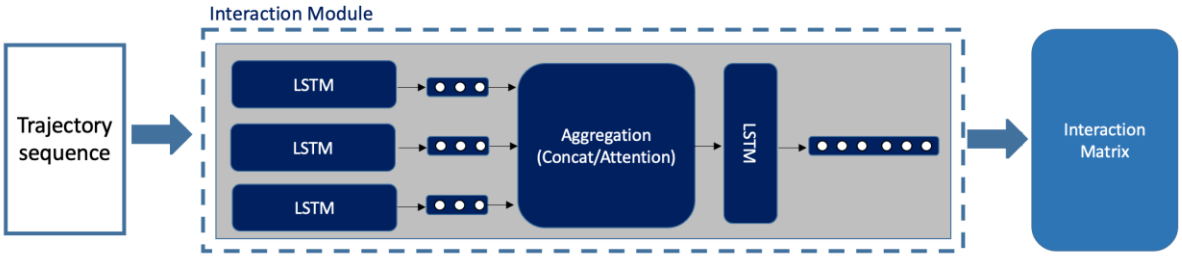

Figure 1 Interaction Module

2.1.2.1 Interactions Embedding

In this section, we discuss a kernel function that embeds the social interactions between pedestrians in a matrix. Kernel functions in general, provide method of representing a data sample through a set of pairwise comparisons. Mathematically, given a data set $X = x_1, x_2, \dots x_n$, a real valued "comparison function" $k = X \times X \rightarrow \mathbb{R}$ is used to represent data set X in form of $n \times n$ square matrix of pairwise comparisons $k_{ij} = k(x_i, x_j)$. Such representation of data in square matrix has several advantages.

First, ML models generally are not independent of the nature of data they are trained on. For instance, images, sets, or sequences can always be transformed into real-valued square matrix before input to any algorithm. Furthermore, this kernel mapping provides modularity by decoupling the data representation from the design of algorithm.

Second, size is fixed. The matrix used to represent a dataset of n objects are always $n \times n$, whatever the nature or the complexity of the objects.

Third, some data structures (e.g., graphs and protein sequences) cannot be explicitly represented as vectors to further be used in a neural network algorithm. Such data can only be represented through some meaningful pairwise comparison methods. Spatial sequences pedestrians in a crowd where movements are influenced by other humans is one such form of data which requires explicit representation in the form of spatial-temporal comparison of trajectories.

2.1.2.2 Graph kernel as measure of similarity

A kernel k represents each object $\boldsymbol{x}$ in space $\mathcal{X}$ as a vector $\phi(\boldsymbol{x}) \in \mathcal{F}$ in another space $\mathcal{F}$ known as *feature space and* computes their dot products. i.e.

$$k(\mathbf{x_i}, \mathbf{x_j}) = \langle \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) \rangle, \text{ for any } \mathrm{x_i}, \mathrm{x_j} \in \mathcal{X} \qquad 2.1$$

Kernels differ from explicit vector representation of objects in that, the mapping $\phi(\boldsymbol{x})$ does not require explicit computation for each point in the dataset, since the pairwise dot products are computed. One intuition is of using kernels as measures of similarity in the sense that $k(\mathbf{x_i}, \mathbf{x_j})$ is "large" when $\mathbf{x_i}$ and $\mathbf{x_j}$ are "similar" in some context. One notion of similarity is the distance and simple dot product $\langle \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) \rangle$ does not straightforwardly fit for this intuition. However, there are some tricks that can close the gap between dot product and distance as a notion of similarity.

The discussion of using distance as notion of similarity is important in trajectory mapping of pedestrians in the presence of other pedestrians. The kernel methods can be used to determine the similarity between pedestrians by calculating distance between them and solving these kernels via dot product to achieve efficiency in computation. We will illustrate this point in following discussion.

Consider two objects $\boldsymbol{x}_1, \boldsymbol{x}_2$ representing coordinates of two pedestrians. These two objects are mapped to two vectors $\phi(\mathbf{x_i})$ and $\phi(\mathbf{x_j})$ in feature space $\mathcal{F}$. The distance between them is defiend as Hilbert distance $d(\mathbf{x}_i, \mathbf{x}_j) \coloneqq \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|$. To express this distance into dot product simply equallty can be exploited as:

$$\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 = \phi(\mathrm{x_i})^{\mathsf{T}}\phi(\mathrm{x_i}) + \phi(\mathrm{x_j})^{\mathsf{T}}\phi(\mathrm{x_j}) - 2\phi(\mathrm{x_i})^{\mathsf{T}}\phi(\mathrm{x_j})$$

Now the distance becomes:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_j) - 2k(\mathbf{x}_i, \mathbf{x}_j)} \qquad 2.2$$

Where k is the kernel function (a dot product in the feature space) as defined in equation 2.1.

In human trajectory mapping, one objective of kernel function is to construct a weighted adjacency matrix where elements of matrix represent the similarity between pedestrians. The dot-product based distance measure defined in equation 2.1, is one straightforward way to model the impact of pedestrians on each other, but this is against the institution that larger the distance between pedestrians, less the influence. One simple fix to this is to use the inverse of distance measure.
Another function that can be considered more appropriate as a kernel function for trajectory mapping task is Gaussian Radial Basis Function (RBF). The RBF kernel on $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X} = \mathbb{R}^n$ is given as:

$$k_{RBF}\left(\mathbf{x_i}, \mathbf{x_j}\right) = \exp\left(-\frac{\mathrm{d}\left(\mathbf{x_i}, \mathbf{x_j}\right)^2}{2\sigma^2}\right) \qquad 2.3$$

Where σ is learnable parameter and d is the Euclidean distance. The RBF kernel can be written as dot product as $k_{RBF} = \langle \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) \rangle$.

From the equation 2.3, note that RBF kernel is a decreasing function the Euclidean distance between points, and therefore has a relevant interpretation (in case of trajectory prediction with social interactions) as measure of similarity: the larger the k_{RBF}, closer the points $\mathbf{x}_i, \mathbf{x}_j$ in feature space $\mathcal{X}$. Using equation 2.1, kernel can be represented in Hilbert space as:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\| \phi(\mathbf{x_i}) \|^2 + \| \phi(\mathbf{x_j}) \|^2 - d\left(\phi(\mathbf{x_j}), \phi(\boldsymbol{x}_j)\right)^2}{2} \qquad 2.4$$

Where d is Hilbert distance and $||.||$ is Hilbert norm ($||u||^2 = \langle u, u \rangle$). This kernel measures the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ as the opposite of the square distance $\mathrm{d}\left(\phi(\mathbf{x_j}), \phi(\mathbf{x_j})\right)^2$ between their representation in the feature space, upto the terms $\| \phi(\mathbf{x_i}) \|^2$ and $\| \phi(\mathbf{x_j}) \|^2$.
This Gaussian RBF kernel forms the basis of our data analysis for human trajectory as this RBF kernel can replace computationally expensive dot-products for nonlinearity and be used as measure of similarity between various objects. Second advantage of defining a general kernel such as RBF is that many classic algorithms that are trained on vectorized data can be applied to other data structures such as graphs without explicitly representing them in vectors.

The distinction between different works in the literatures is in the architectures that used for the embeddings and aggregation of neighbourhood that influence future trajectory of a primary pedestrian. For example, [6] applied a symmetric max-pooling on the LSTM based hidden states of other users in the surrounding whereas [7] and [8] utilised sum-pooling for the aggregation. The common theme among these works is that they aggregate and embed information form the neighbours that are implicit, i.e. consider a geolocation grid with one cell occupied by the primary pedestrian. Then these techniques will consider all of the neighbours that occupy other cells in that grid. However, considering all the neighbours can lead to noise and can deteriorate the performance of the forecasting model. Therefore, it is crucial to design interaction encoding in such a way that minimize the noise and captures the essential neighbouring information. One simple technique is that, for the forecasting trajectory we can

consider only top-k neighbours that influence the most or have the highest interaction with the primary pedestrian. Furthermore, majority of the social pooling techniques with only positional configurations are not able to handle the trajectory collision scenario. Humans are well able to adjust their velocity relative the positions of the others in a crowd to avoid collisions. Most of the studies on human trajectory predictions do not factor in relative velocity of neighbours. In this work, we propose to concatenate relative velocity of neighbours with relative positions. By adding this information interaction module can learn the notion of collision avoidance implicitly.

2.1.3 Overall Model Architecture

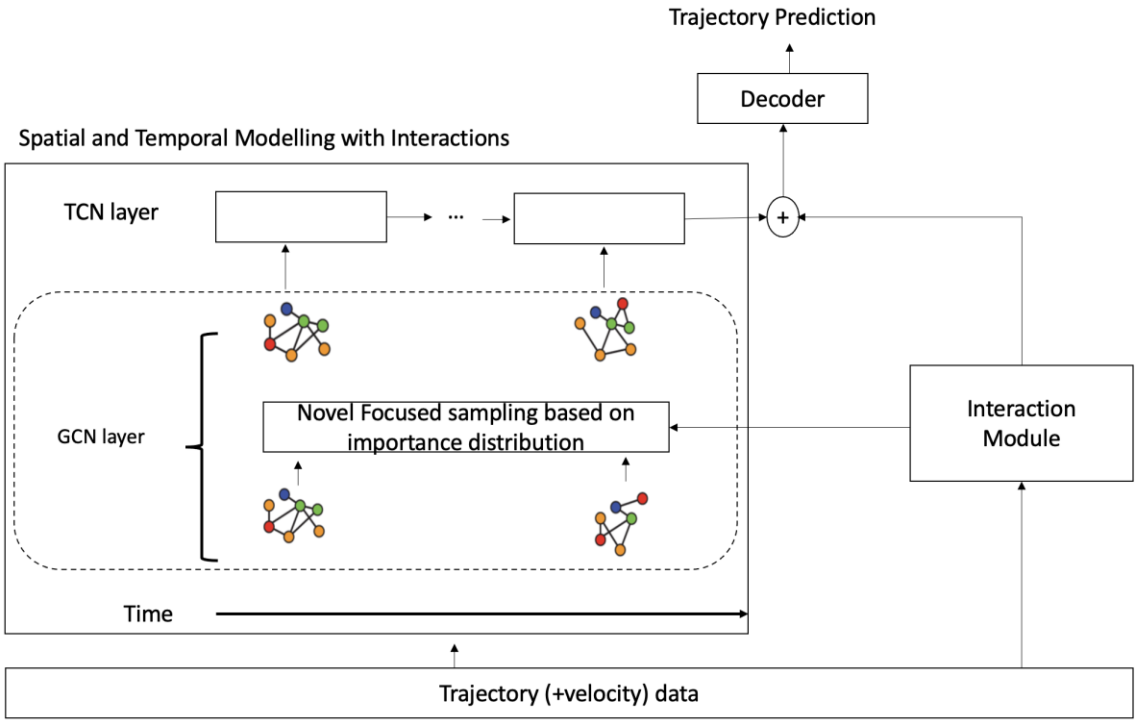

Figure 2 Overall Model Architecture

We formulate the trajectory prediction as a learning task over a graph where each pedestrian at any time t is represented as node v_i^t and two interacting pedestrians are connected via an edge e_{ij}^t in a graph $g_t = (V_i^t, E_{ij}^t)$, where V_i^t is set of nodes and E_{ij}^t is set of edges. The v_i models the sequence representations of the associated pedestrian and edge e_{ij} updates according to the interactions between associated pedestrians. The weights of edges in set E_{ij}^t are determined by the RBF kernel function discussed in section 2.1.2.2. Kernel function determines the similarity between pedestrians as inverse of the distance measured by the L_2 norm in Euclidean space. i.e., $\| v_i^t - v_j^t \|_2$

This can also be replaced by RBF kernel function given in equation 2.42.3 , which can be express in the node vector forms as:

$$\frac{\| v_i^t \|^2 + \| v_j^t \|^2 - \left(\| v_i^t - v_j^t \|_2 \right)^2}{2} \qquad 2.5$$

We will demonstrate performance with both measures. One of the key distinctions between our work and the work of others is that we construct graph from the kernel function. This decouples the interaction module from the rest of the forecasting model and allows to encode different methods to capture social interactions between pedestrians.

2.1.3.1 Spatial Encoding (GCN Layer)

Once graph is constructed, spatial convolution operation is used to aggregate the neighbouring information. Recently, generalizing the CNN to the graph convolutional network (GCN), which can handle arbitrary graph-structured data, has received widespread attention due to its flexibility to handle changes in the underlying graph. The GCN model constructs a filter in the Fourier domain, the filter acts on the nodes of graph and its first-order (denoted by node's degree D) neighbour to capture spatial features between the nodes, and then the GCN model can be built by stacking multiple convolutional layers. Furthermore, GCN layers can be configured for higher order degrees to aggregate information from nodes that are more than one hop away from the primary pedestrian. In our experiments, we observed higher order introduces noise and deteriorate the performance of the trajectory prediction. This is intuitive as other pedestrians that are closer to primary influence have the highest influence on future trajectory.

The details of the Spatial-GCN are given as follows:
In a convolution's operator on a feature map, the output feature map is size as of input feature map with stride 1 and appropriate padding. Now given a convolution of operator with the kernel size $k \times k$ and input feature map f_{in} with number of channels c, the output value for a single channel at spatial location x can be written as:

$$f_{out}(x) = \sigma\left(\sum_{h=1}^{K} \sum_{w=1}^{K} f_{in}\big(\boldsymbol{p}(\boldsymbol{x}, h, w)\big) \cdot \boldsymbol{w}(h, w) \right) \qquad 2.6$$

Where, σ is the activation function and $\boldsymbol{p}$ is sampling function which enumerates neighbours of location x. In case of an image it enumerates over neighbouring pixels, but in our trajectory graph g_t, sampling function is the set of neighbours of the node v_i^t. This set can be fined as a shortest path in the graph that connects two nodes, i.e., $\mathcal{N}(v_i^t) = \{v_j^t | d(v_i^t, v_j^t) \leq D\}$, where $d(v_i^t, v_j^t)$ is the length of shortest path from v_i^t to v_j^t at time t.

The **w** is weight function and provides a weight vector that is used to compute the inner product of input feature vectors. In a 2D grid data such as image, cells or pixels) have a natural spatial order. The weight function in grid data can be implemented by indexing a tensor of (c, K, K) dimensions according to the spatial order. However, in graphs there is no spatial order for nodes which makes defining weight functions trickier. To assign spatial ordering to neighbours of a node in a graph, following three strategies have been proposed in the literature:

- **Uniform ordering** – The simplest strategy is to assign orders is to assign same label to whole set of neighbours of a node.
- **Label based ordering** - To define ordering for the node, one solution proposed by [9] is a labelling process in the ego network for the root node. Similar idea can be extended in our case and instead of assigning unique label to each node, the neighbouring set $\mathcal{N}(v_i^t)$ of a node can be partitioned into a fixed number of subsets and each subset is then assigned a label.
- **Distance based ordering** – another natural way to order nodes in the graph is based on their distance from the root node. In trajectory prediction task root node is the primary pedestrian under investigation.

In our implementation, we adopt uniform and distance-based ordering strategy for the neighbours of a node. With this, we have a mapping for neighbours of the node. We denote mapping as $l_i^t: \mathcal{N}_i^t \rightarrow \{0, \dots, K\}$. Now the refined weight function $\boldsymbol{w}$ by indexting a tensor of (c, K) dimensions.

With sampling and weigh function refined for the pedestrian graphs, we refine convolution operator defined in equation 2.6 over graph as:

$$f_{out}(v_i^t) = \sigma\left(\sum_{v_i^t \in \mathcal{N}_i^t(v_i^t)} \frac{1}{\Omega} f_{in}\left(\boldsymbol{p}(\boldsymbol{v_i^t}, v_j^t)\right) \cdot \boldsymbol{w}(\boldsymbol{v_i^t}, v_j^t)\right) \qquad 2.7$$

Where $\frac{1}{\Omega}$ is normalizing term and equals to the cardinality of corresponding neighbouring set of the node v_i^t.

2.1.3.2 Temporal Encoding (TCN Layer)

In our implementations, we choose Temporal Convolution Networks (TCNs) as time-sequence encoder. This can also be replaced by a simple LSTM. The reason of our preference of TCNs over LSTM is simply due to the factors that TCNs require less memory for training and have demonstrated stable gradients during backpropagation addressing the vanishing or exploding gradient issues seen with RNNs. Furthermore, TCNs do not accumulate the sequential predictions errors like the RNNs do.

From the architecture perspective, TCNs include CNN and RNNS blocks and can take a sequence of a length as input and produces a sequence of the same length. Therefore, TCN uses a 1D fully-convolutional network (FCN) architecture, where each hidden layer is the same length as the input layer, and zero padding of length (kernel size − 1) is added to keep subsequent layers the same length as previous ones. In order to keep the size of output sequence equal to input sequence, the TCN uses *causal convolutions*, convolutions where output at time t is convolved only with elements from time t and earlier in the previous layer. In particularly, a simple casual convolution has fixed size look back linearly saleable with the depth of the

network. In order to increase the look back exponentially, [10] proposed a dilated convolution to enables an output at the top level to represent a wider range of inputs, thus effectively expanding the receptive field of a CNN.

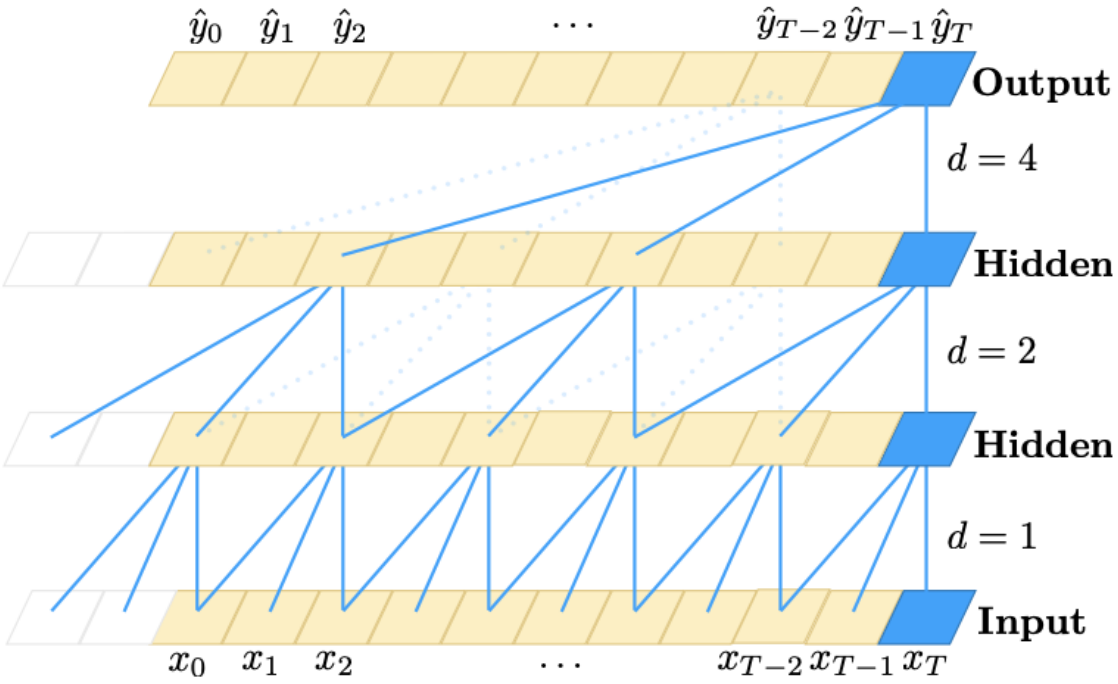

Figure 3: A dilated causal convolution with dilation factors d = 1, 2, 4 and filter size k = 3. (source: [10]).

One major difference in our model and the state-of-the-art is that we represent pedestrians state using their velocities as opposed to absolute coordinates. Switching to velocities increases the generalization power and the architecture can be used for trajectory collision detection.

2.1.3.3 Validation and Evaluation Metrics

Given a set of N pedestrians and their corresponding observed coordinates over a time period T_{obs}, we predict the upcoming trajectories over future time horizon T_{pred} for pedestrian i with his/her trajectories encoded as v_i^t.
The training of the forecasting model aims to minimize negative log-likelihood loss given as:

$$\mathcal{L}_i(w) = -\sum_{t=T_{obs}+1}^{T_{pred}} \log\Big(P(\boldsymbol{v_i^t} \mid \mu_i^t, \sigma_i^t, \rho_i^t)\Big) \qquad 2.8$$

Where w includes all the trainable parameters of the model, μ_i^t is the mean distribution, σ_i^t is the variances, and ρ_i^t is the correlation.

During test time, till time-step period T_{obs},, we provide the ground truth position of all the pedestrians as input to the forecasting model. From time T_{obs+1}, to T_{pred}, we use the predicted velocities (derived from the predicted positions) of each pedestrian as input to the forecasting model and predict the future trajectories of all the pedestrians.

For the evaluation on testing sample, the most commonly used metrics in human trajectory forecasting are:

- Average Displacement Error (ADE) - an average L_2 distance between ground truth and model prediction over the prediction time period.
- Final Displacement Error (FDE) – as the name suggests, this is distance between final position in ground truth and the predicted final placement at the end of the prediction period.

Both measures are unimodal, means they predict next step given a set of previous observations. In our implementation, we output future trajectory as bi-variate Gaussian distributions and ADE and FDE in our case are top-k predicted distribution closest to the ground-truth. More specifically, we use a range (3-20) sample from the predicted distributions as top-k and ADE is the average distance between predicted distributions and the ground-truth. Similarly, FDE averages the distance between final position in the sample predicted distributions and the final position in ground-truth.

$$\text{ADE} = \frac{\sum_{n \in N} \sum_{t \in T_p} \| \widehat{p_t^n} - p_t^n \|_2}{N \times T_p} \qquad 2.9$$

$$\text{FDE} = \frac{\sum_{n \in N} \| \widehat{p_t^n} - p_t^n \|_2}{N}, t = T_p \qquad 2.10$$

The ADE and FDE metrics evaluate models based on distance between predicted and ground-truth. However, these metrics do not evaluate collision detection. For collision detection, we follow the work of [11] to detect the direction of the closest neighbour and measure the angle between neighbour and the primary pedestrian. A neighbour within the angular range of $[180 \pm 15]$ degree is classified as *colliding trajectory*.

2.1.4 Datasets and Experiments

One of the key requirements for the models to detect trajectory collision or provide accurate human movement prediction, is to train them on datasets that have trajectories with interactions among agents. For this purpose, the experiments were conducted on open-source datasets ETH [1] and UCY [2]. Both datasets contain annotated trajectories of pedestrians that are socially interacting with each other. The trajectories are extracted from video frames and sampled every 0.4 seconds totalling in 650 tracks over a period of 25 minutes. The UCY dataset consist of real pedestrian trajectories (composed in three sequences Zara1, Zara2, and UCY) with rich multi-human interaction scenarios captured at 0.4 seconds from the top view. The

goal of models is to predict 3161 human trajectories, observing for each trajectory 8 consecutive ground-truth values (3.2 seconds) i.e., $t-7, t-6, \dots, t$, in 2D plane and forecasting the following 12 (4.8 seconds), i.e., $t+1, \dots, t+12$. For training configuration, we used a batch size of 128 and stochastic gradient descent as learning algorithm. We trained models on 150 epochs with an initial learning rate of 0.01.

2.1.5 Results and Baseline Comparison

Following the norms in human trajectory forecasting work, ADE and FDE are used to evaluate the performance of model quantitively. From the results of experiments, our model competes with the state-of-the-art when compared in isolation based on the ADE/FDE performance and outperforms most of the SoTA on average measures. In the FDE metric, our model performed consistently better than SoTA due to the interaction module defined in equation 2.6. The interaction module considers the similarity as inverse of the distance in the feature space. From the Table 3, it can be observed that our model's performance is not far from LSTM model. This is due the reason that LSTM is used as encoder in the interaction module. Apart from the performance, our model has few parameters when compared to LSTM.

Table 3 ADE and FDE metrics for several models that are similar to ours

Model (with interactions)	**ETH**		**HOTEL**		**UNIV**		**ZARA1**		**ZARA2**		**AVG**	
	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
LSTM [35]	0.71	1.25	0.37	0.74	0.51	1.10	0.41	0.9	0.32	0.7	0.45	0.94
GAT [38]	0.69	1.29	0.49	1.01	0.55	1.32	**0.30**	0.62	0.36	0.75	0.48	1.0
GAN [6]	0.87	1.62	0.67	1.37	0.76	1.52	0.35	0.68	0.42	0.84	0.61	1.21
GCN [37]	**0.64**	**1.11**	0.49	0.85	**0.44**	**0.79**	0.34	0.53	**0.30**	**0.48**	**0.44**	**0.75**
Ours	0.75	1.16	**0.36**	**0.72**	0.48	1.05	0.33	**0.49**	0.31	0.55	0.45	0.87

From the inference and model size perspective, our model has considerably lower number of parameters with 7.9K parameters when compared to SoTA such as LSTM which has 246K and social-GAN which has 46.3K parameters. In terms of inference speed, GCN was previously the fastest method with an inference time of 0.0.002 seconds per inference step. The inference time of GAN is 0.0.098 seconds per inference step. Our model falls in the middle with inference of 0.006 seconds per step.

From qualitative perspective, our model behaviour is dependent on kernel function used for the interaction module. With L2 norm kernel function which calculates the score based on the distance between pedestrian vectors in the feature space, predicted trajectories are not collision free. To avoid the collision, an RBF kernel, or the inverse of the L2 norm distance as kernel function is used which gives higher interaction score to a pair of pedestrians if they are close to each other and from the output we observed that this avoids the collision in trajectories. This is also the case for parallel walking, with RBF kernel pedestrians can maintain a social distance among them while walking in the same direction.

In summary, our model showed graph convolution layer in conjunction with TCN layer can improve trajectory prediction performance from the ADE and FDE metrics perspective. Furthermore, graph convolution and TCN layers reduced the size of the model and improved inference. For the interactions between pedestrians, RBF or inverse of L2 norm distance improved the predictions for collision avoidance and parallel walking.

2.2 Contextualization of Large Urban Mobility Data with Open Street Map Entities

2.2.1 OpenStreetMap elements dataset

Open Street Map (OSM) [12] is a collaborative project that aims at creating free and accessible maps and cartography data of the world. It relies on crowd-sourced data by volunteers and releases the generated mapping data mostly under the Creative Commons Attribution–Share Alike License (CC-BY-SA) license.
OSM provides a wealth of information regarding Physical entities, including their locations, names, addresses, etc.
All physical entities in OSM are represented in OSM as elements of three types [12]:

- *Nodes:* To define points in space. A single node defines a single point in space with latitude, longitude, and a node ID (See Table 4).
- Ways: To define linear features, such as roads, walls, rivers, and boundaries. A way is an ordered list of nodes. It can be closed to define features such as contours and areas, or open to define features such as roads and paths.
- Relations: To define complex entities that are formed by combination of other OSM elements. A relation contains an ordered list of one or more nodes, ways and/or relations.

OSM also provides *Tags*, which are key-value tuples associated to Nodes, Ways and Relations, to provide descriptions, meaning and any relevant contextual information for the various elements. An extensive list of key-value tags is documented in [13].

Table 4 OSM Node fields

name	Description
id	A unique identifier among all nodes.
lat	WGS84 Latitude coordinate in degrees.
lon	WGS84 Longitude coordinate in degrees.
tags	A list of tags associated with this node.

For this deliverable we exploited the availability of OSM elements data for the Tokyo area (see Figure 4), to accompany the OpenPFlow data that also covers the Tokyo metropolitan area.

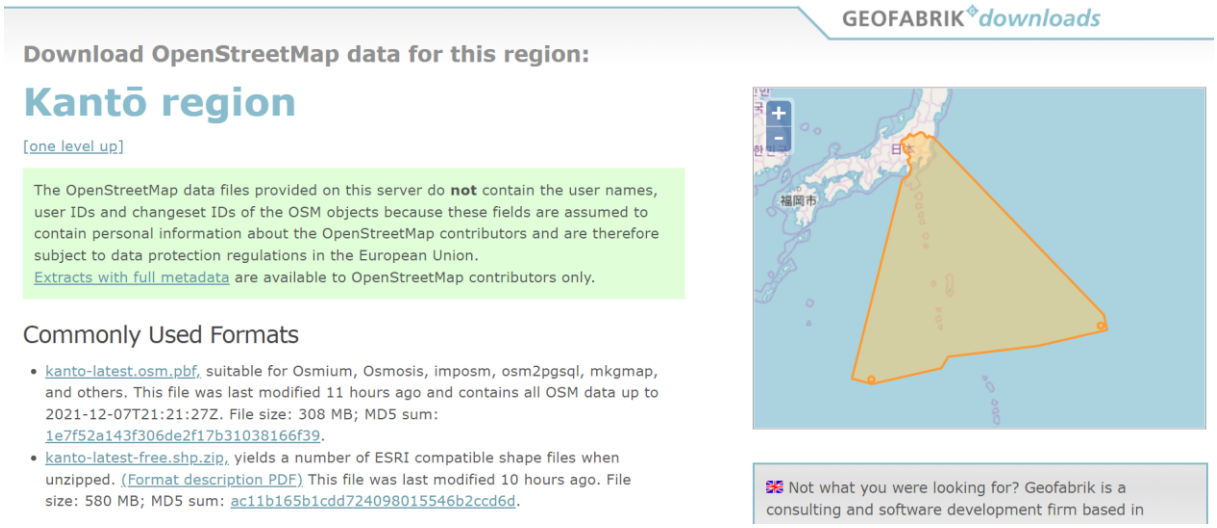

Figure 4: Tokyo OSM data from Geofabrik[14]

2.2.1.1 OpenPFlow Trajectory Dataset

OpenPFlow is an open dataset for typical crowd movements through different transportation modes in urban areas. It consists of continuous spatial temporal locations data for the metropolitan areas Japan. The PFLOW dataset extracted trip surveys combined with other population statistics, transportation data, and commercial GPS traces of a sample of the population. The result is a fully anonymized and synthetised dataset of a minute-by-minute trajectories containing unique id, latitude, longitude, time stamps, transportation mode, and magnification factor. This is a huge dataset that contains more than 660 million data records.

2.2.2 Solution Design and Evaluation

In this work we were interested in providing our sequential autoencoding model, reported in [16], with the ability to consume the multi-modal contextual information coming from OSM elements. This is motivated by the fact that urban trajectory data manifests in a feature-rich environment, as can be seen in Figure 5, characterized by topology, neighbouring structures (shops, street names, etc), and that much of the information about these features is publicly available and proved by data sources such as OSM.

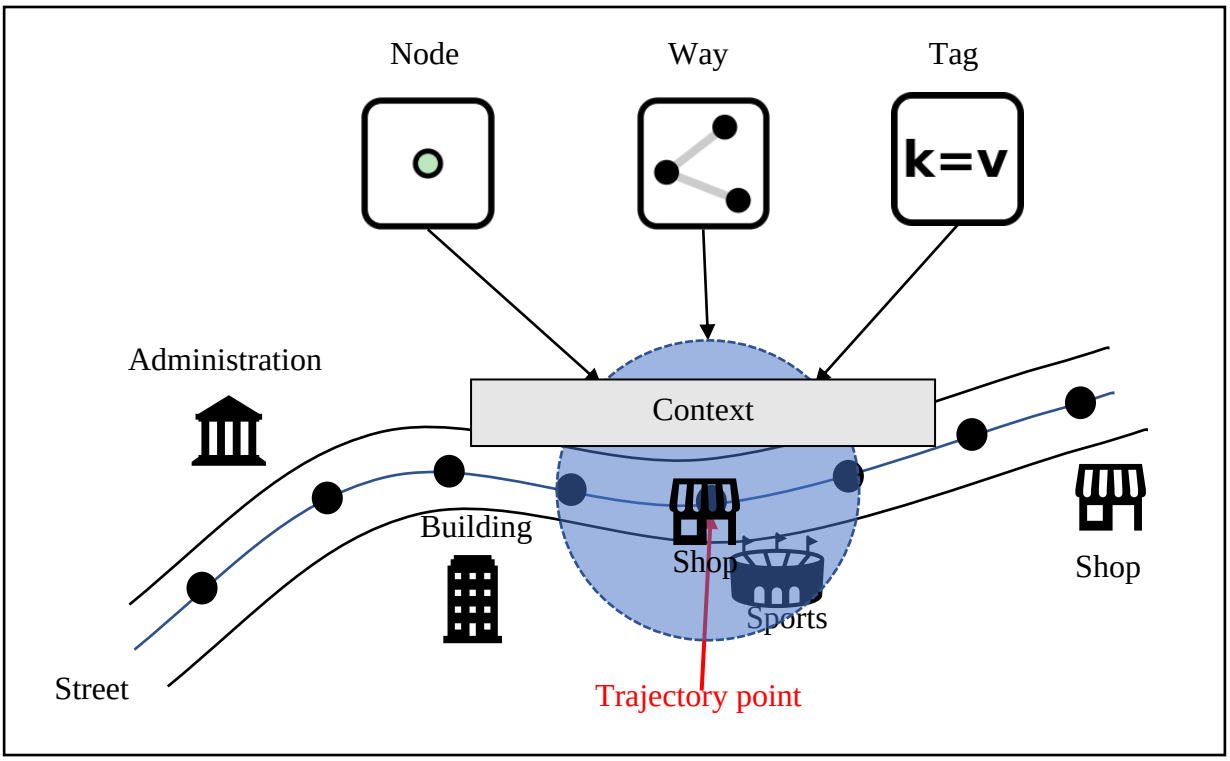

Figure 5: Trajectory Context Information in OSM

Tags information, nodes and ways' locations and shapes, are sources of information of different modalities. We are talking about natural language descriptions, point coordinates, geometric shapes, and relations. Incorporating these modalities in a common pipeline requires finding a common representation.

For this we envisaged a two-step pipeline (see Figure 6) that consists of:

1. Generating vector latent representation of OSM context for a given trajectory point.
2. Concatenating the vector representation of the context with the trajectory tracklets for consumption by the sequential autoencoder.

The two steps involve training two different models separately. The first model will discover the best possible latent representation of the variety of OSM elements, then generates a dataset containing all vector representations for all the elements within the geographic scope of interest. The second model consumes trajectory and context data, to generate the required output.

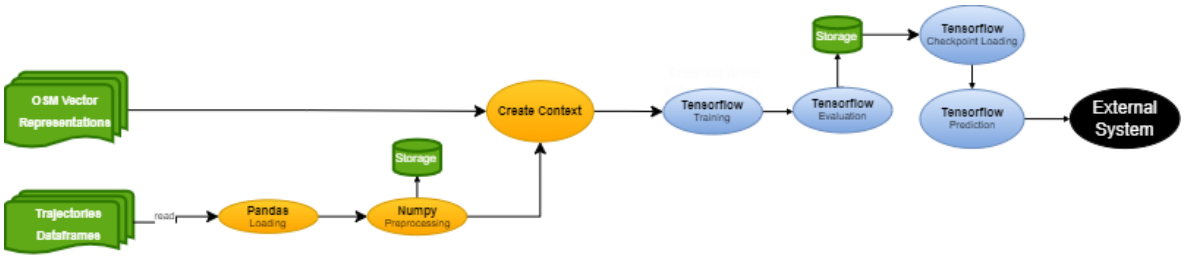

Figure 6: Pipeline with OSM context creation

To create context for every processed tracklet, we first organize all the vector representations in a k-dimensional (KD)Tree structure, with their IDs and Latitude/Longitude locations. The KDTree structure is used to identify for every point in the tracklet, the closest set of OSM entities, after which the corresponding vectors will be concatenated as additional input features.

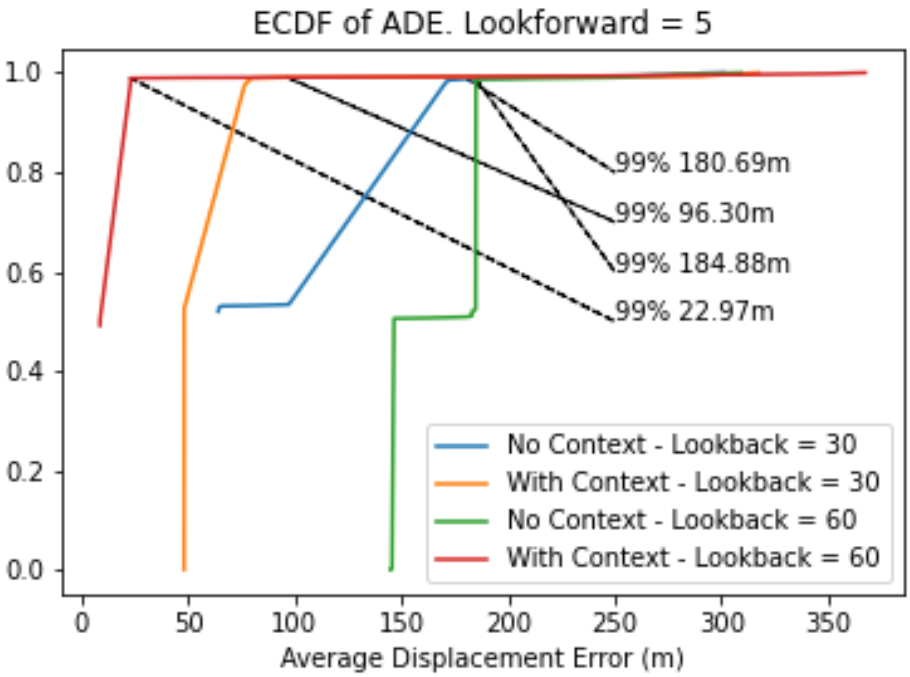

Figure 7 ADE

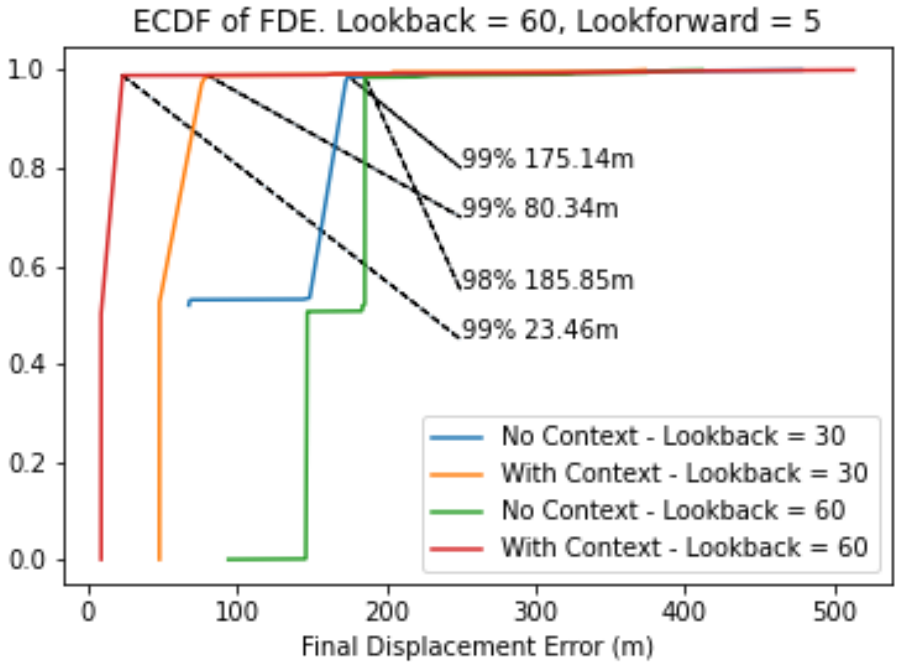

Figure 8 FDE

Following this approach, Figure 7 and Figure 8 show the empirical cumulative distribution functions of Average Displacement Error (ADE) and the Final Displacement Error (FDE), evaluation metrics for model training, using 60 (1 hour) and 30 (30 minutes) points of trajectory history, combined with context information for 1 closest OSM element, compared with the

case without context information. The target is to predict the next 5 trajectory points (5 minutes). This model achieves a best average **ADE** of **22.97 meters** and **96.30 meters** for the lookback of 60 points and 30 points consecutively, and an **FDE** of **23.46 meters** and **80.34 meters** for the same case. The results are a noticeable improvement from the model trained without OSM context data, as can be seen in the plots.

2.2.3 Conclusion

Consuming OSM elements data of if great utility for geolocation analytics and could enhance any geolocation-based analytics service. Nevertheless, consuming OSM data requires handling large amounts of data, hence careful and intelligent data preparation needs to be put in place, to allow the extraction, representation, and fusion of OSM data with trajectory data at an urban scale. We view these capabilities as foundational for modelling urban level and dense analysis of mobility patterns, the next steps will focus on comparative analysis with prior results and implementation of virtualized services with efficient data pipelines to process this kind of multi-modal data.

2.3 Group mobility detection and prediction

In deliverable D5.1, we presented a solution named “Group-In” (also published in [16]), a wireless scanning system to detect static or mobile people groups in indoor or outdoor environments. Some of the content in this section is from the previous deliverables D5.1 and partially in D5.3, such as the group mobility service functional aspects and service aspects. In addition to the existing group mobility detection and prediction based on various clustering algorithms, weakly-supervised machine learning algorithms are implemented for application to the group inference. Along with the existing methodology, this section includes the new methodology based on the weak supervision.

Group-In collects only wireless traces from the Bluetooth-enabled mobile devices for group inference. The key problem addressed in this work is to detect not only static groups but also moving groups with a multi-phased approach based on noisy wireless Received Signal Strength Indicator (RSSIs) observed by multiple wireless scanners without localization support. Group-In has new centralized and decentralized schemes to process the sparse and noisy wireless data, and leverage graph-based clustering techniques for group detection from short-term and long-term aspects.

In addition to the graph-based clustering techniques, a new machine learning algorithm is proposed. The algorithm belongs to the family of weak supervision algorithms.

Solution Design: A scientific paper related to the prototype system is published and presented in the ACM/IEEE Information Processing in Sensor Networks (IPSN’20) conference [16], which is a top-tier venue for fields such as sensing, cyber physical systems and IoT.
Group-In provides two outcomes: 1) group detection in short time intervals such as two minutes and 2) long-term linkages such as a month. Applications of the group inference are

considered for crowd management, smart retail, evacuation modelling, social isolation, and social distancing.
For the first outcome, weakly-supervised machine learning is considered as an alternative. The main advantage of the weakly-supervised machine learning is replacing the need of ground-truth data collection with domain experts through programmatic labelling. The concept of programmatic labelling is proposed in [39] and the tooling has been popularly used and studied in the recent literature.

2.3.1 Dataset Description

To verify the performance of the group inference, we conduct two experimental studies. One consists of 27 controlled scenarios in the lab environments. The other is a real-world scenario where we place Bluetooth scanners in an office environment. Both the controlled and real-world experiments result in high accuracy group detection in short time intervals and sampling times in terms of the Jaccard index and pairwise similarity coefficient.

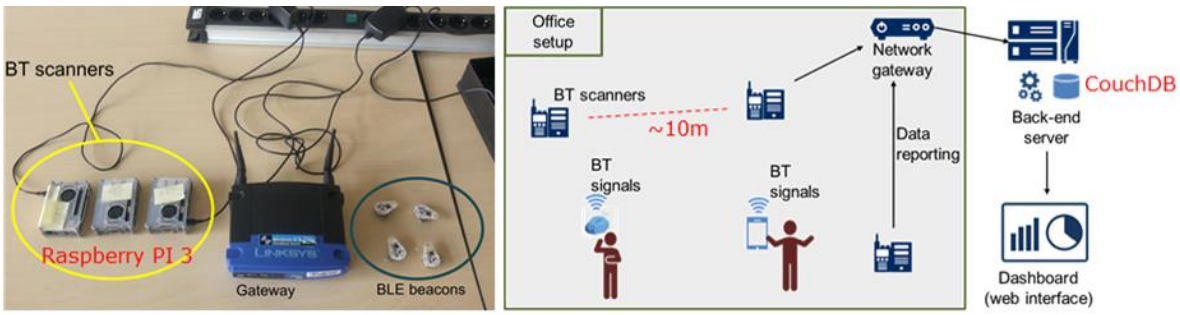

Figure 9 . Experimental setup and basic setup in the office environment

Figure 9-left shows some of the devices used in the initial experiments such as Raspberry PIs, BLE beacons, and wireless gateway. Figure 9-right illustrates the basic system setup for the experiments. For the testing group inference, we leverage the dataset collected in the lab environment. Especially, the controlled scenarios provide a ground-truth for training and measuring the performance through cross-validation.

The dataset contains ground-truth based on the marked groups of beacons in the controlled environment. On the other hand, ground-truth data collection is not straight-forward, especially for real-world “in-the-wild" scenarios. Furthermore, it might be costly and privacy invading to track people groups with images or similar camera-based technologies. For this reason, as an alternative to the supervised machine learning with ground-truth, we consider weakly-supervised ML. The weakly-supervised ML still uses the existing dataset for benchmarking purpose, although in the future operational use the algorithm would not need the manually collected ground-truth data.

2.3.2 Implemented Methods

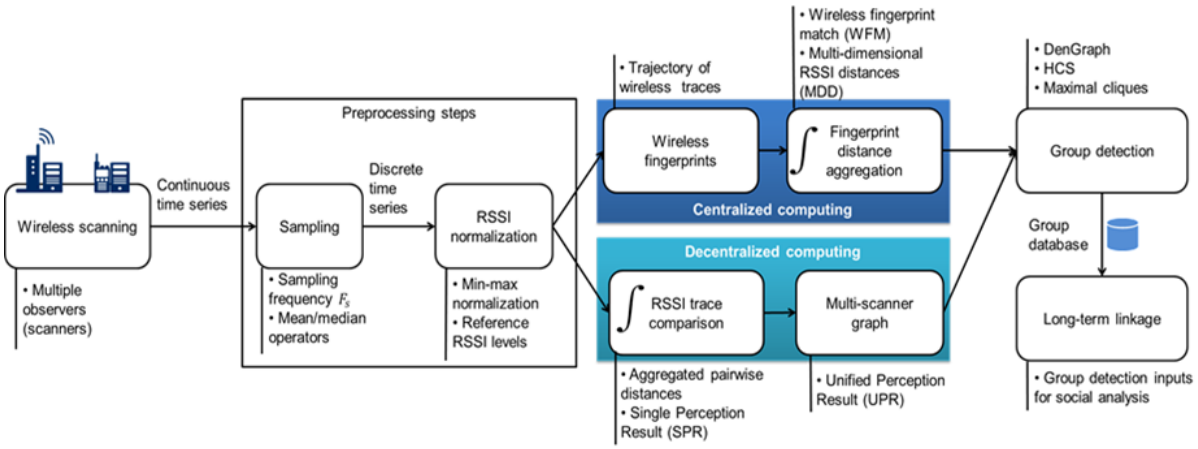

Figure 10 Group-In multi-phased pipeline approach with pre-processing, centralized/decentralized computing, and clustering.

Figure 10 show an overview of the implemented analytics pipeline including steps of data collection, pre-processing, centralized/decentralized data aggregation algorithms and clustering algorithms.

Group-In selected to leverage three clustering-based approaches as the group monitoring problem through wireless fingerprinting is a specific clustering problem. Moreover, we considered the application of Group-In in many setups without much training and testing. The main reason is applicability in real (wild) scenarios where the data collection can be costly and not feasible due to various reasons such as privacy or others (e.g., smart campus, smart city, and smart office). The main idea was to have an easy to deploy system which generates group monitoring insights from the deployment environment without much configuration changes or long data collection. The model selection focused on using *graph clustering approaches* as the data can be modelled easily as a social network, where nodes represent mobile wireless devices (or people), and edges represent the aggregated distance between these devices. The graph clustering would give set of clustered nodes which would represent the people groups in real-world. Various graph clustering algorithms are tested and showcased in the results. The model selection has been made empirically, where three graph clustering algorithms performed best various scenarios. There is no clear winner graph clustering algorithm as from scenario to scenario on of the three models may perform better. Thus, at the moment Group-In employs the following three graph clustering methods.

- DenGraph: Density-based scanning approach on graphs models
- HCS: Clustering highly connected subgraphs
- MaxClique: Clustering fully connected subgraphs

As aforementioned, one of the main challenges of the group prediction is collecting ground-truth in real setups. Although our initial approaches do not require supervised learning, but they rather rely on unsupervised learning through clustering technique, there are two limitations in this approach: 1) The clustering models still require several configuration parameters

to be configured and if possible cross-verified with a ground-truth dataset. 2) The clustering does not take usage of a possible domain expert who can programmatically label the data.

Table 5 A view on the wireless dataset pre-processed for enabling the application of weak supervision. The labels at the end column represents if person_x and person_y belong to same group.

	time	exp_number	person_x	person_y	person_x_s1	person_x_s2	person_x_s3	person_y_s1	person_y_s2	person_y_s3	label
0	1540475280	26	1	2	-75.0	-73.0	-43.0	-78.0	-68.0	-68.0	1.0
1	1540475280	26	1	3	-75.0	-73.0	-43.0	NaN	NaN	NaN	1.0
2	1540475280	26	1	4	-75.0	-73.0	-43.0	NaN	NaN	-65.0	1.0
3	1540475280	26	1	5	-75.0	-73.0	-43.0	NaN	-76.0	-60.0	1.0
4	1540475280	26	1	6	-75.0	-73.0	-43.0	NaN	-67.0	-76.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...
54553	1540476424	28	4	6	NaN	-80.0	NaN	NaN	-88.0	NaN	0.0
54554	1540476424	28	4	7	NaN	-80.0	NaN	NaN	-78.5	NaN	0.0
54555	1540476424	28	5	6	NaN	-81.0	NaN	NaN	-88.0	NaN	1.0
54556	1540476424	28	5	7	NaN	-81.0	NaN	NaN	-78.5	NaN	1.0
54557	1540476424	28	6	7	NaN	-88.0	NaN	NaN	-78.5	NaN	1.0

Table 5 shows a sample data from the group mobility dataset. The last column in this dataset presents the actual ground truth data which are going to be used for benchmarking the new weakly-supervised machine learning algorithm. The dataset is pre-processed and formatted again to make the application of weak supervision through programmatic "labelling functions" possible. Using the Bluetooth dataset, a set of labelling functions are implemented. Currently, the number of labelling functions are limited as the lack of high-modality of the data (e.g., number of features) makes it harder to implement new labelling functions even for a domain expert. For this purpose, we apply a new weak supervision algorithm that considers the limited number of labelling functions that results in a limited portion of dataset labelled. The new algorithm enhances the training of weak supervision especially for such limited scenarios.

Although the work on the weak supervision application to the group mobility is new, the algorithm has been already tested in different machine learning datasets and benchmarked against existing approaches. The goal is to apply this algorithm to the group mobility scenario to produce insights without ground-truth.

2.3.3 Results and Evaluation

The below results apply for our initial tests that are also reflected in D5.1. More complete results can be found in the scientific paper [17]. Although the machine learning pipeline for the weakly-supervised analytics has already been implemented, there have been limited testing and the evaluation results are not yet available.

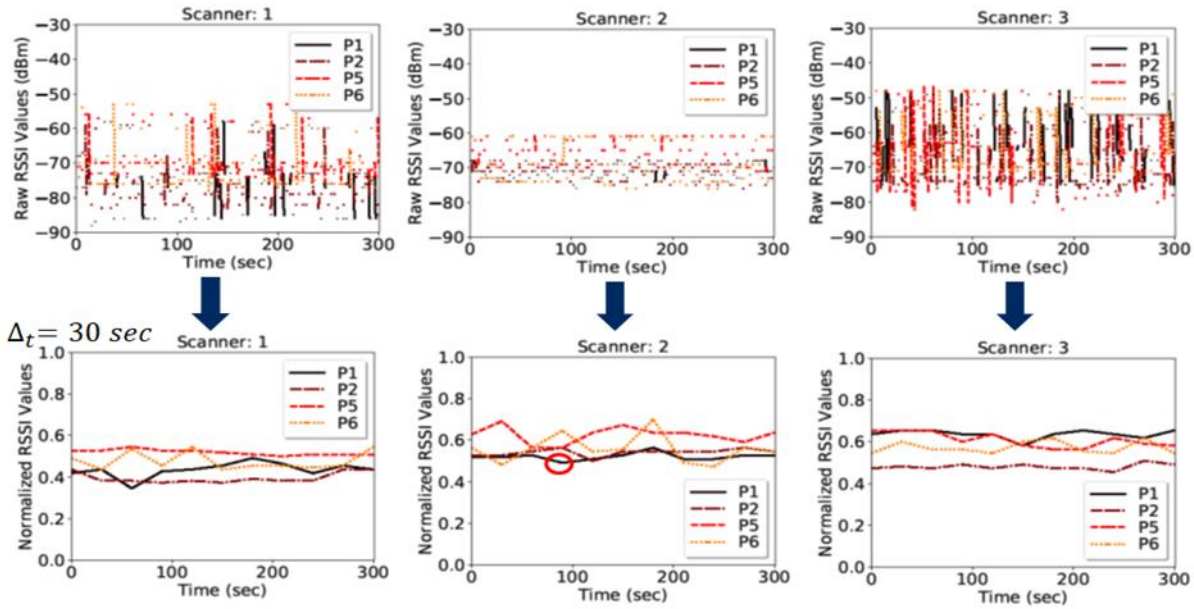

Figure 11 Result of the pre-processing phase: Top: Before pre-processing, bottom: after pre-processing

Figure 11 shows the outcomes of initial tests conducted in the lab setup using three scanners (Scanner 1,2, and 3). The top shows the initial RSSI measurement (before pre-processing) from the same beacons by different scanners. The bottom shows the same measurements after the pre-processing steps with 30sec sampling time.

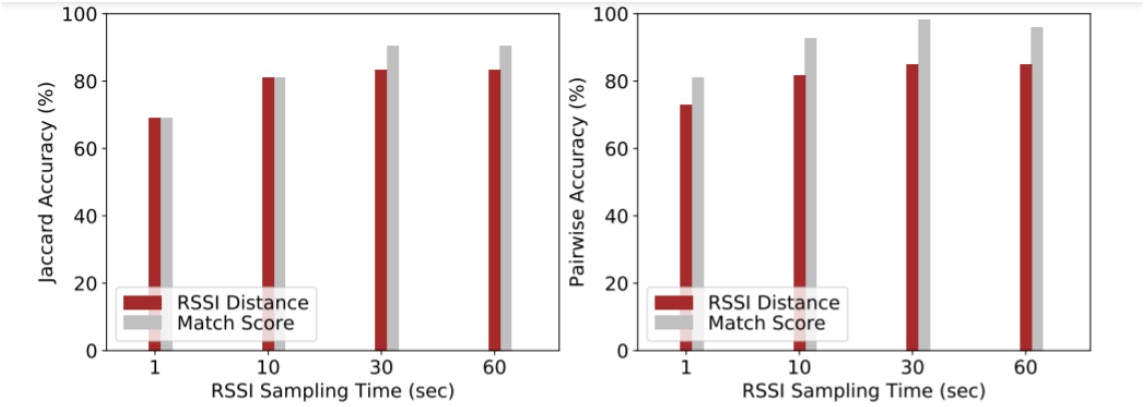

Figure 12 Group monitoring performance using different trajectory matching algorithms (from [16]).

Figure 12 shows the different algorithms in the centralized computing based on RSSI distance matching (e.g., aggregated RSSI trajectories) and match score based on the ordering (ranking) of the scanners through the signal strengths. Both these algorithms provide up to more than 80% group detection accuracy given a sampling time of 30 or 60 sec. The granularity of the dataset might affect the choice of the method. Both the RSSI distance and match score methods are described in detail in [16].

Table 6 Summary of the algorithms used in the experimental evaluation (from [16]).

Analytics algorithms	Approach phases	Parameters	Parameter trial values
DenGraph	Group detection	Cluster distance	(0, 1]
HCS	Group detection	Min edge weight - Threshold	(0, 1], (0, 1.5]
MaxClique	Group detection	Min edge weight - Threshold	(0, 1], (0, 1.5]
WFM	Fingerprint trajectory match	υ: Match score vector	[[3, 100], [2, 25], [1, 5]]
MDD	Fingerprint trajectory match	ζ: Max distance parameter	7 (empirical)
Benchmark algorithms			
Girvan-Newman	Graph clustering (after WFM)	-	-
DBScan	Clustering preprocessed data	ϵ: Epsilon	(0,1]
MeanShift	Clustering preprocessed data	Kernel	(0,1]

Table 6 shows the summary of the analytics algorithms that are implemented and tested for the experimental evaluation. Some of the analytics algorithms require setting a configuration parameter, which can be highly effective for the performance of the clustering. For this purpose, several parameter values are tried empirically on only one scenario, and the best performing value is applied to the whole portion as well as all other experimental scenarios. Even though the parameters are statically set, the system provides high accuracies even in highly-different different scenarios [16]. In Table 6, the algorithms are also mapped to the stages (phases) on the analytics pipeline.

Figure 13 shows the initial experimental results of Group-In for controlled and real-world set-ups. Group-In achieves close to 98% accuracy for 30 sec sampling time and 2 minutes time interval, whereas in the real-world setup the accuracy reaches close to 90% for both metrics. Girvan-Newman (GN), DBScan, MeanShift clustering algorithms are used for comparison. These algorithms are applied directly after the pre-processing phase (replacing centralized or decentralized computing).

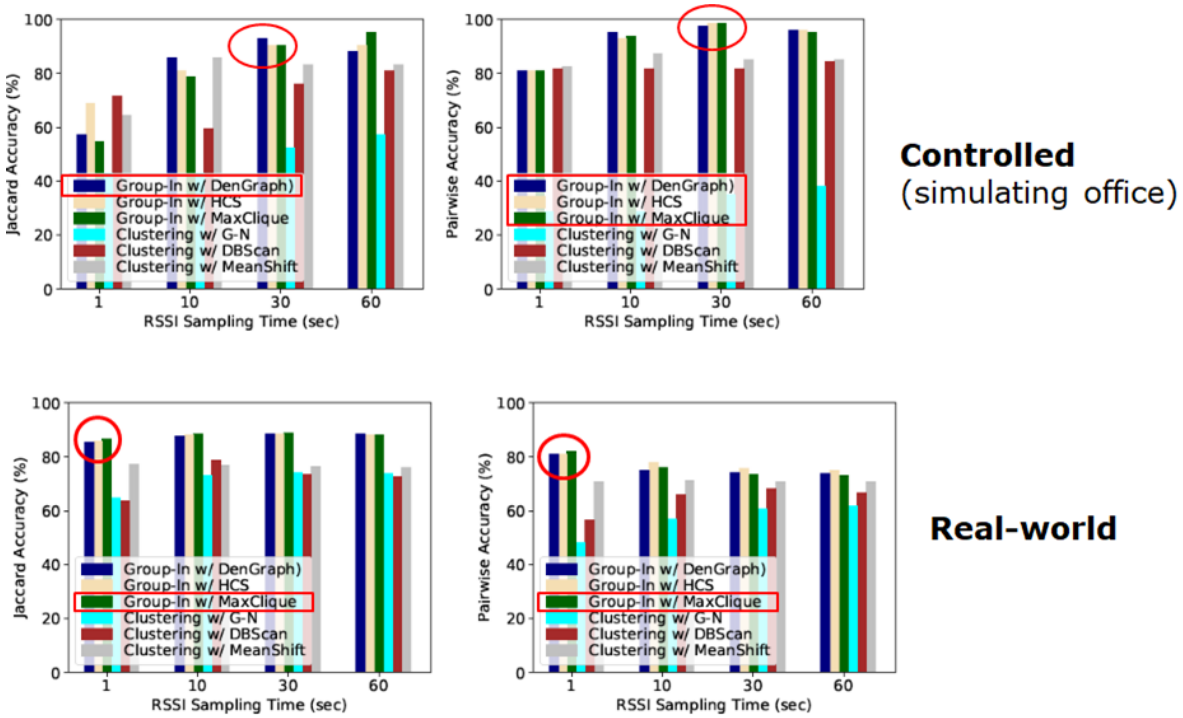

Figure 13 . Experimental results for controlled and real-world office scenario setups. Using the centralized computing approach

Figure 14 shows another set of results of Group-In only for controlled setups. In this experiment, the BLE beacon devices are artificially placed together as two sets of devices, representing two groups separated from each other scenario. These two groups of devices are placed with 10 meters distance from each other. Later, the groups are gradually placed closer to each other until they are only 1 meter apart. The groups are statically placed during the experiments. The results in Figure 14 shows that as long as the groups have around 4 meters distance (even though they are static), Group-In can differentiate the people in different groups from each other with about 80% Jaccard accuracy and pairwise differentiation accuracy. On the other hand, in the scenarios where two groups are too close to each other (e.g., with 1- or 2-meter distance) statically or dynamically (groups walking in parallel to each other), Group-In produces limited group monitoring performance. This experiment shows the feasibility of application of group monitoring through wireless fingerprinting using Bluetooth in various use cases in real scenarios such as in city squares or shopping streets.

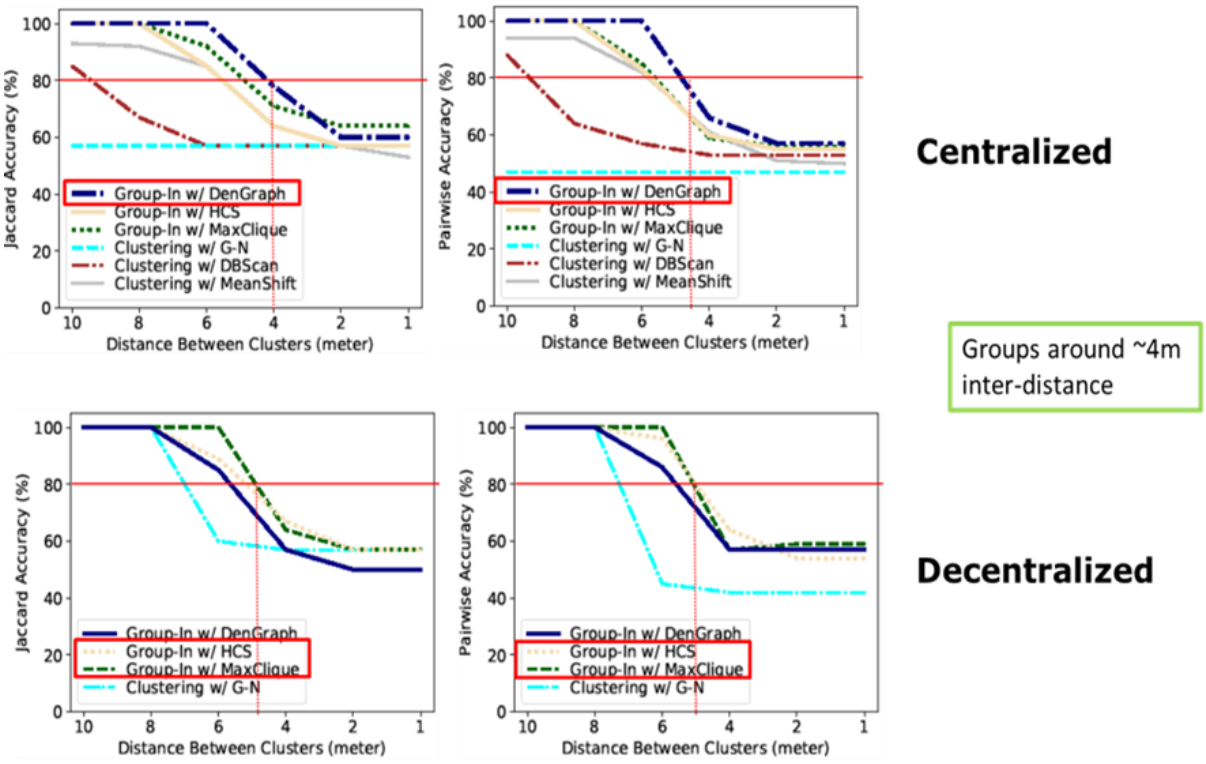

Figure 14 Group monitoring performance of Group-In based on group inter-distances

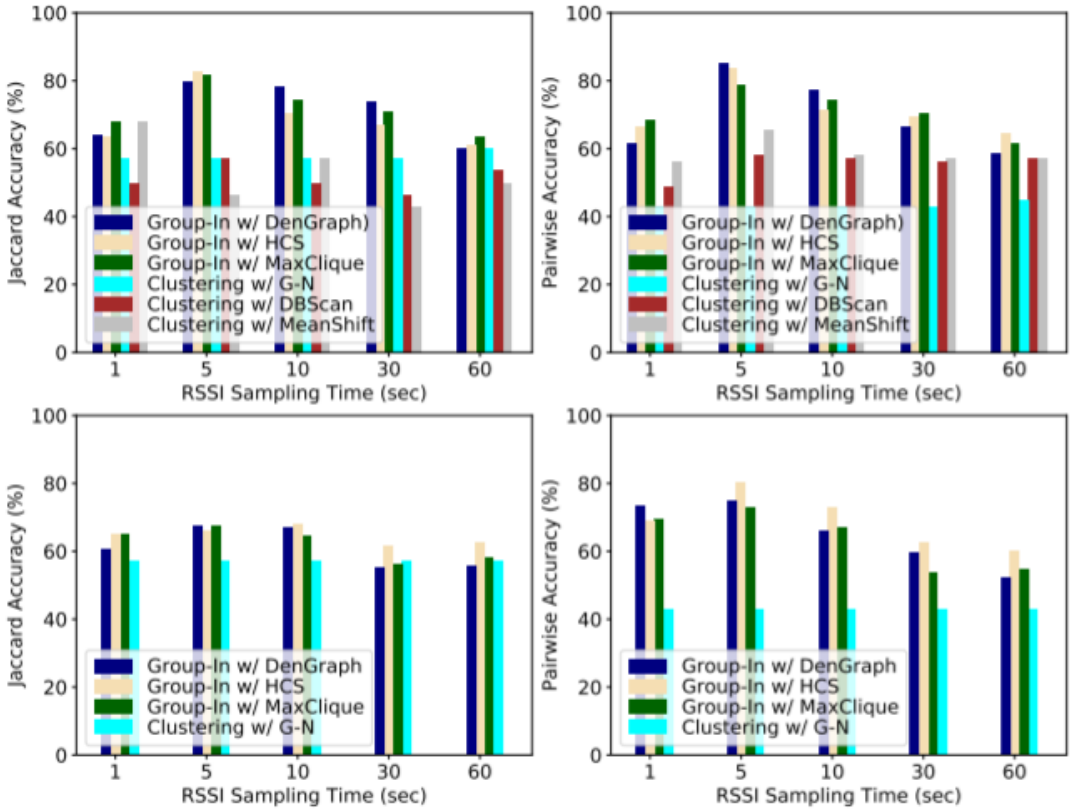

Figure 15 Group monitoring performance of Group-In for the "random-walk" scenario. Top: centralized algorithms, bottom: decentralized algorithms.

Figure 15 illustrates the results in a random-walk scenario, where the Bluetooth beacons are carried in a random walk fashion in a large conference room. The mobility creates a constraint especially for longer sampling times (e.g., 30 or 60 seconds). As sampling reduces the dynamicity in the actual measurements. This expectation is reflected in the real experimental results as shown in Figure 15. The best performing sampling time was 5 seconds where the noise in the wireless signal strengths is partially eliminated and the dynamicity in the signals due to the movement of the people is also captured by the analytics pipeline. In both centralized and decentralized computing, this resulted in higher than 80% pairwise group inference accuracy.

More extensive results about the functionality 1 can be seen in the full paper which was included in ACM/IEEE IPSN'20 proceedings [16].

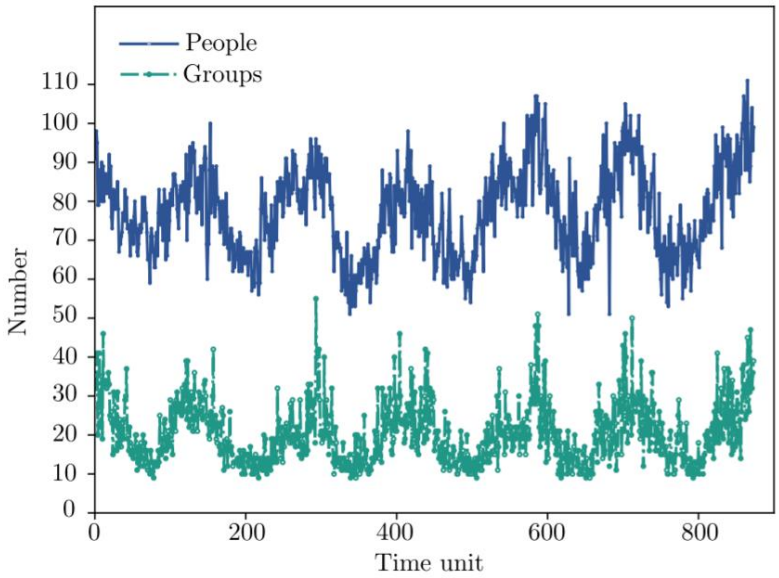

Figure 16 People counts and number of peoples observed every 10 minutes based on wireless activities in a city [18].

As the last set of experiments to be highlighted for Group-In, we consider the scalability of the system in a larger-scale such as smart cities [18]. To understand the feasibility of the analytics in a large-scale domain such as a smart city, we applied Group-In in an anonymized people count dataset which considers the following: Wi-Fi based people counting (Wi-Fi PP counts) and Group-In based group counting. Although the data is not tested against a ground-truth data (due to lack of available ground-truth dataset). For more precise measurements a small amount of ground-truth data would be beneficial. Group-In is observed as computationally feasible for application at larger scales to infer groups among more than 100 people and generate real-time insights. The dataset from the previous PoC study [20] is described in more detail in this deliverable as the multi-modal dataset.

2.3.4 Application in UC & PoCs

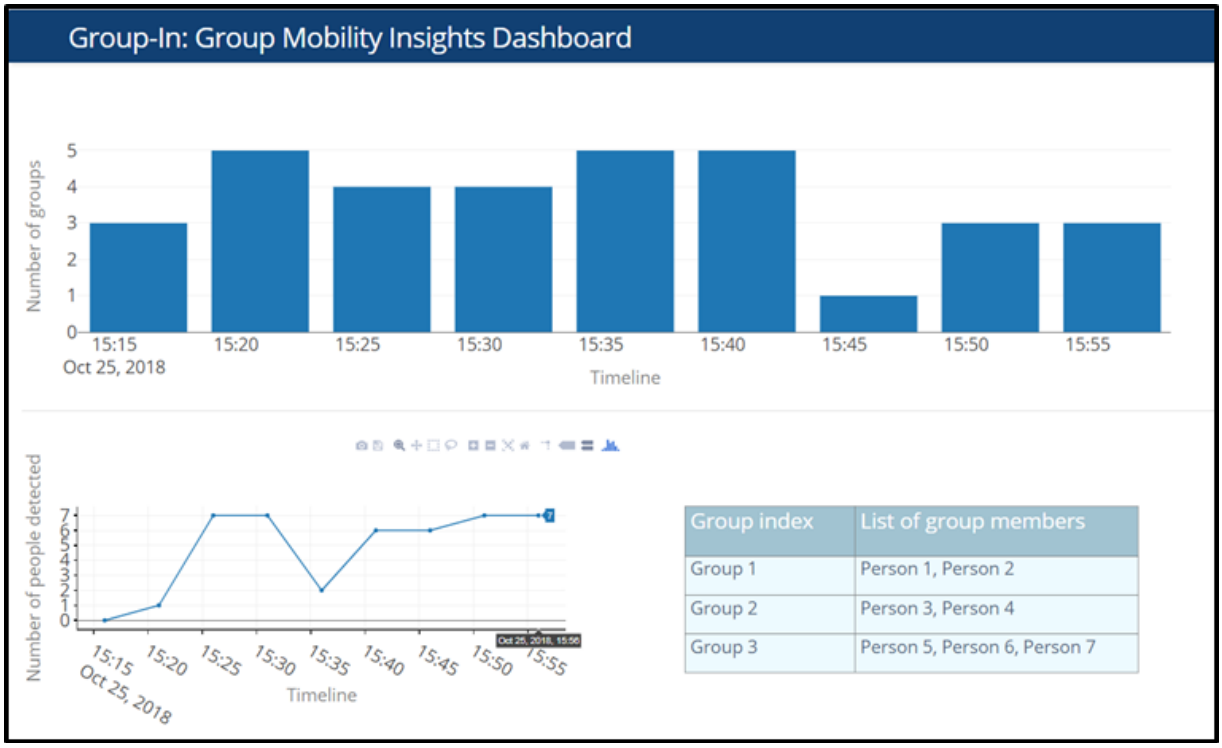

Figure 17 Group-In Web dashboard showcasing the crowd mobility analytics results

The experimental setup included three wireless scanners that are placed 10 meters apart. The wireless scanners can capture wireless packets (Bluetooth advertisement packets) from BLE beacon devices. Wireless scanners (Raspberry PIs) can perform simple processing on themselves and create messages and send the packets to the back-end server through the illustrated network gateway. The back-end server places the data into a NoSQL database that provides fast access to the data through key-value pairs (JSON objects). Lastly, the back-end server of the Group-In process the raw data through its analytics modules and visualizes the analytics results on a web dashboard as shown in Figure 17.
The group mobility functionality using the clustering techniques with high accuracy group detection, including the whole pipeline as well as the dashboard is considered to be presented in the WP6 PoC activity.

2.4 Multi-modal crowd mobility

For group-pattern detection, in this section, we present in a new application “CountMeIn” [21], a wireless scanning system with auxiliary sensors (i.e., stereoscopic people counting cameras) to detect static or mobile people groups in indoor or outdoor environments. Some of the content in this section are from the most recent paper [21], whereas more details can be found in the paper.

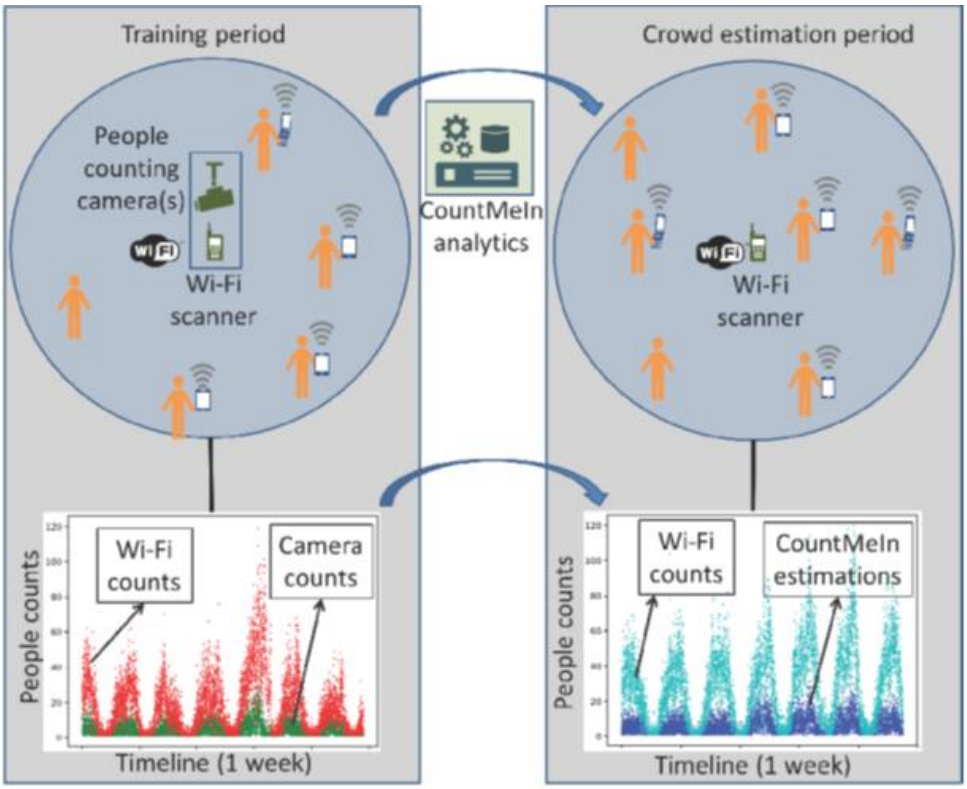

Figure 18 CountMeIn approach of learning correlations between Wi-Fi and auxiliary sensor and applying calibration in the long-term crowd estimation period. [21]

Solution Design: A scientific paper related to the prototype system is accepted to the IEEE *International Conference on Pervasive Computing and Communications* (Percom'22) conference [21], which is a top-tier venue for fields such as pervasive computing and IoT. Moreover, we had a position paper about crowd estimation in smart cities

CountMeIn proposes long-term high accuracy crowd estimation in urban environments such as places in smart cities using Wi-Fi RSSI in the vicinity of the environment as well as auxiliary sensors [19] as illustrated in Figure 18. Applications are considered similar to the Group-In applications, whereas crowd estimation is one of the major topics for smart city applications in general. This topic has been widely researched in research communities. However, most existing literature focus on leveraging computer vision-based solutions.

2.4.1 Dataset Description

The dataset was collected previously in a pilot PoC study. The pilot study details are included in the previous position article [20]. The data used in the CountMeIn included only people counts (integer values) from the auxiliary sensors [20], which served as the near-ground truth. Moreover, Wi-Fi data included anonymized IDs (hashed and salted) signal strength values (RSSIs). The dataset did not involve individual tracking or any (personal) information that can be mapped to a certain individual.

As stated in [21], the data refers to six consecutive months, from February to July, where all the devices operated without interruptions. The previous position article describes the pilot setup and data aspects [20]. The goal is to use the field data for assessing feasibility of CountMeIn in large open areas of a smart city. The testing setup includes 2 Wi-Fi scanners and 2 stereoscopic cameras. The cameras cover two areas: 1) Pedestrian shopping street and 2) Near the beach area. Each of these areas is supported by a custom Wi-Fi scanner built to operate in outdoor environments.

The people count from the auxiliary sensors serve as ground-truth. This assumption was cross-verified by a field test team through manual counting. The performance of camera-based count is between 88% (evening peak times) and 98% (morning). Environmental factors such as rain affect the accuracy as the cameras do not capture multiple people under one umbrella. We suppose that higher accuracy camera counts would enable learning crowdedness patterns more precisely and lead to easier machine learning model convergence, whereas we could not test this because of the lack of video footage or detailed field test data.

In addition to the field data, the results include data from lab setup where an edge device is used to test the machine learning performance in terms of scalability (training time).

2.4.2 Implemented Methods

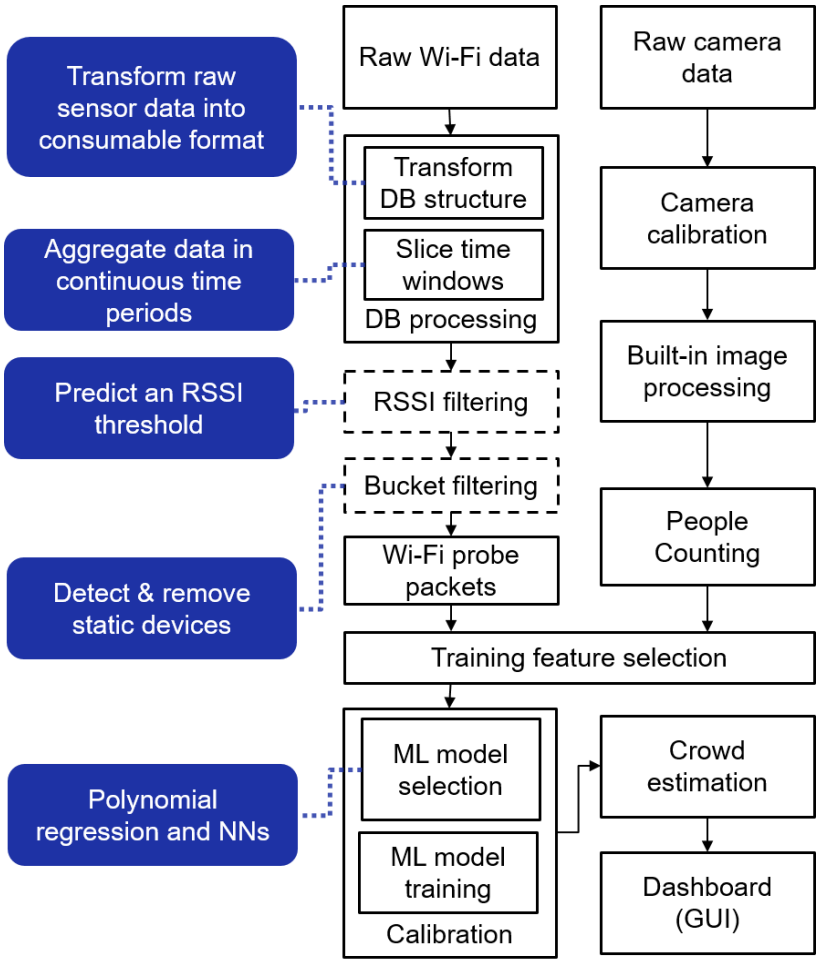

Figure 19 The analytics method of correlation of Wi-Fi and auxiliary sensors for learning and applying in the long-term in smart cities [21].

CountMeIn calibrates the count based on Wi-Fi probe packets (PPs) for realistic crowd estimation. The calibration system is trained within a short period through counts coming from stereoscopic cameras [19]. Figure 19 illustrates the functional view of: 1) Training period to determine the relationship between the Wi-Fi PPs and the near-ground truth people counting data; and 2) Use the calibration system to predict using the trained machine learning model.

The training period involves data collection for raw Wi-Fi and camera data in parallel. As the raw Wi-Fi data is noisy in its nature and off-the-shelf devices do not accurately count people using Wi-Fi, CountMeIn includes the four tasks listed on the left side of Figure 19 before training or using the calibration model. In the next subsections, we provide details about the Wi-Fi calibration steps. The off-the-shelf people counting camera provides a calibration interface, and it has built-in image processing and people counting modules.

The main analytics functionality comes from the machine learning in the calibration module. This module includes different neural networks and polynomial regression models. We include

the neural network architecture with the best performance in the Table 7. The details of regression models and all the analytics steps listed in the architecture can be found in [21].

Table 7 The neural network architecture with the best performance for crowd estimation [21].

Layers	Shape	Number of parameters
Dense layer 1	50	200
Dense layer 2	35	1785
Dense layer 3	22	792
Dense layer 4	14	322
Dense layer 5	6	90
Dense layer 6	1	7

2.4.3 Results and Evaluation

We include several key results from the evaluation. The complete set of results can be found in [21].

Table 8 The performance of CountMeIn, along with the raw Wi-Fi PP count performance and PFA [22] performance.

	Minute-level				Hourly mean			
Model Name	MAD	MAPE	WAPE	RMSE	MAD	MAPE	WAPE	RMSE
Raw Wi-Fi PP counts	14.05	238.75%	205.19%	19.10	11.84	166.97%	188.12%	15.42
Pedestrian Flow Analyzer [22]	4.95	59.46%	66.27%	6.60	1.89	25.38%	43.56%	3.22
CountMeIn w/ Reg. w/o RSSI filtering	2.57	36.26%	34.44%	3.78	0.43	5.56%	10.07%	0.96
CountMeIn w/ Reg.	2.56	36.09%	34.35%	3.75	0.42	5.53%	9.88%	0.94
CountMeIn w/ NN	**2.52**	**34.88%**	**33.81%**	**3.72**	**0.40**	**5.11%**	**9.42%**	**0.91**

The results listed in the Table 8 shows the minutely and hourly crowd estimation in terms of four statistical metrics. CountMeIn with neural networks provides the best outcome in all the statistical metrics. Compared to Wi-Fi PP counts and the state-of-the-art PFA system [22], CountMeIn provides substantial gains thanks to the machine learning from the correlations from the auxiliary sensors.

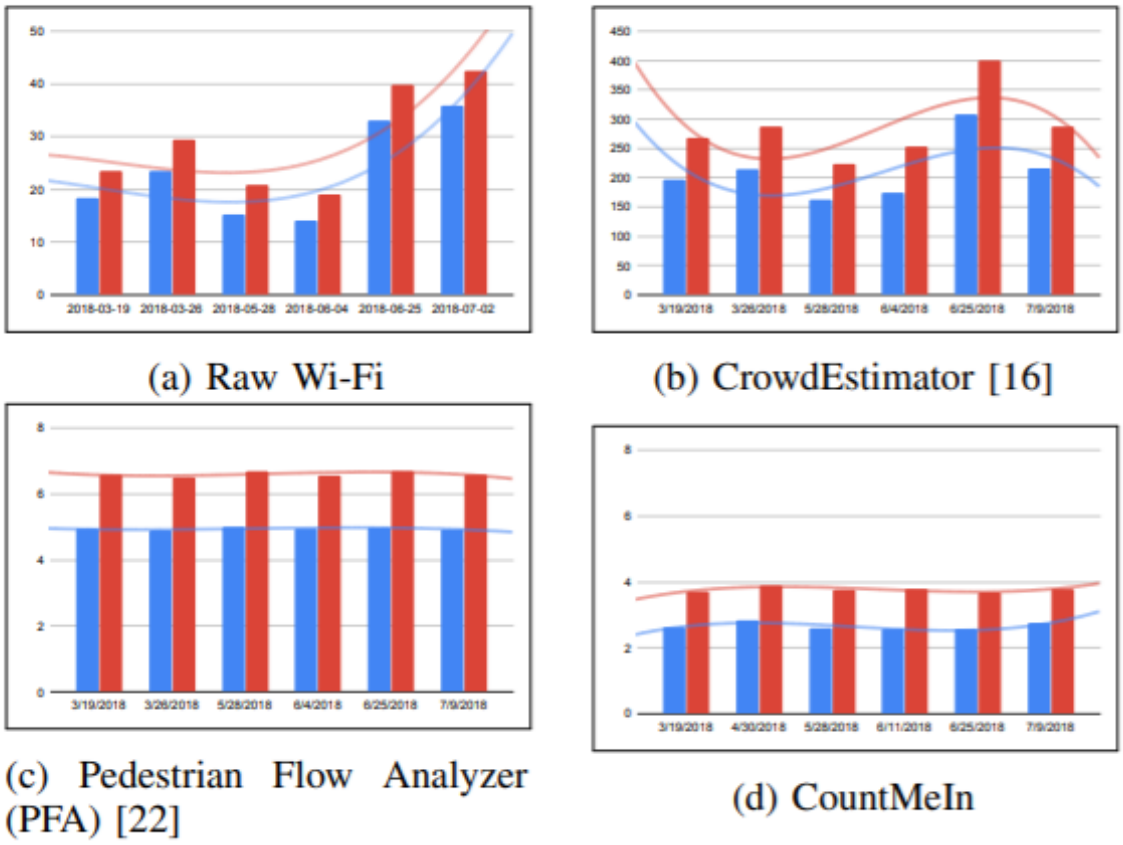

Figure 20 Comparison of error values (Red = RMSE, Blue = MAD) with raw Wi-Fi PP count, the two state-of-the-art approaches CrowdEstimator [23], PFA [22], and CountMeIn [21].

Figure 20 shows the comparison of error values in raw Wi-Fi (without analytics or auxiliary sensor), the state-of-the-art Wi-Fi-based PFA [22] and CrowdEstimator [23] systems. Our previous CrowdEstimator [23] model constantly calibrates a correlation coefficient between camera ground truth and Wi-Fi PP count for crowd mobility solutions. The other model, Pedestrian Flow Analyser (PFA) [21], performs various pre-processing steps to reduce noise in the raw Wi-Fi data before calculating crowd count and performing further crowd analysis. We implement both these approaches and compare their error values with the CountMeIn system.

Training times: In our tests for the polynomial regression model with higher degrees (up to 12) the training takes a maximum of only ~37 sec on 2-week training data, whereas the neural network model takes about 27 min to complete 150 epochs of training over the same data. On the other hand, while time taken by the neural network model is over 43x more than the regression model, the neural network generalizes better and can outperform the regression model in cases where patterns in test data are slightly different from the training data.

2.4.4 Application in UC2

CountMeIn is part of the crowd mobility analytics use case (UC2) in WP5. The system is based on using Wi-Fi and auxiliary sensors, whereas similar methodology could be considered for 4G

or 5G datasets using signal strength as well as other multi-modal features of these datasets. Group-In and CountMeIn together provides key characteristics of crowd mobility analytics through machine learning.

2.5 Analytics service for Vulnerable Road User

2.5.1 Dataset Description

Urbanization has birthed an exponential increase of the population of cities. Today, 55% of the world population lives in urban areas, and this is projected to increase to 68% by 2050 [25]. Therefore, road safety is a critical requirement for vulnerable road users (VRUs) in such environments. User clustering is particularly important for flow monitoring and crowd control and accordingly, it has been studied from different angles in the LOCUS project e.g., Point of Interest.

In future C-ITS system, VRU road safety situation will be detected by different sensors, sensor fusion and AI mechanisms. Fusion of sensing and perception from VRU's device, vehicle and infrastructure will be employed enabling VRU path prediction or VRU profile detection. Data inputs will include

- VRU sensors e.g. motion, compass, gyroscope,
- Vehicle sensors e.g. cameras, radar, LIDAR,
- Roadside infrastructure sensors e.g. cameras, VRU activated equipment such as push-to-cross buttons.

In this section, we extend the study of user clustering by focusing on the clustering benefits on the communication system performance (spectrum usage) where vulnerable users are concerned – recent 5G Automotive Association report [30] identified it is one of the areas where further research is needed. The dataset models the Shibuya intersection in Tokyo, Japan, famous for being the world's busiest intersection, with a square of size 70 m as illustrated in Figure 21. At its busiest period, as many as 3000 people cross in different directions, making it an imperative for a good VRU clustering study scenario [26]. This dataset is simulation based using requirements from the recently completed ETSI ITS VRU standard. User movement at such intersections is dynamic and a robust communications system needs to be designed for the worst-case scenarios. Hence, this research work could be considered as a validation of the clustering requirements provided by ETSI ITS VRU standard as some design recommendations are provided.

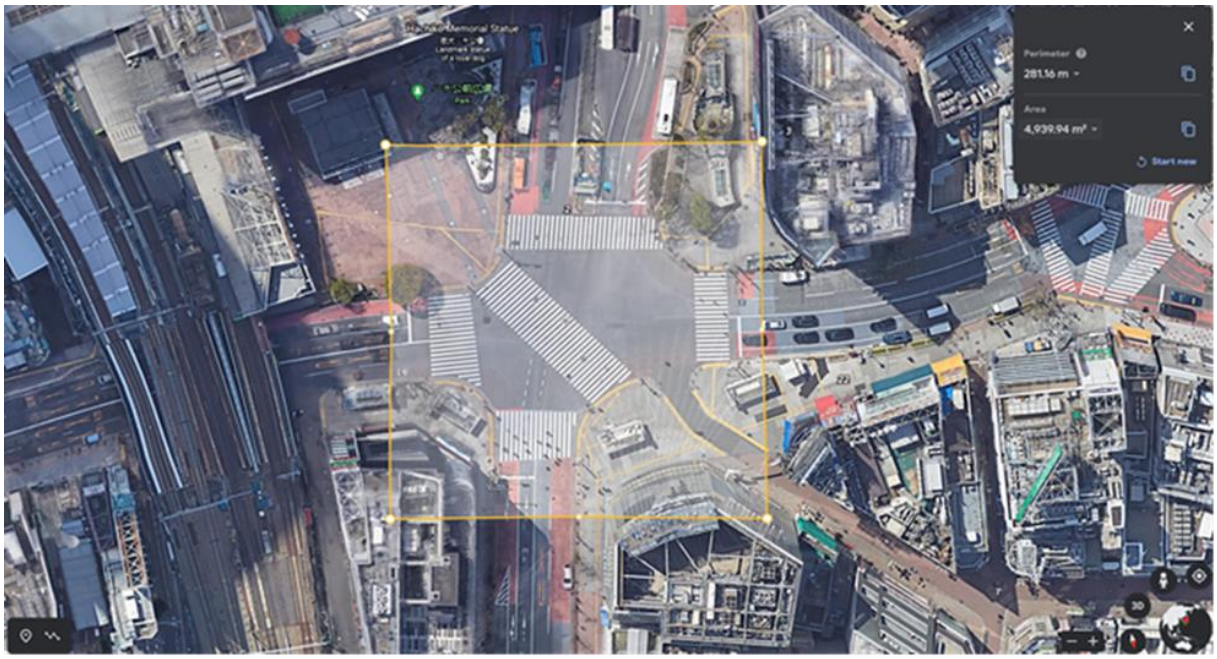

Figure 21 An Overview of the Shibuya Scramble Crossing (Google Earth)

2.5.2 Implemented Methods

At an urban intersection, there is often a high density of VRUs as illustrated in Figure 22. To successfully assess a road safety risk for each VRU in ETSI Cooperative intelligent transport system (C-ITS) system architecture (ETSI EN 302 665), there is a periodic exchange of standard VRU messages required between different VRU devices equipped with intelligent transport system stations (ITS-S) and a roadside ITS (R-ITS) at the crossing.

Figure 22 : VRUs at crossing in urban intersection (adopted from ETSI TS 103 300-1 [27])

In crowded scenarios, this will require a prohibitive amount of communication resources e.g. in a dedicated 5.9 GHz ITS band since separate V2P broadcast messaging sessions are to be initiated for each VRU device in a risk zone causing network congestion and message loss. However, with a clustering solution, VRU devices can be grouped into clusters and only the cluster head VRU will continuously exchange messages with the R-ITS (which may control cluster formation and maintenance), requiring a single session for the cluster. The cluster parameters are communicated using the VRU Awareness message (VAM). Thus, user clustering provides a reduction in the VAM traffic and also as a result, may reduce the energy consumption in the VRU devices.

There are several criteria which need to be satisfied for the formation and maintaining of VRU clusters. These include the relative speed and relative distance between members of a cluster as well as the number of VRUs already existing in a cluster. Detailed operational requirements for VRU clusters are defined in ETSI TS 103 300-3 standard [28], [29].

2.5.3 Results and Evaluation

In this section, we develop a simulation platform to evaluate the performance of homogenous VRU clustering i.e., the grouping of pedestrians at an urban intersection. We consider a wide range of VRU velocity as they pass through the intersection. Velocity ranges from 0.1 to 5 km/h covering both slow-moving pedestrians (e.g., children and the elderly) and fast-moving pedestrians (e.g., runners). At the beginning of the simulation, the VRUs are grouped according to their relative speeds, the maximum speed difference inside a cluster is kept at 5% [30]. The VRUs are tracked until they are out of the risk zone, i.e., they have safely crossed the intersection. The VAM messaging overhead is calculated by the sum of the total number of exchanged messages in the VRU system, which is an ensemble of ITS stations interacting with each other to support VRU crossing, e.g. VRU device ITS-S, vehicle ITS-S and R-ITS. The frequency of the standard VAM messaging generation is chosen as 2000 ms. [30], thus the total number of exchanged messages between a VRU device and the R-ITS is dependent on how much time it takes the VRU to cross the intersection.

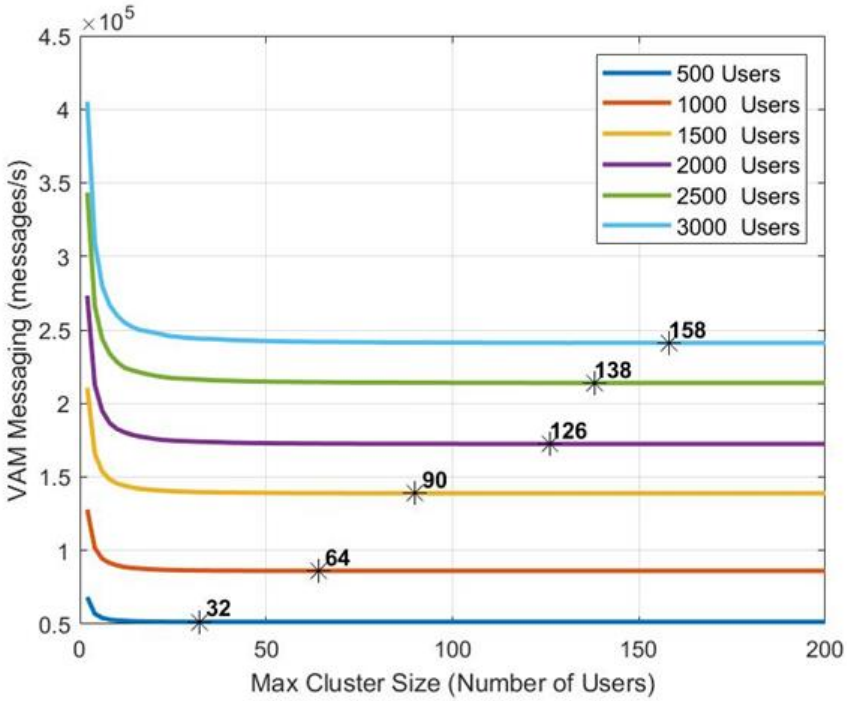

Figure 23 VAM messaging traffic in VRU clustering

Firstly, in Figure 23, we investigate the VAM messaging traffic change whilst varying the constraint on the maximum number of users that can join a cluster. Here, we assume that the users have VRU devices equipped with an ITS-S to enable the exchange of standard messages. It can be seen that there is an inverse relationship between the cluster size and VAM messages traffic since a higher constraint results in a fewer total of VRU cluster heads, and thus a decrease in the messaging overhead. More interestingly, however, we observe a plateauing of the VAM messaging traffic after a certain cluster size for different VRU groups as indicated by the starred points in Figure 23. Specifically, the respective maximum cluster sizes when 500

(average usage) or 3000 (busiest usage) VRUs cross the intersection at a single time are 32 and 158. Given that the clusters are formed by the similarities in their relative speeds, these results indicate that cluster sizes up to the maximum of the indicated starred values would provide some benefit. Nevertheless, we note that even when the VRUs are travelling at similar speeds, other conditions such as the processing power of the VRU cluster head would need to be considered as an additional limiting factor to the cluster sizes. This information could be considered in the deployed system design.

Next, using the information about the maximum cluster size for clustering at the Shibuya intersection with a square of size 70 m covering the whole area of the crossing, Figure 24 depicts the average number and size of VRU clusters formed by grouping the VRUs with their relative speeds. The numbers and sizes of VRU groups increase with increasing VRU numbers. Specifically, at the busiest periods, with 3000 users crossing the intersection, 81 clusters each made of 36 VRUs are formed while at less busy periods, with 500 users, 54 clusters of 9 VRUs each are formed. Note that although 3000 is the current maximum number of users at the Shibuya intersection, some extra capacity would need to be considered when designing the commercial system.

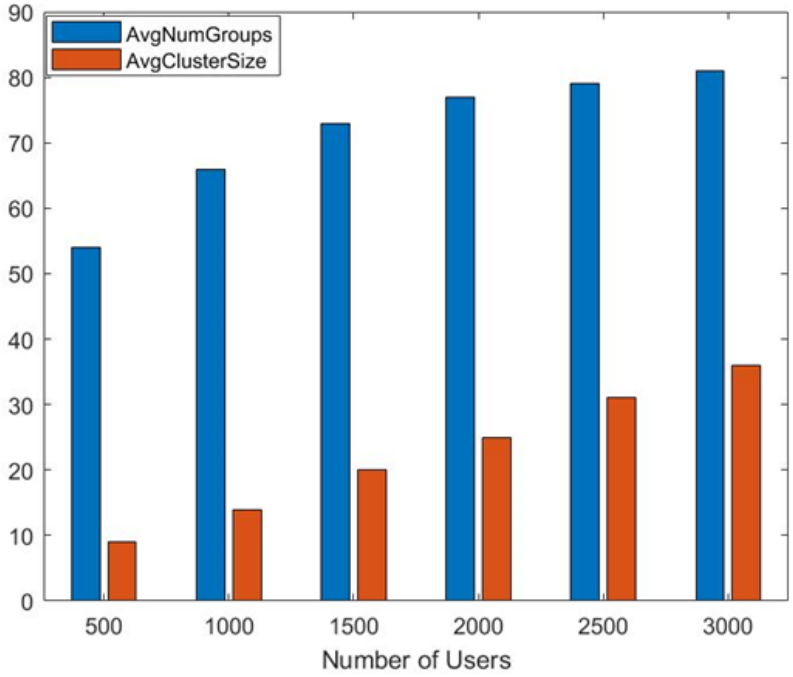

Figure 24 VRU clustering statistics for varying number of users

Finally, we compare the messaging traffic with and without the clustering solution. It can be seen from the Figure 24 that VRU clustering achieves a lower VAM messaging traffic for the same conditions. In particular, the clustering solution provides reduction in the traffic and corresponding communication resources and spectrum usage by approximately 47% for lower VRU numbers and 60 % for higher VRU numbers. These results are important, indicating that

VRU clustering solutions are especially beneficial during the busiest traffic periods at the intersection.

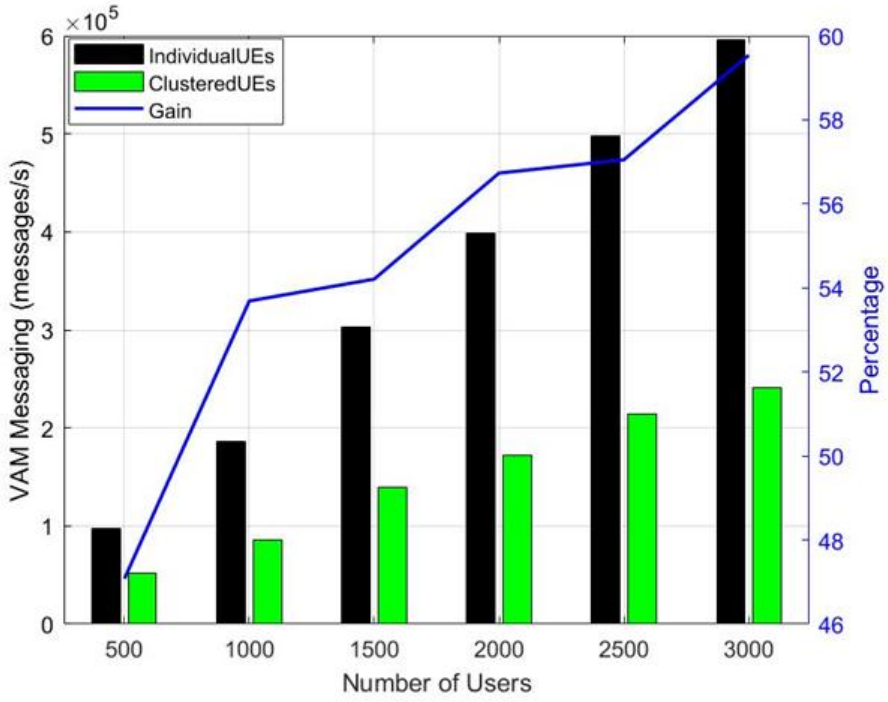

Figure 5-5: VRU clustering impact on VAM traffic

2.6 Logistics in a seaport terminal using AGVs

2.6.1 Remotely controlled AMR/AGV for logistics

Mobile robots for logistics shuttle materials between working areas and warehouses in any flexible sequence carrying materials to the right place just in time.

Until recently, traditional automated guided vehicles (AGVs) were the only option for automating transportation tasks. Today, however, AGVs are being challenged by the more sophisticated, flexible, and cost-effective technology of autonomous mobile robots (AMRs). While AGVs and AMRs both move materials from one place to another, that is where the similarities cease.

An AGV has minimal on-board intelligence and can only obey simple programming instructions. To navigate, it needs to be guided by wires, magnetic strips, or sensors, which typically require extensive facility updates to install, during which time production could be stopped. This phase can require some weeks.

The AGV is obliged to follow these pre-determined routes. Any change in the routes requires a new installation phase.

The AGV can detect obstacles in front of it, but it is not able to navigate around them, so it simply stops until the obstacle is removed.

In contrast, the AMR navigates using maps that its software constructs on-site or via pre-loaded facility drawings. This capability can be compared to a car with a GPS and a pre-loaded

set of maps. When it is taught the owner's home and work address, it generates the most direct path based on simple positions on the map. This is similar to the way the AMR is taught locations to pick up and drop of parts. The AMR uses data from cameras, built-in sensors and laser scanners as well as sophisticated software that enables it to detect its surroundings and choose the most efficient route to the target. It works completely autonomously and if fork-lifts, pallets, people, or other obstacles occur in front of it, the AMR will safely manoeuvre around them, using the best alternative route. This optimizes productivity by ensuring that material flow stays on schedule.
This autonomous operation also makes an AMR far more flexible than an AGV. AGVs are limited to following strict routes that are integrated into the facility, whereas an AMR can easily implement new routes with some software adjustments.

2.6.1.1 Automated Guided Vehicle

An Automated Guided Vehicle (AGV) system is a flexible way to provide a transport. It repeats transports safely and with high precision with vehicles especially designed for the transport. The vehicles need to find their way between loads and unload points. On their way they should not harm people or collide with other AGV's or stuff.
There is no typical AGV. The vehicles are as varied as the goods they carry. It can be a large vehicle carrying 20 tons steel rolls in a steel work or minimal carrier for microchips in a clean room environment.
As in a city full of cars, we need to decide where we will allow the vehicles to drive in the factory. To avoid traffic jams, we should also decide the traffic rules for the factory. The system designer does this by drawing a map with drive paths. The system designer will also decide driving directions that are allowed and what rules should apply in path crossings.
An AGV is equipped with a vehicle controller where the map is stored. The controller calculates and supervises the path to drive in order to reach the destination point. The drive and steer encoders report the distance and the direction that the vehicle has driven to the vehicle controller. This feedback is used by the vehicle controller to guide the vehicle. This method works as long as the motors and drive encoders are well trimmed. But mostly this is a quite inaccurate method if not used together with a supplementary position update method.
To secure that the vehicle maintains the desired path, complementary navigation methods are used:

- LIDAR Navigation
- Spot Navigation
- Magnetic Tape Navigation
- Inductive Wire Navigation
- Range Navigation
- Multi-navigation

Different navigation methods have different update rates where the laser navigation method together with wire navigation has the highest update rate followed by range navigation and spot navigation. The system designer draws the map with a layout definition tool. The system designer imports the factory drawing into the layout definition tool and draws the segments (drive paths). On each segment the system designer can control the vehicle behaviour, the speed can be adjusted and the driving direction has to be set.

Most AGVs are three wheelers with two supporting wheels in the back and a steering and driving wheel in the front. This is called a Steer-Drive (SD) vehicle. A QUAD vehicle has a more flexible movement pattern while it is equipped with two or more steering wheels and a number of supporting wheels. It can change direction without rotating and move sideways. The vehicle follows a pre-established path designed on a mapping tool. When it has to turn, it takes as reference point the point A in Figure 25 .

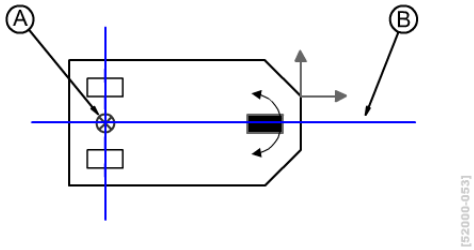

Figure 25 Vehicle (A=vehicle reference point, B=symmetry axis).

The vehicle receives information from the remote-control system about which segment in the layout it should drive next. The vehicle controller continuously updates its estimated position with the information from the position sensor (bearing/range scanner or spot/magnetic/inductive sensors) and encoders. The difference between a laser/range navigating vehicle and a spot/wire navigation vehicle is that the former can immediately drive the segments if there are reflectors/walls in the area where the vehicle is supposed to drive. A spot/wire navigating vehicle needs spots or magnetic/inductive wire placed in the floor along the vehicle path before it can drive the downloaded segments. Navigation based on reflector/walls provides therefore a more flexible solution when there are continuous changes in the layout. When a vehicle follows the track, it will adjust its position.
By using one or more of these navigation methods the estimated position of the vehicle is updated using the known landmarks (reflectors, walls, magnetic spots etc). If the updated position gives that the reference point is beside the guide path the vehicle computer will correct the vehicle position by controlling the steer and drive motors. For safety reasons the vehicle controller stops the vehicle if the reference point is too far away from the segment.
A vehicle can use more than one navigation method during its navigation The PLC program selects which navigation method to use, for example based on the segment the vehicle is running. This is called multi-navigation.

Table 9 Vehicle Navigation Combination

Navigation methods and combinations	Laser	Spot	Magnetic Tape	Range
Laser		x	x	x
Spot	x			x
Magnetic Tape	x			x
Range	x	x	x	

2.6.1.2 Navigation Systems

As mentioned above, laser navigation is the most flexible navigation system. Usually there is no need to move reflectors when adjusting drive segments in the layout. Spot navigation or wire navigation is used either in environments where there is no wall to mount reflectors on or the load the vehicle will carry prevents using a laser scanner on top or the layout is quite static. Spot or wire navigation is also a good choice in areas with deep stacking stations where the surroundings are dynamic. Range navigation can be used in environments like corridors and cargo compartment of a truck. Multi-navigation is the best approach when the requirements vary over different parts of the operation area.

2.6.1.3 Navigation Level

The navigation level is a confidence number ranging from 0 to 100 that the navigation system reports. When the vehicle controller receives measurements that are in line with its estimated position it increases the number. The number is reduced if no observation has been made for a time or if the measurements contradict the estimated position. When the navigation level reaches 0 the vehicle is geographical lost, and it will stop driving in automatic mode. To continue driving a new initialization of the navigation system is required. Only laser navigation can automatically reinitialize the navigation system. The other navigation methods require supervision by an operator.

2.6.1.4 LIDAR Navigation

In the environment of a reflector guided system reflectors are strategically placed in the operation area. They are all identical and consist of retro-reflective tape. The reflectors as well as the guide path (segments) are part of what it is called the layout. The layout data including position (X,Y coordinate) and angle (direction which the reflector is facing) of all reflectors are known to the vehicle controller.
A reflector guided vehicle uses a laser scanner to detect reflectors mounted in the environment. The laser scanner continuously measures the angles to the reflectors. These measurements are used in combination with steer and drive encoder signals to determine the position (X, Y) and angle of the vehicle relative to the global coordinate system.

In the installation work, the mounting position of the reflectors must be determined with high accuracy. It is recommended that a Reflector Surveyor software is used, or a surveyor is consulted. The surveyor must determine the position and angle of the reflectors. This is done automatically by a Reflector Surveyor SW. The reflector data is then imported to the layout definition tool and the information is downloaded to the vehicle controller.

2.6.1.4.1 Navigation Precision

The precision of a Laser navigated vehicle is dependent on several factors. Factors improving navigation precision are:

- Good distribution of visible reflectors
- Well measured reflector positions
- Good vehicles mechanics

- Good vehicle tuning
- Good floor conditions.

The maximum vehicle speed depends on the same things as for navigation precision. Driving too fast in the navigation area might result in a navigation failure.
There are different kinds of errors sources for the laser navigation.
Some examples are:

- Pipes and metal objects can reflect the laser beam with a signal similar to the reflector.
- With mirrors a reflector can reflect the laser beam from the laser scanner via the mirror and therefore be mistaken for a real reflector.
- In a corridor, reflectors that are close to each other but at a distance in depth can be mistaken for one reflector.
- Reflectors hidden by inventory.

2.6.1.5 Spot Navigation

A Spot guided vehicle uses landmarks placed in the floor. These landmarks are referred to as spot magnet and consist of magnetic markers. The spot magnets are placed in drilled holes in the floor. The vehicle is equipped with two or more magnetic sensors for detecting the spot magnets so that it can update its position when passing a spot magnet. Between the spot magnets the vehicle is guided using the feedback from encoders. If there is a deviation between the measured position of the spot magnet and the position in the map, the vehicle controller will adjust its position estimate.
Spot guidance is especially useful when reflector placement is difficult or if it is difficult to mount a laser scanner on the vehicle.
For instance, the layout could contain:

- Deep-stacking station (where reflectors on the wall is temporary hidden by load).
- Big open areas (larger than the range of the Laser scanner).
- Harsh environment areas.

The position of the spot magnets should be measured relative to a reference point. This information is entered into the layout definition tool. The finished layout is transferred to the vehicle controller. Normally, a vehicle is equipped with a front and a rear spot sensor. As the sensor can’t identify which spot magnet it is detecting there is a risk that the vehicle controller will use the wrong spot magnet to update its position. To minimize this risk there are safety functions.
The precision of a spot navigated vehicle depends on a number of factors.
Factors improving the navigation precision are:

- Short distance between spot magnets
- Well measured spot magnets for the spot magnet map
- Good vehicles mechanics
- Good vehicle tuning
- Good floor conditions

The maximum vehicle speed depends on a number of factors:

- Spot magnet size

- Spot magnet field strength
- Magnetic sensor height above the floor
- Sample period of the magnetic sensor.

Driving too fast over small magnetic fields, or having the sensor at the wrong height, would mean that the magnetic sensor might not detect the spot magnet causing navigation failure. Normally the spot system is insensitive to disturbance. An issue that could occur could be magnetic objects close or relatively close to a spot magnet. A magnetic object close to the spot magnet might cause abnormal signal detected by the antenna. This would result in navigation failure.
For safety, while using spot magnets for navigation the vehicle controller checks the following conditions:

- For all detected spot magnets there must be a spot in the map at the same position within a given tolerance
- Two consecutive spot detection from the same antenna must be associated to two different spots in map.
- The vehicle may not travel longer than a predefined distance without detecting any spot.

If any of the above does not hold true, then the navigation level will become zero and the vehicle stops driving in automatic mode. A system event is also generated.

2.6.1.6 Magnetic Tape Navigation

In magnetic tape navigation the vehicle makes use of magnetic tapes on the floor. The vehicle is equipped with two or more magnetic antennas for detecting the tapes and keeping the vehicle on track when driving in the layout. The vehicle may also have one or more digital sensors. The digital sensors detect markers placed along the magnetic tape which are used for lengthwise position updates. Magnetic guidance is especially useful when reflector placement is difficult in the environment or if it is difficult to mount a laser scanner on the vehicle. This navigation method is similar to spot navigation and very similar to inductive wire navigation.
For a good performance it is necessary that the layout provided to the planning tool and the placement of the tape on the floor have a good correspondence.
The precision of a magnetic navigated vehicle depends on a number of factors. Factors improving navigation precision are:

- Distance Markers for position updates along the path
- Well measured magnetic tape for the magnet map
- Good vehicles mechanics
- Good vehicle tuning
- Good floor conditions.

Distance markers are used, as described above, for position updates along the path. The distance markers are placed in the environment and drawn in the layout design tool. The best use of distance markers occurs when they are placed before curves and stations. These markers need to be surveyed as well as the sensor needs to be tuned for accurate position update.

Crossroads should be used as little as possible. This because when the antenna is near multiple navigation wires, it is not possible to determine which magnetic tape it is that generate the magnetic field. When the antenna is close to multiple magnetic tapes the measurements are ignored.
The maximum vehicle speed depends on a number of factors:

- The distance sensor height above the floor
- The distance sensor tuning accuracy.

Driving too fast might result in no updates of distance markers (length update) and might result in navigation failure. A magnetic object close to the magnetic tape or distance marker might cause abnormal signal detected by the antenna. This would result in navigation failure. For safety, while using magnet tape for navigation, the vehicle controller checks the following conditions:

- For all detected magnetic tapes there must be a navigation wire in the map at the same position with a specified tolerance
- For all detected distance markers there must be a marker in the map at the same position with a specified tolerance
- All drawn navigation wires in the layout plan that the antennas pass over must be detected
- The vehicle may not travel longer than a specified distance without detecting any magnetic tape.

If any of the above does not hold, the vehicle stops, and a system event is generated.

2.6.1.7 Inductive Wire Navigation

Wire navigated vehicles uses inductive wires in the floor for navigation. The vehicle is equipped with two or more antennas for detecting the wires so that the vehicle knows it is on track when driving in the layout. To update the position the vehicle uses markers on the floor that trigger a signal on the digital sensor when passing over the marker. Wire navigation is especially useful when either reflector placement is difficult in the environment or if it is difficult to mount a laser scanner on the vehicle or the layout is static. This navigation method is similar to magnetic tape navigation and, since the wires are in the floor, it is less sensitive to mechanical damages. Inductive wire navigated vehicles tracks its global position using antennas for detecting inductive wires in the floor. By comparing the detected wires with the expected navigation wires in the layout, the position can be determined. To improve the accuracy of its position estimate, the vehicle controller uses marker on the floor that trigger a signal on the digital sensor when passing over the marker.
The layout can be divided into frequency groups. This is a must if there is any fork in the layout. By using separated frequencies, the vehicle can choose another route when at a fork in the layout. The wires need to have different frequencies when they must be close or to share the same path in the floor to avoid disturbing each other.
The precision of a magnetic navigated vehicle depends on several factors. Factors improving navigation precision are:

- Distance Markers for position updates along the path

- Adjusted layout to the navigation wire
- Good vehicles mechanics
- Good vehicle tuning
- Good floor conditions.

Distance markers are used for position updates along the path. The distance markers are placed in the environment and drawn in the layout design tool. The best use of distance markers is when they are placed before curves and stations. These markers need to be surveyed as well as the sensor needs to be tuned for accurate position update.
The size of the distance marker depends on the speed at which the vehicle will pass the marker.
The maximum vehicle speed depends on:

- The distance sensor height above the floor
- The distance sensor tuning accuracy.

Driving too fast might result in no updates of distance markers (length update) or not detecting crossing wires with side antennas and might result in navigation failure.
An electric object close to the navigation wire or the distance marker might cause abnormal signal detected by the antennas. This might result in navigation failure.
For safety, using inductive wire for navigation, the vehicle controller checks the following conditions:

- For all detected inductive wire there must be a navigation wire in the layout map at the same position with a specified tolerance
- For all detected distance markers there must be a marker in the layout map at the same position with a specified tolerance
- All drawn navigation wires in the layout that the antennas pass over must be detected
- All drawn distance markers in the layout that the digital sensors pass over must be detected
- The vehicle may not travel longer than a specified distance without detecting any inductive wire.

If any of the above doesn’t hold, the vehicle stops and a system event is generated.

2.6.1.8 Range Navigation

In range navigation one or more range scanning sensors are used to measure distances to the surrounding environment. The layout should contain navigation walls describing the position of walls or other flat surfaces visible to the range scanning sensor.
Measurement points that are considered to be distances towards navigation walls are filtered. This is done by predicting the position of the wall relative the vehicle using dead reckoning based on encoder data. Measurements points inside a validation gate around the predicted position of the navigation wall are viewed as observation of the wall and used to correct the estimated position of the vehicle.
The precision of a range navigated vehicle is dependent on several factors. Factors improving navigation precision:

- Two or more visible non-parallel navigation walls
- Well measured positions of navigation walls
- Good vehicles mechanics
- Good vehicle tuning
- Good floor conditions.

The maximum vehicle speed depends on the same things as for navigation precision. Driving too fast in the navigation area might result in a navigation failure.
There are different kinds of errors sources for range navigation. Some examples are:

- Clutter in front of walls used for navigation
- The range scanner not horizontal
- Uneven floor
- Black surfaces.

The navigation level is updated based upon how well the measured range data correspond to the estimated position of the AGV and the navigation walls in the map. If no walls are visible or no measurement points could be used for correcting the position estimation, then the navigation level is reduced. When there are measurement points in the range scan that form lines that correspond to navigation walls in the map the navigation level is increased.

2.6.1.9 Multi-navigation

A vehicle can, while driving, switch between navigation methods. This is called multi-navigation. It gives the possibilities to use the flexibility of a laser navigation in factories with one of the other navigation methods in areas where there is not possible to mount reflectors.
The switch of navigation method is controlled by a PLC program, usually with the help of segment triggers. When changing navigation type, it is important to have an overlapping area that supports both navigation types. Some care is needed to make sure that the two navigation methods do not differ too much in the positions they report.
When switching to laser navigation the vehicle controller starts matching incoming bearings from the laser scanner with reflectors in the map. If the bearings can be coupled with known reflectors the measurements will be used to update the estimated position and the vehicle will switch using reflector without stopping. However, if it is not possible to find matching reflectors the navigation level will start to decrease and when it reaches zero the vehicle stops and tries to reinitialize the navigation system. If the navigation initialization succeeds and the new estimate of the vehicle position is still within the safety-zone around the predefined path, then the vehicle will start driving again.
Going from a reflector area to a spot magnet area is a more critical transition than the opposite.
When entering a wire navigation area, the vehicle should have a position such that the antennas are above the wire before switching to wire navigation. This will allow the vehicle to adjust according to the wire before docking at a station or negotiating a curve.
When entering a range navigation area, the vehicle should have a position such that the range detector is able to detect visible and known walls before switching to range navigation.

2.6.2 *Autonomous Mobile Robot*

Autonomous Mobile Robots (AMRs) are an evolution of AGVs. Respect to AGVs they don't need any auxiliary device on the floor or on the walls to position themselves in the environment and navigate. They rely just on odometry, LIDARs, cameras and similar onboard sensors. An advanced AMR can require a complex data processing to navigate. A promising choice is to move this processing on a remote cloud. The requirement to be fulfilled in this case is that the control loop between the AMR and the remote controller has an RTT not greater than 100 ms., possibly 50-80 ms. This RTT is short enough to guide any vehicle up to 50 km/h (that is 1.3 m every 100 ms.). This is also stated in the use cases reported by ISO TC 204 and ETSI TC ITS, that are leading the standardization process. Usually operating speeds are much lower from 6 to 10 km/h.
So, for navigation purposes a refresh rate lower than 100 ms. is enough. In any case, as it occurs for AGVs, AMRs have some safety functions onboard to stop the vehicle in case of emergency. For instance:

- The vehicle stops in case of loss of contact with the remote controller or uses a local program to go back to a specific position
- The vehicle stops in case a proximity sensor detects an obstacle too close

Moving the control function remotely, the AMRs include only low-level controls, sensors and actuators. The control services can be put in a local cloud running on dedicated hosts or datacentres.

2.6.2.1 Sensors characteristics

For measuring distances and detect obstacles motion detectors, ultrasonic sensors, LIDARs, mono and stereo cameras can be used. The choice of the sensors depends on the application and is a trade-off between performance and cost. Sensor's characteristics are summarized in Table 10:

Table 10 Sensors characteristics

Sensor	Precision	Coverage	Rate	Cost	Environment
Motion Detector	Very low (~50cm)	~100° x 3-7m	~50Hz	Low	Indoor/Outdoor
Ultrasonic Range	Low (~20cm)	~15° x 2-400cm	~40Hz	Low	Indoor/Outdoor
Stereo Camera	Medium (~10cm)	~90° x 2-10m	~15Hz	High	Indoor/Outdoor
RGB-D Camera	High (~5cm)	~90° x 70-500cm	~25Hz	Medium	Indoor
LIDAR 1D	Very High (~1mm)	4cm-40m	~10Hz	Very Low	Indoor
LIDAR 2D	High	240° x 6cm-20m	~50Hz	High	Indoor/Outdoor

	(~2cm)				
LIDAR 3D	High (~2cm)	270°-360° x 10-20° x 0.5m-60m	~100Hz	Very High	Indoor

A stable navigation requires the fusion of information from multiple sensors like the wheel encoders, a LIDAR and cameras.
Wheel encoders, providing information on wheels speed and position, can be a primary source of information for tracking the path of the robot, but they need to be integrated with further sensors due to non-negligible effects of slippery that can affect wheels when moving. A second source of information can be a sonar or a LIDAR. Sonars have a low precision and can be adopted mainly for short range movements. LIDARs, especially 2D LIDARs, can be a good compromise between cost and accuracy for mid and long-range navigation and for collision avoidance purposes, both indoors and outdoors, offering an angle width of 240° with an operating range from 6 cm to 20 m.
However, LIDARs are sensitive to sunlight causing the detection of fake targets that can confuse the navigation functionality, causing the loss of the positioning. Cameras can help in solving such situations, having the possibility of detecting reference patterns, objects and scenes. Combining cameras and LIDAR, that provides depth information, stereoscopy is not necessary, simplifying the data processing. Stereo-images are anyway a good option for navigation tasks. Real-time image processing is a difficult task when operating in a generic environment where light conditions can drastically change time by time. Pre-processing of the image for colour normalization, and image equalization are typically required. Pattern matching can be based on several techniques requesting the extraction of image descriptors and the search in databases for classification.
Video analysis requires a huge amount of computation. The latter can be easily made exploiting the computation capabilities the cloud can offer. However, acting this way, latency can become an issue for the stability of the robot navigation.
In perspective, having no restrictions in computational power, vision can become the primary sensor, but still a lot of work for having a reliable detection, recognition and positioning of objects with cameras in all light operating conditions is matter of investigation.
An aspect to be taken into account is the amount of bandwidth required by the different sensors. In Table 11, some reference values are reported.

Table 11 Examples of data bandwidth of provided by common sensors for robotic localization and navigation. Data can change according to several factors, such as frequency, resolution, etc.

Data	Bandwidth
Laser Scan	60 kbps
Odometry	15 kbps

High resolution cameras require much more bandwidth. It depends on several factors like resolution, frame rate and coding. The rate can vary from few hundred kbps to about 10 Mbps

per camera. For navigation purposes a high quality and high-resolution images are required, so the bandwidth is usually close to 10 Mbps.

2.6.2.2 Remote control

The control system for an AMR requires a set of functionalities, including data analytics and vehicles coordination management, the vehicle navigation service, the vehicle data processing and pattern recognition services. These services can be placed in cloud under the control of a cloud management service.
The management service is used to coordinate AMRs activities and requests for services by users, assigning vehicles to requested tasks according to an optimization criterion like minimization of paths length in case of navigation. The navigation service takes care of calculating the track the vehicle should follow to reach its destination based on its current position and sensors data. It is also in charge of the collision avoidance mechanism. The data processing and pattern recognition services are used to detect the context and the AMR position in the environment.
Control services can be divided between time critical and non-critical. The time critical ones, like navigation, sensors processing and pattern recognition, must be placed in the local edge cloud near the base-station serving the working area, to keep latency low, guaranteeing stability and a proper reaction time. Time non-critical services, like the management function, can be placed remotely since their action don't affect real time behaviours.

2.7 Positioning and Flow Monitoring for Controlling COVID-19

Most recently, with the COVID-19 spreading, national authorities have enforced several countermeasures, including restrictions on the people mobility, to slow down the disease. The human mobility is the dominant factor that drives the evolution of the contagion. It is therefore necessary to develop mathematical models that can predict and provide information about the possible space-time evolution paths of the pandemic considering the influence of human mobility. The need of such models emerges for adopting efficient, specific, and timely countermeasures. From this perspective, the models could be used as a support to national authorities and medical care systems in planning in advance resources allocation as well as operational plans.

The objective of this work is to develop a stochastic epidemiological model able to predict the evolution of the COVID-19 spreading. The human mobility plays a central role, therefore a specific model describing in probabilistic terms the people flows has been developed. The whole stochastic model is therefore composed of two different parts: i) epidemiological model; and ii) stochastic model for human mobility. The former has been developed as an extension of the classical compartmental susceptible-infectious-removed (SIR) model by leveraging the Kermack-McKendrick theory [31], [32] . In particular, six compartments are defined: susceptible, infectious, detected infectious, hospitalized, quarantined, and removed. Such compartments are related by the epidemiological parameters. The human mobility model has been chosen via model order selection to identify the model which best fits the

characteristics of the people flows among different locations. Being the human mobility intrinsically stochastic, the whole model is mathematically defined in terms of non-canonical stochastic differential equations (SDEs).

The disease dynamics are related to the variation, over the time, of the size of the compartments. However, given the high mathematical complexity of the model and the lack of closed form solutions, stochastic analysis has been leveraged and a numerical solution is obtained by using the general explicit Runge-Kutta numerical methods. Furthermore, being the trade-off between prediction accuracy and computational complexity a crucial aspect to consider for the solution of the SDEs, as well as the capability to cover all the possible trajectories that an epidemic could take, the numerical solutions are obtained by performing a second order explicit Runge-Kutta numerical method through a Montecarlo simulation.

Case studies are presented for two scenarios in which the prediction of the COVID-19 outbreak is computed for both Emilia-Romagna and Lombardy regions in Italy. The period used for the prediction consists of 30 days (from 02/24/2020 to 03/25/2020). The transition probabilities (associated to the mobility between provinces in the two regions) are estimated by using maximum likelihood estimation over the dataset provided in [4] , which is composed of triples having the following structure: source province, destination province, and a time series reporting the percentage of population moving between source and destination provinces at each day. In particular, the data records are generated following a specific processing pipeline: first, an anonymized set of active users (those with at least one data record with anonymization consistent with European GDPR) is created by analysing the users' activities in given pre-outbreak period. Then, the mobility data are processed as follows: i) using a 5-minutes window, a time aggregation is computed to remove the short-time dynamics and the users' location in such a time-window is computed as the geometric center of all recorded users' (latitude, longitude) pairs within such a window; ii) a spatial aggregation is performed to assign a source province and a destination province to each user with the source province is chosen as the most frequently visited location within the time interval 00:00-6:00, while the destination province is chosen as the province where the user remained for most of the time (at least 1 hour); and iii) for each day, a source-destination matrix is computed where each entry (i, j) of the matrix represents the number of users that moved from the province i to the province j.

The epidemiological parameters of the model are derived from [33], [34]. We chose a pessimistic assumption in which the parameters are kept fixed and describe the worst pandemic scenario in the whole period of simulation. In [33] a mathematical framework to estimate the epidemiological parameters of a classical SIR model is given, while in [34] a deterministic model to describe the COVID-19 evolution in Italy is offered. Furthermore, such a model provides useful information on the epidemiological parameters of the disease. The results illustrate the predicted evolution, for Emilia-Romagna and Lombardy regions, of the infectious and hospitalized compartments compared to the real data provided by the Dipartimento della Protezione Civile, Italy.

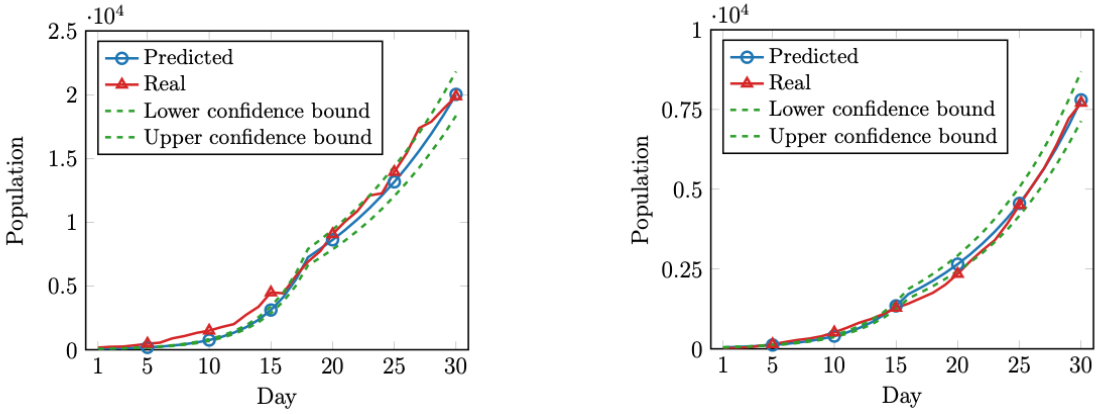

Figure 26 (Left) Infectious evolution for Lombardy - (Right) infectious evolution for Emilia Romagna

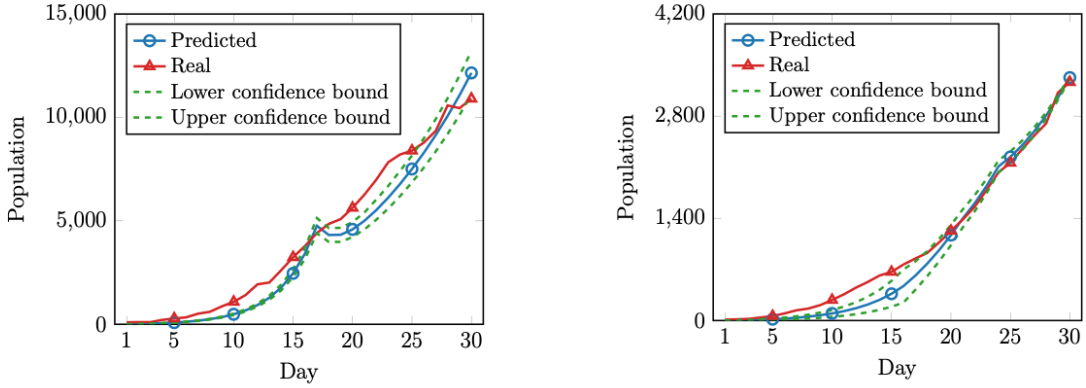

Figure 27 (Left) Hospitalized evolution for Lombardy - (Right) Hospitalized evolution for Emilia-Romagna

As example results obtained, Figure 26-left and Figure 26-right show the evolution of the infected compartment of Lombardy and Emilia-Romagna. It can be noticed that both the predicted curves have a similar trend compared to the real ones. Furthermore, starting from the 15th day we can observe a small deviation of the prediction with respect to the real data. This is attributed to the enforcement of countermeasures (i.e., lockdown and limitations on human mobility) that impact significantly on the dynamics of the disease. Figure 27-left and Figure 27-right show the predicted hospitalized compared to the real ones.

It is important to notice that, thanks to the flexibility and the parametrization of the proposed model, the latter can be adapted to follow the current epidemiological situation, as well as to provide useful information in evaluating the possible trajectories that the epidemic could take due to the enforcement of alternative countermeasures.

In addition, for each figure, lower (5%) and upper (95%) confidence bounds are represented in green. It can be noticed that as more the simulation time grows, as more the amplitude of

the confidence interval increases. This is coherent with the theoretical results. Indeed, using a second order Runge-Kutta method, the approximation error is an asymptotic of the third order. As more the simulation time increases, as more the global approximation error, which includes the error of each iteration, grows up.

This work proposed a stochastic model for predicting and evaluating the possible space-time evolution paths of the COVID-19 pandemic by exploiting human mobility data. Case studies are presented for two scenarios in which the prediction of the COVID-19 outbreak is computed for Emilia-Romagna and Lombardy regions in Italy. The results show that such a model provides, since the outbreak, reliable predictions despite of the disruptive growth of the pandemic. From this perspective, the model could be used as a support to national authorities and medical care systems in planning in advance resources allocation as well as operational plans.

2.8 Transportation optimization based on the identification of traffic profiles

Continuing the work presented in previous deliverable (D5.1) Incelligent proposes and advanced solution for the problem of trajectory prediction. The solution is compared to two baseline methods, which are considered state-of-the-art for the problem. Additionally, this solution is extended for the task of trajectory profiling, by fitting a supervised learning method to mobility analytics extracted by the trajectories.

2.8.1 Dataset Description

For the first task, we train and evaluate the three different deep learning architectures (2 baselines and the proposed model) on two different datasets. The first, ETH-UCY dataset, is a publicly available pedestrian dataset that contains 3161 pedestrian trajectories from 5 different mobility scenes. The data is captured at 2.5Hz frequency, resulting in time-delta of 0.4s between trajectory steps.
In order to generalize the solution, the models were trained and evaluated in another dataset, simulated by Incelligent's simulator, containing both pedestrian and car trajectories, in a realistic mixed indoor-outdoor mobility scenario taking place in McArthurGlen shopping center in Athens. The simulation produces realistic mobility patterns by utilizing a probabilistic model for the connection of the different areas of the scene and generates trajectories of stochastic velocity and movement.
Before the split of the datasets to train and test sets, a pre-process step of data augmentation is performed to generate sufficient training data. This is implemented with the sliding window method for trajectories of more than 20 steps. Thus, after the augmentation procedure, all distinct trajectories contain 20 steps, 8 of which are used for training and 12 for inference and evaluation. After this pre-process step there are 65000 and 225000 trajectories in the ETH/UCY and simulated dataset respectively.
For the second task of trajectory profiling, experiments were conducted in the Incelligent simulated dataset as well as another publicly available dataset, OpenPflow, which also contains different mobility agents, like pedestrians, bicycles, vehicles, trains and buildings.

Figure 28 Snapshot from the Incelligent simulator with the mobility areas (roads, parking lots, shops)

2.8.2 Implemented Methods

Task 1: Trajectory Prediction

For this task 3 different deep neural architectures were implemented, trained, and evaluated. Two of them were chosen from the current bibliography as state of the art and the third, which was inspired by the baseline models, is the proposed architecture.

Social GAN

The first one is Social GAN, one of the first GAN architectures proposed for the task and is the only socially aware approach presented in this contribution. Socially aware is a predictive model that takes other agents' trajectory into account into training and inference stage. It is composed by two different neural networks, the generator and the discriminator, that train each other in parallel in an adversarial way.

The generator is an Encoder Decoder model with LSTM layers, which is trained to generate new trajectories. Between the Encoder and the Decoder there is a Pooling Module, which is ultimately a Max Pooling operator and is responsible for modelling of social interactions between agents. This helps the generator to produce non colliding trajectories.

The Discriminator is fed with both real and generated trajectories and classifies them as real or fake as well as socially acceptable or not. Thus, the generator model's loss is increasing whenever the discriminator correctly classifies a fake trajectory or decides that a generated trajectory is not socially acceptable due to a conflict with another agent's trajectory.

Social GAN's architecture is presented in Figure 29.

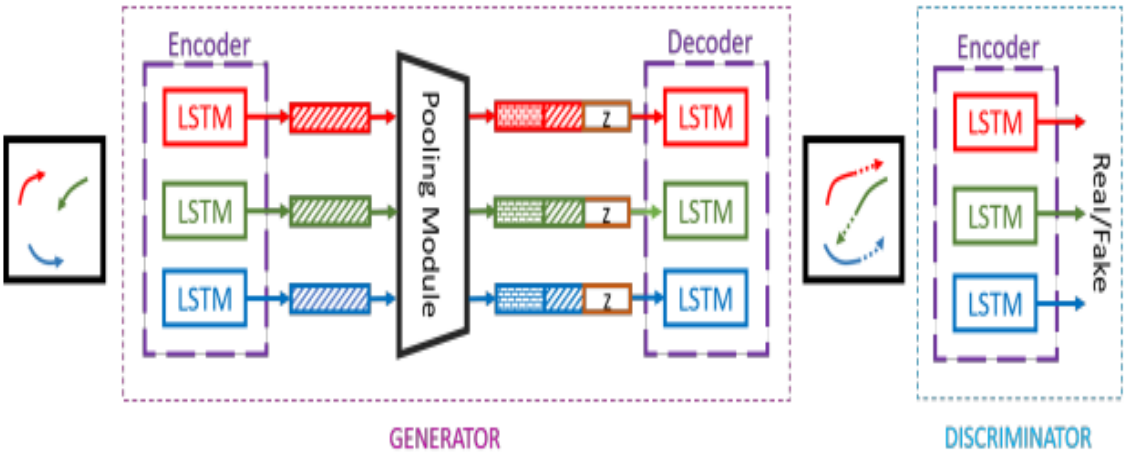

Figure 29 Social GAN architecture

Trajectory Transformer

The second model, Trajectory Transformer, is also an Encoder Decoder architecture but relies entirely on Attention modules, without using recurrent layers in any stage of the computation. Attention mechanism is a way of calculating the interdependence of different components of a timeseries. Within the attention module, each sequence entry, named Query (Q) is compared with every other entry, named Key (K), by a dot product scaled with the dimensionality d of both Q and K. This dot product vector is used to weight the respective sequence entries, now called Values (V).

The same attention mechanism is present in the decoder too, but in the form of masked attention, to mask out the future timesteps of the trajectory. It is also important to highlight that, despite the O(n^2) complexity of the calculation, attention is a fully parallelizable, unlike LSTM units.

Before trajectories are fed to the network, there is a positional encoding steps, which adds the timestep feature as an extra dimension to the trajectories. Both the encoder and the decoder are composed by similar blocks of attention modules, followed by feedforward layers. All blocks implement residuals connections between the input and output tensors.

Trajectory Transformer architecture is presented in Figure 30.

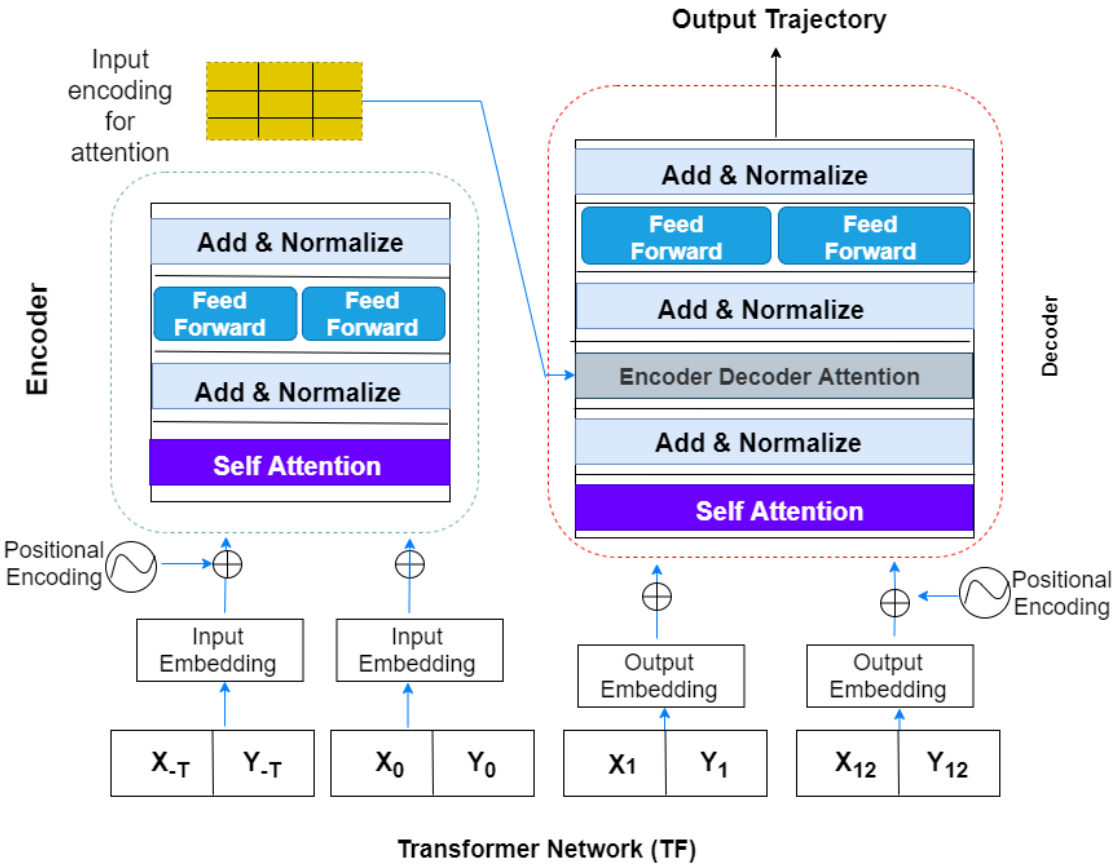

Figure 30 Trajectory Transformer Architecture

Proposed architecture

The model presented in this paragraph is the proposed architecture for the task of trajectory prediction. It takes components from both baseline architecture. The main goal of this contribution is to compete the performance of the state-of-the-art models on the task, by reducing the computational complexity of the solution, since SotA models need huge resources and training periods to produce satisfactory results.

The proposed architecture follows the encoder decoder scheme with GRU as the core processing units. An attention module was added between the encoder and the decoder as it proved to add value to time series forecasting in general. For the attention calculation, Query, Key and Value vectors can be directly obtained by the hidden and output states of the GRU layer. The output of the Attention module is then fed to the decoder, which consists of another GRU layer followed by a Dense layer with Rectified Linear Unit (ReLU) activation. The final output sequence is produced by the output Linear layer of the network. Mean squared error (MSE) was used as the loss function for the training procedure. The proposed architecture is shown in Figure 31.

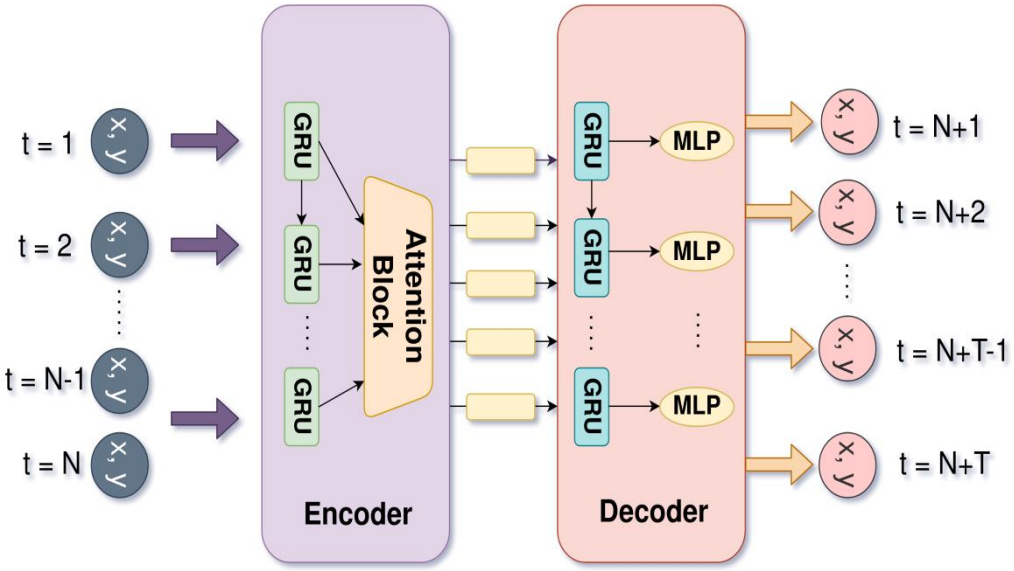

Figure 31 GRU with Attention Architecture

2.8.3 Results and Evaluation

Task 2: Trajectory Profiling

To implement the trajectory profiling solution, some analytics are extracted from the trajectory datasets. Three features are calculated, maximum distance covered, the maximum velocity and the maximum acceleration achieved per trajectory. These features should be indicative and able to separate agents in distinct mobility classes. A new dataset is constructed with these three as the input features and the mobility class as the target variable. Different classification algorithms were evaluated during the experimental process, with XGBoost classifier being selected for the final solution as it achieved the best evaluation metrics in a very reasonable training time

Results and Evaluation

For the evaluation of the trajectory prediction models the two metrics used are:

- **Average Displacement** Error (ADE), the mean Euclidean error of the real and predicted trajectory. It can be calculated as per equation 2.9

- **Final Displacement** Error (FDE), the Euclidean distance between the destination point of the real trajectory and the final point of the prediction. It can be measured using equation 2.10.

Table 12 shows the achieved results for the ETH/UCY dataset. It is prominent that the proposed model performs very well on the task, as it outperforms both baselines. Figure 32 shows these results in a form of bar-plot.

Table 12 Results in ETH/UCY dataset

Model	FDE	ADE	MDE
Social GAN	1.24	0.60	0.92
Transformer	0.95	0.43	0.69
GRU	**0.88**	**0.41**	**0.64**

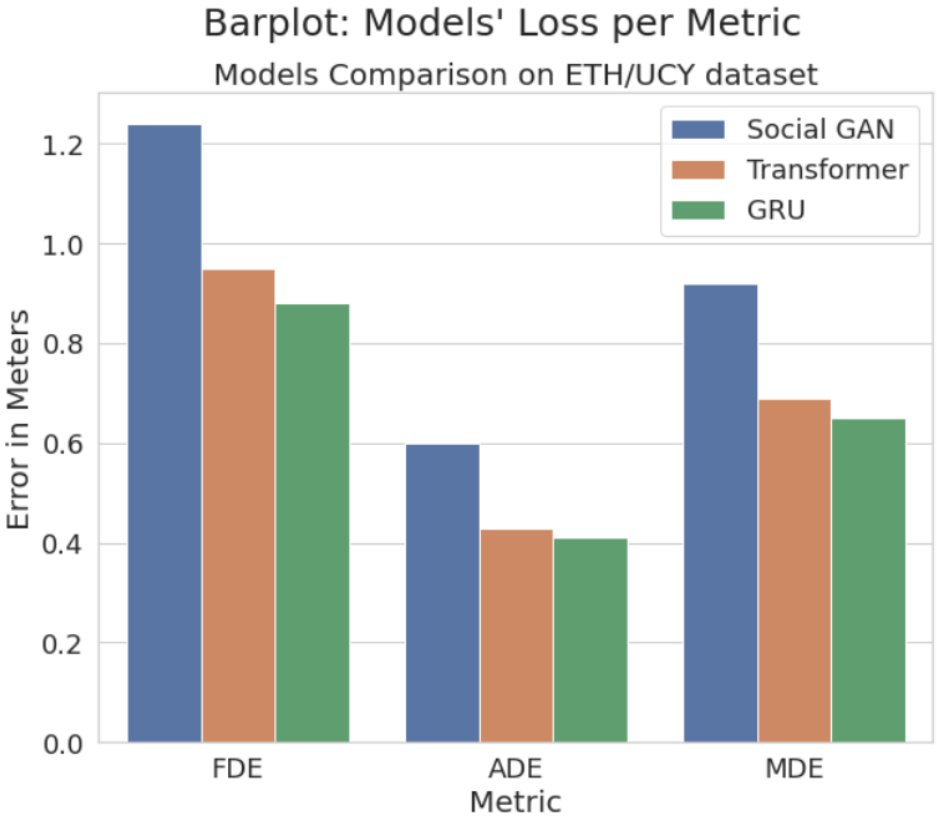

Figure 32 Results in ETH/UCY dataset

Table 13 shows the achieved results for the ETH/UCY dataset. Again, the proposed architecture outperforms baselines, with a large gain in FDE. Figure 33 shows these results in a form of bar-plot.

Table 13 Results in Incelligent simulated dataset

Model	FDE	ADE	MDE
Social GAN	6.96	3.54	5.25

Transformer	1.5	0.62	1.06
GRU	**0.77**	**0.44**	**0.61**

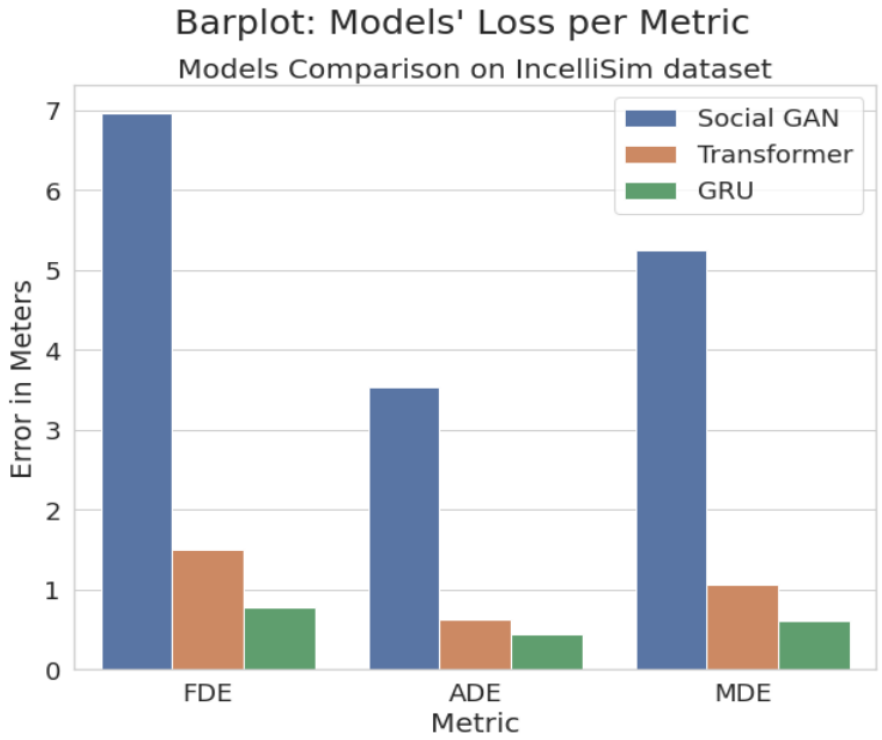

Figure 33 Results in Incelligent simulated dataset

Finally, Table 14 presents the computational comparison of the three different architectures, in terms of number of hyperparameters. This highlights the main advantage of the proposed architecture, which can be implemented and trained with very limited computational resources.

Table 14 Computational Comparison of the three models

Model	**Number of parameters**	**Ratio to GRU**
GRU	43.736K	1x
Social GAN	4008.387K	92x
Transformer	44156.667K	1010x

Table 15 presents the classification results for the trajectory profiling model on the Open-PFlow dataset.

Table 15 Classification results for trajectory profiling

Class	Precision	Recall	F Score	Support
Pedestrian	0.81	0.94	0.87	15437
Vehicle	0.87	0.85	0.86	8500
Train	0.75	0.59	0.66	2450
Bicycle	0.85	0.69	0.76	5901

2.8.4 Application in PoCs

Figure 34is presented a proposed solution to be included in the PoC3. It combines functionalities from other WPs, like fingerprinting position from WP3 with trajectory prediction / profiling, to provide a notification system for possible collision between moving objects (pedestrian, vehicles etc), which can have applications in various scenarios taking place in mixed indoor-outdoor environments.

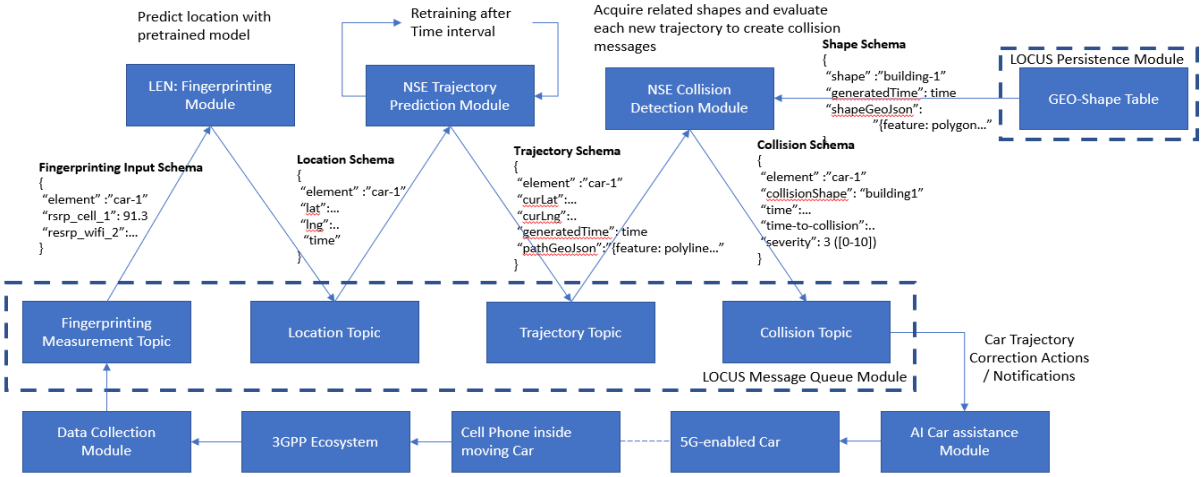

Figure 34 Solution integration flow diagram for PoC3

3 Virtualization of Analytics Services and Pipelines

This chapter provides a description of the experimental work carried out for the virtualization of the NSE UCs machine learning pipeline starting from the results reported in the deliverable D5.3; in particular, the progresses achieved with the Open-Source frameworks Seldon and Kubeflow have been exploited to deliver a complete Pipeline as a Service performing all the steps needed, from the training of a machine learning model to the deploying of the latter in a virtualized environment.
As described in D5.1 and D5.3, the UC1 Functionality-1 for identifying crowd mobility patterns machine learning pipeline software code has been used to carry out all the experiments done for the virtualization of the machine learning pipelines. Figure 35 shows the referred machine learning pipeline.

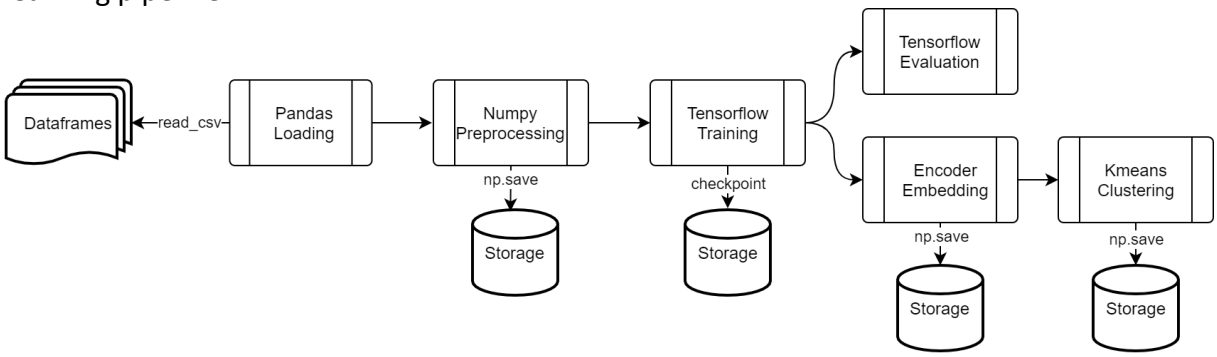

Figure 35 Reference Machine Learning pipeline

In D5.3 Seldon was used to perform the serving of pre-trained models generated from the execution of the reference machine learning pipeline code; one model for the encoding of the input data and one for the clustering prediction, both in a single-model serving mode and with an inference graph serving mode. Kubeflow, instead, was used for the virtualization of all the steps of the reference machine learning pipeline, except for the serving step; each step was a self-contained user code executed in the Kubernetes virtualized environment. The natural continuation of the work reported in D5.3 consists in the integration of the two aforementioned tools and the results achieved through them: so, the realization of a completely virtualized machine learning pipeline using Kubeflow and inserting, as last step, the integration with Seldon in order to serve the models trained in the previous virtualized steps.
Kubeflow, as mentioned in D5.1, comes with a number of services that could be used in the realization of machine learning workflows, among all these services also Seldon is provided: the idea is to exploit the Seldon deployed with Kubeflow to realize the serving step of the virtualized machine learning pipeline. Furthermore, since in D5.3 the chained serving of the encoder and clustering models, realizing an inference graph, was successfully achieved, the targeted Seldon deployment file to be instantiated in the Kubeflow serving step is the one used to deploy the Seldon graph, thus generating automatically, in the Kubernetes cluster where Kubeflow and Seldon are deployed, a pod containing all the services that were described in D5.3 and are needed to provide the serving of the models that compose the graph. The integration of the models serving with Seldon in the virtualized Kubeflow pipeline made clear the need to store the models trained in the previous step of the pipeline in a storage service that could be accessed by Seldon. Once the graph deployment file is applied in the last

step of the virtualized Kubeflow pipeline in the Kubernetes environment, the trained models can be automatically fetched and then be deployed and chained forming the Seldon inference graph. To supply this, an s3-compatible object storage, MinIO in our case, was deployed in the Kubernetes environment where also Kubeflow, and Seldon within it, is deployed. For the sake of completeness of the document, the single node Kubernetes cluster environment, provided by Microk8s, used to validate the Kubeflow approach is shown in Figure 36. The architecture is similar to the one reported in D5.3: compared to the previous one, the minio-system and models namespaces were created in order to be used, respectively, to deploy the MinIO object store instance and to deploy the models served by Seldon.

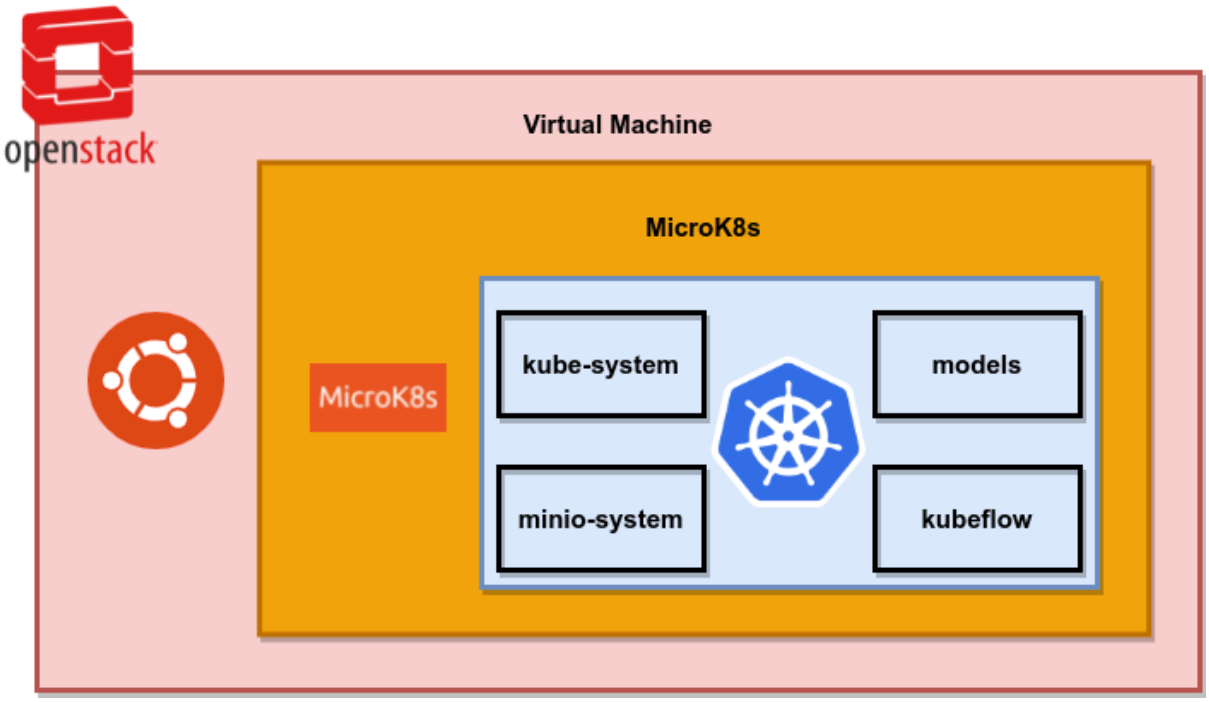

Figure 36 Microk8s - Kubeflow environment

The introduction of the need to save the trained models in an external storage entity in order to be used in the deploying step implies having to supply the pipeline of an interaction function with the MinIO instance: like the other steps of the Kubeflow pipeline already documented in D5.3 (data loading, pre-processing of data, etc.…) this function has been implemented as a reusable Kubeflow pipeline component; this is also particularly relevant since the models to be persisted in the object storage are two, encoder and clustering, opening to the possibility to show the reusability of the Kubeflow components either within the same pipeline or across different pipelines.
The Kubeflow pipeline generated from the reference machine learning pipeline including all the steps described in D5.3 plus the new steps introduced as reported above is shown in Figure 37.

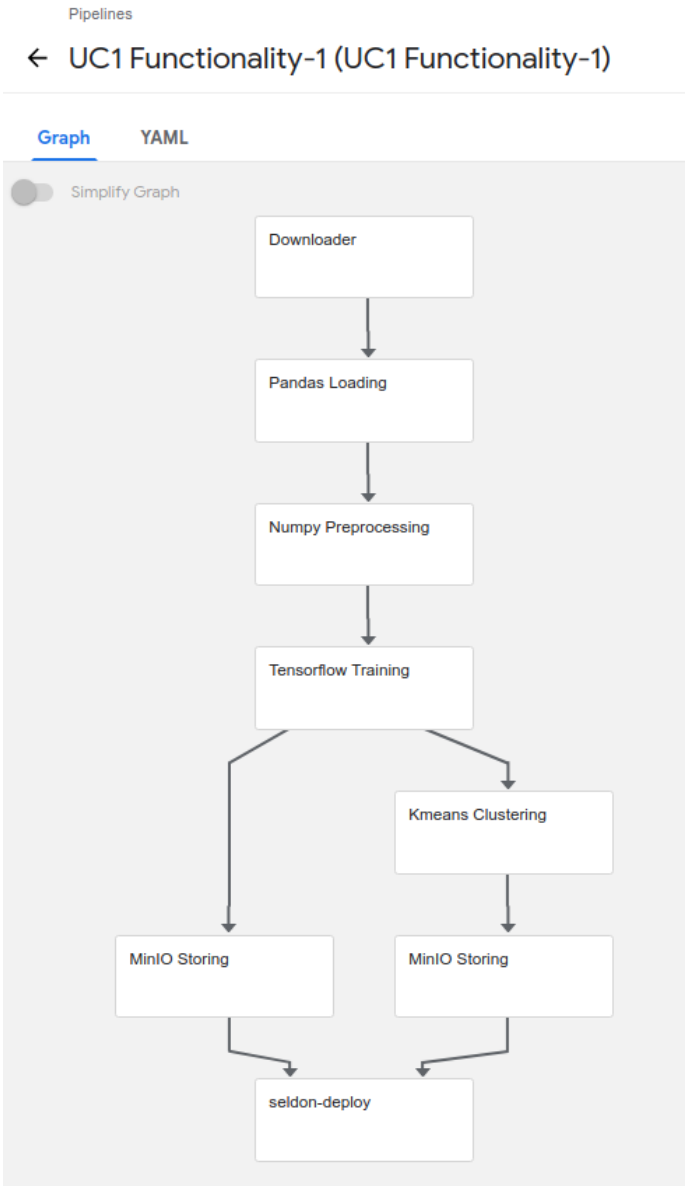

Figure 37 Fully virtualized machine learning pipeline

3.1 Kubeflow pipeline: Model Storing

Following the same approach described in D5.3 and summarized in Figure 38 for the realization of a Kubeflow pipeline component, the pipeline step for the persistence of the trained models has been implemented; the MinIO python client has been used to implement the logic of the component that handle the storing of the model in the deployed object storage instance.

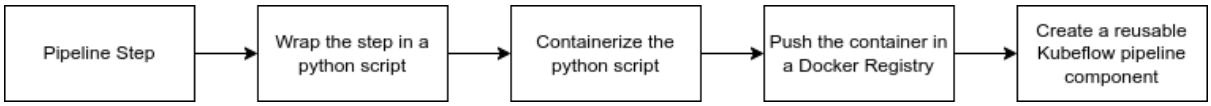

Figure 38 : Create a reusable Kubeflow pipeline component

As aforementioned, this Kubeflow pipeline component has been imported in the code that describes the Kubeflow pipeline following the same approach described in D5.3. Here, it is integrated only once but has been used twice in order to save the two models generated by the reference machine learning pipeline that was virtualized. In order to reuse the model storing Kubeflow component in the pipeline, the component itself was built in a way that the model to be stored in the MinIO object storage can be passed as an argument of the Kubeflow component as well as the credentials needed to access the MinIO instance; the latter are also arguments of the virtualized pipeline itself. As described in D5.3 each Kubeflow pipeline component could have a number of inputs and outputs (described in the Kubeflow pipeline component definition and used in the python code that implements a given step of the reference machine learning pipeline), in the case of this Model Storing component the model that have to be persisted is, both times, an output of a previous step of the virtualized pipeline: Tensorflow Training component for the encoder model and Kmeans Clustering for the clustering model, as shown in Figure 37.

```yaml
apiVersion: machinelearning.seldon.io/v1
kind: SeldonDeployment
metadata:
  name: locus-graph
  namespace: models
spec:
  name: locus-graph
  predictors:
  - componentSpecs:
    - spec:
        containers:
        - image: mikeno/clustering:latest
          name: clustering
          imagePullPolicy: Always
    graph:
      name: encoder
      implementation: TENSORFLOW_SERVER
      envSecretRefName: seldon-init-container-secret
      modelUri: s3://locus-bucket/locusencoder
      children:
        - name: clustering
          type: MODEL
    name: default
    replicas: 1
```

Figure 39 Seldon Graph deployment file

Figure 39 shows the Seldon Graph deployment file that was used in D5.3 and that will be used in the last step of the virtualized reference machine learning pipeline. From D5.3 we can recall how the serving of the encoder model was performed by one of the defaults serving servers made available by Seldon, Tensorflow Server. This was possible because the default Tensorflow Server was suitable to serve the generated encoder model: the server offers the possibility to load a model from an external object store and the encoder model expects the format of the data to make a prediction on to be in the same format that are passed to the server itself. For the clustering model, instead, the default servers made available by Seldon were not suitable to handle the serving, so, an ad-hoc Seldon server was built using the seldon-core-microservice python wrapper; in particular, a non-reusable model server was built (Figure 40).

Figure 40 Non-reusable Model Servers

In a non-reusable model server, the model to be served and exposed is built-in in the image of the server itself so, for every model that could be served by this server or for every new version of the same model a new build of the image of the server is needed. Since in the virtualized Kubeflow pipeline the clustering model is generated and trained at runtime the use of a non-reusable model server was not suitable and, instead, a reusable model server was built (Figure 41).

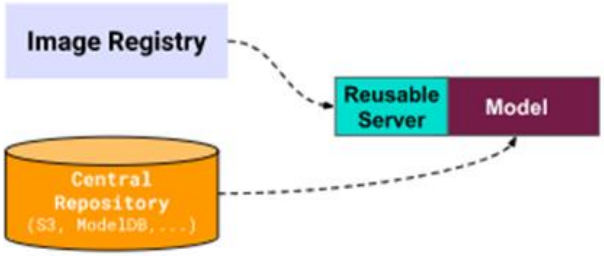

Figure 41 Reusable Model Servers

Reusable model server could be used to serve similar models, that could be completely different models or different versions of the same one, fetching the model that has to be exposed from a central model repository. Our central model repository is provided by the MinIO object store, where the clustering model will be saved automatically by the reusable Kubeflow pipeline component when the pipeline is executed. In order to build the reusable model server for the clustering model the non-reusable model server reported in D5.3 for the clustering model was taken as starting point: in fact, of the two functions that make up a model server, *__init__* and *predict*, only the initialization function has been changed in order to load the model that has to be served from the MinIO instance in which was saved by the Kubeflow Model Storing component. Figure 42 shows the use of the python MinIO client to retrieve the model from the object store.

```
def __init__(self):
    client = Minio('10.30.8.38:30745', 'minioadmin', 'minioadmin', secure = False)
    try:
        response = client.get_object("clustering", "clustering.joblib")
        with open('clustering.joblib', 'wb') as fd:
            for d in response:
                fd.write(d)
    finally:
        response.close()
        response.release_conn()

    self._model = load('clustering.joblib')
```

Figure 42 Initialization function for reusable Model Server

3.2 Kubeflow Pipeline: Seldon Deployment

The last step of the Kubeflow pipeline, for the virtualized UC1 Functionality-1 machine learning pipeline, is the deployment of the Seldon inference graph, shown in Figure 39. This deployment step, unlike all the previous of the Kubeflow pipeline, was not realized building a reusable Kubeflow pipeline component but directly making use of the Kubeflow Pipeline SDK (Kubeflow, 2021). The Kubeflow Pipeline SDK allows you to interact and manipulate the resources of the Kubernetes cluster where Kubeflow is deployed: operations such as *get*, *create*, *apply*, *delete*, *replace* and *patch* can be performed targeting the Kubernetes cluster, thus allowing to apply automatically the Seldon graph deployment file. Figure 43 shows the use of the python Kubeflow Pipeline SDK to apply the Seldon graph deployment file that was previously loaded inside the code of the Kubeflow pipeline; to be noticed that this last step is executed only after the persistence of the encoder and clustering models in the MinIO instance.

```
# Start Graph Seldon Deployment
seldon_deploy = kfp.dsl.ResourceOp(
    name = "Seldon Deploy",
    k8s_resource = deployment_dict,
    action = 'apply'
).after(encoder_model_storing, clustering_model_storing)
```

Figure 43 Seldon graph deployment code

Figure 44 shows the execution of all the steps of the virtualized reference machine learning pipeline. The execution of the pipeline was started from the Kubeflow Pipeline Dashboard that was directly mapped to a node port with a custom Kubernetes Service due to an issue that prevented the creation of new pipeline runs from the dashboard itself. Figure 45 shows the running inference graph deployed by the last step of the virtualized reference Kubeflow pipeline in the Kubernetes cluster where also Kubeflow is deployed. Once all the containers in the POD are running, an HTTP POST request can be issued to trigger a prediction. Issuing a request to the inference graph will start the execution of the predict function of the encoder model generating an output that will be automatically forwarded as input to the clustering model; the response of the clustering model will be used as response body. Figure 46 displays the response to a test HTTP POST request submitted to the deployed Seldon inference graph.

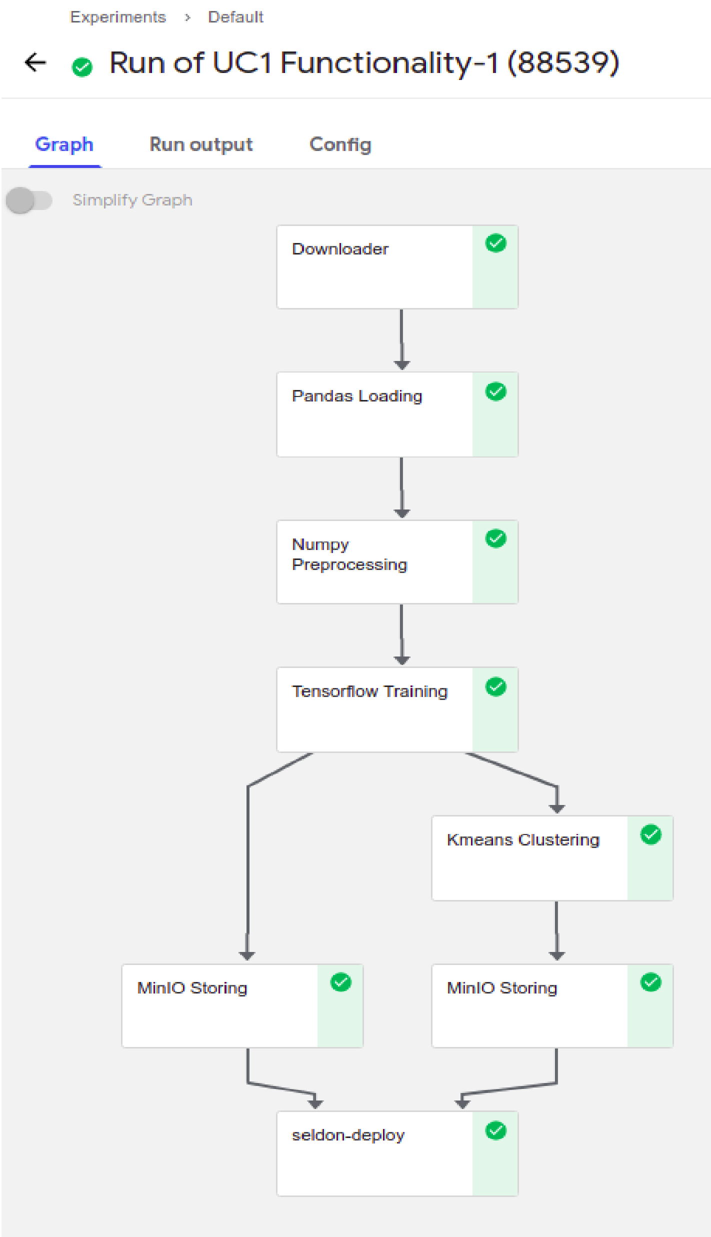

Figure 44 UC1 Functionality-1 Kubeflow Pipeline execution

```
NAME                                                READY   STATUS    RESTARTS   AGE
locus-graph-default-0-clustering-encoder-6fcdcc9466-77sjm   4/4     Running   0          55m
```

Figure 45 Seldon inference graph deployed in Kubernetes cluster

Figure 46 Seldon Graph prediction

3.3 Context-Aware Virtualized Machine Learning Pipeline

As described in 2.2, the contextualization of large urban mobility data with the usage of Open Street Map entities, i.e., the incorporation of different modalities like tags information, nodes, locations and shapes, in the UC1 Functionality-1 for identifying crowd mobility patterns machine learning pipeline, results in an improvement of the trained model as shown in Figure 7 and Figure 8. In order to reflect this new approach for the training of the models also in a virtualized machine learning pipeline, new Kubeflow components have been developed and components already developed for the UC1 Functionality-1 for identifying crowd mobility patterns machine learning virtualized pipeline have been reused since the code-base used as starting point for the development of this context-aware pipeline is the one of the UC1 Functionality-1 for identifying crowd mobility patterns machine learning pipeline that was already virtualized with Kubeflow as reported in the deliverable D5.3 and completed in sections 3.1 and 3.2 of this document.
The code of the reference machine learning pipeline depicted in Figure 6 of section2.2.2 has been split up following the same approach described in the deliverable D5.3 and in section 3.1 and 3.2 of the current document in order to generate the Kubeflow components that form the context-aware virtualized machine learning pipeline. Figure 47 shows the results of this experimental work.

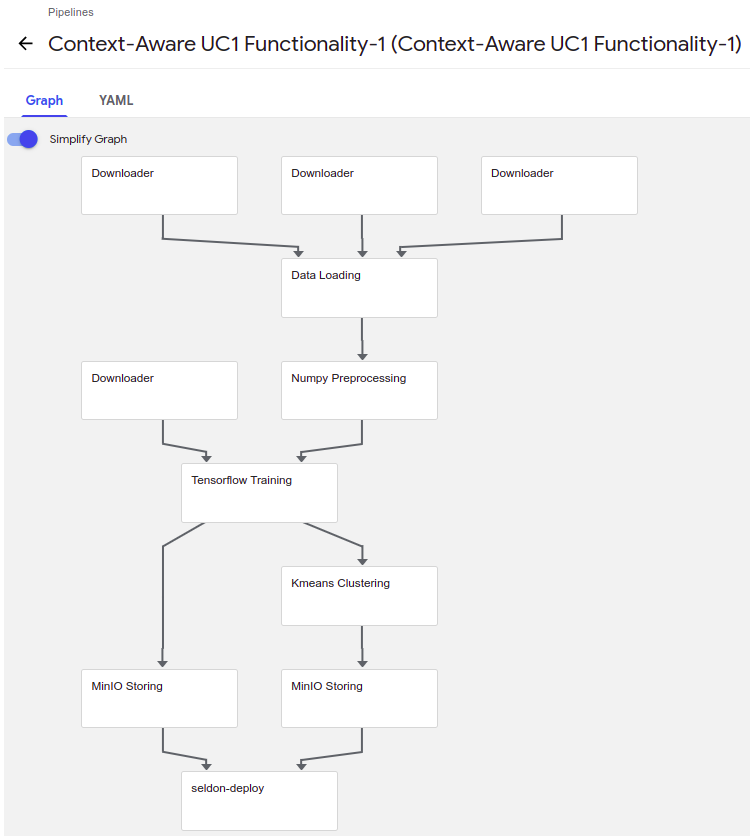

Figure 47 Context-Aware virtualized machine learning pipeline

The major changes from the original UC1 Functionality-1 for identifying crowd mobility patterns machine learning pipeline, and so from the first virtualized machine learning pipeline, are reflected in the Data Loading and Tensorflow Training steps. The Data Loading step takes

care to retrieve and process the OpenPFlow, Open Street Map, nodes.npz and tree.pkl datasets and files used by the context-aware UC1 Functionality-1 code. The Tensorflow Training step uses the newly available data to generate a new machine learning model. The remaining steps of the context-aware virtualized machine pipeline are exactly the same already described in the sections 3.1 and 3.2 of this document, demonstrating the possibility to reuse the developed Kubeflow pipeline component also across different pipelines simply referencing the " .yaml" files that represent the components in the code of the pipeline itself. Since experimental works are still in progress for this version of the context-aware virtualized machine learning pipeline demonstration of the execution of the developed steps will be reported in the next deliverable of the WP5.

3.4 Next steps

The main idea for the continuation of the work for the virtualization of analytics services and pipelines, starting from the results reported in the previous sections, in order to achieve a better cohesion and integration within the LOCUS Platform, is identified in the realization of a continuous integration system where the models are trained on-demand every time new training data are available and deployed on the virtualization platform as last step of an ad-hoc Kubeflow Pipeline. The Kubeflow Pipeline SDK provides a client that can be used to connect to the Kubeflow Pipeline service and thus used to perform a number of different operations, among them, the execution of pipelines can be triggered. Figure 48 shows a brief sketch of the architecture that could realize this continuous integration of machine learning deployed models.

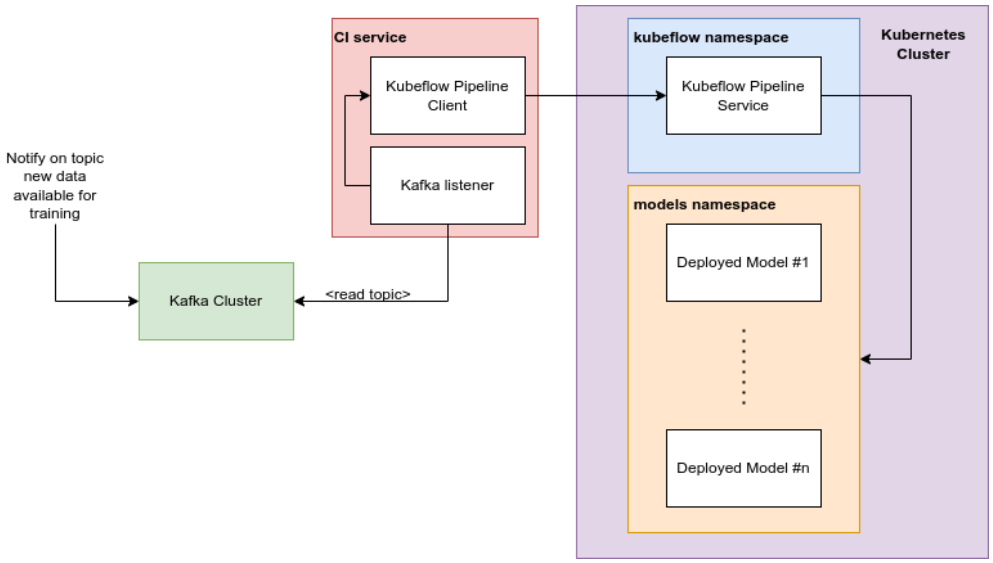

Figure 48 Continuous Models Integration architecture

The idea would be to make use of a publish-subscribe service, Apache Kafka for instance, to be aware of when new training data are available and consequently trigger, using the Kubeflow Pipeline client, a new run of a given virtualized machine learning Kubeflow pipeline in order to generate and train new models with the new training data and deploy the models themselves so that they are available to make predictions, the latter triggered by HTTP requests.

4 Conclusions and Future work

The first deliverable D5.1 highlighted key analytics functionalities for NSE UC1-6, covered state-of-the-art from UC relevant domains, identified potential data sources, and described solution designs for each of the UCs while providing initial results from implementation of the solutions. The second deliverable D5.3 from WP5 defined a generic ML pipeline for the purpose of virtualisation while unifying some functions common across UCs such as data preprocessing steps, model storage, and version control. The key approach in our virtualisation strategy is to provide to localisation & analytics services that provide access to services independent of its internal implementation. Another aspect in our approach to implementation of analytics functions and services is to gain overall agility in terms of adopting changes quickly and scalability. With this in focus, we extended core algorithms underpinning the analytics services and aimed to achieve accurate predictions, faster inference, and storage optimisation. Furthermore, a set of algorithms were trained and evaluated on heterogenous datasets for better generalisation. The result of this approach is that models can be adopted with minimal changes for various scenarios across multiple UCs. To achieve these following advancements have been made to ML models.

- For NSE-UC1,3, and 4, in human trajectory prediction work, we infused acontextual information from two fronts: i) In dense urban environments, human trajectories are influenced by others in the surrounding and to capture this information, we designed a set of kernel functions that embed the interactions among trajectories providing more accurate inference of future trajectories. In particular, our proposed kernel function considers distance between pedestrians and agents and infers interaction score based on the distance. Our experiments showed that in contrary to previous works, inverse of distance (RBF kernel or inverse of L2 norm distance) as a measure interaction score performed better than simple distance measure. This verifies our intuition that in human mobility, closer two pedestrians (or any other moving objects) are, more impact. They have on each other's trajectory.
- From the model's architecture perspective, we extended RNN based neural network architecture to LSTM, GRU, and TCN for the temporal encoding and future steps prediction. In particular our combination of kernel function with GCN for spatial encoding provided the best results in terms of size of the model and inference speed. Although the performance of graph-based architecture was not far from LSTM encoder with interactions [35], but our architecture improved upon multiple fronts such as 30x improvement on size of model and 500x improvement over the inference speed. These improvements are crucial for deployment of model as virtualised functions for LOCUS virtualization platform.
- In addition to considering the other trajectories, we infused contextual information by concatenating the trajectory feature vectors with contextual information vectors acquired from the OSM. Our experiment results showed that contextual information improved performance in terms of ADE and FDE metrics over the models with no contextual information.

- From group mobility pattern detection (NSE-UC2) perspective, a new solution "CountMeIn" which leverages auxiliary sensors to count the number of people in a group.
- For the NSE-UC6, proposed a stochastic model for predicting and evaluating the possible space-time evolution paths of the COVID-19 pandemic by exploiting human mobility data. The results show that such a model provides, since the outbreak, reliable predictions despite of the disruptive growth of the pandemic.

- From the perspective of providing these analytics models as services, ML pipelines defined in D5.3, are further extended with new trained models making them part of ML pipelines that will output the results for a POC by chaining a set of microservices and workflows.

Regarding future work, current work in human trajectory prediction can be extended on multiple fronts. From the project perspective, developed models as localisation analytics services can be benchmarked against KPIs such as latency requirements for various verticals in a 5G environment. Since the last period of the LOCUS project focuses on integration of technical developments for the POCs, this will be an opportunity to deploy these models as VNFs by deploying ML pipelines. This will be reported in the last deliverable (D5.4) of WP5 which is due in July 2022.

From the research perspective, although our model for human trajectory forecasting outperformed stat-of-the-art computationally and also improved the inference speed. However, we believe there is need to define custom evaluation metrics when it comes the evaluating the prediction accuracy of trajectory forecasting models. The majority of the work focuses on the evaluation metrics (ADE and FDE) which are distance based and do not factor in other factors such as trajectory collision, walking in parallel, or following other trajectories. We believe these factors require appropriate evaluation metrics which could paint more complete picture. Secondly, most of the literature focuses on the relative positions of other pedestrians when encoding trajectories into vectors. However, from our experiments, we observed relative velocity information can further improve trajectory predictions and aid in solving the trajectory collision scenarios. Furthermore, if the output from the trained models is to be used in emergency scenarios or situations where results are used for human supervision, we believe these results must accompany with possible explanations. For this, Layer-wise Relevance Propagation (LRP) [36] technique is a useful investigative tool to gain insights regarding the decision-making process of neural networks. In our last deliverable, we aim to investigate more techniques that can explain the effects of surroundings on the predicted trajectories.

5 References

[1] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In European conference on computer vision, pages 452–465. Springer, 2010.

[2] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In Computer Graphics Forum, volume 26, pages 655–664. Wiley Online Library, 2007

[3] Lohan, E. S., Torres-Sospedra, J., Leppäkoski, H., Richter, P., Peng, Z., & Huerta, J. (2017). Wi-Fi crowdsourced fingerprinting dataset for indoor positioning. Data, 2(4), 32.

[4] E. Pepe, P. Bajardi, L. Gauvin, F. Privitera, B. Lake, C. Cattuto, and M. Tizzoni, "COVID-19 outbreak response, a dataset to assess mobility changes in Italy following national lockdown," Scientific Data, vol. 7, no. 1, p. 230, Jul. 2020. [Online]. Available: https://doi.org/10.1038/s41597-020-00575-2

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[6] Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2255–2264, 2018.

[7] A. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," 2018.

[8] I. Hasan, F. Setti, T. Tsesmelis, A. D. Bue, F. Galasso, and M. Cristani, "Mx-lstm: Mixing tracklets and vislets to jointly forecast trajectories and head poses," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6067–6076, 2018.

[9] Niepert, M., Ahmed, M., & Kutzkov, K. (2016, June). Learning convolutional neural networks for graphs. In International conference on machine learning (pp. 2014-2023). PMLR.

[10] Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprintarXiv:1803.01271.

[11] Kothari, P., Kreiss, S., & Alahi, A. (2021). Human trajectory forecasting in crowds: A deep learning perspective. IEEE Transactions on Intelligent Transportation Systems.

[12] OpenStreetMap, «OpenStreetMap,» [Online]. Available: https://wiki.openstreetmap.org/wiki/Main_Page. [Accessed online on 20 01 2022].

[13] OpenStreetMap, «Elements,» [Online]. Available: https://wiki.openstreetmap.org/wiki/Elements. [Accessed online on 20 01 2022].

[14] OpenStreetMap, «Map features,» [Online]. Available: https://wiki.openstreetmap.org/wiki/Map_features. [Accessed online on 20 01 2022].

[15] Geofabrik, «OpenStreetMap Data Extracts,» [Online]. Available: http://download.geofabrik.de/. [Accessed online on 20 01 2022].

[16] Deliverable D5.1, "Design and implementation of virtualization technologies and pattern recognition mechanisms for physical analytics, preliminary version", LOCUS Project, url: https://www.locus-project.eu/results/deliverables/, 2021.

[17] G. Solmaz, J. Furst, S. Aytac and F.-J. Wu, “Group-In: Group Inference from Wireless Traces of Mobile Devices,” *19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN),* pp. 157-168, 2020
[18] S. Bartoletti et al., "Location-Based Analytics in 5G and Beyond," in IEEE Communications Magazine, vol. 59, no. 7, pp. 38-43, July 2021, doi: 10.1109/MCOM.001.2001096.
[19] HELLA Aglaia People Sensing Technologies, “Advanced People Sensor APS-180E,” http://people-sensing.com/, 2017.
[20] Solmaz, Gurkan, et al. "Toward understanding crowd mobility in smart cities through the internet of things." IEEE Communications Magazine 57.4 (2019): 40-46.
[21] G. Solmaz, P. Baranwal, and F. Cirillo, “CountMeIn: Adaptive Crowd Estimation with Wi-Fi in Smart Cities”, Accepted to 20th IEEE International Conference on Pervasive Computing and Communications (PerCom), 2022.
[22] Y. Fukuzaki, M. Mochizuki, K. Murao, and N. Nishio, “Statistical analysis of actual number of pedestrians for Wi-Fi packet-based pedestrian flow sensing,” in ACM UbiComp, 2015, pp. 1519–1526.
[23] F.-J. Wu and G. Solmaz, “Crowdestimator: Approximating crowd sizes with multi-modal data for internet-of-things services,” in ACM MobiSys, 2018, pp. 337–349.
[24] Ratner, Alexander, et al. "Snorkel: Rapid training data creation with weak supervision." Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases. Vol. 11. No. 3. NIH Public Access, 2017.
[25] United Nations, Revision of World Urbanization Prospects, May 2018 [online] Available: https://www.un.org/en/desa/2018-revision-world-urbanization-prospects
[26] Shibuya Scramble Crossing, https://en.wikipedia.org/wiki/Shibuya_Crossing
[27] ETSI Technical Report TR 103 300-1, “Intelligent Transport System (ITS) Vulnerable Road Users (VRU) awareness Part 1: Use Cases definition (Release 2)”, v2.1.1, September 2019.
[28] ETSI Technical Report TR 103 300-2, “Intelligent Transport System (ITS) Vulnerable Road Users (VRU) awareness Part 2: Functional Architecture and Requirements definition (Release 2)”, v2.1.1, May 2020.
[29] ETSI Technical Report TR 103 300-3, “Intelligent Transport Systems (ITS) Vulnerable Road Users (VRU) awareness Part 3: Specification of VRU awareness basic service (Release 2)”, v2.1.1, November 2020.
[30] 5G Automotive Association ‘Vulnerable Road User Protection’ whitepaper, V1.0, Aug 2020
[31] W. O. Kermack and A. G. McKendrick, “Contributions to the mathematical theory of epidemics—I,” Bulletin of Mathematical Biology, vol. 53, no. 1, pp. 33–55, Mar. 1991. [Online]. Available: https://doi.org/10.1007/BF02464423
[32] W. O. Kermack and A. G. McKendrick,, “Contributions to the mathematical theory of epidemics—II. The problem of endemicity,” Bulletin of Mathematical Biology, vol. 53, no. 1, pp. 57–87, Mar. 1991. [Online]. Available: https://doi.org/10.1007/BF02464424
[33] G. C. Calafiore, C. Novara, and C. Possieri, “A modified SIR model for the COVID-19 contagion in Italy,” 2020.

[34] G. Giordano, F. Blanchini, R. Bruno, P. Colaneri, A. Di Filippo, A. Di Matteo, and M. Colaneri, “Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy,” Nature Medicine, vol. 26, no. 6, pp. 855–860, Jun. 2020. [Online]. Available: https://doi.org/10.1038/s41591-020-0883-7

[35] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social lstm: Human trajectory prediction in crowded spaces,” 2016 IEEE Conference on Computer Vision and Pattern Recog- nition (CVPR), pp. 961–971, 2016.

[36] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. Müller, and W. Samek, “On pixel-wise explanations for non-linear classifier deci- sions by layer-wise relevance propagation,” PLoS ONE, vol. 10, 2015.

[37] Mohamed, A., Qian, K., Elhoseiny, M., & Claudel, C. (2020). Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 14424-14432).

[38] Kosaraju, V., Sadeghian, A., Martín-Martín, R., Reid, I., Rezatofighi, S. H., & Savarese, S. (2019). Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. arXiv preprint arXiv:1907.03395.

[39] Ratner, Alexander, et al. "Snorkel: Rapid training data creation with weak supervision." Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases. Vol. 11. No. 3. NIH Public Access, 2017.