Autonomous self-adaptive services for TRansformational personalized inclUsivenesS and resilience in mobiliTy

D3.2 Advanced internal and external sensing system.v1

Lead beneficiary	AviSense.AI	**Lead author**	Nikos Piperigkos
Reviewers	Theofilos Christodoulou (CERTH), Gor Hakobyan (Waveye GmbH)		
Type	R	**Dissemination**	PU
Document version	V1.0	**Due date**	31/08/2025

Co-funded by the European Union

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI

KIAT 한국산업기술진흥원
Korea Institute for Advancement of Technology

AutoTRUST project has received funding from CINEA under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101148123.

Project information

Project title	**Autonomous self-adaptive services for Transformational personalized inclUsivenesS and resilience in mobility**
Project acronym	**AutoTRUST**
Grant Agreement No	101148123
Type of action	Research and Innovation Actions (RIA)
Call	HORIZON-CL5-2023-D6-01
Topic	HORIZON-CL5-2023-D6-01-01 - User-centric development of vehicle technologies and solutions to optimise the on-board experience and ensure inclusiveness (CCAM Partnership)
Start date	1 May 2024
Duration	36 months

Document information

Associated WP	WP3
Associated task(s)	T3.2, T3.3, T3.4
Main Author(s)	Nikos Piperigkos (AviSense.AI), Alexandros Gkillas (AviSense.AI)
Author(s)	Aris Lalos (AviSense.AI), Dimitrios Tsiktsiris, Theoktisti Marinopoulou, Dimitrios Manolakis, Theofilos Christodoulou, Christos Basdekis, Dimitrios Triantafyllou, Eleftherios Blitsis, Theofanis Siamatras, Thomas Kopalidis, Evangelos Athanasakis, Panagiotis Lepentsiotis, Stefanos Georgiadis, Angeliki Zacharaki, Antonios Lalas, Konstantinos Votis (CERTH), Leonidas Katellaris, Nearchos Stylianidis (UCY), You-Jun Choi (KATECH), Sooji Yeom (MORAI), Sajad Hamzenejadi (UNIGE), Gor Hakobyan (Waveye GmbH), Pierre Chehwan (BETI), Bruna Franceschini (CARITAS)
Reviewers	Theofilos Christodoulou (CERTH), Gor Hakobyan (Waveye GmbH)
Type	R — Document, report
Dissemination level	PU — Public
Due date	M16 (31/08/2025) – Extended to M17 (30/09/2025)
Submission date	30/09/2025

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI
KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

Document version history

Version	Date	Changes	Contributor (s)
v0.1	28/05/2025	Initial table of contents	Nikos Piperigkos (AviSense.AI), Alexandros Gkillas (AviSense.AI)
v0.2	12/06/2025	Contribution in Section 1	Nikos Piperigkos (AviSense.AI), Alexandros Gkillas (AviSense.AI)
v0.3	04/07/2025	Contribution in Section 2 and Section 3	All authors
v0.4	18/07/2025	Contribution in Section 3 and Section 4	All authors
v0.5	24/07/2025	Contribution in additional Sections	All authors
v0.6	01/08/2025	Version ready for review	Nikos Piperigkos (AviSense.AI), Alexandros Gkillas (AviSense.AI)
v0.7	26/08/2025	Review by CERTH	Theofilos Christodoulou (CERTH)
v0.7.5	27/08/2025	Refinement based on review comments	Nikos Piperigkos (AviSense.AI), Alexandros Gkillas (AviSense.AI)
v0.8	28/08/2025	Review by Waveye GmbH	Gor Hakobyan (Waveye GmbH)
v0.8.5	29/08/2025	Refinement based on review comments	Nikos Piperigkos (AviSense.AI), Alexandros Gkillas (AviSense.AI)
v0.9	19/09/2025	Final review and refinements	Theofilos Christodoulou (CERTH), Antonios Lalas (CERTH)
v1.0	30/09/2025	Final version for submission	Antonios Lalas (CERTH)

Disclaimer

Copyright message

Contents

List of acronyms and abbreviations

Abbreviation	Description
AC	Air conditioning
ADAS	Advanced Driver Assistance Systems
AI	Artificial Intelligence
AMOTA	Average multi object tracking accuracy
AMOTP	Average multi object tracking precision
ANN	Artificial neural network
AOS GraphLap CoMOT	All in one stage Graph Laplacian CoMOT
APE	Absolute pose error
AR	Augmented Reality
ASR	Automatic Speech Recognition
ATC	Adapt then combine
AV	Autonomous Vehicle
BSM	Basic Safety Message
CAM	Cooperative Awareness Message
CAV	Connected and autonomous vehicle
CFD	Computational Fluid Dynamics
CL	Cooperative localization
CNN	Convolutional neural network
CoMOT	Cooperative MOT
DENM	Decentralized Environmental Notification Message
DP	Differential privacy
EAR	Eye Aspect Ratio
ECU	Electronic control unit
ETSI	European Telecommunications Standards Institute
FAR	False acceptance rate
FIR	False injection rate
FL-DU	Federated learning based deep unrolling
FN	False negative
FP	False positive
GDPR	General data protection regulation
GNSS	Global Navigation Satellite System
GT	Ground truth
HA	Hungarian algorithm
HE	Homomorphic encryption
HMI	Human Machine Interface
HQS	Half quadratic splitting
IDSW	Identity switches

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI
KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

Abbreviation	Description
IMU	Inertial Measurement Unit
IoT	Internet of Things
IOU	Intersection over union
ITS	Intelligent Transportation Systems
KF	Kalman filtering
LiDAR	Light detection and ranging
LLM	Large Language Model
LSTM	Long-short term memory
MAR	Mouth Aspect Ration
ML	Machine Learning
MLP	Multi Layer Perceptron
MOT	Multi object detection and tracking
MOTP	Multi object tracking precision
MT	Mostly tracked
MTCNN	Multitask Cascaded Convolutional Neural Network
PM	Particulate matter
PSM	Personal Safety Message
QoE	Quality of experience
QoS	Quality of service
RADAR	Radio detection and ranging
RANS	Reynolds Averaged Navier Strikes
RNN	Recurrent neural network
RSA	Roadside Alert
sAMOTA	scaled AMOTA
SCNN	Spiking Convolutional Neural Networks
SIMPLE	Semi Implicit Method for Pressure Linked Equations
SLAM	Simultaneous localization and mapping
SNN	Spiking Neural Networks
SPAT	Signal Phase and Timing
SR	Super-resolution
STT	Speech to text
TSA GraphLap CoMOT	Two stages of association Graph Laplacian CoMOT
TTS	Text to speech
V2I	Vehicle to infrastructure
V2V	Vehicle to vehicle
V2X	Vehicle to everything
VAE	Variational autoencoder
VAS	Virtual Assistant System
VIL	Vehicle in the loop

Co-funded by the European Union

Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI

KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

Abbreviation	Description
ViT	Vision Transformer
VLM	Vision Language Models
VOC	Volatile Organic Compounds
VRU	Vulnerable Road User

List of figures

List of tables

Executive summary

The deliverable D3.2, “Advanced Internal and External Sensing System.v1,” presents the technical advancements and initial validation of the integrated internal and external sensing systems developed within the AutoTRUST project, which aims to advance the state of the art in autonomous, safe, and inclusive mobility.

The internal sensing system is designed to ensure passenger safety, comfort, and personalized interaction inside vehicles. The system combines visual, audio, and environmental sensors to provide a comprehensive view of the cabin and its occupants. Key features include real-time monitoring of driver attention and well-being, detection of risky behaviors and abnormal sounds, as well as privacy-preserving facial and emotion recognition. An intelligent virtual assistant interprets data from multiple sources and enables natural, context-aware interactions between users and the vehicle, all running locally for enhanced privacy and reliability. Furthermore, environmental sensors and advanced simulation models are deployed to continuously assess and optimize air quality and thermal comfort, ensuring a healthy and comfortable cabin environment for all passengers.

The external sensing system supports safe and resilient mobility by enabling vehicles to accurately perceive and understand their surrounding environment, both individually and cooperatively. The system delivers reliable detection and assessment of road conditions, including the identification and prioritization of hazards such as potholes and damaged infrastructure. This information is intuitively conveyed to drivers using advanced visualization techniques, such as augmented reality overlays, which provide early and easily interpretable warnings. Crucially, AutoTRUST enables vehicles to securely share information about hazards, road conditions, and dynamic objects with each other and with traffic infrastructure. This collaborative approach extends the awareness of each vehicle beyond its own sensors, improving responsiveness and safety in complex, dynamic environments. By allowing vehicles to collaboratively learn from local data without exchanging sensitive information, AutoTRUST improves perception and localization accuracy while ensuring compliance with data privacy requirements. The use of privacy-preserving techniques, such as secure aggregation and encryption, is a core part of the system, reflecting a strong commitment to user trust and data protection.

All developed modules have been validated using a combination of real-world data collection, custom test scenarios, and large-scale simulation environments. The deliverable also outlines ongoing and future work towards full-scale pilot demonstrations and continued integration of

user feedback, with the goal of delivering a robust, user-centric, and ethically sound autonomous mobility platform.

In summary, D3.2 establishes the foundation for advanced, trustworthy sensing and perception systems in autonomous vehicles, combining technical excellence with a user-centered, privacy-first approach. The work described in this deliverable directly contributes to safer, more comfortable, and more inclusive mobility solutions for all road users.

It should be underlined that this is the first version of the Deliverable and the AutoTRUST sensing technologies, which will continue to evolve in the second period of the project and will be described in its final version of the Deliverable D3.4 “Advanced internal and external sensing system.v2”, due to M29, and in accordance with the project’s description of work.

Co-funded by the European Union

Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI

KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

1. Introduction

The Deliverable D3.2, "Advanced Internal and External Sensing System.v1," provides a comprehensive overview of the sensing architectures, methodologies, and initial results achieved within the AutoTRUST project. This document describes the core technologies that enable autonomous vehicles to perceive, understand, and respond to both the in-cabin environment and the external driving scene, supporting the project's vision for safe, inclusive, and resilient mobility. The primary objective of this deliverable is to present the design and implementation of integrated sensing solutions that advance user safety, comfort, and trust in autonomous systems. The report covers a wide range of technical topics, including occupant monitoring, driver attention analysis, environmental quality assessment, road condition detection, cooperative awareness, and privacy-preserving data management. Both the internal and external sensing systems are discussed in detail, with a focus on their architectural foundations, technical innovations, and real-world validation. This deliverable is intended as a reference for project partners, reviewers, and stakeholders who seek to understand the state-of-the-art sensing technologies developed in AutoTRUST, their role within the overall project architecture, and their contribution to the project's strategic objectives. The document also establishes the foundation for future development, large-scale pilot deployments, and the continuous improvement of user-centered autonomous mobility solutions.

1.1. Purpose and structure of the document

The purpose of this document is to provide a detailed account of the methodologies, technologies, and results underpinning the development of the advanced internal and external sensing system within the AutoTRUST project. This deliverable aims to guide both the technical implementation and the integration of sensing solutions that ensure safe, inclusive, and user-centric autonomous mobility.

This report documents the activities and decisions taken throughout the design, development, and validation phases of the sensing system. It captures the rationale behind key technical choices, highlights innovative approaches to occupant and environment monitoring, and sets out the measures adopted to address privacy, security, and ethical considerations.

The structure of the document is as follows:

- **Section 1** introduces the objectives and relevance of the deliverable, providing an overview of its purpose and intended audience.

- **Section 2** presents the internal sensing system for in-cabin monitoring, including occupant safety, comfort analysis, behavioral understanding, and environmental sensing.
- **Section 3** describes the external sensing system, covering approaches for contextual scene analysis, road condition assessment, cooperative perception, and multi-agent information sharing.
- **Section 4** addresses privacy, security, and data trustworthiness within the sensing system, outlining both technical and organizational safeguards, with a focus on privacy-preserving learning and secure data aggregation.
- **Section 5** summarizes the key conclusions and sets the direction for ongoing work and future milestones.

Each section is organized to provide both a high-level overview and detailed technical insights, ensuring the document is accessible to a diverse readership, including technical experts, project partners, and broader stakeholders. The report also serves as a foundational reference for subsequent deliverables and the continuous improvement of the AutoTRUST platform.

1.2. Intended Audience

The Deliverable D3.2, "Advanced Internal and External Sensing System.v1," is intended for both public use and for the AutoTRUST consortium, including all project partners and affiliated stakeholders. It serves as a comprehensive reference, providing detailed guidance on the implementation and integration of advanced sensing systems developed within the project. The document is designed to support technical contributors, project managers, and consortium members throughout the project's duration, ensuring a shared understanding of methodologies, system architectures, and validation approaches. In addition, it provides transparency and accountability for external reviewers, European Commission evaluators, and the wider research and innovation community interested in state-of-the-art solutions for autonomous, inclusive, and resilient mobility.

1.3. Interrelations

The AutoTRUST project brings together a multidisciplinary consortium with expertise spanning academia, industry, and research organizations. This diversity ensures a holistic approach to developing advanced, AI-driven solutions for personalized, inclusive, and resilient connected and automated mobility (CCAM). With sixteen partners across ten EU member states and associated countries—including Norway, Switzerland, the United Kingdom, Korea, and Japan—

the project leverages a wide range of perspectives and resources to address challenges related to security, privacy, well-being, health, and user assistance in automated vehicle environments.

Within the project structure, AutoTRUST is organized as a Research and Innovation Action (RIA), segmented into six interconnected work packages (WPs), each further divided into targeted tasks. The internal and external sensing systems described in this deliverable (D3.2) form a cornerstone for several technical and user-centered activities across these work packages. Specifically, the methodologies and results presented here build on the user requirements and best practices established in earlier deliverables, and provide essential technological input for the development and integration of perception, decision-making, and user interaction components addressed in subsequent work packages (WP4, WP5). Furthermore, the sensing architectures and privacy frameworks detailed in D3.2 support the system-level evaluation, demonstration, and validation activities planned for later stages of the project (WP5). This ensures that the sensing systems are not only technically robust but also aligned with broader project objectives relating to inclusiveness, safety, and trustworthiness.

Overall, D3.2 is positioned at the intersection of user needs, technological innovation, and ethical compliance, and serves as a key reference point for ongoing collaboration and knowledge exchange among AutoTRUST partners.

2. Internal sensing system of in-cabin monitoring for occupant safety and comfort

In the pursuit of enhanced safety, comfort, and personalization within automotive environments, the integration of intelligent in-cabin monitoring systems has become a focal point of research and development. Towards this end, the research and technological developments of AutoTRUST focus on multi-modal monitoring and advanced AI-driven analytics. The system architecture encompasses multiple interconnected modules that work synergistically to provide real-time assessment of occupant behavior, environmental conditions, and potential safety risks. Through the integration of computer vision, artificial intelligence, sensor networks, and advanced signal processing techniques, these systems create a holistic understanding of the cabin environment that extends far beyond traditional monitoring capabilities.

The following sections detail the technical aspects of the system, focusing on the adopted methodologies, performance characteristics, and practical applications of each component. Specifically, Section 2.1 explores the cabin environment and occupant behaviours through strategic sensor placement and machine learning techniques, addressing areas such as facial recognition, emotion and distraction detection, and virtual assistance. Section 2.2 highlights innovative approaches for saliency-based obstacle detection and augmented reality solutions for proactive hazard awareness. Section 2.3 investigates air quality and thermal comfort using smart sensors and computational fluid dynamics, Section 2.4 introduces visual analysis methods leveraging object detection and cutting-edge vision-language models, while, finally, Section 2.5 presents our concluding remarks.

2.1. Analysis of cabin environment and occupant behaviors

The in-cabin monitoring architecture of AutoTRUST, presented in Figure 1, integrates multiple sensor modalities with AI-driven analytics to ensure driver and passenger safety, comfort, and inclusiveness. The system combines side- and front-facing cameras, a microphone array, and an event-based sensor, which continuously capture multimodal data streams inside the vehicle. These streams are processed on the Jetson Orin AGX platform, where lightweight and optimized models execute real-time tasks such as face identification, object and forgotten-item detection, abnormal sound recognition, and facial emotion analysis. While drowsiness detection and event-based driver distraction detection are already developed and running on edge, they have not yet been connected to the LLM reasoning layer—a task that is scheduled for future integration. All other detections are fused through a shared memory layer, enabling downstream reasoning modules and a large language model (LLM) functioning as a virtual assistant. The LLM supports natural interaction via speech-to-text and text-to-speech modules, offering context-aware alerts, feedback, and personalized communication. Camera and microphone placement will be further analyzed in the following subsections, while the event-based sensor—being the newest addition to the system—will have its optimal positioning determined during the next demo. This architecture is fully edge-deployed to meet low-latency and privacy requirements, while maintaining scalability for different cabin setups and use cases.

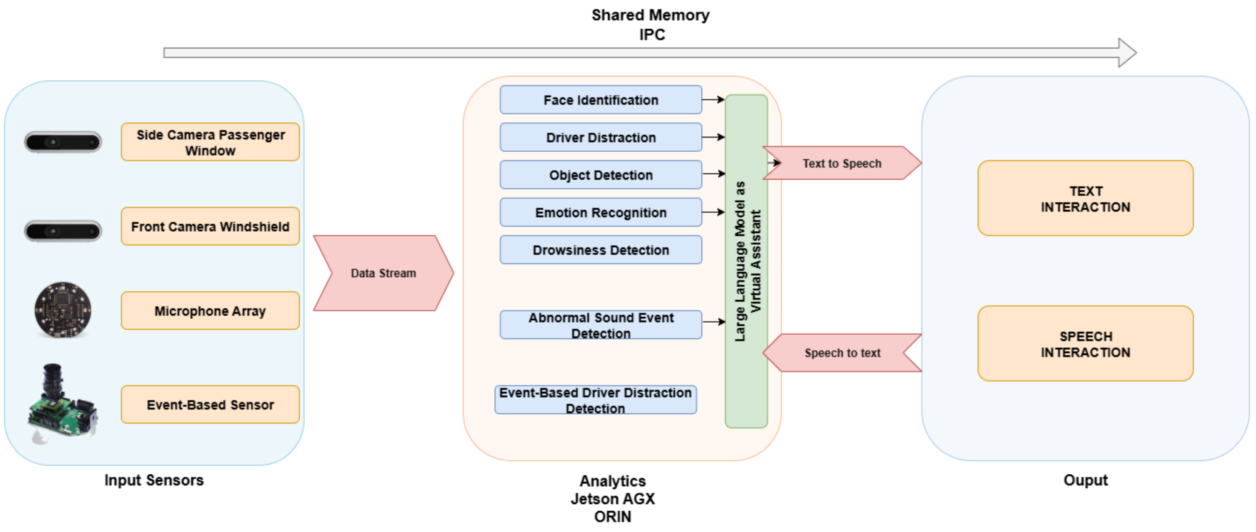


Figure 1: Architecture of the in-cabin monitoring system

2.1.1. Camera placement

In the current implementation two primary cameras are deployed to monitor the driver's state and behaviour. The front-facing driver camera, mounted on the dashboard or windshield, is used to continuously assess the driver's head pose, gaze direction, eye openness, and facial expressions. This camera supports the face identification, facial emotion recognition, drowsiness detection and object detection modules. Complementing this, a side-facing camera, typically mounted at the passenger-side corner of the windshield, provides an additional angle on the driver's face and upper body. In the implementation showcased at EUCAD, a 360-degree camera was also installed, intended for use in future deployments, particularly in larger vehicles such as buses. The inclusion of such a camera can further enhance monitoring coverage and improve the overall robustness of the system's results. The figures (Figure 2, Figure 3, Figure 4) below illustrate the camera placement used in the two demonstration setups.

Co-funded by the European Union

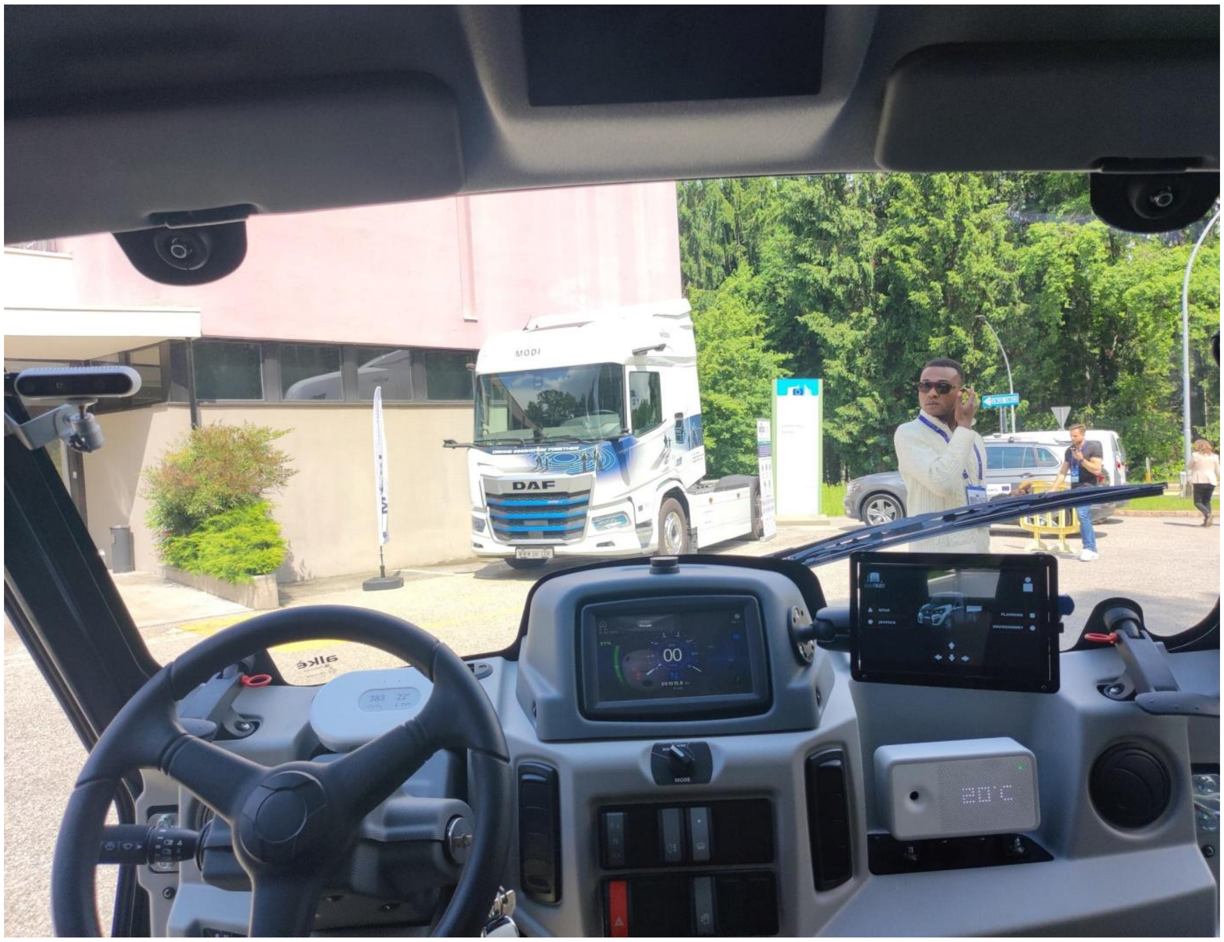


Figure 2 Front-facing camera - EUCAD demo

2.1.2. Microphone placement

The current implementation includes two microphones, each serving a distinct function within the system. The first microphone is dedicated to abnormal sound detection and is strategically placed near the Jetson Orin hardware to ensure clear audio capture of environmental sounds from within the vehicle cabin (Figure 4). Its positioning allows for efficient real-time processing of potentially critical audio cues mentioned in the Abnormal Sound Event detection module. The second microphone is used for interaction with the Virtual Assistant System (VAS). In the showcased setup, this is the built-in microphone of a laptop, enabling basic voice interaction capabilities between the user and the assistant.

Co-funded by the European Union

Project funded by

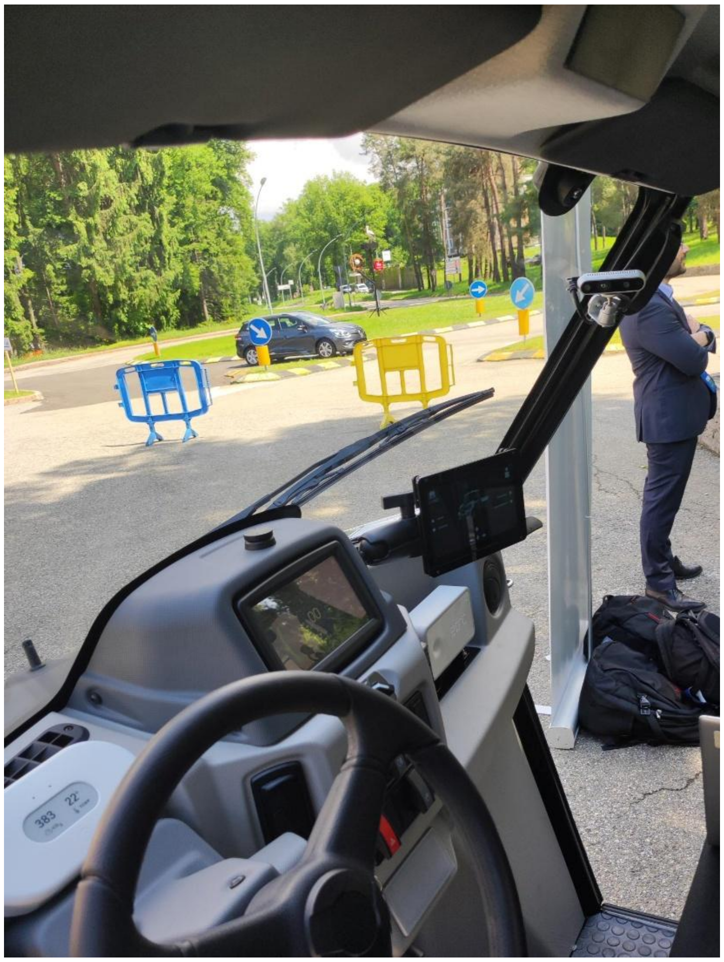


Figure 3: Side camera - EUCAD demo

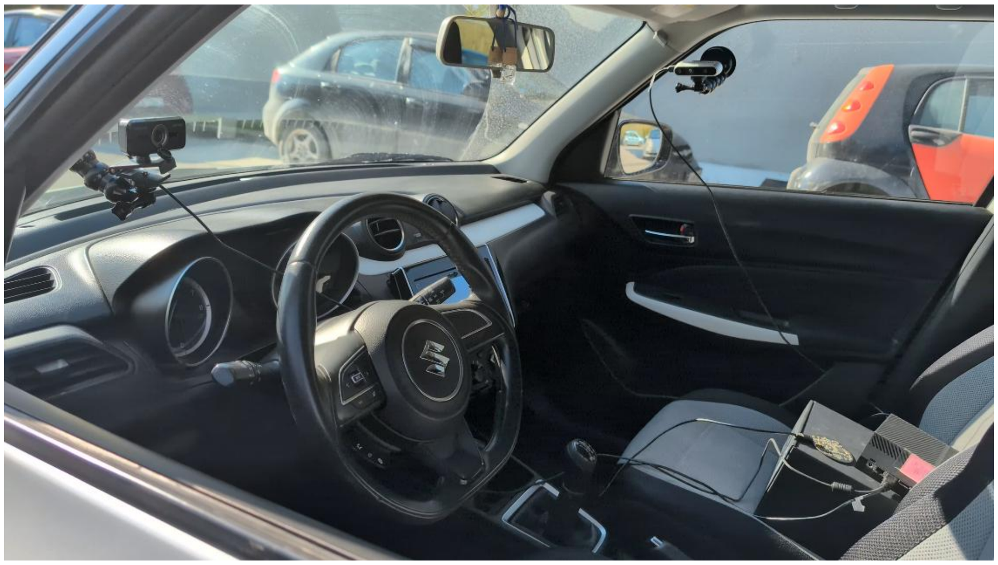


Figure 4: Setup of the first demo

2.1.3. In-cabin object detection

The object-detection module continuously monitors the in-cabin optical environment to detect occupants and their personal items—person, phone, bag, laptop—in real time within a vehicle cabin, all while operating under a 15 W power envelope. One-stage YOLO detectors have been shown to outperform two-stage frameworks (e.g., Faster R-CNN) on both latency and accuracy when deployed on automotive edge hardware. Moreover, the YOLOv8[1] architecture's anchor-free head and CSP backbone deliver higher mAP than YOLOv5 without increasing model size. Accordingly, the COCO-pretrained YOLOv8-s model was adopted and leveraged only INT8 post-training calibration—avoiding any on-device training—while still supplying downstream subsystems with safety-critical context.

Model Architecture

The detector core is Ultralytics YOLOv8-s1, an 11 M-parameter network that fits entirely in the on-chip SRAM of modern embedded GPUs. The architecture of the model is presented in the Figure 5. We export the model from PyTorch to ONNX (FP32), then compile it into an INT8 TensorRT engine using ~500 representative in-cabin frames for calibration. At inference, the network outputs bounding-box coordinates, objectness scores, and class logits for the five target categories.

Pre-processing

Each inference cycle begins with a 640 × 480 RGB frame, which is letterboxed to 640 × 640 (to preserve aspect ratio), normalized to the [0, 1] range, and reshaped into a batch of size N with dimensions 3 × 640 × 640 in FP32. A zero-copy CUDA pipeline then feeds this tensor directly into the TensorRT engine bindings, eliminating redundant host–device transfers and ensuring minimal latency on Jetson-class hardware.

Deployment

The in-cabin object-detection pipeline runs on the Jetson Orin AGX via ONNX Runtime with CUDA support. Each 640 × 480 frames from the inward-facing camera are processed by the ONNX-format YOLOv8-s1, classifier which outputs class predictions and confidence scores in real time. These detections are written to the shared memory fusion layer, where downstream perception and reasoning modules—including those that leverage the onboard LLM—consume them to drive driver-alert functionality. Additional features such as seat-occupancy monitoring and forgotten-object detection will be integrated in the next stages of development, especially

[1] https://docs.ultralytics.com/models/yolov8/

in vehicles where these functions are particularly useful, such as buses. In Figure 6, the object detection results from the first demo are presented.

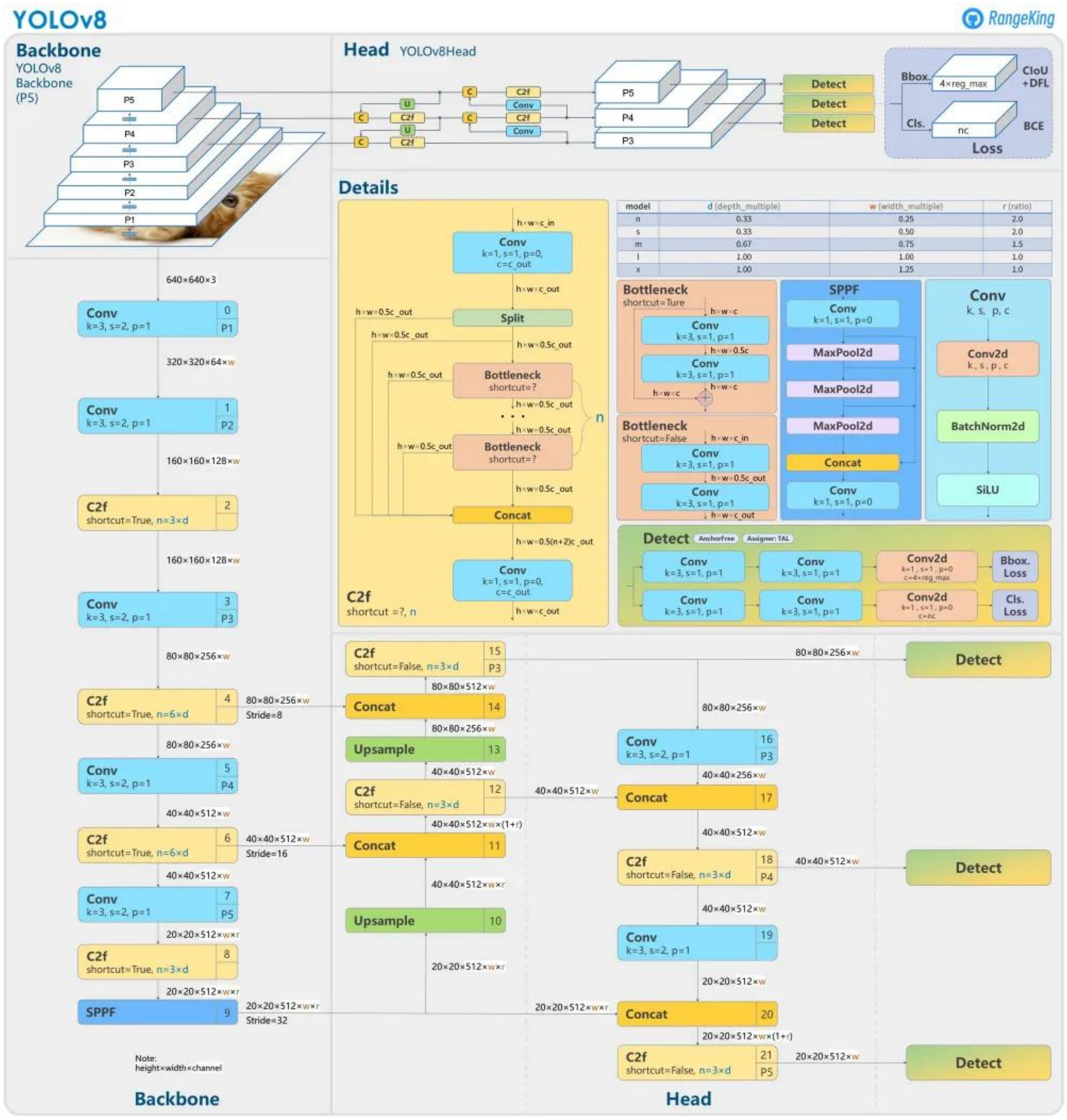
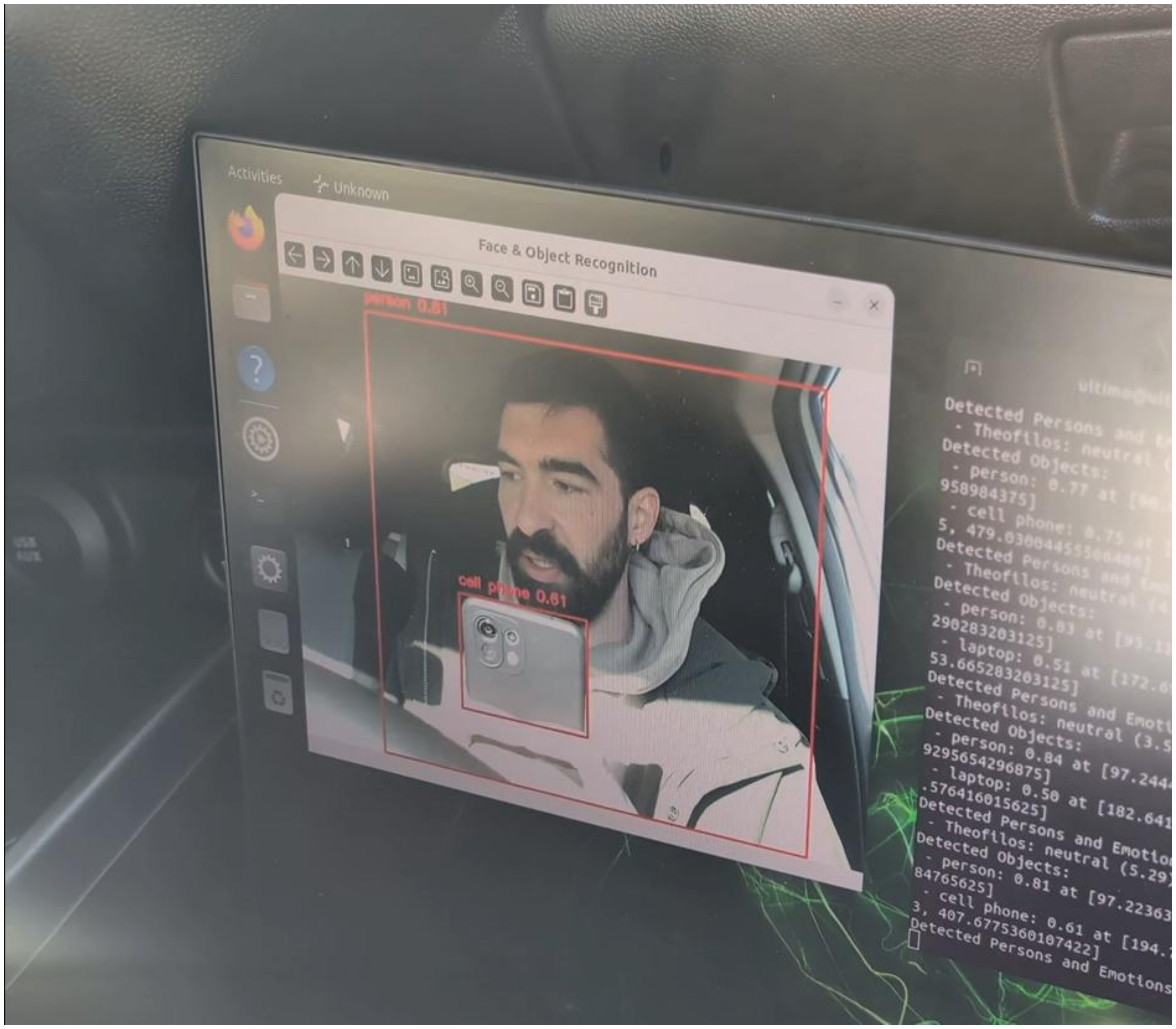


Figure 5: YOLOv8 Architecture[2]

[2] https://docs.ultralytics.com/models/yolov8/



```
Detected Persons and Em
 - Theofilos: neutral (
Detected Objects:
 - person: 0.77 at [90.
958984375]
 - cell phone: 0.75 at [
5, 479.03004455566406]
Detected Persons and Emo
 - Theofilos: neutral (4.
Detected Objects:
 - person: 0.83 at [95.13
290283203125]
 - laptop: 0.51 at [172.6
53.665283203125]
Detected Persons and Emoti
 - Theofilos: neutral (3.5
Detected Objects:
 - person: 0.84 at [97.244
9295654296875]
 - laptop: 0.50 at [182.641
.576416015625]
Detected Persons and Emotio
 - Theofilos: neutral (5.29)
Detected Objects:
 - person: 0.81 at [97.22363
84765625]
 - cell phone: 0.61 at [194.
3, 407.6775360107422]
Detected Persons and Emotions
```

Figure 6: Person and Cellphone Identification from the first demo

2.1.4. Facial recognition for passengers' identification

The facial recognition module allows for real-time detection and identification of previously known individuals, and supports personalization, access control, and driver profile management. The facial recognition module runs continuously and fully offline, on embedded hardware, which enables privacy-preserving identification that requires no internet connection, and no cloud inference.

Method Overview

The identification module runs a two-stage face-based identification pipeline that has been optimized for real-time performance on edge platforms:

- **Face Detection**
  The detection of faces is performed using a modified version of the Multi-task Cascaded Convolutional Neural Network (MTCNN) pipeline [1] [2], accelerated using CUDA and tuned for in-cabin operation. The detection stage provides tight bounding boxes along with facial landmarks, which are used to align and crop face images (see Figure 7).

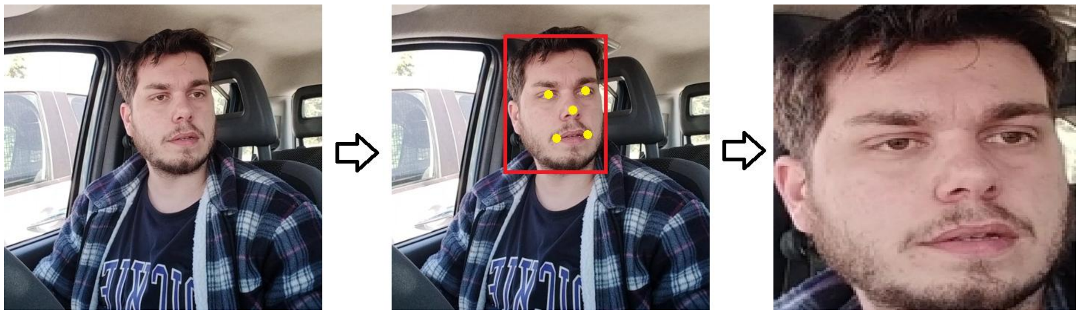


Figure 7: Face detection and preprocessing using the modified MTCNN pipeline. From left to right: original image, detected face with bounding box and facial landmarks, and cropped/aligned face used for embedding

- **Face Embedding and Matching**
  The cropped faces are resized to 160×160 pixels and passed to an ONNX-exported version of FaceNet (Inception-ResNet v1) [2], providing a 512-dimensional embedding vector for each face. This model uses pretrained weights from VGGFace2, which provides diverse coverage of poses, ages, and lighting conditions, ensuring robust embeddings for identification tasks (see Figure 8).

Figure 8: Example images from the pretraining datasets (VGGFace2) showing variations in pose, age, and illumination

During enrolment, the occupants are enrolled by creating prototype embeddings by averaging multiple samples of a person in multiple poses and lighting conditions. During inference, the incoming embeddings are compared to the stored prototypes using cosine similarity, and an identification decision is made if the similarity is greater than a calibrated threshold of 0.55, which was selected to balance false leaves and false rejects.

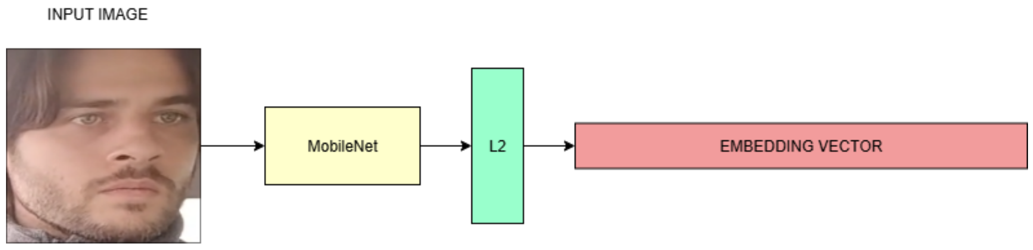


Figure 9: Face embedding extraction and matching using FaceNet (Inception-ResNet v1). The cropped/aligned face (left) is converted into a 512-dimensional embedding vector (middle) and compared to stored prototypes using cosine similarity to produce an ide

Evaluation and Performance

The module went through evaluation in genuine deployment settings with pre-enrolled occupants in different seated, postures, and ambient illuminations. The performance aspects, after training, were:

- **Recognition accuracy**: 96.5% (across seated occupants, with varied poses)
- **False acceptance rate (FAR)**: < 2%
- **False rejection rate (FRR)**: < 4%
- **Inference latency**: < 50 ms per identity (Jetson Orin, CUDA-accelerated)
- **Throughput**: up to 10 identity checks per second

The system also performs well with head tilt, partial occlusion (e.g., eyeglasses, facemasks), and illuminated from the side. The cosine similarities from recommendation works well on a well-separated embedding space and the rapid matching allows extensibility to new users and little re-training.

Deployment and Integration

The deployment is fully wrapped on the NVIDIA Jetson Orin AGX platform built with ONNX Runtime. All computations happen on the edge with no facial images or embeddings being transmitted anywhere. Enrolment as well as the creation of prototypes is also via a secure local interface.

Output identity tags are:

- Published to the shared fusion layer

- Used to facilitate personalized in-cabin experiences (e.g., music preferences, seat configuration)
- Logged (with permission) for occupancy analysis and session continuity

The system is designed for plug-and-play extensibility. New users can be enrolled by capturing a short sequence of faces and generating a profile and does not require retraining of the base FaceNet model.

2.1.5. Facial Emotion Recognition

The facial emotion recognition module allows the in-cabin system to detect the emotional state of occupants in real-time. Recognizing emotions, such as happy, sad, or angry, is possible, and the in-cabin system can use this knowledge to provide further context to the virtual assistant or vehicle control layer helping the system adapt accordingly. Furthermore, the module supports inclusivity and personalization for the autonomous vehicle (AV), by raising user awareness for their well-being in the vehicles environment.

Dataset and model Selection

Moreover, the emotion recognition module and model are based on the ResEmoteNet architecture [3], a convolutional neural network designed for general and robust expression classification in non-constrained environments (see Figure 10). The model has been pre-trained on the AffectNet-7 [4] dataset which includes over one million labeled facial images categorized into seven universal emotion classes: *happy, surprise, sad, anger, disgust, fear, and neutral*. The model has been pre-trained on the AffectNet-7 dataset, which includes over one million labeled facial images categorized into seven universal emotion classes: *happy, surprise, sad, anger, disgust, fear, and neutral* (see Figure 11).

Additionally, there was concern over proper testing and realization within the in-cabin environment and to avoid overfitting any emotion or facial datasets not recorded within a vehicle environment, testing of the model was completed based on cabin footage collected during integration tests on the system, with no fine-tuning completed.

Co-funded by the European Union

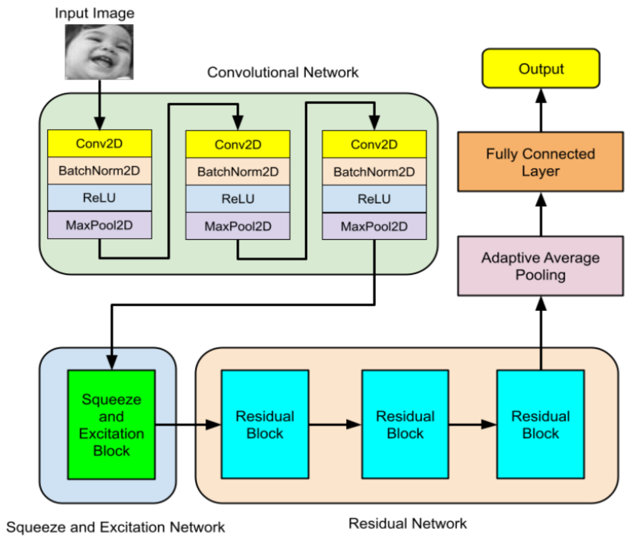


Figure 10: Overall architecture of the ResEmoteNet model

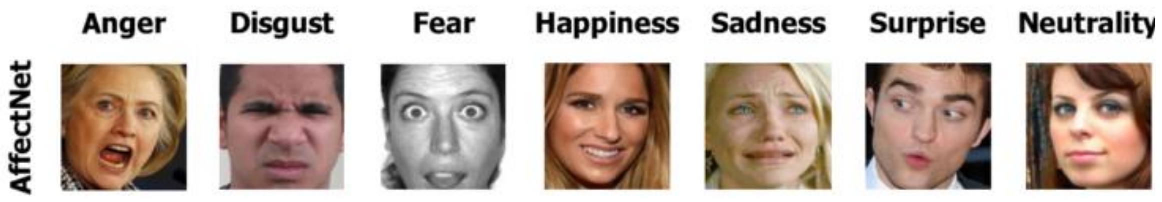


Figure 11: Example emotion-labeled faces from the AffectNet-7 dataset (seven universal classes).

Methodology

The ResEmoteNet model includes:

- Several 3 x 3 convolutional layers featuring squeeze-and-excitation (SE) gating for channel-wise attention
- **Residual** connections [5] for support stable gradient propagation
- **Adaptive pooling** for variable sized images
- A final **softmax classifier** over the 7-class output space

The face regions are detected via the MTCNN pipeline, cropped to a resolution of 160 x 160, then normalized, and passed through the ResEmoteNet model as presented in the Figure 12. The face regions are detected via the MTCNN, cropped to a resolution of 160 x 160, then normalized, and passed through the ResEmoteNet model. The predicted emotion probabilities were straightforwardly smoothed using a temporal majority-vote method, across a 2-second sliding window, to eliminate flickering predictions.

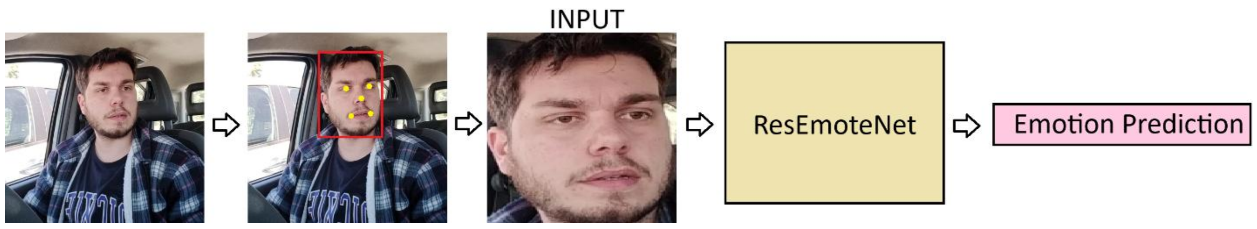


Figure 12: Facial Emotion Prediction Pipeline.

Evaluation and Performance

The ResEmoteNet model was evaluated on the AffectNet-7 validation split, and obtained the following performance:

- **Top-1 accuracy**: 72.9%
- **Balanced F1 score**: 0.71
- **Inference latency**: ~30 ms per frame (ONNX Runtime, Jetson Orin AGX)
- **Memory footprint**: < 150 MB (quantized FP16)

In practice, the model was able to work robustly in varying lighting conditions and head poses, especially when the side-facing camera input was included.

Deployment and Integration

The facial emotion recognition module is deployed as an ONNX model containerized via Docker and executed on the Jetson Orin AGX platform. It publishes emotion scores per detected face to the central fusion layer, which in turn makes these signals available to downstream components such as:

- The virtual assistant (Section 2.1.6), enabling context-aware interaction
- The driver behavior monitoring, which can correlate emotion with distraction
- The risk detection system, supporting escalation strategies (e.g. repeated sadness or fear triggers notifications)

This module is designed to be fully privacy-preserving: all processing is done locally, no face images are stored, and all outputs are anonymized confidence scores.

2.1.6. Driver Distraction Detection

The driver distraction detection module allows for monitoring upper body and hand movements over a lengthy period in order to find behaviours that would ascertain that the driver is not driving safely. This module is capable of processing and delivering results in real time by utilizing a side-mounted camera and an embedded inference pipeline, which is part of the driver behaviour analysis framework.

Dataset Description

To train the distraction detection model, a custom in-cabin dataset was created. This dataset was generated using a combination of front-facing and mid-to-side view cameras, capturing synchronized multi-view video data in real-time. The mid-to-side view cameras introduced subtle angular variations to enhance the model's ability to generalize across different viewing angles.

The dataset consists of 653 annotated video clips, each lasting 3 seconds (72 frames), as detailed in Table 1. These clips were captured under diverse lighting conditions and involved multiple subjects. Each video clip was annotated according to six distinct behavior classes: normal driving, texting, phone calls, drinking, smoking, and reaching (e.g., toward the dashboard or side compartments).

To ensure dataset variability, camera angles, occupant clothing, object appearances, and seating positions were deliberately diversified. Additionally, every frame within each video clip was annotated to support precise clip-level supervision during training. Representative frames for three behavior classes and multiple individuals are shown in Figure 13.

Table 1: Distribution of annotated behaviour classes in the custom in-cabin distraction dataset.

Behavior Class	Number of Videos	Percentage (%)	Duration (mm:ss)
Drinking	103	15.77%	05:09
Safe Driving	124	18.99%	06:12
Smoking	62	9.49%	03:06
Talking on Phone	104	15.93%	05:12
Talking to Passenger	129	19.75%	06:27
Texting on Phone	131	20.06%	06:33
Total Duration	32 minutes and 39 seconds		

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI
KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

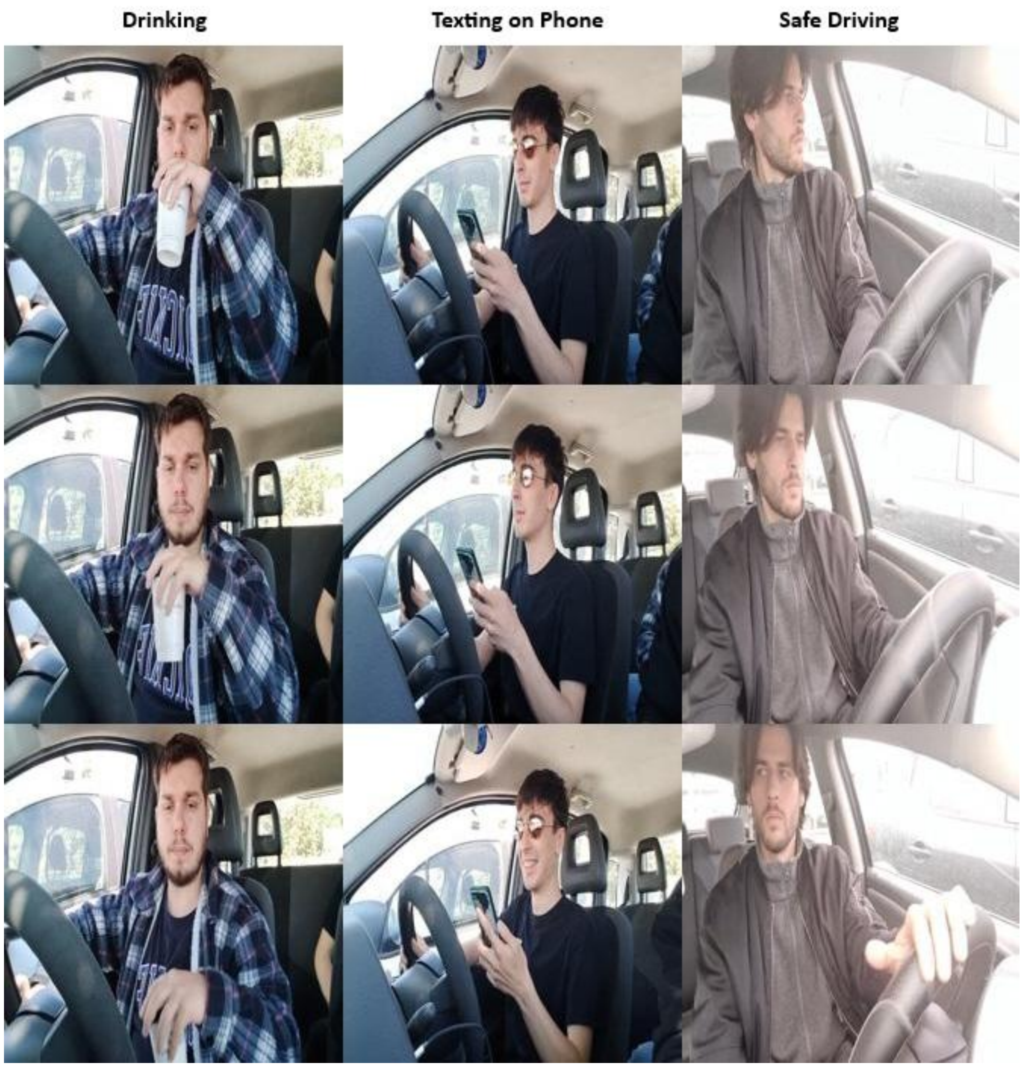


Figure 13: Example frames from the custom in-cabin distraction dataset. Each column shows a behaviourbehaviour class (Drinking, Texting, Safe Driving) and each row shows a different individual performing that behaviourbehaviour.

Methodology

The distraction detection model is seeded on top of MoViNet-A3, a lightweight spatiotemporal model for video optimized for edge inference, an optimal use of computer vision at the edge (see Figure 14). The model was designed to take in sequences of RGB frames and learn features based on motion content in the video to provide decision-making information across similar upper-body gestures temporally.

Notable model features:

- Temporal convolutional layers to encode motion
- Depth-wise separable convolutions for reduced computations

- Partial fine-tuning on the last layers of training the dataset for training purposes only
- Input resolution: 224×224 RGB
- Sequence length: 72 frames (3 seconds)

Data augmentations: random cropping, brightness, frame jitter, and horizontal flipping to help with generalization.

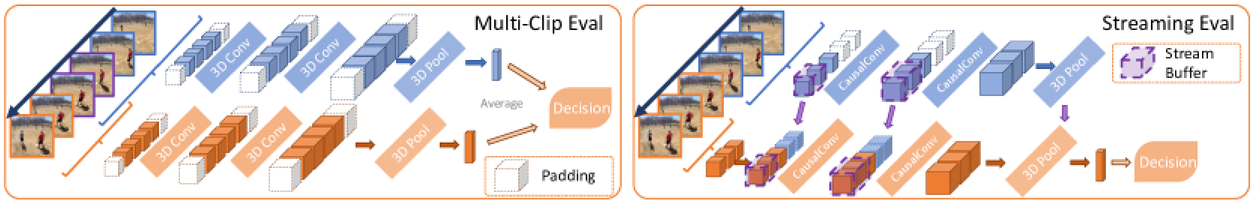


Figure 14: MoviNet Architecture

Training

Splitting the dataset into training and testing subsets was accomplished through an 80/20 per-class random shuffle. This reproducible process guarantees balanced class representation in both sub-sets and protects against data "leakage" for consistent and reliable evaluations. The model was trained using TensorFlow and exported to ONNX format for deployment.

Training configuration:

- Optimizer: Adam
- Learning rate: 0.0002
- Batch size: 8
- Epochs: 20
- Loss: Categorical cross-entropy
- Hardware: NVIDIA RTX 4090 GPU (training), Jetson Orin AGX (inference)

In Figure 15, the training validation performance was stable around epoch 12 with the early stopping and learning rate reduction used to counter overfitting, and the best model was exported to TensorRT (the optimal format for embedded deployment).

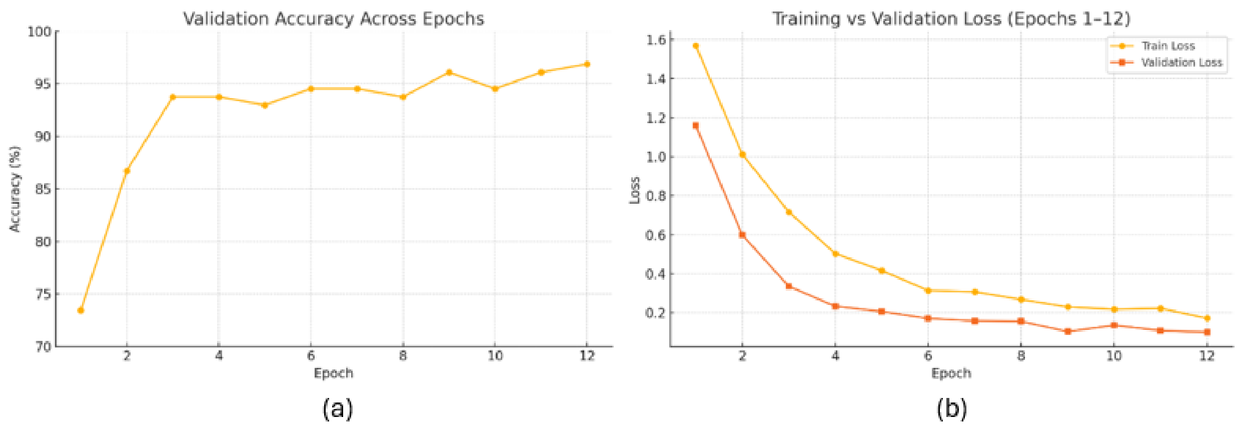


Figure 15: Training and deployment configuration for the distraction detection model.

Experimental Results

The model was tested on a held-out test set with the following metrics:

- **Clip-level classification accuracy**: 89%
- **Average F1-score (macro)**: 0.88
- **Per-class F1 range**: score ranged ±5% from the mean
- **Inference latency**: < 100 ms per video clip (on Jetson Orin AGX)
- **Memory footprint**: < 300 MB (FP16)

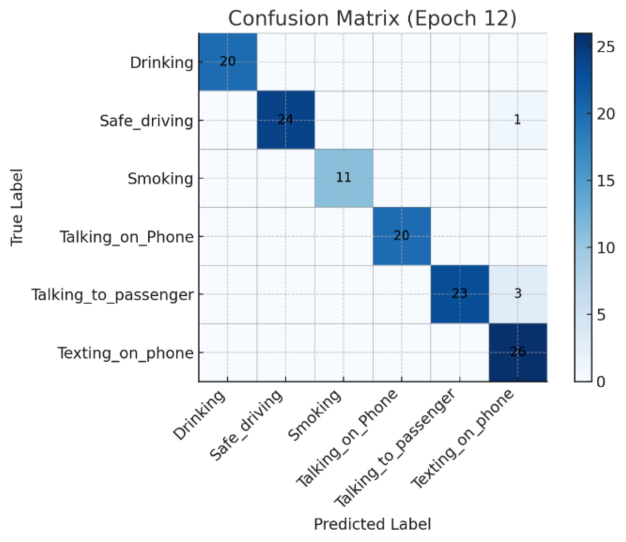


Figure 16: Confusion matrix illustrating the model's classification performance across behaviour classes on the test set

The model exhibited high temporal stability and correctly differentiated visually similar behaviours, such as "reaching" vs. "drinking", especially with the combination of inputs from front and side camera views. Confusion was lowest for high-motion behaviours (such as phone use), and highest for overlapping gestures (such as reaching vs. adjusting) as demonstrated in Figure 16 and Table 2.

Table 2: Per-class performance metrics of the distraction detection model on the test set.

Class	Precision	Recall	F1 Score	Accuracy	Correct / Total
Drinking	1.00	1.00	1.00	1.00	20 / 20
Safe_driving	0.89	0.96	0.92	0.96	24 / 25
Smoking	1.00	1.00	1.00	1.00	11 / 11
Talking_on_Phone	1.00	1.00	1.00	1.00	20 / 20
Talking_to_passenger	0.96	0.88	0.92	0.88	23 / 26
Texting_on_phone	1.00	1.00	1.00	1.00	26 / 26

Deployment and Integration

The whole distraction detection module is run solely on the Jetson Orin AGX and was containerized to allow module integration; it takes a live feed from the side-facing driver camera, and outputs class probabilities every 3 seconds.

The predictions of behaviour from this module will:

- Be forwarded to the risk evaluation layer
- Be used in real-time by the virtual assistant (e.g., distraction warning)
- Be Logged for a session-level behavior monitoring

This module can run offline, has low power consumption, supports configuration with YAML for class definitions and threshold levels.

2.1.7. Event-Based Driver Distraction Detection

In addition to RGB cameras, the Driver Distraction Detection task is addressed using event-based sensors and Spiking Neural Networks (SNNs). Event-based cameras capture changes in light intensity rather than full image frames, effectively suppressing static backgrounds and emphasizing motion, an advantage for accurately classifying driver activities. Furthermore, SNNs represent a new generation of neural networks that are biologically inspired, highly energy-efficient, and well-suited to processing event-based data.

Dataset Pre-processing

Due to the limited availability of event data, techniques that can convert RBG images to event-based representations for SNN input are evaluated. The dataset described in 0 are converted to spike trains using the random coding approach, first presented in [6]. Using this method, a fixed value TS, referred to as the spike emission duration or number of timesteps per image, is defined. For each pixel in a normalized RGB image (values scaled to the [0,1] range), a spike is emitted at a given timestep if the pixel value exceeds a randomly sampled number from a uniform distribution over [0,1]. Consequently, brighter pixels (e.g., with intensity 0.9) have a higher probability of firing approximately 90% across the TS timesteps, while darker pixels spike

From each video (Table 1), a total of 75 frames were sampled from the middle part. Each frame was then converted to grayscale and resized to a resolution of 128×128 pixels. The final preprocessing step involved converting the frames into spike trains using a total of TS = 110 timesteps. Figure 17 Illustrates an RGB frame from the Driver Distraction Detection dataset converted into an event-based format using the described method. Moving forward, event-based sensors will be employed to capture driver distraction datasets, enabling more efficient data representation and further enhancing the performance of SNN models. Figure 18 provides an example of distracted person using his phone, captured with the Prophesee event-based sensor.

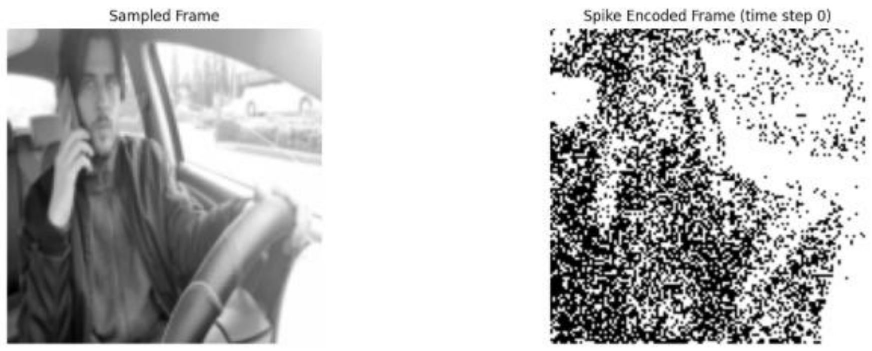


Figure 17: An RGB frame from the Driver Distraction Detection dataset (left) converted into an event-based representation using spike encoding (right)

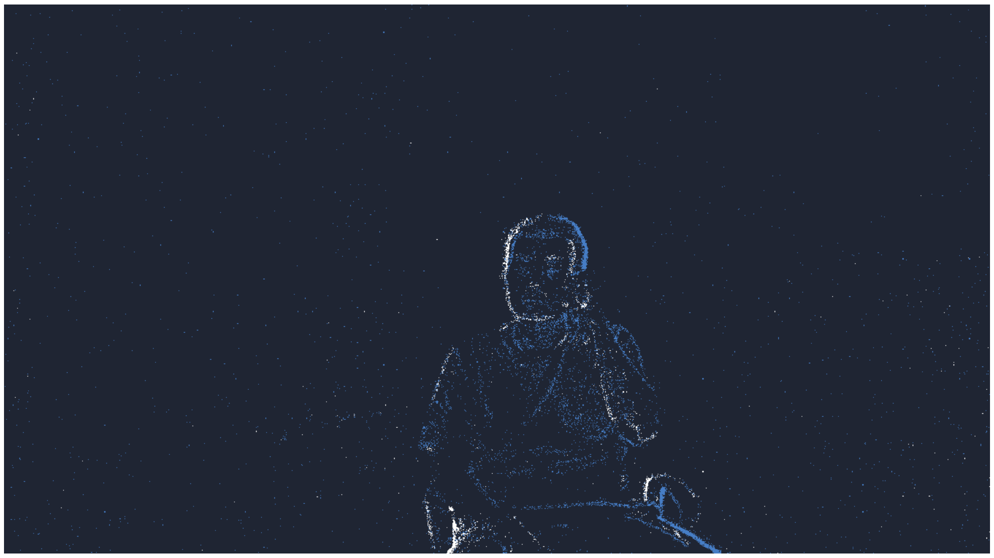


Figure 18: Driver distracted by phone, captured with a Prophesee event-based sensor

Methodology

This section presents the architecture of the 3D Spiking Convolutional Neural Network (SCNN). The core building block of the model is the 3D Spiking Block, which comprises a Convolutional layer, a Spiking layer, and a Max Pooling layer.

- Spiking Neuron Model

The Leaky Integrate-and-Fire (LIF) neuron is a fundamental, biologically inspired model commonly used in Spiking Neural Networks (SNNs). It mimics the behaviour of real neurons by integrating incoming signals over time, gradually leaking some of the accumulated potential, and firing a spike once a predefined threshold is exceeded. The membrane potential at time step t, denoted as U[t], evolves according to the following dynamics:

$$U[t] = \beta U[t-1] + WX[t] \qquad \text{Eq. 1}$$

where $\beta = e^{-1/\tau}$ is the membrane decay rate, W is a learnable weight and X[t] represents the input to the neuron.

A spike is emitted when U[t] exceeds the threshold U_{thr}. Following the spike, the membrane potential is reset. In our implementation, we use a subtractive reset mechanism. The spiking dynamics are thus defined as:

$$U[t] = \beta U[t-1] + WX[t] - S[t-1]U_{thr} \qquad \text{Eq. 2}$$

$$S[t] = H(U[t] - U_{thr}) \qquad \text{Eq. 3}$$

where $S[t] \in \{0,1\}$ denotes the spike output at t, $H(\cdot)$ denotes the Heaviside step function which is defined as $H(x) = 1$ for $x \geq 0$ and $H(x) = 0$ for $x < 1$. In our experiments, we initialize $beta$ to 0.9 but treat it as a learnable parameter (for all spiking layers except the final one), while U_{thr} is fixed to 1.

- 3D Convolutional Spiking Block

The proposed 3D SCNN architecture is designed to capture the spatiotemporal dynamics of event-based video data. In 3D convolutional layers, the kernel operates across three dimensions, namely height, width, and temporal depth, enabling the model to learn motion patterns over time alongside spatial features. Figure 19 illustrates the architecture of the model, which consists of 2 3D convolutional layers, followed each by a spiking neuron and a max pooling layer. Both 3D convolutional layers are configured with a kernel size of 3x3x3 and a stride of 1 in all dimensions. A fully connected layer followed by a spiking output neuron produces the final class prediction.

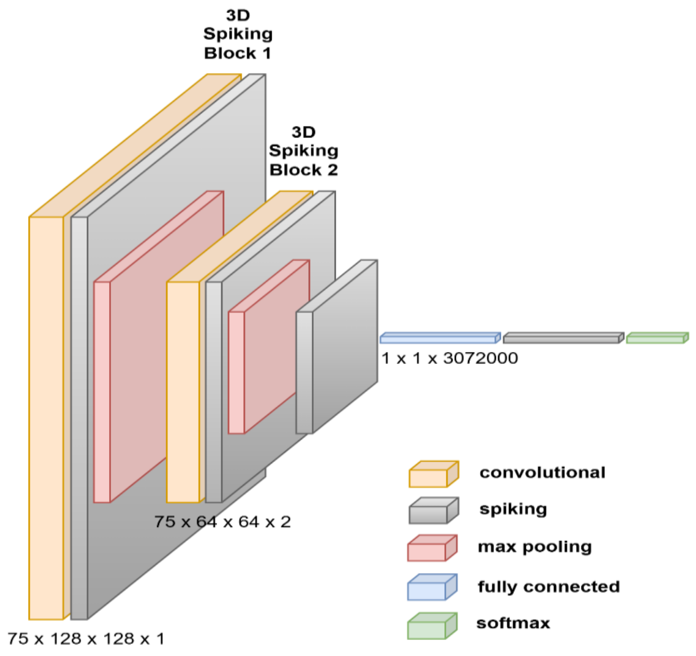


Figure 19: Overview of the proposed 3D CSNN architecture

Training

All models were implemented on PyTorch and trained on a NVIDIA GeForce RTX 3080 Ti GPU with 12 GB memory. Training was carried out for a maximum of 100 epochs, with an early stopping mechanism applied to stop training if no improvement in validation loss was observed. Lastly, we utilized the Adam optimizer with a 10^{-3}.

Experimental Results

Using a held-out evaluation set, we experimented with different neural network configurations, focusing on optimizing both the architecture and hyperparameters. The first experiment concerns selecting the optimal loss function, specifically comparing three commonly used loss functions for spiking neural networks: Spike Count loss, Max Membrane loss, and Spike Rate loss.

The architecture of Figure 19 was trained using each loss function and then tested on the evaluation set. Table 3 contains the accuracy, the macro and weighted average F1 scores when using each loss function. Among the evaluated options, Max Membrane loss consistently outperformed the other loss functions across all metrics, achieving the highest accuracy (0.78), macro-average F1 (0.77), and weighted-average F1 (0.78). In contrast, Spike Rate loss yielded the weakest results, indicating its limited effectiveness for this task.

Table 3: Evaluation results for different loss functions

Loss Function	Accuracy	Macro Average F1	Weighted Average F1
Spike Count loss	0.55	0.48	0.52
Max Membrane loss	0.78	0.77	0.78
Spike Rate	0.31	0.17	0.21

Next, the impact of architectural components of the model on performance was evaluated. More specifically, based on the architecture of Figure 19 which contains 2 convolutional layers and 3 spiking layers, configurations with different number of convolutional and spiking layers were trained and evaluated on the testing set. All models use the Max Membrane loss. contains the evaluation results for 4 different architectures.

Table 4: Evaluation results for different number of layers

Architecture	Accuracy	Macro Average F1	Weighted Average F1
3 Convolutional + 3 Spiking	0.82	0.81	0.82
3 Convolutional + 4 Spiking	0.81	0.79	0.81
3 Convolutional + 5 Spiking	0.61	0.56	0.59
4 Convolutional + 5 Spiking	0.63	0.56	0.61

The results indicate that architectures with 3 convolutional layers and 3 to 4 spiking layers yield the best performance, achieving accuracies above 0.80. Increasing the number of spiking layers beyond 4 or adding an additional convolutional layer led to a notable decline in performance across all metrics.

In conclusion, the best performing model utilizes the Max Membrane loss function, 3 convolutional and 3 spiking layers, achieving an accuracy of 82%. Figure 20 shows the confusion matrix for the model's predictions on the test dataset. While the performance is promising, there is still room for improvement, particularly through further adapting of the architecture to better suit the dataset and task. Most importantly, since SNNs are inherently better suited for event-based data, developing a dedicated event-based driver distraction detection dataset is an important next step.

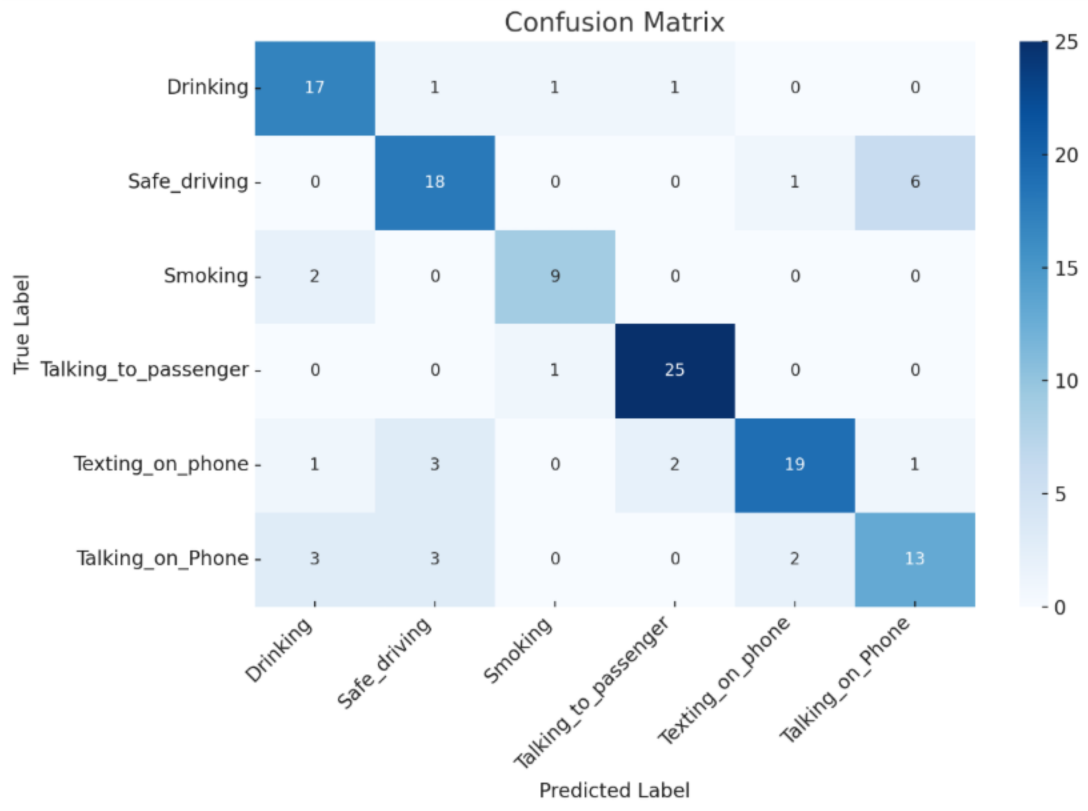


Figure 20: Confusion matrix for the model's predictions on the test dataset

2.1.8. Drowsiness Detection

This module proposes a real-time driver drowsiness detection system that detects symptoms of exhaustion using basic biometric rules, the Eye Aspect Ratio (EAR) and the Mouth Aspect Ratio (MAR), camera input, and a combined pre-trained ONNX model for facial landmark detection. When drowsiness is detected, the system communicates with a virtual assistant to issue auditory alerts, helping drivers/users to prevent fatigue-related accidents.

Co-funded by the European Union

Architecture

Two essential components are merged in the solution's combined ONNX model. The face detection model used in this module is InsightFace's SCRFD_10G_KPS. For the landmark detection, the 2D106Det model was integrated.

PyTorch is used to wrap these up with specific preprocessing and postprocessing processes before they are exported to ONNX for expedited processing. In the pre processing stage, the image is resized and padded to 1920x1920 pixels, converted from BGR to RGB and transposed from HWC to CHW format. Afterward, in the post processing phase, Non-Maximum Suppression is applied to face results, the detected faces are normalized and aligned via transformations and then cropped to 192x192 for landmark input. Finally, an inverse transformation recovers landmarks in the original frame.

Drowsiness Detection Methods

When the facial landmarks are detected, the key facial regions (eyes and mouth) are extracted to compute:

- EAR: Indicates eye openness. Low EAR means eye closure and micro-sleep.
- MAR: Measures mouth openness. Here, high MAR means yawning in combination with low EAR.

The technique also employs a rolling average over the previous 30 frames for both EAR and MAR values to increase stability and reduce noise. These rolling averages are compared to predetermined thresholds. Based on this comparison, the system classifies the driver into one of two states: Awake or Drowsy, as it is shown in Figure 21. If the EAR falls below the threshold, or if both the EAR and MAR reach critical levels, the driver is flagged as Drowsy, and an alert is issued.

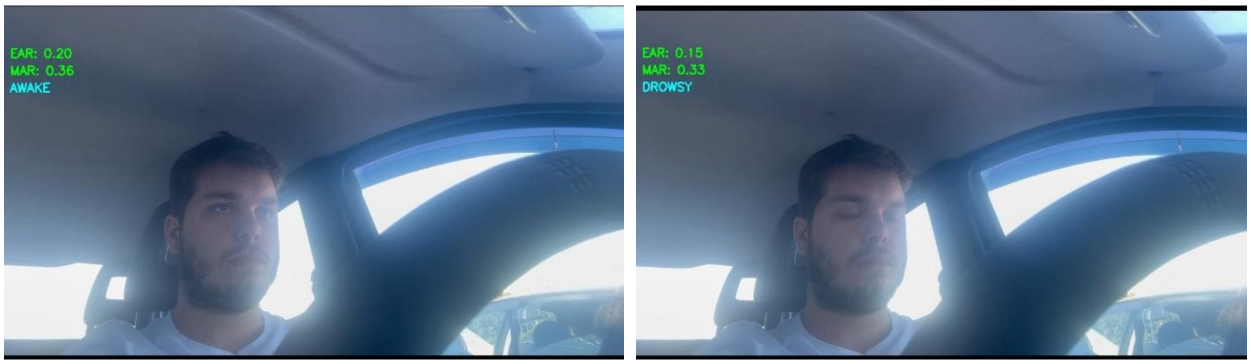


Figure 21: Binary classification of driver's drowsiness state

Deployment

The drowsiness detection module operates entirely on the Jetson Orin AGX, leveraging ONNX Runtime with CUDA acceleration for real-time performance. Video input from a USB RGB camera is processed by a combined ONNX model and then as shown in Figure 22, the drowsiness state of the driver/user is obtained. In case there is a drowsiness event, alerts are published to a shared memory fusion layer and sent to an embedded large language model (LLM) (quantized LLaMA 3–8B). The LLM then issues voice prompts to warn the driver and suggest corrective actions in real time.

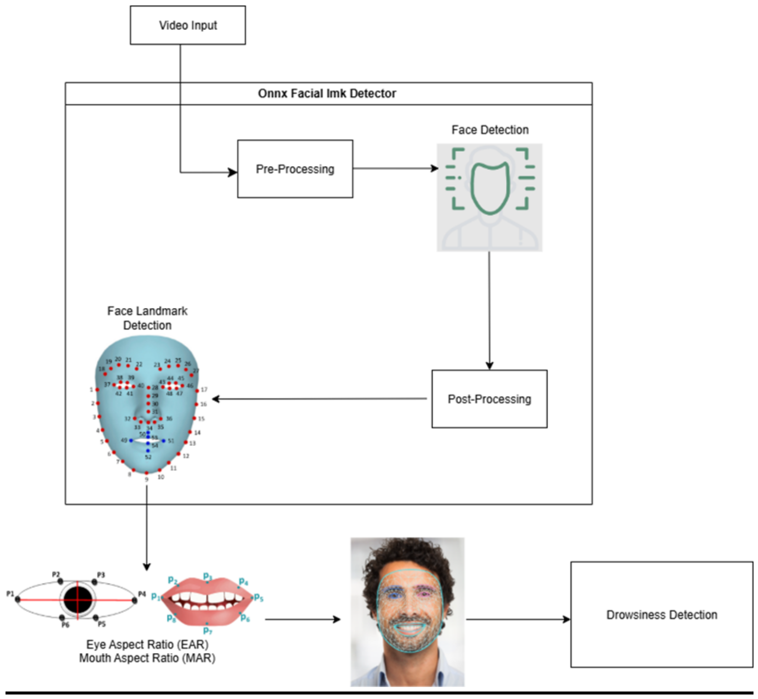


Figure 22: The Drowsiness Detection System's workflow diagram.

2.1.9. Abnormal Sound Event Detection

The abnormal sound event detection module continuously monitors the in-cabin acoustic environment to detect and classify auditory events that may signal distress, distraction, or safety-critical conditions.

Dataset Description

The training dataset includes eight classes of semantically relevant in-cabin acoustic events: *Baby cry, Noise, Scream, Siren, Snoring, Speech, Traffic noise, and Vehicle Horn*. These categories were selected to capture a range of situations that may affect driving performance including cognitive distractions, indicators of driver fatigue and external hazards requiring prompt attention as highlighted in the Driver Distraction Detection study (A safety-oriented framework for sound event detection in driving scenarios). Table 5 summarises the dataset composition, data sources, sampling specifications and total duration of each class. Several indicative waveform samples of a scream and a vehicle horn recording are illustrated in Figure 23 and Figure 24 respectively. The dataset was compiled from a diverse mix of publicly available sources, including environmental recordings, user-submitted content, speech corpora and multimedia collections. In total, the dataset comprises 3 hours and 20 minutes of labelled audio. This diversity in origin, format, and acoustic conditions enhances the model's robustness and supports generalization to real-world in-cabin scenarios.

Table 5: Summary of the acoustic event dataset used for training, including class names, source datasets, sampling characteristics and total duration per class.

Class	Dataset	Description	Sampling Rate	Total Duration (min)
Baby Cry	Donate-a-Cry[3]	User-submitted infant sounds	8 kHz	25.56
Noise	MIVIA [7]	Multi-SNR noise samples	32 kHz	25.08
Scream	Deeply Nonverbal[4] & Nonspeech [8]	Expressive vocalizations	16 - 32 kHz	25.22
Siren	Emergency Sirens [5]	YouTube/Google emergency vehicle sounds	44.1 kHz	25.12
Snoring	Snoring Dataset [9]	Compiled online snoring clips	44.1 - 48 kHz	25.00
Speech	LibriSpeech [10]	Audiobook speech from LibriVox	16 kHz	25.05
Traffic Noise	Background Noise Dataset [6]	Environmental samples from Kaggle	16 kHz	25.09
Vehicle Horn	UrbanSound8k [11] & HornBase [12]	Freesound & controlled recordings	16 - 96 kHz	26.33
Total Duration	3 hours and 22 minutes			

[3] https://github.com/gveres/donateacry-corpus
[4] https://github.com/deeplyinc/Nonverbal-Vocalization-Dataset
[5] https://www.kaggle.com/datasets/vishnu0399/emergency-vehicle-siren-sounds
[6] https://www.kaggle.com/datasets/moazabdeljalil/back-ground-noise

All audio clips were standardized to 16 kHz mono WAV format and normalized to ensure consistent amplitude scaling. Silent or low-energy segments were excluded using a threshold-based filter. Each clip was padded or trimmed to exactly 5 seconds.

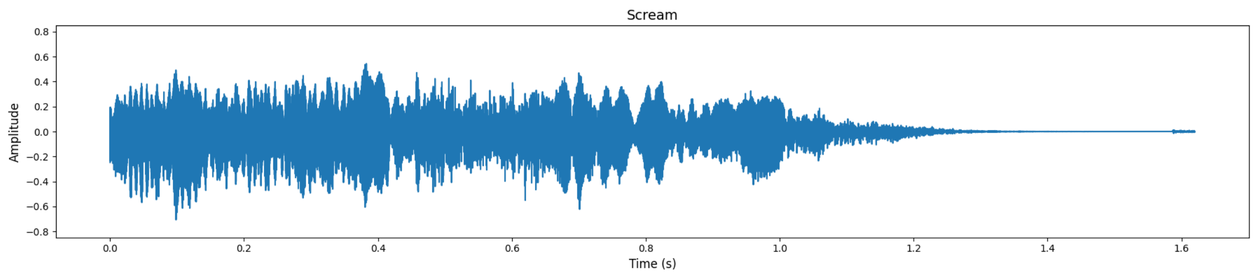


Figure 23: Waveform sample of 'Scream' class

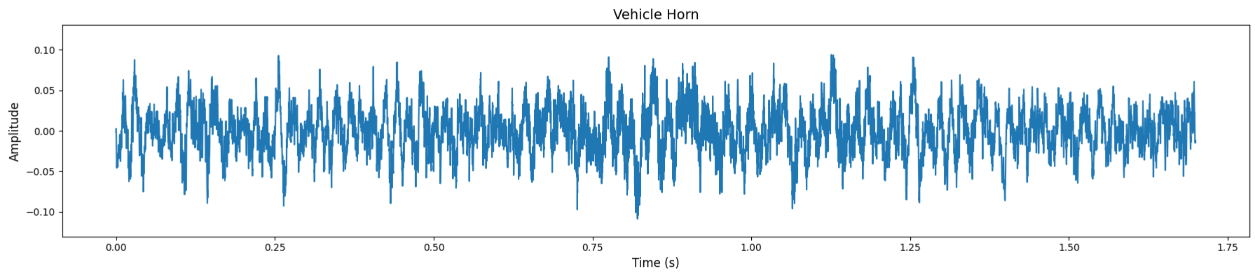


Figure 24: Waveform sample for 'Vehicle Horn' class

Methodology

The system adopts a two-stage architecture based on transfer learning. In the first stage, pretrained audio embeddings are extracted using YAMNet, a convolutional neural network trained on the large-scale AudioSet dataset [13]. This model converts raw waveforms into 1024-dimensional embeddings that encode high-level semantic representations of the acoustic input. By leveraging YAMNet as a fixed feature extractor, the system benefits from robust, general-purpose audio understanding without the need to train from scratch. In the second stage, a custom multi-layer perceptron classifier (MLP) is trained on top of these frozen embeddings to perform task-specific classification across the eight selected sound categories. The architecture of the classifier, is illustrated in Figure 25.

Training

The dataset was split into 70% training, 20% validation, and 10% testing using stratified sampling to maintain class balance. Training was conducted for up to 200 epochs using the Adam optimizer (learning rate = 0.001) and sparse categorical cross-entropy loss. To prevent overfitting and ensure stable convergence, early stopping was applied with a patience of 15 epochs, while learning rate reduction on plateau was triggered with a factor of 0.1 and patience of 10 epochs. Training employed a batch size of 32 with prefetching enabled for efficient data loading. Training and validation accuracy and loss curves are illustrated in Figure 26 showing stable convergence around epoch 50 and consistently high generalization performance on the validation set.

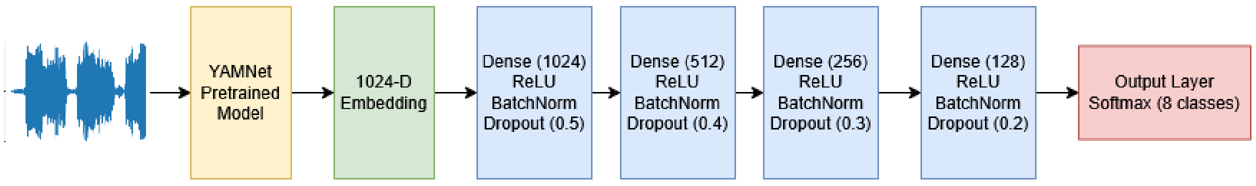

YAMNet Pretrained Model
1024-D Embedding
Dense (1024) ReLU BatchNorm Dropout (0.5)
Dense (512) ReLU BatchNorm Dropout (0.4)
Dense (256) ReLU BatchNorm Dropout (0.3)
Dense (128) ReLU BatchNorm Dropout (0.2)
Output Layer Softmax (8 classes)

Figure 25: Architecture of the abnormal sound event detection model

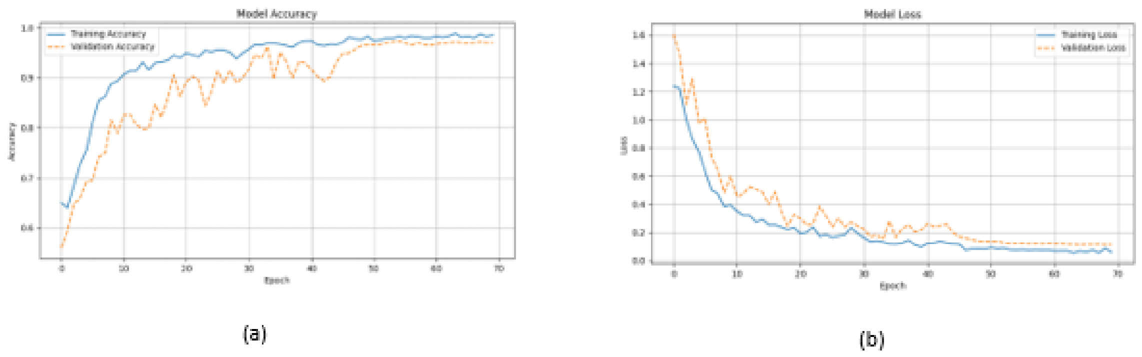


Figure 26: Training and validation (a) accuracy and (b) loss curves over epochs for the abnormal sound event detection model.

Experimental Results

The model was evaluated on a held-out test set comprising 522 audio samples evenly distributed across the eight target classes. It achieved a macro average accuracy of **94%**, with strong performance across all metrics. Table 6 presents the detailed classification report, including precision, recall, and F1 scores per class. Most categories exhibited F1 scores above 0.90, indicating consistent and reliable performance.

The macro-averaged F1 score reached 0.94, confirming the model's ability to generalize well across diverse sound types. Minor performance drops, such as in the Scream class (F1 = 0.87), are attributable to higher intra-class variability and potential overlap with Speech or Noise samples. Figure 27 demonstrates the confusion matrix, visualising the distribution of predictions across all eight classes. The diagonal dominance indicates high agreement between predicted and true labels, while misclassifications are sparse and largely limited to acoustically similar classes.

Table 6: Classification report for the abnormal sound event detection model on the test set.

Class	Precision	Recall	F1-Score	Support
Baby Cry	1.00	0.86	0.93	22
Noise	1.00	0.88	0.94	51
Scream	0.97	0.79	0.87	48
Siren	0.94	1.00	0.97	51
Snoring	0.97	0.99	0.98	151
Speech	0.94	0.98	0.96	51
Traffic Noise	0.84	1.00	0.91	51
Vehicle Horn	0.92	0.92	0.92	97
Macro avg	**0.95**	**0.93**	**0.94**	**522**
Weighted avg	**0.95**	**0.94**	**0.94**	**522**

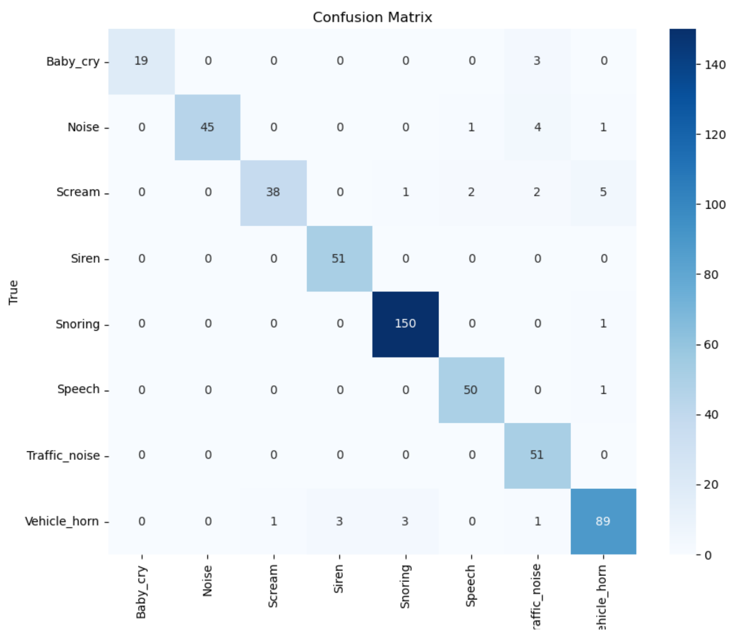


Figure 27: Confusion matrix for the abnormal sound event detection model.

Deployment

The abnormal sound event detection module runs entirely on the Jetson Orin AGX using ONNX Runtime with CUDA acceleration. Audio is captured via the ReSpeaker Mic Array V2.0, processed by the pretrained YAMNet model[7] for embedding extraction, and classified by a lightweight MLP classifier. The model outputs the predicted class and confidence score in real

[7] https://github.com/tensorflow/models/tree/master/research/audioset/YAMNet

time, publishing them to the shared memory fusion layer. These outputs are also forwarded to the embedded LLM (quantized LLaMA 3.1–8B), enabling context-aware reasoning and adaptive interaction. For instance, when such events are detected, the LLM can issue appropriate commands to notify the driver and assist in handling the situation.

2.1.10. Virtual assistant for occupant interaction

A context-aware Virtual Assistant has been developed to enable real-time, naturalistic interaction between occupants and the vehicle. The virtual assistant enhances driver comfort and safety by providing behavioural feedback related to the driver's cues of observable behaviour, intuition or emotional changes, and environmental sounds or alerts (car horns etc.). The rapid processing of these signals allows the assistant to provide adaptable behaviour with responses relevant to context and superior to pre-defined responses that characterize traditional computer assisted driving.

The virtual assistant relies on a LLM that conforms to the Meta architecture of LLaMA-3-8B-Instruct [14]. The development of the virtual assistant also involved the optimization of the LLM to operate on edge devices (NVIDIA Jetson platform) with int4 quantization using NanoLLM[8] and compiled with TVM via MLC[9] formalism. This allowed a significant reduction in operational memory requirements and processing while maintaining high quality processing language comprehension and production. The system consists of two components to support spoken interaction:

• Speech-to-text (STT) is done using NVIDIA Riva [10] offering low latency and accurate transcription using GPU accelerated pipelines.

• Text-to-speech (TTS) is done using Piper[11], a lightweight neural TTS system suitable for embedded platforms, providing natural and expressive voice generation.

The underlying services are all containerized (using Docker) to ensure modularity, scalability, and reproducibility across various environments. The Assistant is managed through a Python web UI to coordinate the audio I/O, response generation, and management of contextual states.

8 https://dusty-nv.github.io/NanoLLM/
9 https://llm.mlc.ai/
10 https://developer.nvidia.com/riva
11 https://github.com/rhasspy/piper

Co-funded by the European Union
Project funded by

Multimodal Input & Context-Aware Behaviour

The Virtual Assistant currently receives input exclusively from three key perception modules: Facial Emotion Recognition (Section 2.1.4), Driver Distraction Detection (Section 2.1.5), and Abnormal Sound Event Detection (Section 2.1.6). These modules operate independently, providing high-level semantic insights regarding the driver's emotional state, cognitive load, and relevant auditory events within the cabin. In addition, the assistant uses a dedicated microphone solely for speech-based user interaction and does not directly process other continuous sensor inputs.

Each contributing module performs its own noise-robust analysis and forwards pre-processed output to the assistant. To improve reliability, decisions are smoothed using majority voting across a temporal window, minimizing false positives or overreaction to brief, transient signals. Once a significant event or change is detected, such as a shift in emotional state, distraction behaviour, or alarming sound, a concise contextual summary is generated and provided as a prompt to the embedded language model.

The assistant's responses are formulated to be brief, context-aware, and minimally disruptive, aimed at enhancing driver situational awareness without introducing distraction. Additional modules described in Section 2.1 (e.g., drowsiness detection, passenger identification, object detection) are planned for integration in future development stages to further enhance the assistant's situational understanding and interaction capabilities.

Model Optimization with NanoLLM and TVM

Deploying large models on edge devices is extremely challenging because the devices have very limited computational and memory resources. To overcome these limitations, NanoLLM[8] was adopted, a framework for efficiently running transformer models on a diverse range of hardware. Using MLC[9] and TVM, LLaMA3 [14] model was converted into an optimized runtime format with int4 quantization resulting in substantial decreases resource utilization while maintaining response quality. With these resources optimizations, the assistant can run fully offline, there-by enabling low-latency inference and improved reliability in embedded spaces such as automotive platforms. The modularity of the NanoLLM framework also allows support to quickly transition to other hard-ware platforms as needed.

Real-Time Speech Recognition with NVIDIA Riva

The system's ASR (Automatic speech recognition) engine is NVIDIA Riva[10], which operates in a hosted service environment. Riva has the advantage of providing fast, GPU-accelerated transcription with very high accuracy under noisy in-car conditions. Riva has the ability to operate via gRPC-based streaming which supports continuous speech capture and transcription. The entire process executes locally in a Dockerized container on Jetson, and with this architecture, it required no remote background processing. It also melded seamlessly with the requirements of DGI and edge deployment for rapid response with no internet access in a vehicle.

Natural Text-to-Speech using Piper

With Piper[11] incorporating natural text to speech functionality for responding audibly became feasible. Modern neural TTS engines designed for edge performance, like Piper, fully phoneme text-to-speech systems utilizing compact and optimized voice models to synthesize speech. The selected voice had been precompiled which reduces the need to dynamically compile voices during playback resulting in fast synthesis.

The TTS pipeline is tightly integrated with the assistant's main logic, so that responses are vocalized instantly, preserving the sense of real-time conversation. Importantly, Piper is designed to run fully offline, avoiding cloud dependency and making it well-suited for the automotive domain, where latency and reliability are critical.

2.2. Cooperative saliency-based pothole detection and AR rendering solutions

This task focuses on enhancing road safety, driver awareness, and overall in-cabin experience by integrating cooperative perception and intuitive visualization mechanisms. The system under development leverages a saliency-based approach for detecting negative road anomalies (e.g., potholes) and employs Augmented Reality (AR) to render these hazards directly in the driver’s field of view. Emphasis is placed on real-time responsiveness, resource efficiency, human interpretability, and cooperative communication within connected vehicle networks.

Application Context and Benefits

The solution contributes in two key directions:

- **Enhanced Safety and User Experience**: By delivering timely and intuitive AR cues about hazardous road conditions, the system improves situational awareness and supports proactive driving behavior. It facilitates safer human-vehicle interaction within XR-enabled automotive ecosystems and is further reinforced through immersive training simulations.
- **Human-Centered Automotive Innovation**: The system integrates structured user feedback to refine the Human-Machine Interfaces (HMIs), ensuring that AR-based Advanced Driver Assistance Systems (ADAS) align with real-world expectations and

ergonomic needs. This inclusive design approach targets improved trust, comfort, and adoption across diverse user groups.

2.2.1. Saliency-based potentially hazardous obstacle detection

At the core of the system lies a saliency-aware detection framework that processes 3D LiDAR point clouds to identify road surface anomalies. Each point is assigned a saliency score derived from its local geometric and spectral properties, highlighting deviations from typical road topology. High-saliency regions often correspond to sharp geometric discontinuities—indicative of potholes or bumps.

Detected potholes are classified into “Far,” “Close,” and “Detected” states based on spatial thresholds relative to the ego vehicle. A rule-based filtering mechanism considers vehicle speed, obstacle severity (e.g., estimated volume), and proximity to prioritize information. This ensures that only the most critical cues are surfaced to the driver—minimizing distraction and preserving cognitive load. A colour-coded system indicates the severity of the detected potholes.

Visualization Enhancements Include:

- A **color-coded severity scale** (e.g., Green = minor, Orange = moderate, Red = severe) (Figure 28).
- Dynamic **size and elevation** adjustments of rendered pothole markers based on proximity and depth.
- **Selective rendering** logic to omit minor imperfections at high speeds or when not relevant to trajectory.
- This adaptive interface is critical to preventing tunnel vision or visual overload—recognized risks in AR-HUD designs.

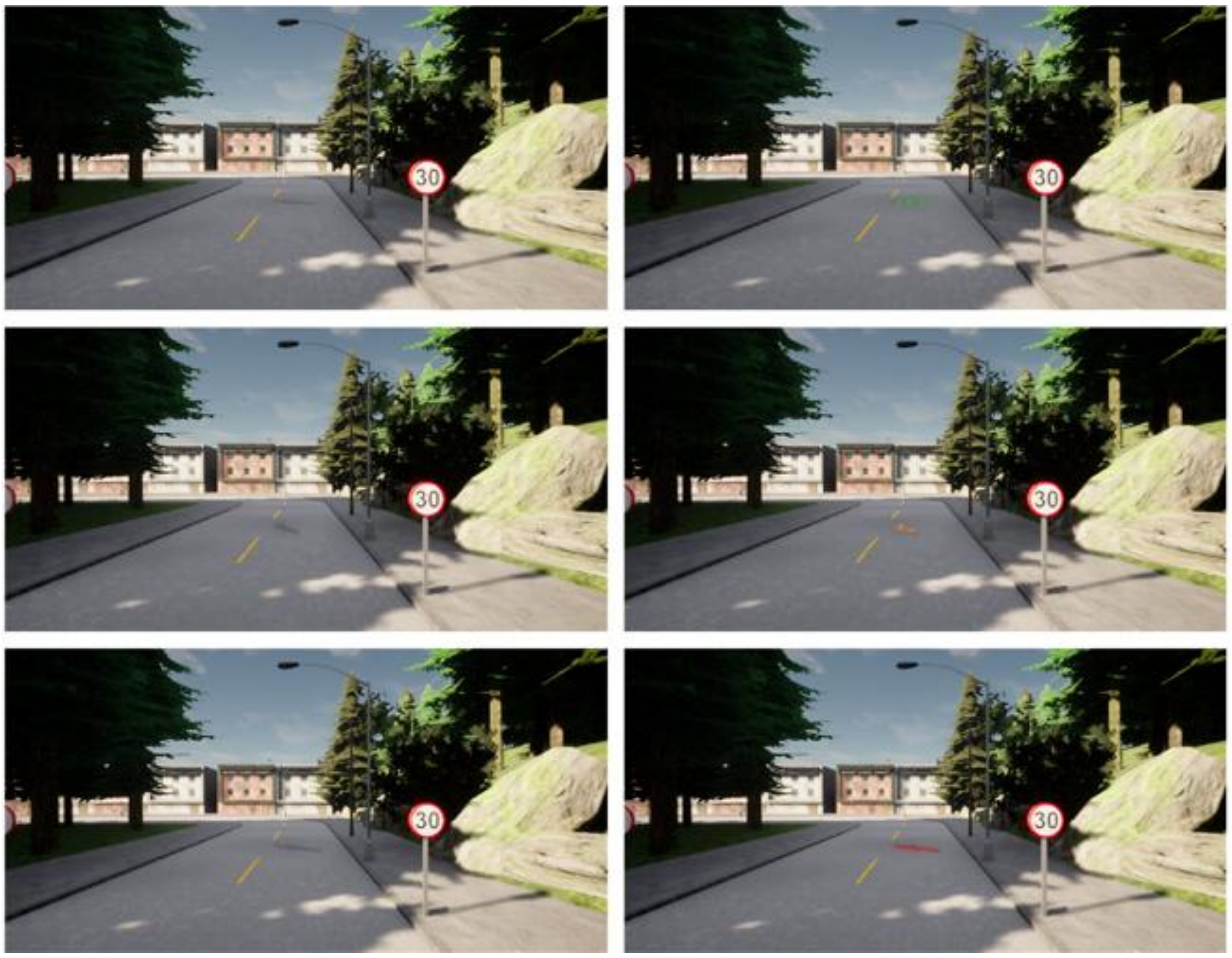


Figure 28: Overlay visualization of the pothole with different colours based on their severity. Different severity prioritization of potholes based on their size and volume.

2.2.2. Augmented Reality rendering & early hazard awareness

Upon detection, the AR module activates to project the pothole's location and severity directly onto the windshield interface. When the vehicle is within a 40-meter radius, a semi-transparent overlay appears above the pothole along with a floating warning icon. The rendering system is:

- **Non-intrusive**: Graphics are conformal and preserve scene continuity.
- **Scalable**: Supports visualization of multiple hazards in real-time.
- **Context-sensitive**: Tailored to driver behavior, road topology, and vehicle trajectory.

This approach enhances early awareness and affords drivers sufficient time to adapt speed or path, without diverting attention from the road.

2.2.3. Cooperative information sharing

To further support situational awareness beyond a single vehicle's perception capabilities, a **cooperative communication layer** has been integrated into the system. Detected pothole data is shared with other connected vehicles and roadside infrastructure, enabling a shared understanding of the driving environment across a network of vehicles. This cooperative approach enhances **system redundancy and robustness** and is crucial for forming a **collective environmental model** that can support more informed autonomous decision-making.

2.2.4. Technical Advancements and Implementation

The visualization of the potholes in the application is completed in two main phases: the creation phase and the visualization phase.

In the creation phase, a 3D mesh of the road with the pothole is constructed using Blender, an open-source 3D modelling application. After creating the pothole mesh, the road is imported in the Unreal Engine 5, where the application is developed.

Once the mesh is imported, we move into the visualization phase. It is assumed that the scanning and detection process have already been completed by another car with the needed sensors. Knowing the location and the shape of the hole have been recorded, this information is used to display a transparent marker in the position of the hole along with a warning sign above it, both of which are only visible in the windscreen display and are not rendered on the rest of the scene.

When the driver's car enters the detection range (default value is 40m), both the sign and the transparent marker become visible in the windscreen display of the car to alert the driver. Additional important detail is that the transform (location specifically on the z axis, rotation and scale) of the warning sign changes dynamically depending on the distance from the obstacle to ensure that it remains clearly visible to the driver.

2.2.5. Visual Material

The presentation includes visual demonstrations of:

- **Pothole visualization from ego-vehicle perspective** (Ego 1).
- **Early warning scenario** (Ego 2), where AR-enhanced overlays guide the driver with clear, in-situ road hazard cues (Figure 29).
- **Recognition phase showing dynamic detection states** (Detected, Turn, Close, Far) (Figure 30).

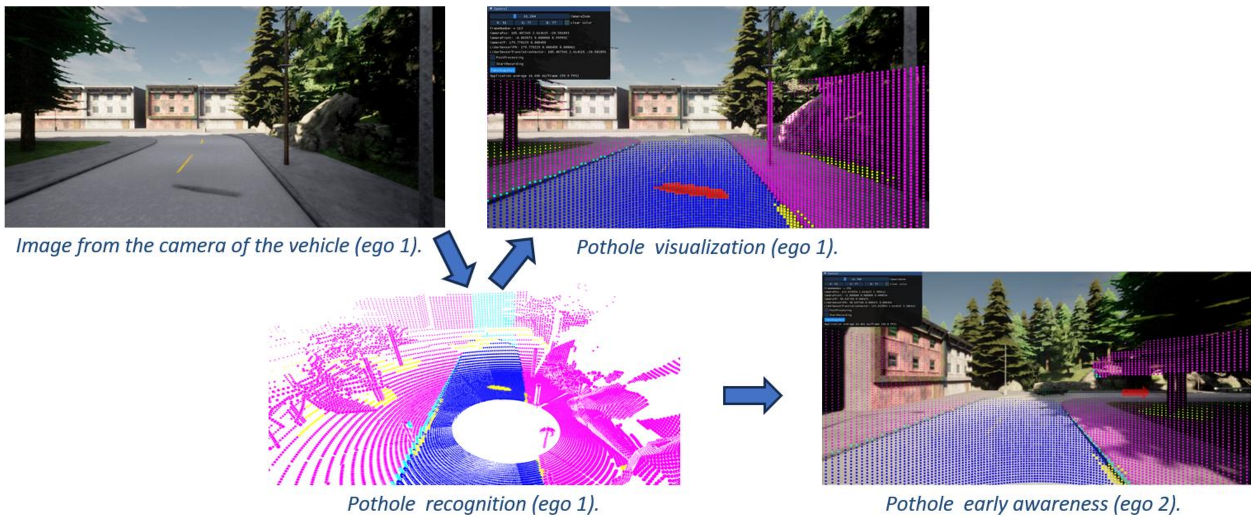


Image from the camera of the vehicle (ego 1).
Pothole visualization (ego 1).
Pothole recognition (ego 1).
Pothole early awareness (ego 2).

Figure 29: Pothole recognition and visualization for early awareness.

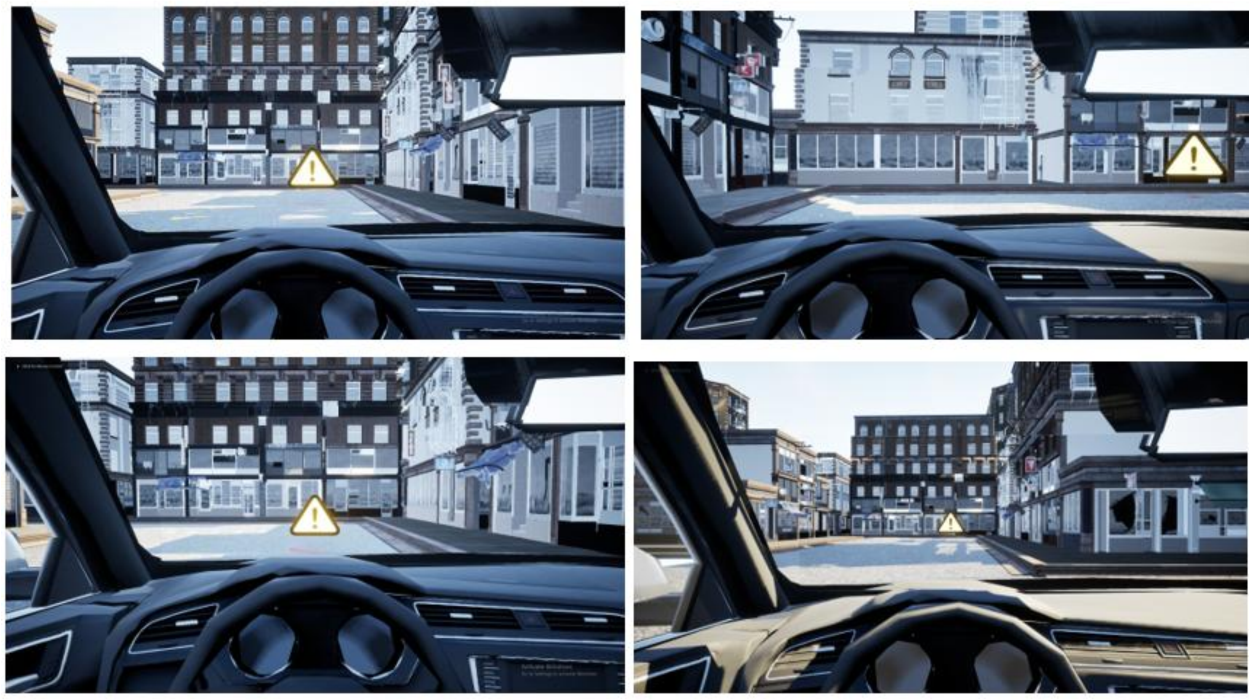


Figure 30: Visualization of pothole recognition in different distances.

2.2.6. Next Steps

Ongoing efforts are focused on further optimizing the detection latency, enriching the visualization interface with additional cues (e.g., bump severity levels), and integrating the system into full-stack CAV simulators. Future iterations will also incorporate feedback from human-centered evaluations to ensure ergonomic compliance and maximize driver acceptance.

2.3. In-vehicle air quality and thermal comfort analysis

Ensuring a comfortable, healthy, and safe in-vehicle environment for passengers relies heavily on the real-time monitoring, reporting, and analysis of air quality and thermal comfort conditions. The analysis of this data provides valuable insights into the in-cabin environment, facilitates the detection of abnormal conditions, and supports the initiation of corrective actions. These actions can be carried out by the driver or passengers, such as adjusting the operational settings of the air conditioning system to improve temperature conditions or opening windows to enhance ventilation. Furthermore, the development of spatiotemporal in-vehicle models enables detailed analysis of environmental conditions under various testing scenarios, contributing to the advancement of predictive and diagnostic capabilities aimed at preventing uncomfortable or unhealthy situations.

2.3.1. Smart Sensors Deployment

Sensors Properties

Two different types of sensors will be used for analyzing in-vehicle environmental conditions; a) advanced stationary smart sensors fixed in designated cabin locations and b) sensors integrated into mobile devices such as smartphones. Both sensor types are capable of measuring a variety of environmental parameters, including temperature, relative humidity, volatile organic compounds (VOCs), and particulate matter (PM), such as PM2.5 and PM10. Furthermore, the data collection frequency can be manually adjusted based on the conditions and the passenger on boarding rate, ranging from 2 to 5 mins—allows for continuous and dynamic monitoring, which is essential for promptly identifying abrupt environmental changes or discomforting conditions.

Sensors Placement

The strategic placement of advanced stationary sensors within the vehicle cabin is essential for ensuring accurate, reliable, and representative environmental measurements. These sensors must be installed in areas that are not directly influenced by dynamic and continuous airflow changes—such as those near air conditioning inlets or vents—as such locations may distort readings. Instead, the sensors should be positioned in zones that reflect the actual exposure of passengers, particularly near seating areas, where individuals spend most of their travel time.

To capture spatial variability effectively, stationary sensors will be installed at occupant head height and distributed across different sections of the bus—specifically at the front, middle, and rear of the cabin. This approach ensures a comprehensive spatial profile of air quality and thermal comfort conditions throughout interior space. In contrast, mobile sensors will not be fixed in place but rather carried by selected passengers. This allows for dynamic spatial mapping of environmental conditions as the sensors move through the cabin. When combined with data on passenger movement, these mobile measurements can offer deeper insights into personal exposure levels and localized environmental variations that fixed sensors may not fully capture.

It is also important to note that, for stationary sensors, a smart hub will be deployed to manage wireless connectivity and data transmission. This hub will be connected to a power source and strategically positioned near the driver's area, ensuring stable communication with all sensors and reliable data flow to the central platform at all times during bus operation.

Integration and Results Visualization

Both types of sensors—stationary and mobile—are seamlessly integrated with an IoT platform, which serves as the central system for data aggregation, processing, and management. Through the smart hub and standardized wireless communication protocols, real-time sensor data is transmitted to the platform, where it is securely stored, analyzed, and processed using advanced algorithms.

Following processing, the data is transmitted to an external visualization interface usable by relevant stakeholders, including drivers, passengers, and system operators. This interface allows for real-time monitoring of in-vehicle environmental conditions, as well as retrospective analysis through the exploration of historical trends. In addition to visualization, the platform supports anomaly detection, automatic alert generation, and control action suggestions—such as adjustments to the air conditioning system to restore or improve thermal comfort and air quality. An example of the visualization platform, along with the installed sensors, is presented in Figure 31.

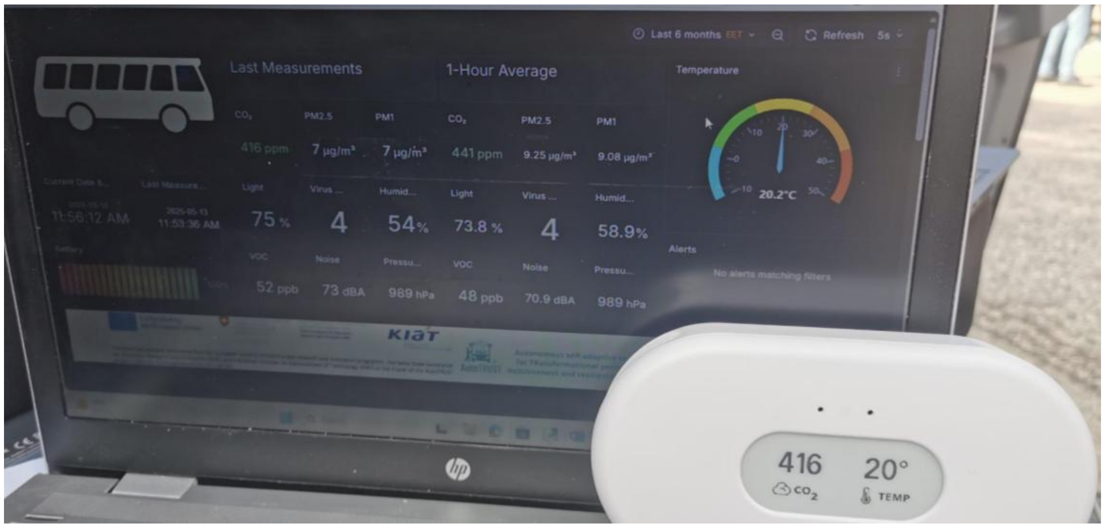


Figure 31: Example of the in-vehicle environment stationary sensor and visualization interface.

2.3.2. Spatiotemporal of in-vehicle environment using 3D Computational Fluid Dynamics

Even though real-time sensor data provides valuable point-based measurements, it cannot fully capture the complex behavior of airflows, the distribution of temperature or pollutant dispersion across the entire cabin space. Therefore, a high-fidelity three-dimensional Computational Fluid Dynamics (3D CFD) model is developed as a complementary tool to simulate and analyze the spatiotemporal behavior of the in-vehicle environmental parameters under various operating conditions.

This numerical virtual model provides a full-field view of the interior conditions, helping identify critical areas such as stagnation zones, thermal discomfort regions, or areas prone to pollutant accumulation. These insights are critical for diagnosing environmental inefficiencies and guiding improvements or operational adjustments to mechanical systems such as the AC units.

For numerical simulations, Reynolds-Averaged Navier–Stokes (RANS) equations are used for solving and modelling steady–state airflow and heat transfer within the vehicle cabin. Moreover, the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) algorithm was implemented for analyzing the pressure velocity coupling, while the standard k–ε turbulence model was considered for the turbulence effects [15], [16], [17], [18]. Furthermore, the geometry of the bus cabin was accurately reconstructed, and appropriate boundary conditions—such as inlet velocities, temperature profiles, and surface properties—were applied based on realistic operational data.

An example of the CFD simulation results is presented in Figure 32, showcasing the temperature distribution and airflow patterns throughout the bus interior. These results highlight the capability of CFD to analyze in detail environmental behaviors that cannot be detected by sensors alone—for example, the formation of hot or cold zones, inadequate ventilation regions, or recirculation areas.

Additionally, to the detail analysis of different in-vehicle environmental parameters, the CFD model can also be employed in various theoretical testing scenarios, including investigations of contamination events or evaluations of alternative mechanical system designs, such as various air conditioning (AC) unit configurations. This is because the model allows the simulation of different operating conditions, including variations in HVAC settings, window openings, passenger occupancy levels, and even localized pollutant sources—such as coughing or sneezing—relevant to airborne disease transmission. These scenarios provide valuable support for predictive diagnostics, allowing for the proactive identification of potentially uncomfortable or unhealthy cabin conditions.

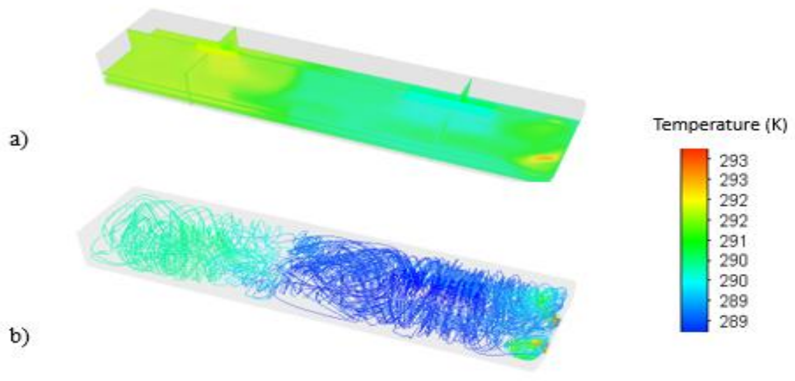


Figure 32: Example of In-vehicle CFD simulation for a) temperature distribution and b) air flow patterns

2.3.3. In-cabin air quality forecasting

This module provides real-time forecasting of in-cabin air quality by leveraging sensor data on CO_2, PM2.5, TVOCs, humidity, and temperature. Using a LightGBM-based predictive model trained on the GAMS Indoor Dataset, it continuously adapts to the latest environmental conditions to deliver short-term, minute-level forecasts. These predictions are integrated into a dashboard and connected with an onboard LLM to enable context-aware reasoning, early warnings, and personalized recommendations for healthier cabin environments.

Dataset

The GAMS Indoor Dataset [12] is used for model training. It contains high-frequency (1-minute) indoor air quality (IAQ) measurements from real environments, including CO_2, humidity, PM2.5, temperature, and TVOCs (Table 7). The dataset spans Nov 21, 2016 – Mar 28, 2017 (~135,099 data points).

Table 7: Format of the GAMS Indoor Dataset

Factor	Description	Units	Sample Value
CO_2	Carbon Dioxide concentration	ppm	708.0
Humidity	Relative Humidity	%	72.09
PM2.5	Particulate Matter ≤2.5 µm	µg/m³	9.0
Temperature	Air temperature	°C	20.83
TVOCs	Total Volatile Organic Compounds	ppm	0.062

[12] Twairball. (2017). GAMS indoor air quality dataset. GitHub. https://github.com/twairball/gams-dataset

Methodology

To forecast IAQ, environmental factors (CO_2, PM2.5, TVOCs, temperature, humidity) and time features (hour, day, month) were used. Short-term trends were captured by including pollutant measurements from the previous five minutes.

The LightGBM regressor [19] was selected due to its efficiency and accuracy, trained for 100 boosting rounds using mean squared error (MSE) as the loss function.

The forecasting task was set to a 1-minute prediction horizon, where the last five measurements are used as inputs. This setup is directly connected to real-time data from the sensor, ensuring that the model continuously adapts to the most recent environmental conditions.

Training and Results

Data was split chronologically to prevent leakage:

- Training**:** Before March 3, 2017 (~80%)
- Testing: From March 3, 2017 (~20%)

This ensures evaluation on future, unseen data to emulate real-world forecasting. The results of the training are shown in the Table 8.

Table 8: Evaluation Metrics for IAQ Parameters

Variable	RMSE	MAE	R^2
CO_2	6.62	3.59	0.9997
VOC	0.009	0.003	0.9921
PM2.5	0.64	0.44	0.9946

Deployment

Models are deployed on the Jetson Orin AGX using Python with CUDA for real-time inference. Sensor data is streamed to generate short-term air quality forecasts, which are integrated into a dashboard, as one can see in Figure 33, that visualizes both real-time and predicted values. The dashboard provides interactive charts, dynamic alarms, and session summary reports highlighting pollutant trends, threshold exceedances, and improvement tips. Forecast outputs are also processed in parallel with a quantized LLaMA 3–8B model to enable context-aware reasoning and personalized feedback, supporting preventive actions and healthier indoor conditions.

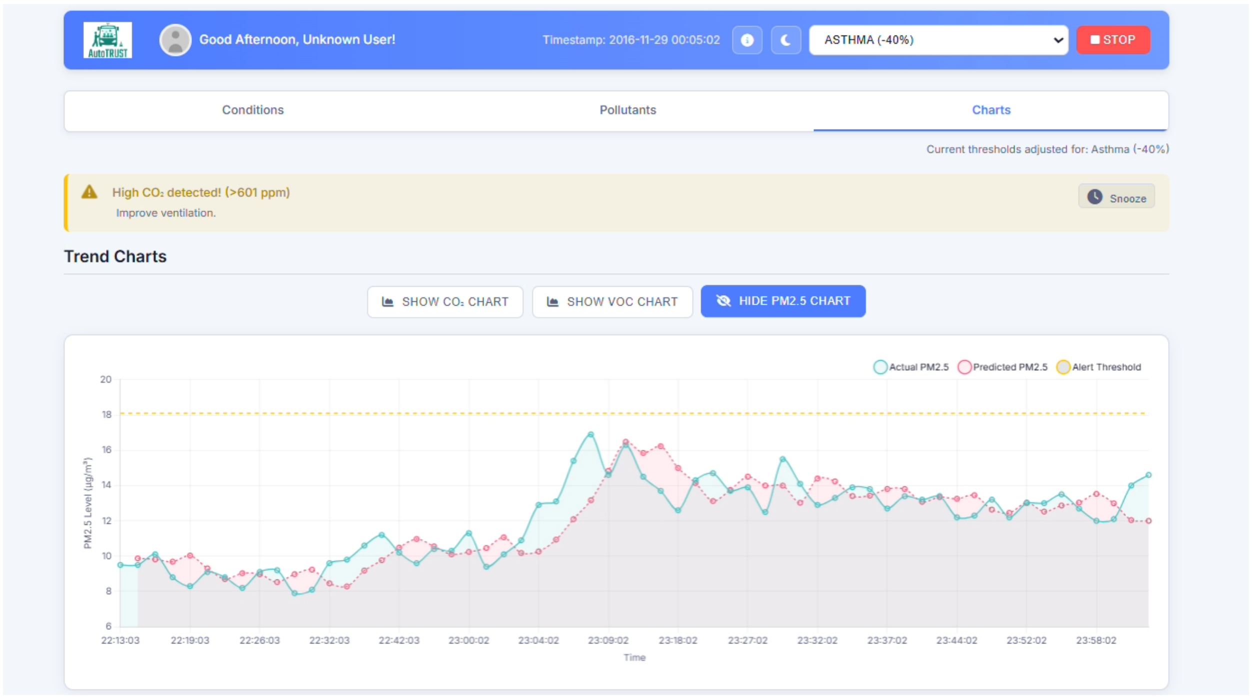


Figure 33: Real-Time IAQ Forecasting Dashboard

2.4. Visual analysis with object detection and vision-language models in public transport vehicles

An in-cabin monitoring system integrates multiple computer vision and machine learning components to assess passenger behavior, occupancy, and safety. The proposed system that is under development can potentially combine object detection, activity recognition, and vision-language models to interpret dynamic, multi-modal visual environments such as those inside public transport vehicles. Object detection modules, such as those based on the YOLO family of models, are employed to localize and classify individuals or objects within the scene, offering foundational spatial awareness. For more complex tasks such as identifying aggressive behavior, temporal video classification models are used to differentiate between violent and non-violent interactions using datasets tailored to transportation contexts. However, object detection and action classification alone cannot capture nuanced contextual information. Hence, Vision-Language Models (VLMs) are introduced to bridge this gap. VLMs generate descriptive narratives and answer questions about visual content, enabling high-level scene understanding and safety assessments. This multi-layered architecture creates a robust analytical pipeline capable of transforming raw video data into interpretable insights, making it suitable for scalable deployment in intelligent public transportation systems. Herein we describe the proposed components. However, further optimizations and refinements still need to be investigated to make sure that these components can be feasibly integrated into the pilot scenarios.

2.4.1. Object Detection in buses

In this case, we leveraged YOLO object detection models to monitor the inside environment of a bus. See Section 2.1.3 for the details about YOLO architecture and functionality. Such models can be initially applied to a video, capturing the interior of a bus, to detect passengers that are standing or seated. YOLOv11 which is pretrained on the COCO dataset was used for this case with Figure 34 showing its results. The model was able to detect passengers in the bus to monitor occupancy. With bounding boxes around each person, the model provides a detailed visual representation of the spatial arrangement of passengers inside the bus.

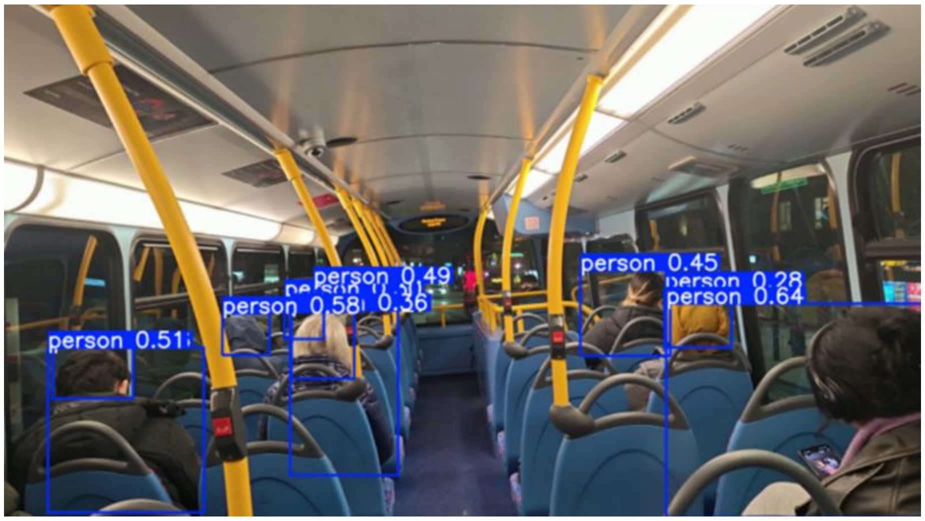


Figure 34: YOLOv11 detections of people.

Experiments were also carried out using the YOLO World model (Figure 35) which works by conditioning the detection output to a given text, representing the classes to be detected. In effect, this allows recognizing different objects that the model has not been explicitly trained on. Object classes could include "person," "seated person," "standing person," and "empty/vacant seat". However, the model was not that reliable in detecting these classes, therefore the approach was simplified to focus only on the "person" class. This first attempt provided a basic structure for object detection and classification within the bus environment.

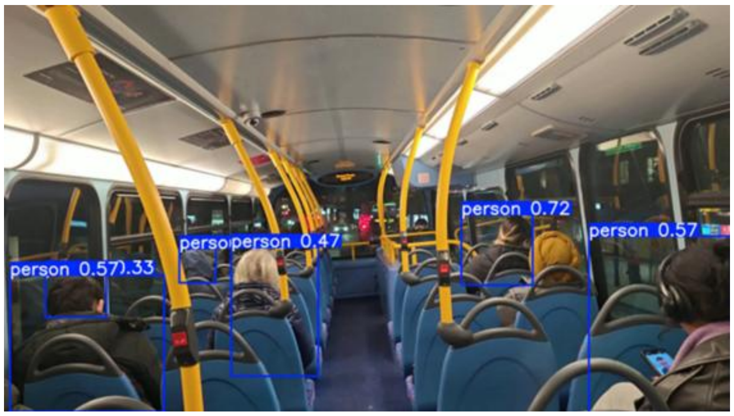


Figure 35: Example of detections using the YOLO World model.

2.4.2. Violence/Non-Violence Video Classification

As mentioned in D3.1, the Bus Violence dataset based on can be used in a bus monitoring system to train and evaluate models for automated violence detection in real-time onboard video feeds. By providing short, annotated video clips of both violent and non-violent interactions recorded from multiple camera angles inside a moving bus, the dataset enables the development of supervised learning models (e.g., CNN-RNN hybrids, 3D CNNs, transformers) that can learn temporal and spatial patterns associated with aggressive behaviours. Its balanced class distribution (700 videos classified as violent and 700 videos classified as non-violent) make it suitable for both training and benchmarking violence detection algorithms under realistic conditions, such as crowded scenes, occlusions, and camera motion.

Figure 36 below shows a snapshot of the video classification output during inference. In the top-left corner, the model's prediction, either violent or non-violent, is clearly displayed. The video classification model was trained for 100 epochs using the Bus Violence dataset, with ResNet18. serving as the backbone architecture reaching at the end an 0.95 accuracy rate.

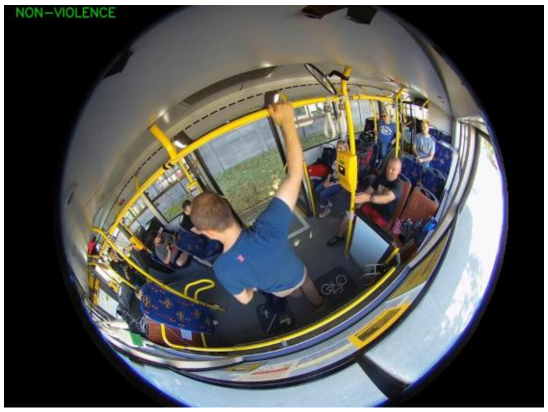
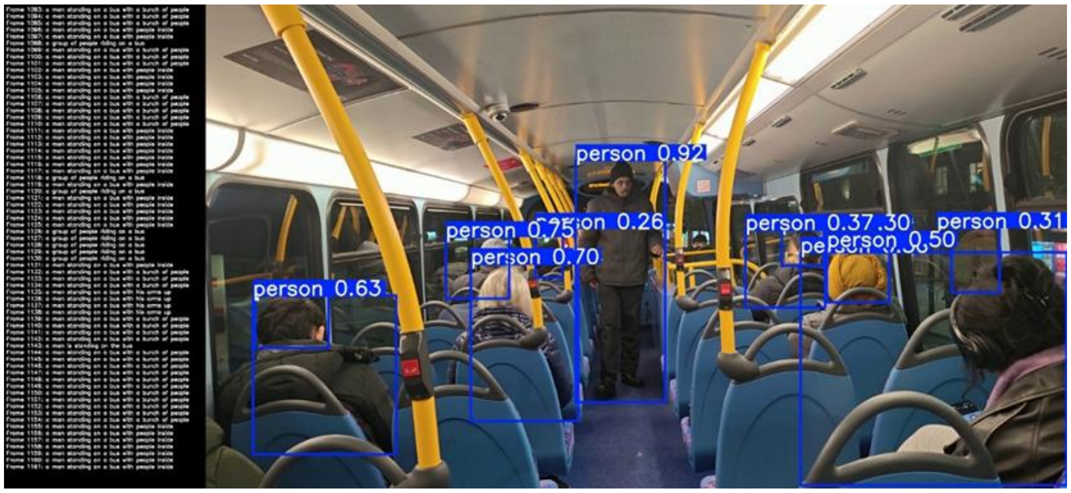


Figure 36: Model prediction output during inference on the Bus Violence dataset.

2.4.3. Vision Language Models for bus passengers identification

While object detection provides information, such as the types of objects and localization within an image, it does not provide information regarding higher level activities within a scene. For this reason, in this task, we exploit the combined usage of object detection with a VLM to describe the activities captured in each video frame. The VLM in this case, processed the visual data (namely each frame of the video) and generated textual descriptions of what is happening in the bus. These descriptions provided contextual insights, complementing the visual detections and offering a narrative of the scene. Figure 37 gives an example of the output of the processed video.

The final output is a combined video designed to present both visual detections and textual descriptions. The right side of the video displays the YOLOv11-generated bounding boxes overlaid on the frames of the video, highlighting detected objects such as the passengers of the bus. On the left side, the VLM-generated descriptions were displayed, providing a detailed account of the activities in each frame. A text prompt was further provided to the VLM as it processed the video to generate an appropriate response. For example, when given the prompt: *"What is happening on the bus, and is there any danger in it?"*, the model analysed each frame and responded: *"The bus is filled with passengers, and no visible danger is present."*



```
Frame 1093: a man standing on a bus with a bunch of people
Frame 1094: a man standing on a bus with a bunch of people
Frame 1095: a man standing on a bus with a bunch of people
Frame 1096: a man standing on a bus with people inside
Frame 1097: a man standing on a bus with people inside
Frame 1098: a group of people riding on a bus
Frame 1099: a man standing on a bus with a bunch of people
Frame 1100: a man standing on a bus with a bunch of people
Frame 1101: a man standing on a bus with a bunch of people
Frame 1102: a man standing on a bus with people inside
Frame 1103: a man standing on a bus with people inside
Frame 1104: a man standing on a bus with people inside
Frame 1105: a man standing on a bus with people inside
Frame 1106: a man standing on a bus with a bunch of people
Frame 1107: a man standing on a bus with a bunch of people
Frame 1108: a man standing on a bus with a bunch of people
Frame 1109: a man standing on a bus with a bunch of people
Frame 1110: a man standing on a bus with a bunch of people
Frame 1111: a man standing on a bus with people inside
Frame 1112: a man standing on a bus with people inside
Frame 1113: a man standing on a bus with people inside
Frame 1114: a man standing on a bus with people inside
Frame 1115: a man standing on a bus with people inside
Frame 1116: a man standing on a bus with people inside
Frame 1117: a man standing on a bus with people inside
Frame 1118: a group of people riding on a bus
Frame 1119: a man standing on a bus with people inside
Frame 1120: a group of people riding on a bus
Frame 1121: a man standing on a bus with people inside
Frame 1122: a man standing on a bus with people inside
Frame 1123: a man standing on a bus with people inside
Frame 1124: a man standing on a bus with people inside
Frame 1125: a man standing on a bus with people inside
Frame 1126: a group of people riding on a bus
Frame 1127: a group of people riding on a bus
Frame 1128: a group of people riding on a bus
Frame 1129: a group of people riding on a bus
Frame 1130: a group of people riding on a bus
Frame 1131: a man standing on a bus with people inside
Frame 1132: a man standing on a bus with a bunch of people
Frame 1133: a man standing on a bus with a bunch of people
Frame 1134: a man standing on a bus with a bunch of people
Frame 1135: a man standing on a bus with his arms up
Frame 1136: a man standing on a bus with his arms up
Frame 1137: a man standing on a bus with his arms up
Frame 1138: a man standing on a bus with his arms up
Frame 1139: a man standing on a bus with a bunch of people
Frame 1140: a man standing on a bus with a bunch of people
Frame 1141: a man standing on a bus with a bunch of people
Frame 1142: a man standing on a bus with a bunch of people
Frame 1143: a man is standing on the bus
Frame 1144: a man standing on a bus with a bunch of people
Frame 1145: a man standing on a bus with a bunch of people
Frame 1146: a man standing on a bus with a bunch of people
Frame 1147: a man standing on a bus with a bunch of people
Frame 1148: a man standing on a bus with a bunch of people
Frame 1149: a man standing on a bus with a bunch of people
Frame 1150: a man standing on a bus with a bunch of people
Frame 1151: a man standing on a bus with a bunch of people
Frame 1152: a man standing on a bus with a bunch of people
Frame 1153: a man standing on a bus with a bunch of people
Frame 1154: a man standing on a bus with a bunch of people
Frame 1155: a man standing on a bus with people inside
Frame 1156: a man standing on a bus with people inside
Frame 1157: a man standing on a bus with people inside
Frame 1158: a man standing on a bus with people inside
Frame 1159: a man standing on a bus with people inside
Frame 1160: a man standing on a bus with people inside
Frame 1161: a man standing on a bus with people inside
```

Figure 37: YOLOv11 detections of people in the bus with the descriptions of the VLM model.

Investigation of Prompting Techniques

Various VLMs models were tested with different prompts to analyse activities inside the bus and assess the level of safety and comfort based on the provided images or video footage. The following text prompts were used to extract insights from scenes depicting passengers inside the bus:

- “Explain this image in detail.”/ “Describe this image.”
- “What can you see on this image?” / “What can you say about this image?”
- “Can you describe thoroughly frame by frame what is going on in this video?” / “Can you explain thoroughly the activity of this video?”
- “Describe this image and state whether the passengers feel safe.”
- “Is there is any danger in the image?”
- “What is going on in this image?”
- “Make a summary of what is happening in the bus.”

The VLM models effectively described the images and video frames in detail, generating responses of two to three paragraphs while accurately identifying most objects in the scene. When prompted to assess passenger safety in a bus video, the model responded: *"The video does not show any danger in the bus."*

Exploration of Different VLM Models Optimized for Edge Deployment

VLM models require significant memory resources due to their ability to process both visual and textual data using large-scale transformer architectures. Running inference on multiple images and text sequences simultaneously further increases memory demands, making these models unsuitable for edge deployment.

Most VLM models contain over a billion parameters, contributing to their high memory consumption. However, advancements in computer vision have led to the development of more lightweight open-source alternatives. One such example is SmolVLM [13], which offers two compact models: SmolVLM-256 [14] and SmolVLM-500 [15], with 250M and 500M parameters, respectively. These are currently the smallest VLM models available showing competitive results when evaluated on different benchmarks.

In addition to testing the models on real images, as previously described, artificially generated images (shown in Figure 38) were also evaluated to assess their performance.

Both models successfully provided detailed descriptions of the images. However, some of the generated information was inaccurate, highlighting the limitations of smaller models in capturing the full contextual details of an image. Larger models, particularly those with around 7 billion parameters, demonstrated greater accuracy in this task, suggesting that a higher

[13] https://huggingface.co/HuggingFaceTB/SmolVLM-Instruct
[14] https://huggingface.co/HuggingFaceTB/SmolVLM-256M-Instruct
[15] https://huggingface.co/HuggingFaceTB/SmolVLM-500M-Instruct

number of parameters enhances the model's ability to interpret complex visual scenes more reliably.

Figure 38: AI generated images of people in a public bus.

Further Experimentation

Further experimentation involved fine-tuning VLMs using a dataset from Roboflow[16]. The figure below illustrates a prediction from Florence2[17], a 0.23B parameter model capable of caption generation, optical character recognition (OCR), segmentation, and object detection (shown in Figure 39). In this instance, it successfully detected the faces of bus passengers. Beyond these tasks, the model was also fine-tuned for Visual Question Answering (VQA). For example, when asked, “How many people are sitting in the bus?”, it responded, “The number of people sitting in the bus is 21.” After fine-tuning, the model was evaluated on a test set and demonstrated strong precision, with predictions closely aligning with the ground truth. In one case, when asked the same question, it predicted “The number of people sitting in the bus is 22,” while the ground truth was 23.

[16] https://universe.roboflow.com/cuenta-6ibit/cuenta-fujbt
[17] https://huggingface.co/microsoft/Florence-2-base

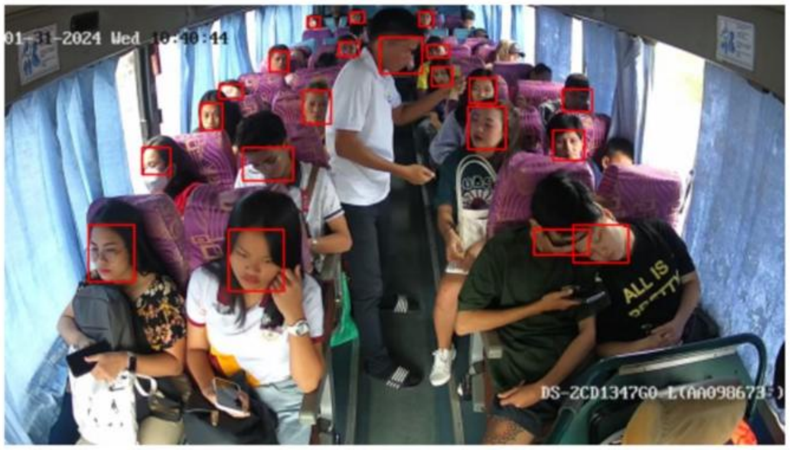


Figure 39: Florence2 detections on image.

The SmolVLM-Base[18] model (2.2B parameters) was further fine-tuned for visual question answering (VQA) using synthetic image data. Synthetic generated images paired with corresponding questions and answers, exemplifies the type of data employed during the fine-tuning process. Following fine-tuning, the model was evaluated on a test set and showed accurate performance. For instance, in response to the question, "How many people are sitting?", the model correctly predicted, "There are 5 people sitting in the bus," matching the ground truth. Alongside fine-tuning efforts, extensive experimentation with prompt engineering was conducted to determine which models best describe visual scenes or answer image-based questions. Platforms such as the Roboflow Playground[19] have proven useful in this regard, enabling users to test some of the most prominent models for these tasks.

Future Steps

The next steps involve gathering comprehensive visual data from buses operating in Cyprus to support the development of an intelligent monitoring system. This will be followed by integrating YOLO-based object detection models with VLM to enable accurate object recognition and generate contextual visual descriptions of bus environments. The aim is to develop a bus monitoring pipeline capable of real-time detection and scene understanding. In parallel, methodologies for deploying this system efficiently at the edge will be explored, ensuring low-latency performance and scalability for real-world applications.

[18] https://huggingface.co/HuggingFaceTB/SmolVLM2-2.2B-Instruct
[19] https://playground.roboflow.com/open-prompt?utm_campaign=Newsletter+-+4%2F10%2F2025+-+%5Bplayground%5D&utm_content=Newsletter+-+4%2F10%2F2025+-+%5Bplayground%5D&utm_medium=email_action&utm_source=email

2.5. Concluding remarks

This section has presented the AutoTRUST comprehensive and integrated approach to in-cabin monitoring and occupant interaction system, emphasizing the role of multimodal sensing, edge computing, and real-time analytics in enhancing vehicle safety, personalization, and user well-being. Through detailed descriptions of camera and microphone placements, sensor modalities, and deep learning-based inference pipelines, the section has demonstrated the feasibility and effectiveness of deploying complex AI models on embedded platforms under constrained conditions. From driver behavior and drowsiness detection to facial recognition, emotion understanding, and abnormal sound event classification, each module contributes to a layered perception framework that enables context-aware responses and autonomous support mechanisms. The integration of a virtual assistant powered by a quantized LLM further exemplifies how semantic-level understanding and human-like interaction can be embedded directly into the vehicle environment, minimizing reliance on cloud connectivity and improving responsiveness. In conclusion, the work outlined in this section lays the technical and conceptual foundation for a new generation of intelligent, human-centered vehicle cabins—capable of interpreting, adapting to, and actively supporting the occupants in a safe, private, and contextually rich manner. In the following section, we move from the interior of the vehicle to its exterior and present AutoTRUST innovative approach.

Co-funded by the European Union
Project funded by
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI
KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

3. External sensing system of 4D cooperative situational awareness for scene analysis, safe and inclusive mobility

The evolution of cooperative perception and decision-making systems has become a cornerstone in enabling safe, context-aware, and intelligent mobility solutions for connected and automated vehicles (CAVs). This chapter outlines the development and integration of AutoTRUST's advanced external sensing system designed to deliver 4D cooperative situational awareness, enhancing both safety and inclusivity in modern mobility ecosystems. To this end, a layered approach was followed, beginning with single-agent scene analysis modules (such as road condition assessment and traffic sign detection) that operate at the edge, using real-time vision-based perception models deployed on embedded platforms like the NVIDIA Jetson Orin AGX. The focus then expands toward multi-agent systems that enable collaborative awareness and localization through standardized V2X messaging, distributed learning, and cooperative tracking strategies, incorporating techniques such as federated learning, Kalman filtering, and graph signal processing, laying out a comprehensive framework for situational awareness that is scalable, robust, and communication-efficient. The chapter is structured as follows. Section 3.1 covers single-agent scene analysis, emphasizing contextual perception through road condition assessment and traffic sign detection. Section 3.2 expands the scope to multi-agent cooperation, showcasing the use of standardized V2X messaging, cooperative localization, and distributed optimization techniques for multi-object detection and tracking. Finally, our concluding remarks are provided in Section 3.3.

3.1. Single-agent scene analysis for contextual understanding

3.1.1. Road condition assessment

The road-condition assessment module continuously monitors the external environment, by using an internal outward camera in front of the driver to detect potholes on the road.

Dataset Description

The Road Pothole Images dataset[20] comprises real-world roadway photographs annotated with surface-defect bounding boxes to facilitate pothole detection. The original release contains two subsets—a "Simplex" set with 3 403 images and a "Complex" set with 8094 images—capturing

[20] https://www.kaggle.com/datasets/sovitrath/road-pothole-images-for-pothole-detection

diverse lighting, pavement textures, and pothole shapes. In this work, we leverage the Simplex subset, which includes 3 126 training images and 277 test images. Each image is annotated via plain-text files (simpleTrainFullPhotosSortedFullAnnotations.txt and simpleTestFullSizeAllPotholesSortedFullAnnotation.txt), where each line specifies an image filename, the number of potholes it contains, and the pixel-coordinates (x_min, y_min, width, height) of each pothole instance.

The Table 9 summarizes the composition and organization of our processed dataset. All annotations were parsed and converted into YOLO-compatible label files (one class: "pothole"), and the images were reorganized into three splits—3 126 for training, 277 for validation, and 277 for testing—via the custom preparation script.

Table 9: Composition and YOLO-compatible organization of the processed pothole dataset.

Subset	Source Folder	Images	Annotation Files
Train	Dataset 1 (Simplex)/Train data/Positive data	3126	simpleTrainFullPhotosSortedFullAnnotations.txt
Validation	Dataset 1 (Simplex)/Test data	277	simpleTestFullSizeAllPotholesSortedFullAnnotation.txt
Test	Dataset 1 (Simplex)/Test data	277	simpleTestFullSizeAllPotholesSortedFullAnnotation.txt

- Image format: JPEG, varying resolutions
- Classes: 1 ("pothole")
- Annotation format: text file with image_id, num_potholes, and repeated (x_min, y_min, width, height) entries per pothole
- Splits: reorganized into train/, valid/, and test/ directories with separate images/ and labels/ subfolders
- Label conversion: coordinates normalized and formatted as class x_center y_center width height in YOLO style by custom script.

Model Architecture

The detector core is Ultralytics YOLOv8-s[21], an 11 M-parameter network that fits entirely in the on-chip SRAM of modern embedded GPUs. We export the model from PyTorch to ONNX (FP32). At inference, the network produces bounding-box coordinates, objectness scores and class logits for the five target object categories.

[21] https://docs.ultralytics.com/models/yolov8/

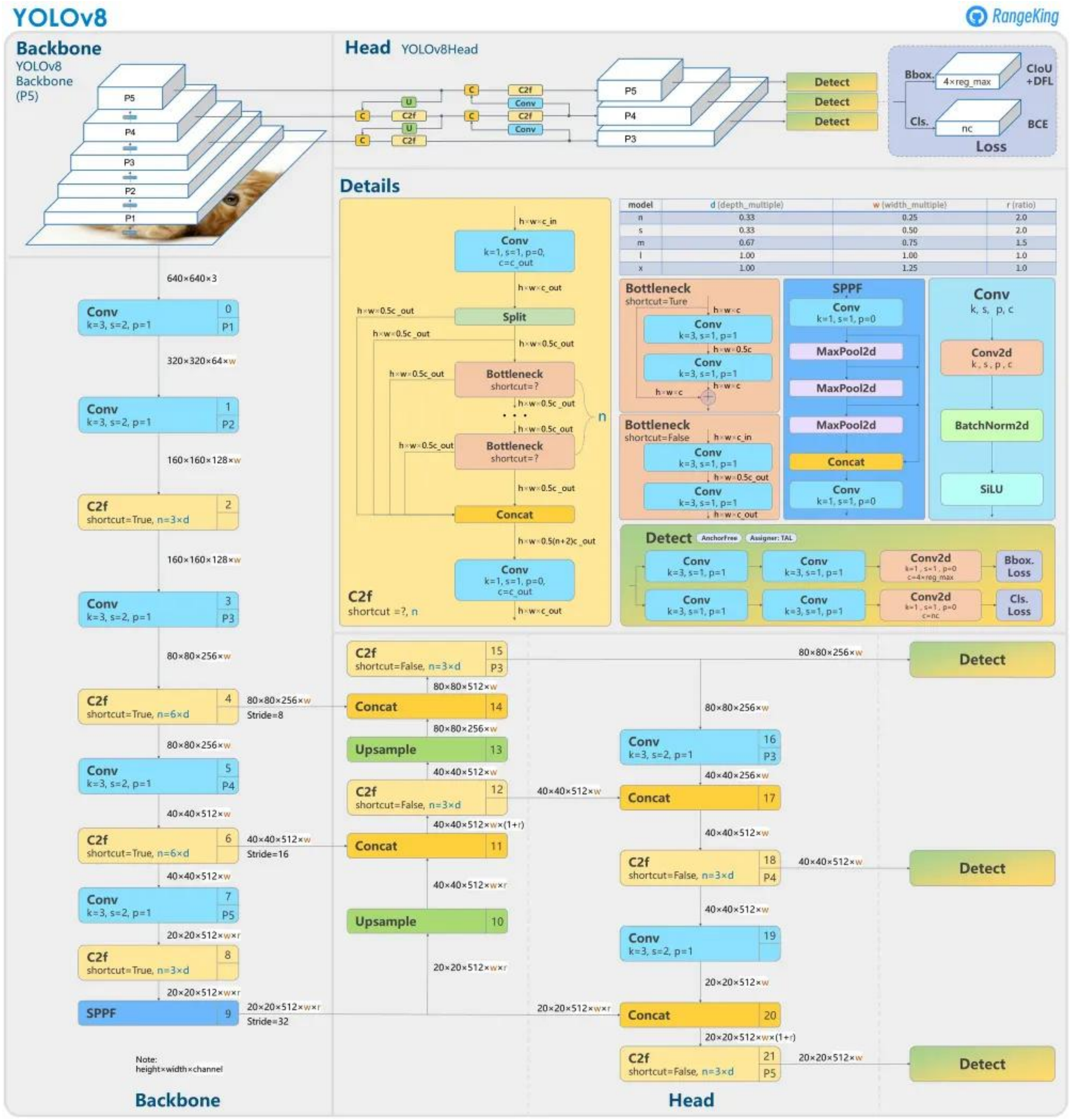


Figure 40 YOLOv8 Architecture[22]

Training

The dataset was pre-partitioned into 3 126 for training, 277 for validation, and 277 for test images. Training proceeded for 20 epochs using stochastic gradient descent (initial learning rate = 0.01; momentum = 0.937; weight decay = 0.0005) with a binary cross-entropy focal loss, and a confidence threshold of 0.2. Automatic mixed precision (AMP) was enabled to accelerate

[22] https://blog.roboflow.com/what-is-yolov8/

convergence and reduce memory footprint. To guard against overfitting and improve generalization to varied pavement textures and lighting, the standard YOLOv8 augmentation pipeline (mosaic, mix-up, random affine, HSV jitter) was applied on-the-fly. A batch size of 8 was used at 640 × 640 resolution, with data prefetching to maximize GPU utilization on Jetson hardware. Training and validation box/cls/DFL losses along with precision, recall, and mAP@0.5 metrics are plotted in Figure 41, showing stable convergence by epoch 15 and minimal overfitting through epoch 20.

Testing

The pothole detector was evaluated on the held-out test split of 277 images. At the conclusion of training (epoch 20), the model achieved a precision of 0.63, recall of 0.50, mAP@0.5 of 0.53, and mAP@0.5–0.95 of 0.23.

In the Figure 42 is illustrated a selection of validation images with predicted bounding boxes and confidence scores overlaid, demonstrating robust localization of larger pothole instances under varied lighting and pavement textures.

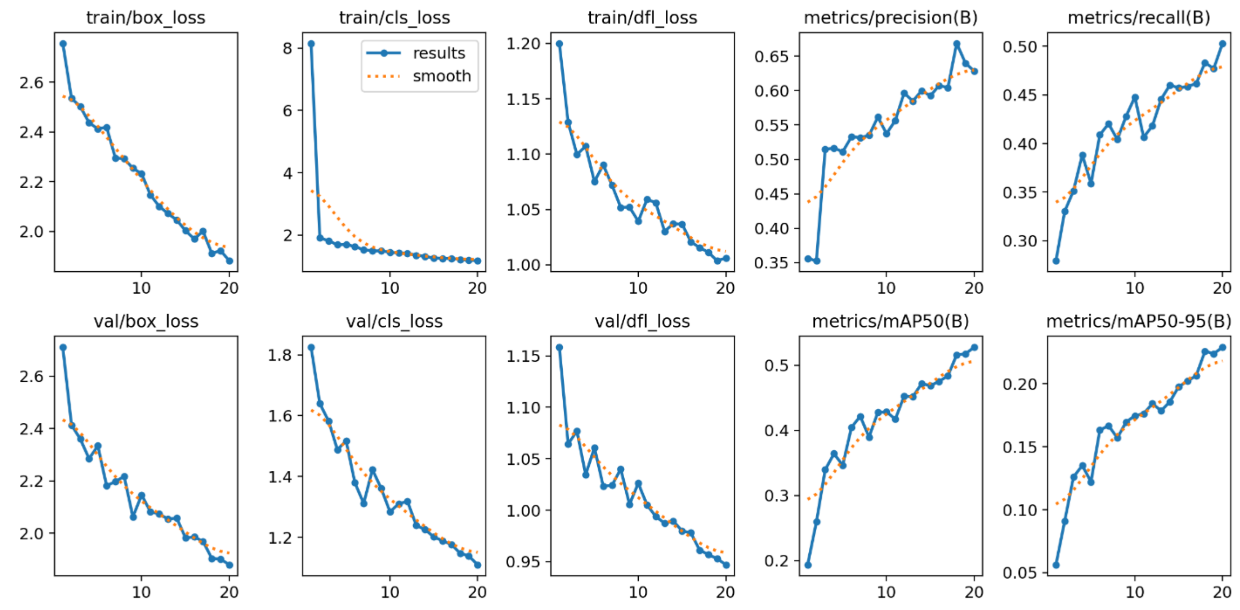


Figure 41: Training and validation performance metrics demonstrating stable convergence and minimal overfitting of the YOLOv8 pothole detection model.

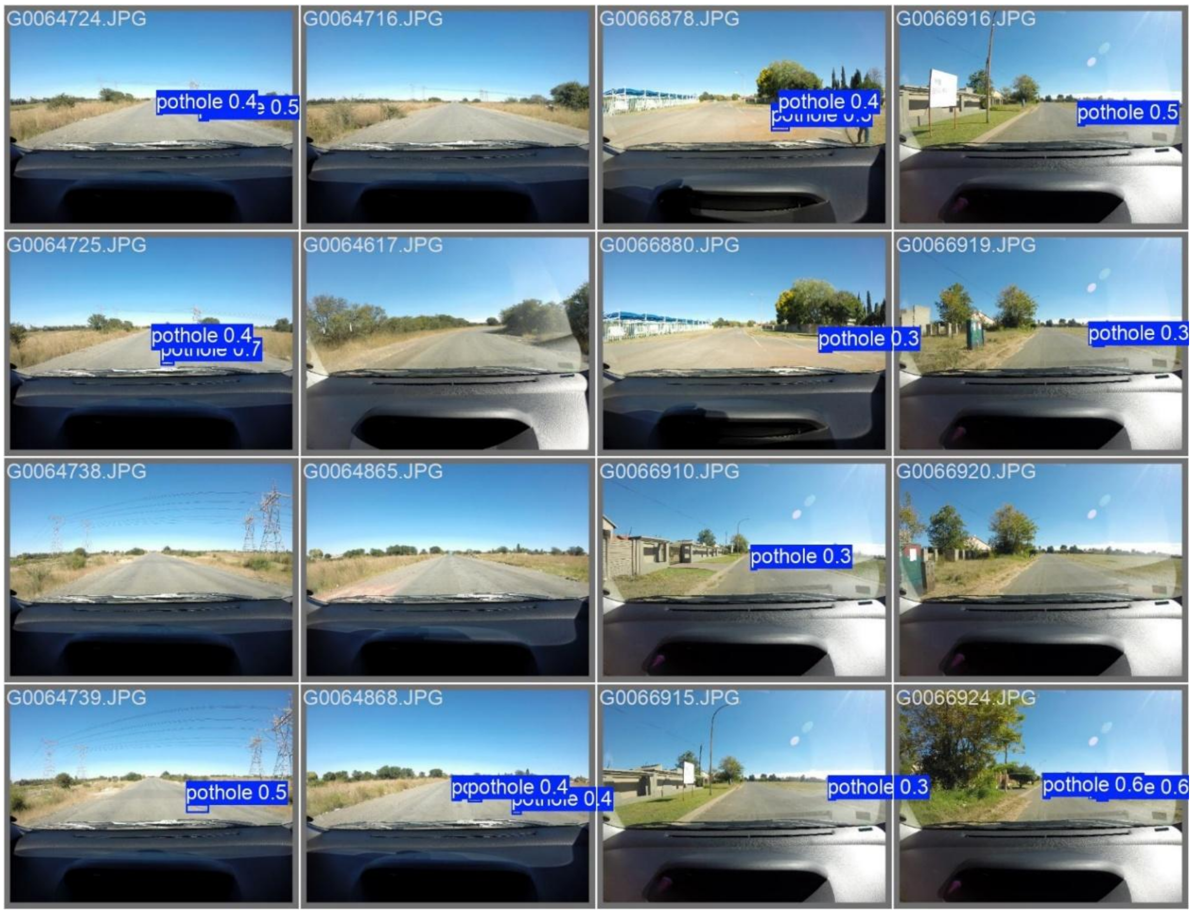


Figure 42: Sample validation images with predicted bounding boxes and confidence scores.

The confusion matrix in Figure 43 summarizes true/false positives and negatives across the “pothole” vs. “background” classes. Of 1 306 ground-truth pothole annotations, the model correctly detected 562 (true positives) but missed 744 (false negatives), yielding the moderate recall observed. It also generated 177 false positives on background regions, contributing to the measured precision.

Overall, the detector reliably identifies prominent potholes but struggles with smaller or low-contrast defects. Future work will explore targeted augmentation (e.g., scale jittering, contrast variations) and additional hard-negative mining to boost recall while maintaining precision.

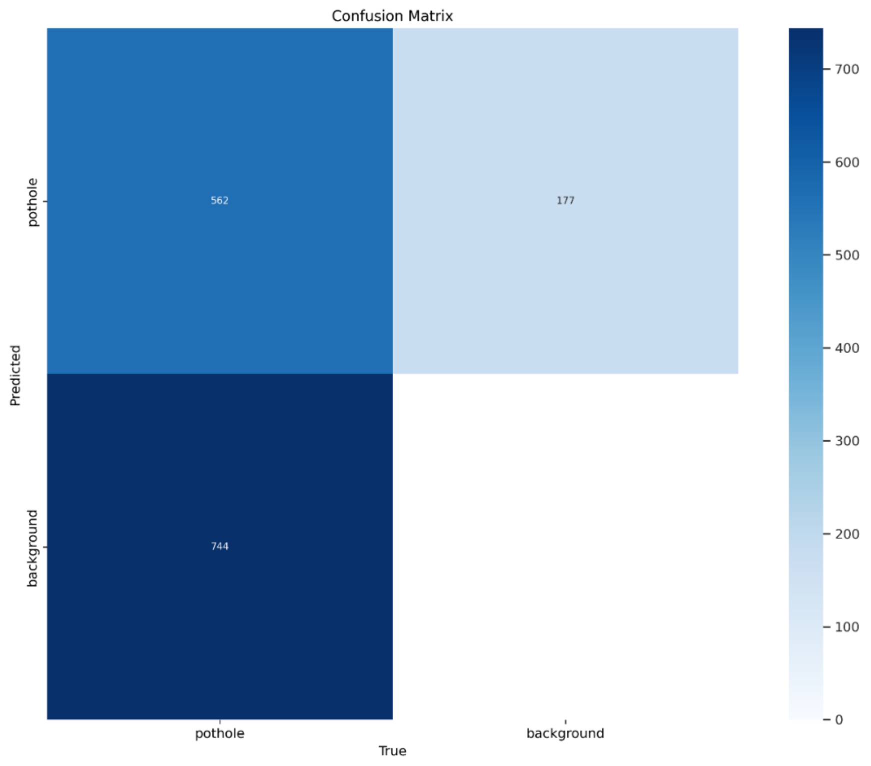


Figure 43: Confusion matrix for "pothole" vs. "background" classes.

Deployment

The road-condition assessment pipeline executes on the Jetson Orin AGX via ONNX Runtime with CUDA support. Each frame from the forward-facing camera is sent through the ONNX-format classifier, which produces a class prediction and confidence score in real time. These results are written to the shared-memory fusion layer and simultaneously passed to the onboard LLM (quantized LLaMA 3–8B). This enables the system to reason for detected road conditions in context for example, triggering driver alerts or issuing corrective guidance whenever a hazard is recognized.

3.1.2. Traffic Sign Detection

The traffic sign detection module continuously monitors the external environment, by using an internal outward camera in front of the driver to detect traffic signs on the road.

Dataset Description

The Traffic Signs Detection dataset contains real-world road scenes annotated with bounding boxes for 15 sign classes (Green Light, Red Light, Stop, and Speed-Limit signs from 10 km h^{-1} to 120 km h^{-1}). The public release already follows a three-way split, totaling **4969 JPEG images** (Table 10):

Table 10: Traffic Signs Detection dataset

Subset	Source folder	Images	Annotation files
Train	train/images	3471	train/labels/*.txt
Valid	valid/images	998	valid/labels/*.txt
Test	test/images	500	test/labels/*.txt

- Image format: JPEG, variable resolutions
- Classes: 15 (see list above)
- Annotation format: YOLO text lines class x_center y_center width height (values normalized 0-1)
- Splits: organized as train/, valid/, test/, each with images/ and labels/ sub-folders
- Preparation: the accompanying Python script (prepare_traffic_signs.py) downloads the Kaggle archive, recreates the folder layout and edits data.yaml so that Ultralytics YOLO recognizes the three splits.

Model Architecture

The detector core is once again Ultralytics YOLOv8-s. We export the model from PyTorch to ONNX (FP32). At inference, the network produces bounding-box coordinates, objectness scores and class logits for the five target object categories.

Training

The predefined splits (3 471 / 998 / 500) were retained. Training ran for **20 epochs** with stochastic gradient descent (initial LR = 0.01, momentum = 0.937, weight-decay = 5×10^{-4}). A *binary-cross-entropy focal loss* handled the multi-class logits, and detections below a confidence of 0.25 were ignored during loss computation. Automatic mixed precision (AMP) reduced memory footprint.

To improve robustness to scale and illumination, the standard YOLOv8 augmentation stack (mosaic, mix-up, random affine, HSV jitter) was supplemented with random motion-blur and Gaussian-noise layers—crucial for the small, high-contrast glyphs of speed-limit signs. Training used a batch size of 16 at 640 × 640 resolution with NVIDIA DALI pre-fetching on Jetson hardware. The evolution of box/cls/DFL losses plus precision, recall and mAP@0.5 metrics is plotted in Figure 45, showing convergence by epoch 32 and negligible over-fitting thereafter.

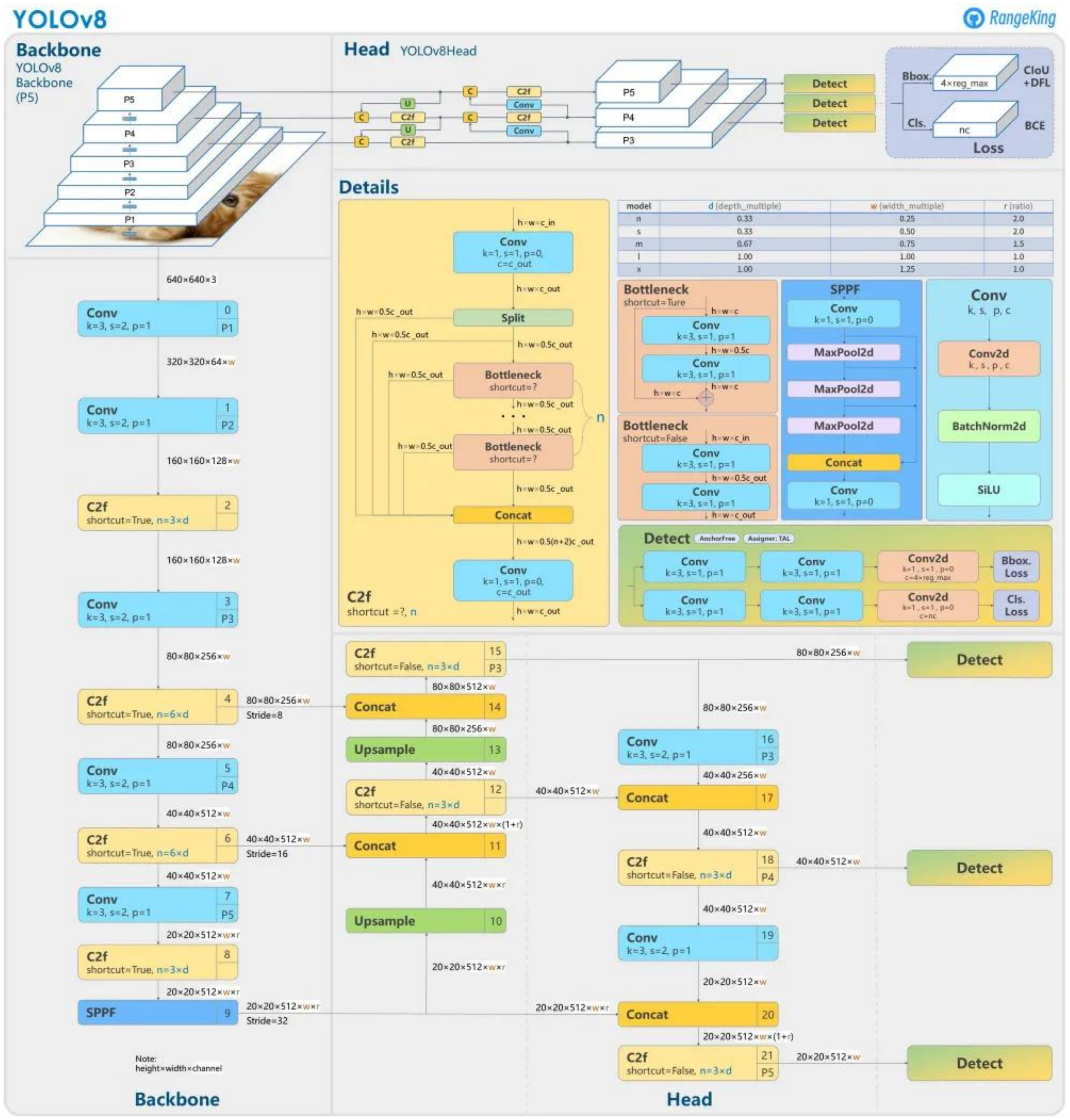


Figure 44 YOLOv8 Architecture[23]

Testing

The traffic sign detector was evaluated on the held-out test split of 500 images. At the conclusion of training, the model achieved a precision of 0.81, recall of 0.74, mAP@0.5 of 0.79, and mAP@0.5–0.95 of 0.46.

[23] https://blog.roboflow.com/what-is-yolov8/

In Figure 46 is illustrated a selection of validation images with predicted bounding boxes and confidence scores overlaid:

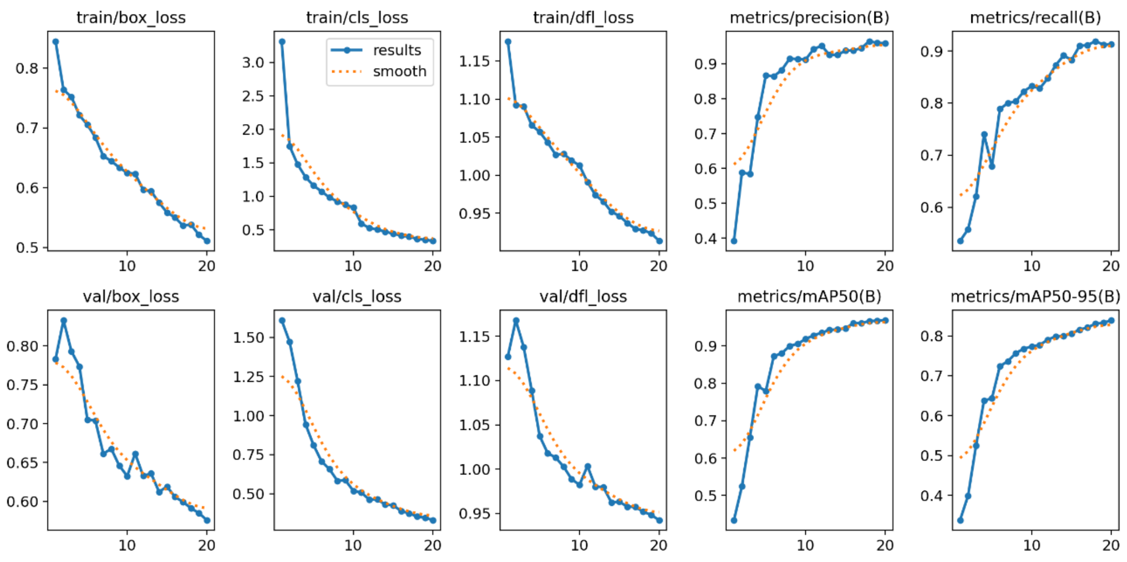


Figure 45: Training and validation performance of the model architecture.

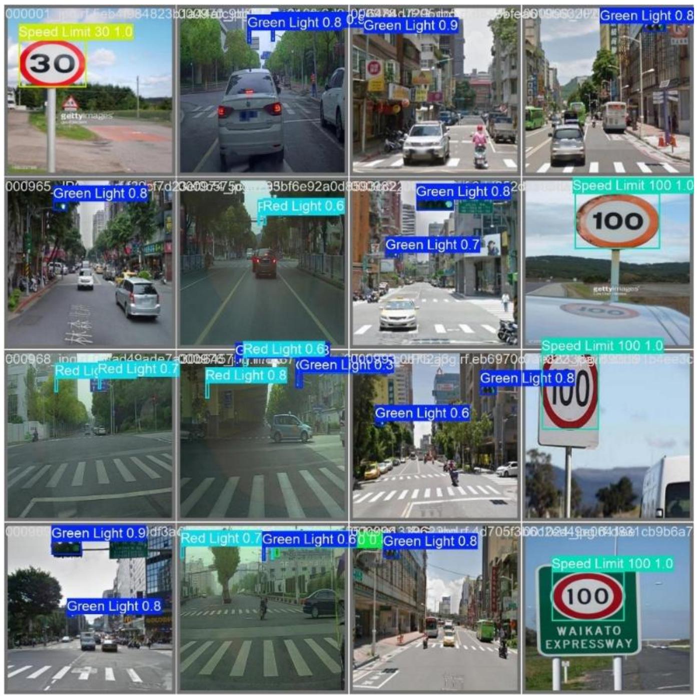


Figure 46: Validation images with predicted bounding boxes and confidence scores.

The confusion matrix in Figure 47 highlights occasional misclassifications between adjacent speed-limits (e.g., 70 ↔ 80 km h^{-1}) due to nearly identical glyphs; Stop and light classes remain highly separable.

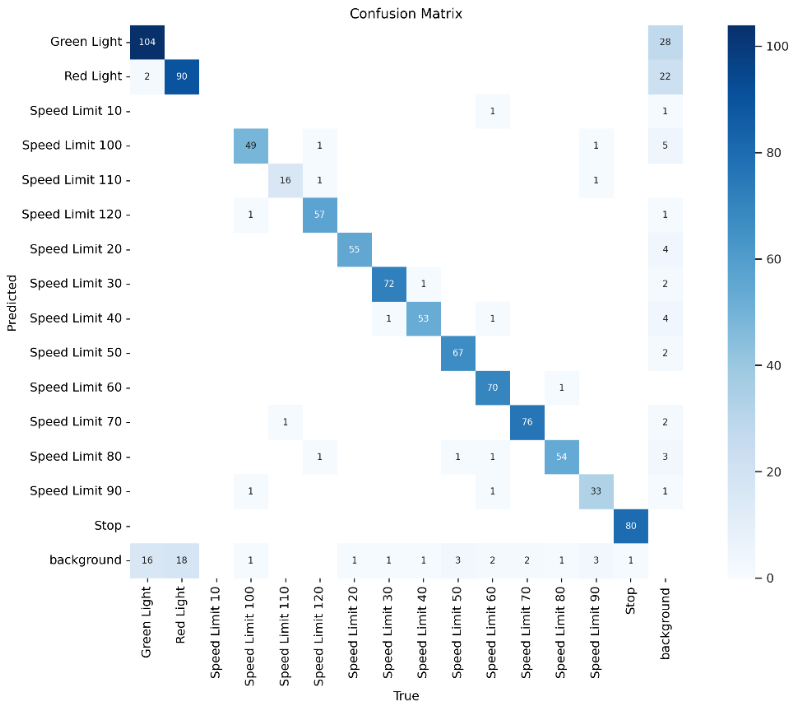


Figure 47: Confusion Matrix highlighting occasional misclassifications.

Overall, the detector delivers robust multi-class performance but would benefit from additional fine-grained shape or font augmentations to further disambiguate neighboring speed-limit signs.

Deployment

The traffic-sign pipeline executes on the **Jetson Orin AGX** via **ONNX Runtime + CUDA**. Each camera frame is resized, forwarded through the ONNX graph (~6 ms latency at 60 FPS) and filtered at 0.25 confidence. Detections are written to the shared-memory fusion layer and ingested by the onboard **LLaMA 3-8B** LLM, enabling context-aware actions—for instance, emitting a speed-limit-exceeded alert or reminding the driver to stop at an upcoming sign.

Future work will explore class-balanced focal loss to further curb speed-limit confusions, and lens-distortion augmentation for wide-angle dashboards.

3.2. Multi-agent cooperative situational awareness and scene analysis

3.2.1. Realize cooperative situational awareness using standard V2X messages format

This subsection describes the list of relevant standardized vehicle to everything (V2X) messages to test and validate cooperative situational awareness scenarios, that is applicable in US/North America and European. It also describes a simulation-based test and validation method for cooperative situational awareness scenarios using V2X messages.

Standardized V2X messages based on SAE J2735 Standards

SAE J2735 is standard that defines V2X messages sets to support interoperability among V2X applications for cooperative connected automated driving system [24] and it is mostly used in North America and Asian countries. It includes various V2X messages to exchange road safety information between vehicles (V2V), vehicle and infrastructure (V2I), and vehicle and pedestrian (V2P).

The basic safety message (BSM) is used in a variety of vehicle safety applications to exchange vehicle safety data. It periodically broadcast the status information such as vehicle's location, speed, acceleration, direction, brake status, etc. so that surrounding vehicles and Vulnerable Road Users (VRU), such as pedestrians, cyclists, or road workers, can understand the situational awareness in real time, and use it to prevent potential collisions for improving driving safety. This message is mainly divided into two parts; while part 1 includes essential data including basic vehicle status information, part 2 contains auxiliary data necessary to perform the service.

The Personal Safety Message (PSM) is used to broadcast safety data regarding the kinematic state of various types of VRU, such as pedestrians, cyclists, or road workers. It includes safety data of VRUs such as location, moving speed, direction, path prediction, etc to allow surrounding road users to recognize them and reduce the risk of collision.

The signal phase and timing (SPAT) message is used to convey the current status of one or more signalized intersections, so that cooperative connected automated driving system or human driver can recognize current movement status of each active phase information in advance when approaching an intersection and drive safely. This message includes information on the current status (i.e. red, green, etc.) and expected changing time (remaining time, etc.) of traffic lights at one or more the intersection. Especially, along with the MAP message, that describes a

[24] https://www.arc-it.net/html/standards/standard17.html

geometric layout of an intersection, the road user of this message can determine the state of the signal phasing and when the next expected phase will occur.

MAP message conveys geographical road information of intersections and road structures. It includes lane positions, directions, connection relationships, boundary information, etc., allowing vehicles to accurately recognize road structures.

RoadSideAlert (RSA) is used to broadcast alerts for nearby hazards to surrounding road users. This message provides simple alerts to road users including mobile hazards (i.e., construction zones, and roadside events are expected to be most frequently used.

Standardized V2X messages based on ETSI ITS Standards

ETSI ITS Message, a European standard, is a V2X message standard for V2V and V2I communication defined by the European Telecommunications Standards Institute (ETSI). It includes cooperative situational awareness information such as real-time location, speed, and hazardous situations those are key components for implementing C-ITS in Europe.

The Cooperative Awareness Message (CAM) is a safety message that periodically transmits current status information of vehicle such as the location, speed, direction, and size to help surrounding vehicles and infrastructure recognize the current status of the vehicle in real time [25].

Decentralized Environmental Notification Message (DENM) is an event-based warning message. It is used to broadcast emergency notifications to surrounding vehicles when abnormal or dangerous situations such as accidents, sudden braking, and obstacles on the road occur [26].

VRU Awareness Messages is the safety message to enable communication between cooperative automated driving systems and VRUs, such as pedestrians and cyclists. It is used to improve road safety by alerting driver/or cooperative automated driving system to periodically broadcast the presence and location of VRUs, allowing them to take necessary precautions [27].

Car2X based test and validation of cooperative situational awareness scenarios

Vector's Car2x is a simulation tool that allows to exchange of status information with vehicles or infrastructure based on standardized V2X messages [28]. It also supports data exchange with

[25] https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.02_60/en_30263702v010302p.pdf
[26] https://www.etsi.org/deliver/etsi_en/302600_302699/30263703/01.03.01_60/en_30263703v010301p.pdf
[27] https://www.etsi.org/deliver/etsi_ts/103300_103399/10330003/02.01.01_60/ts_10330003v020101p.pdf
[28] https://www.vector.com/gb/en/products/products-a-z/software/canoe/option-car2x/

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI
KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

PC5 based V2X communication modules and CAN communication module through the Vector's VN4610 device that is linked with the Car2x via the IEEE 802.11p or 3GPP C-V2X PC5 as well as CAN. It allows to generate risk scenarios between road users based on standardized V2X messages including J2735 and ETSI ITS standards and to verify cooperative situational awareness scenarios in vehicle in the loop (VIL). Additionally, the Morai Sim that is the digital twin-based cooperative automated driving simulator supports VIL simulation by linked with Car2X via UDP communication to display all status information of virtual and real objects as depicted in Figure 48.

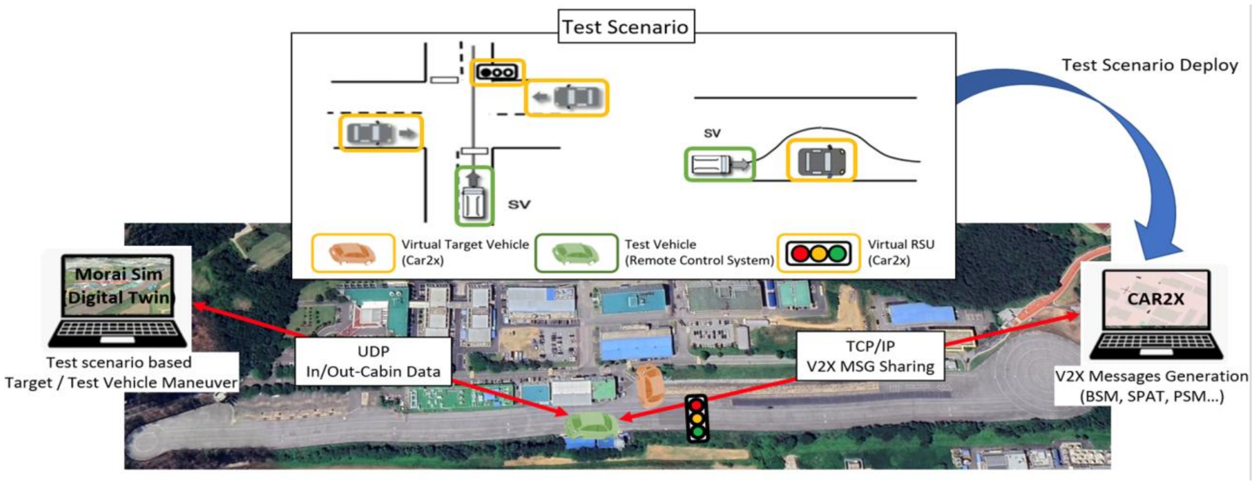

Test Scenario
Test Scenario Deploy
SV
SV
Virtual Target Vehicle (Car2x)
Test Vehicle (Remote Control System)
Virtual RSU (Car2x)
Morai Sim Digital Twin
Test scenario based Target / Test Vehicle Maneuver
UDP In/Out-Cabin Data
TCP/IP V2X MSG Sharing
CAR2X
V2X Messages Generation (BSM, SPAT, PSM...)

Figure 48: VIL simulation in Morai Sim showing status information of virtual and real objects.

The Car2x as an VIL simulation mode with cooperative situational awareness test scenarios, generates relevant V2X messages corresponding to various hazard or dangerous accident situations among virtual objects in virtual environment such as vehicles, VRUs, and traffic lights. After that the Car2X exchanges them with the real test vehicle in real test environment in real time so that test vehicle can recognize its virtual surroundings to test and validate cooperative situational awareness functions as shown in Figure 49.

At the same time, status information of real test vehicle such as position, speed, heading angle, etc. is shared with the Car2X simulator via CAN bus. Therefore, the Car2X can monitor and report all V2X messages of virtual object as well as status information of the test vehicle in real time through the MAP window of the Car2X for analysing developed cooperative situational awareness functions as depicted in Figure 50.

In the following two Sections, the functionality of V2X messages for situational awareness of AVs will be exploited in order to realize novel cooperative localization (CL) and multi-object detection and tracking (MOT) approaches, which rely on the exchange of relevant information and measurements among the involved traffic agents.

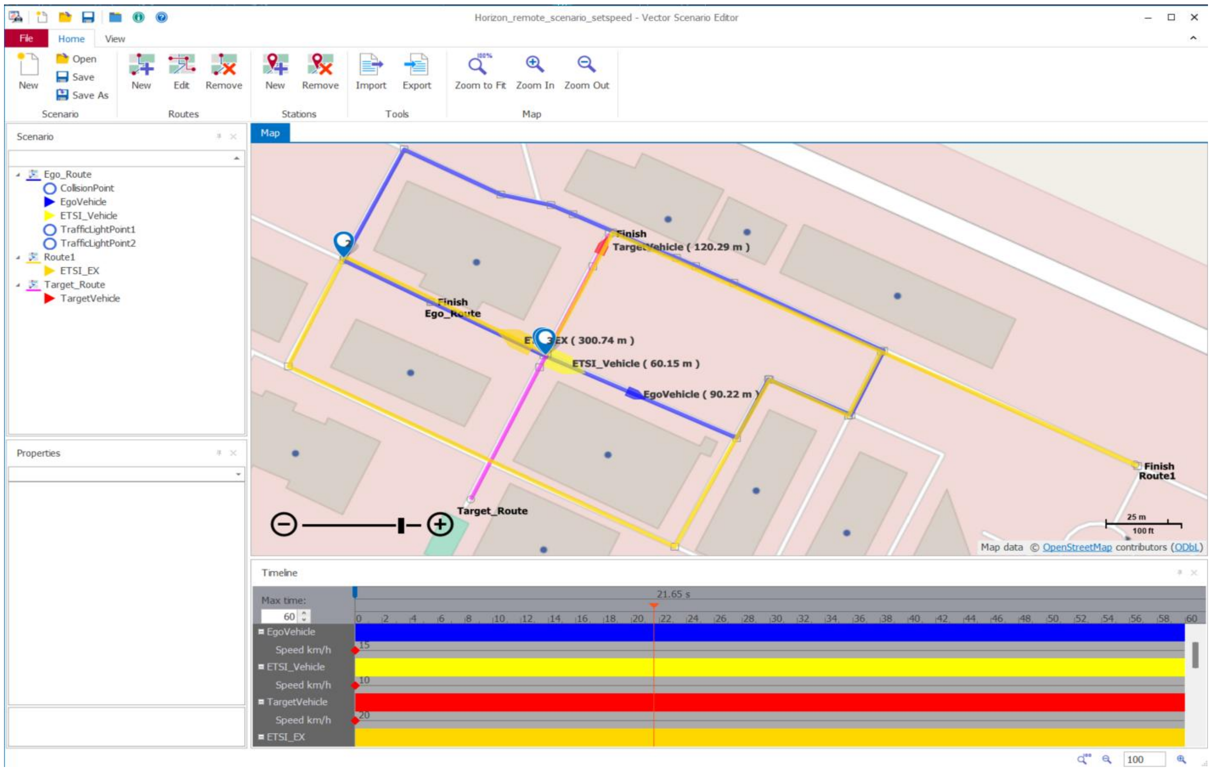


Figure 49: Cooperatives situational awareness test scenario generation by Car2X.

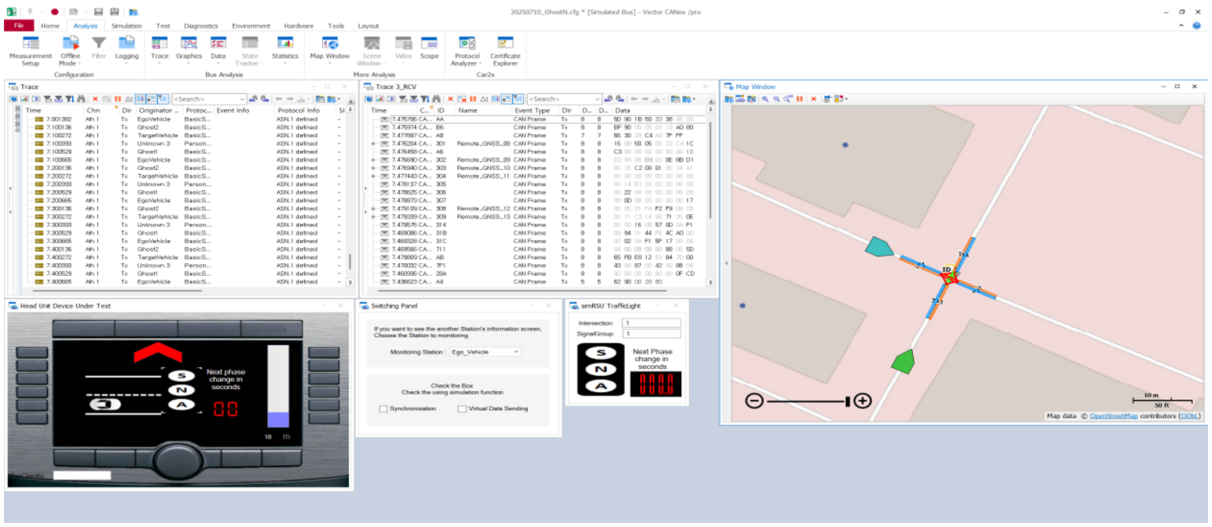

20250710_GhostN.cfg * [Simulated Bus] - Vector CANoe /pro
File
Home
Analysis
Simulation
Test
Diagnostics
Environment
Hardware
Tools
Layout
Trace
Trace 3_RCV
Map Window
Head Unit Device Under Text
Next phase change in seconds
Switching Panel
If you want to see the another Station's information screen, Choose the Station to monitoring
Monitoring Station
Ego_Vehicle
Check the Box
Check the using simulation function
Synchronization
Virtual Data Sending
simRSU TrafficLight
Intersection
SignalGroup
Next Phase change in seconds
Map data © OpenStreetMap contributors (ODbL)

Figure 50: Analysing cooperative situational awareness functions by Car2X.

3.2.2. Cooperative localization via joint distributed learning and decision-making

Preliminaries

Consider a swarm of moving connected and autonomous vehicles (CAVs) at time instant t (Figure 51-(a)), consisting of an (arbitrary) ego vehicle i, its direct neighbors j and non-neighbors

n. In practice, we will exploit a star graph topology $\mathcal{N}_{\mathrm{i}}^{(\mathrm{t})}$, with i at its center, and neighboring vehicles $\mathrm{j} \in \mathcal{N}_{\mathrm{i}}^{(\mathrm{t})}$ as leaves (Figure 51-(b)). Note that: i) each CAV possesses sensing, computation and computation capabilities, ii) i and j are considered neighbors if their communication and sensing range is below a threshold (i.e., 30m), and iii) the swarm might change dynamically over time, i.e., neighbors of ego vehicle i doesn't remain fix. The state of ego vehicle, as well as the others, is represented by its 3D position $\mathbf{x}_{\mathbf{i}}^{(\mathbf{t})} = [\mathrm{x}_{\mathrm{i}}^{\mathrm{t}} \ \mathrm{y}_{\mathrm{i}}^{\mathrm{t}} \ \mathrm{z}_{\mathrm{i}}^{\mathrm{t}}]^{\mathrm{T}} \in \mathbb{R}^3$. As mentioned earlier, vehicles have sensing capabilities to sense ego-motion information, as well as extract relative measurements towards their neighbors. For instance, using IMU and GNSS, we can define the motion and self positioning models, which are also degraded by additive zero-mean Gaussian noise [20]:

- Motion model:

$$\boldsymbol{x}_{\boldsymbol{i}}^{(\boldsymbol{t})} = f\left(\boldsymbol{x}_{\boldsymbol{i}}^{(\boldsymbol{t-1})}, \boldsymbol{u}_{\boldsymbol{i}}^{(\boldsymbol{t})}\right) + \boldsymbol{e}_{\boldsymbol{i}}^{(\boldsymbol{t})} \qquad \text{Eq. 4}$$

where $\boldsymbol{e}_{\boldsymbol{i}}^{(\boldsymbol{t})} \sim \mathcal{G}(0, \sqrt{\boldsymbol{R}_{\boldsymbol{i}}^{(\boldsymbol{t})}})$. Function $f(\cdot)$ takes the form of the constant velocity motion model [23], due to its simplicity: $f = \boldsymbol{A}\boldsymbol{x}_{\boldsymbol{i}}^{(\boldsymbol{t-1})} + \boldsymbol{B}\boldsymbol{u}_{\boldsymbol{i}}^{(\boldsymbol{t})}$, where $\boldsymbol{A} = \mathbb{I}_3$ and $\boldsymbol{B} = diag(dt, dt, dt)$. Control input vector $\boldsymbol{u}_{\boldsymbol{i}}^{(\boldsymbol{t})} = \left[u_i^{(x,t)} \ u_i^{(y,t)} \ u_i^{(z,t)}\right]^T \in \mathbb{R}^3$ comprises 3D velocity as measured by IMU sensor.

- Self-positioning measurement model:

$$\tilde{\boldsymbol{z}}_{\boldsymbol{p},\boldsymbol{i}}^{(\boldsymbol{t})} = \boldsymbol{x}_{\boldsymbol{i}}^{(\boldsymbol{t})} + \boldsymbol{n}_{\boldsymbol{p}}^{(\boldsymbol{t})}, \qquad \boldsymbol{n}_{\boldsymbol{p}}^{(\boldsymbol{t})} \sim \mathcal{G}(0, \sqrt{\boldsymbol{\Sigma}_{\boldsymbol{p}}}) \qquad \text{Eq. 5}$$

Moreover, the visual sensors of camera, LiDAR and RADAR can detect and identify the bounding box of each nearby vehicle j, and extract relative observations with respect to the 3D centroid:

- Relative measurement model:

$$\tilde{z}_{o,ij}^{(t)} = h(\boldsymbol{x}_{\boldsymbol{i}}^{(\boldsymbol{t})}, \boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{t})}) + n_o, \qquad n_o \sim \mathcal{G}(0, \sigma_o) \qquad \text{Eq. 6}$$

$$h(\cdot) = \begin{cases} ||\boldsymbol{x}_{\boldsymbol{i}}^{(\boldsymbol{t})} - \boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{t})}||, & o = d \\ arctan2(y_j^{(t)} - y_i^{(t)}, \quad x_j^{(t)} - x_i^{(t)}), & o = az \\ \arccos \dfrac{z_j^{(t)} - z_i^{(t)}}{||\boldsymbol{x}_{\boldsymbol{i}}^{(\boldsymbol{t})} - \boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{t})}||}, & o = in \end{cases} \qquad \text{Eq. 7}$$

Deterministic function $h(\cdot)$ actually models the relative distance, azimuth and inclination angles ($o = \{d, az, in\}$) between i and j, as measured by i, encoding the 3D positions of the corresponding pair.

In addition to the measurement models, KalmaNet [21] has the potential to enhance ego vehicle localization by modelling system dynamics, as well as state and measurement covariance matrices through an RNN based deep learning architecture. To do that, it builds upon traditional KF to design a data driven approach that will enable the estimation of state $\widetilde{\boldsymbol{x}}_i^{(t)} \epsilon\, \mathbb{R}^3$ and its covariance $\widetilde{\boldsymbol{\Sigma}}_i^{(t)} \epsilon\, \mathbb{R}^{3x3}$. More specifically, KalmaNet's architecture aims to estimate the uncertainty matrices of KF algorithm. These include Kalman gain matrix $\boldsymbol{K}_i^{(t)} \epsilon\, \mathbb{R}^{3x3}$, state transition covariance matrix $\boldsymbol{R}_i^{(t)} \epsilon\, \mathbb{R}^{3x3}$, predicted state covariance matrix $\overline{\boldsymbol{\Sigma}}_i^{(t)} \epsilon\, \mathbb{R}^{3x3}$, and matrix $\boldsymbol{W}_i^{(t)} \epsilon\, \mathbb{R}^{3x3}$. The latter is defined as $\boldsymbol{W}_i^{(t)} = \boldsymbol{H}_i^{(t)}\, \overline{\boldsymbol{\Sigma}}_i^{(t)} \boldsymbol{H}_i^{(t)^{T}} + \boldsymbol{Q}_i^{(t)}$, where $\boldsymbol{H}_i^{(t)}$ corresponds to the jacobian matrix of a generic measurement function with respect to predicted $\overline{\mathbf{x}}_i^{(t)}$ (by motion model Eq. 4). When only GNSS measurement is available, measurement vector $\widetilde{\boldsymbol{z}}_i^{(t)} = \widetilde{\boldsymbol{z}}_{p,i}^{(t)}$ and $\boldsymbol{H}_i^{(t)} = \mathbb{I}_3$. Measurement covariance matrix is denoted by $\boldsymbol{Q}_i^{(t)} \epsilon\, \mathbb{R}^{3x3}$. Using these estimated matrices, KalmanNet computes the updated state estimate $\widetilde{\boldsymbol{x}}_i^{(t)}$ and its covariance $\widetilde{\boldsymbol{\Sigma}}_i^{(t)}$ following the standard KF equations. The network takes as input the current and previous measurement vectors $\widetilde{\boldsymbol{z}}_i^{(t)}$ and $\widetilde{\boldsymbol{z}}_i^{(t-1)}$, the previous state estimates $\widetilde{\boldsymbol{x}}_i^{(t-1)}$, $\overline{\mathbf{x}}_i^{(t-1)}$, and $\overline{\mathbf{x}}_i^{(t-2)}$, the control input vector $\boldsymbol{u}_i^{(t)}$, and the time interval dt. Therefore, the equations of KF can be reformulated as:

$$\boldsymbol{K}_i^{(t)}, \boldsymbol{R}_i^{(t)}, \overline{\boldsymbol{\Sigma}}_i^{(t)}, \mathbf{W}_i^{(t)} = KalmanNet_{\theta_i}(\cdot) \qquad \text{Eq. 8}$$

$$\widetilde{\boldsymbol{x}}_i^{(t)} = \overline{\mathbf{x}}_i^{(t)} + \boldsymbol{K}_i^{(t)}(\, \widetilde{\boldsymbol{z}}_i^{(t)} - \overline{\mathbf{x}}_i^{(t)}) \qquad \text{Eq. 9}$$

$$\widetilde{\boldsymbol{\Sigma}}_i^{(t)} = \overline{\boldsymbol{\Sigma}}_i^{(t)} - \overline{\boldsymbol{\Sigma}}_i^{(t)} \boldsymbol{K}_i^{(t)} \qquad \text{Eq. 10}$$

The network's architecture consists of three gated recurrent units (GRUs) interconnected with fully connected layers, that allow in a cascaded manner to capture $\boldsymbol{R}_i^{(t)}$, $\overline{\boldsymbol{\Sigma}}_i^{(t)}$ and $\mathbf{W}_i^{(t)}$. Apparently, the learned $\boldsymbol{R}_i^{(t)}$ is used to capture $\overline{\boldsymbol{\Sigma}}_i^{(t)}$, which in turn obtains $\mathbf{W}_i^{(t)}$, while both $\overline{\boldsymbol{\Sigma}}_i^{(t)}$ and $\mathbf{W}_i^{(t)}$ are used to produce $\mathbf{K}_i^{(t)}$. The latter, being equal to $\mathbf{K}_i^{(t)} = \overline{\boldsymbol{\Sigma}}_i^{(t)}\, \mathbf{H}_i^{(t)^{\mathrm{T}}} \left(\boldsymbol{W}_i^{(t)}\right)^{-1}$, actually summarizes in a compact manner the underlying uncertainty of the system. Therefore, by exchanging and aggregating only the specific weights that are used to compute Kalman gain matrix, instead of the whole number of network's parameters, each vehicle will be capable of efficiently train its local model to learn Kalman gain in a distributed manner. Thus, based on measurement models Eq. 4-Eq. 6, as well as KalmaNet's architecture and Kalman gain properties, we will formulate in the following the proposed joint distributed learning and decision-making scheme, that will enable ego vehicle i to estimate its own as well as its neighbors' 3D positions, in the form of state vector $\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t})} \epsilon\, \mathbb{R}^{3|\mathcal{N}_i^{(t)}|}$, realizing the goal of continual cooperative situational awareness.

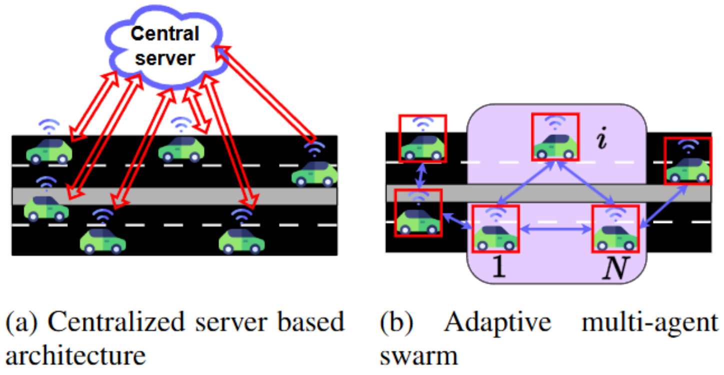

Central server
i
1
N
(a) Centralized server based architecture
(b) Adaptive multi-agent swarm

Figure 51: Distributed optimization in learning and decision-making is realized by adaptive multi-agent swarms, where each agent cooperates with its neighbors to address individual limitations. Note at the right, the star topology that is formulated by the (arbitrary) ego vehicle i, based on communication and sensing range.

Learning stage via federated learning

In this Section, the distributed learning stage of the proposed methodology will be formulated. Instead of a traditional server-based federated learning (FL) process, ego vehicle will now act as the aggregation center enabling a serverless learning approach. Moreover, since Kalman gain captures the underlying uncertainty of the system, only the layers that are used to learn Kalman gain will be transmitted to the ego vehicle, keeping the others intact. In that way, a personalized FL strategy will also be formulated, facilitating a continual and cost-efficient learning process, since transmitted personalized layers consist of a very small number of parameters.

Serverless personalized training

To establish the serverless personalized training, it is assumed that each vehicle belonging to $\mathcal{N}_i^{(t)}$ participates also to the distributed learning process. More specifically, each $j \,\epsilon\, \mathcal{N}_i^{(t)}$, utilizes its local training dataset $\mathcal{D}_j = \{\widetilde{\boldsymbol{Z}}_{\boldsymbol{j}}^{\boldsymbol{1:T_j}}, \boldsymbol{X}_{\boldsymbol{j}}^{\boldsymbol{1:T_j}}\}$, which contains an input trajectory as measured over time by its own sensors (GNSS, IMU, etc.), as well as the corresponding ground truth (or target) trajectory. More specifically, T_j is the length of training trajectories, input $\widetilde{\boldsymbol{Z}}_{\boldsymbol{j}}^{\boldsymbol{1:T_j}} = \left[\tilde{\boldsymbol{z}}_{\boldsymbol{j}}^{(\boldsymbol{1})} \dots \tilde{\boldsymbol{z}}_{\boldsymbol{j}}^{(\boldsymbol{T_j})}\right] \epsilon\, \mathbb{R}^{6xT_j}$ contains the noisy 3D positions and velocities for the corresponding training trajectory, while target $\mathbf{X}_{\mathbf{j}}^{\mathbf{1:T_j}} = \left[\mathbf{x}_{\mathbf{j}}^{(\mathbf{1})} \dots \mathbf{x}_{\mathbf{j}}^{(\mathbf{T_j})}\right] \epsilon\, \mathbb{R}^{3\mathrm{x}\mathrm{T_j}}$ contains the corresponding ground truth 3D trajectory. Note that input $\tilde{\mathbf{Z}}_{\mathbf{j}}^{\mathbf{1:T_j}}$ is derived from measurement models Eq. 4 and Eq. 5.

Each vehicle trains its local KalmaNet model using its own private dataset, following Eq. 8-Eq. 10 and the corresponding KalmaNet architecture from Figure 52. In order to address potential limitations of local datasets regarding different noise statistics (e.g., GNSS operation in urban environments or urban canyons, etc.), an FL strategy that enables the collaboration of vehicles so that can efficiently learn a more robust KalmaNet has been already proposed in [22]. However, it requires a central node and a group of vehicles to be synchronize offline so that the whole process is effectively orchestrated. Based on that limitation, we will derive the much more dynamic approach of PFedKalmaNet, where an ego vehicle will now act as the central node performing models' aggregation, while its connected neighbors transmit only specific layers of the original architecture so as to facilitate the communication exchange. Those personalized layers of original network are used to capture the most important information about the underlying uncertainty, i.e., Kalman gain, and for that task two scenarios will be investigated: i) personalized layer that corresponds to the fully connected layer of Kalman gain $\mathrm{FC_{KG}}$(with pink color), ii) personalized layers that correspond to GRU 1, 2 and 3 (with red color in Figure 52), used for learning matrices $\mathbf{R}_{\mathbf{i}}^{(\mathbf{t})}$, $\bar{\mathbf{\Sigma}}_{\mathbf{i}}^{(\mathbf{t})}$ and $\mathbf{W}_{\mathbf{i}}^{(\mathbf{t})}$. The first scenario focuses on aggregating around 10K or 50% of the total number of parameters by focusing directly to Kalman gain, while in the second one around 5K parameters or 20% of the total number of parameters are to be exchanged and aggregated.

Adaptation step

At the adaptation step, each vehicle j relies on the traditional KF algorithm to learn the Kalman gain $\mathbf{K}_{\mathbf{j}}^{(\mathbf{k})}$ through local KalmanNet $\mathrm{KalmanNet}_{\theta_j}(\cdot)$. Note that learning $\mathbf{K}_{\mathbf{j}}^{(\mathbf{k})}$ inherently incorporates the learning of additional uncertainty matrices (see Eq. 8 and Figure 52). In the proposed framework, the local KalmaNet is trained end-to-end using the local dataset. In more detail, let $\boldsymbol{\theta}_{\mathbf{j}}$ denote the trainable parameters of the local network, and γ_j be a regularization coefficient. Each agent employs an ℓ_2-regularized mean-squared error loss and stochastic gradient descent [21] to optimize its local model, defined as follows:

$$\ell_j(\boldsymbol{\theta}_j) = \frac{1}{T_j}\sum_{k=1}^{T_j}\left|\left|\tilde{\boldsymbol{x}}_j^{(k)}\left(\tilde{\boldsymbol{z}}_j^{(k)};\boldsymbol{\theta}_j\right) - \boldsymbol{x}_j^{(k)}\right|\right|_2^2 + \gamma_j\left|\left|\boldsymbol{\theta}_j\right|\right|_2^2 \qquad \text{Eq. 11}$$

where $\tilde{\boldsymbol{x}}_j^{(k)}\left(\tilde{\boldsymbol{z}}_j^{(k)};\boldsymbol{\theta}_j\right)$is the output state estimation of the local KF parametrized by $\tilde{\boldsymbol{z}}_j^{(k)}$ and $\boldsymbol{\theta}_j$. However, the latter is rather composed of two sets of personalized and non-personalized parameters:

$$\boldsymbol{\theta_j} = \{pers(\boldsymbol{\theta_j}), non - pers(\boldsymbol{\theta_j})\} \quad \text{Eq. 12}$$

where as mentioned before, $pers(\boldsymbol{\theta_j})$ contains the parameters of either FC_{KG} or GRU 1,2 and 3 layers:

$$pers(\boldsymbol{\theta_j}) = \begin{cases} FC_{KG}\ (scenario\ 1) \\ GRU\ 1{,}2 \text{ and } 3 \text{ (scenario 2)} \end{cases} \quad \text{Eq. 13}$$

In practice, Eq. 13 state that end-to-end training is performed in a twofold manner: initially, personalized layers are frozen and only non-personalized layers are updated through Eq. 11, and then, personalized layers are also fine tuned (typically with much less epochs). We adopt this training process, aiming to increase the performance, as well as the ability of the local model to generalize across multiple datasets, by exchanging and aggregating only the personalized layers of each $j \epsilon \mathcal{N}_i^{(t)}$. After all participating vehicles have updated their local KalmaNets using Eq. 11, the combination step is following.

Combination step

The goal of this step is to develop a model that captures underlying system dynamics and accurately estimates uncertainty matrices using local datasets in a communication efficient manner and without the need of exchanging the whole number of network's parameters. As such, each $j \epsilon \mathcal{N}_i^{(t)} \backslash i$, transmits to the ego vehicle their personalized layers $pers(\boldsymbol{\theta_j})$ (from Eq. 13). Ego vehicle, along with its own personalized layers, aggregates the local layers using the averaging fusion rule: $pers(\boldsymbol{\theta_g}) = \frac{1}{|\mathcal{N}_i^{(t)}|} \sum_{\forall j \in \mathcal{N}_i^{(t)}} pers(\boldsymbol{\theta_j})$, where $pers(\boldsymbol{\theta_g})$ denotes the global personalized layers that will be common across the vehicles of the swarm. Ego vehicle broadcasts those parameters to the connected neighbors. Each vehicle then initializes only the personalized layers of the local KalmaNet model with those parameters, keeping the other layers intact. This iterative process is repeated for M communication rounds, enabling the local models to continuously improve through the global personalized layers. This is done in a highly efficient communication manner, since even with a very small number of parameters that are exchanged and fused, local models are capable of achieving comparable performance if the whole network's parameters are exchanged. In the same context, note also that very limited communication rounds are required for attaining high accuracy.

Thus, the serverless PFedKalmanNet-L method is realized by a two step process: *adapt*, i.e., training the local KalmaNet using KF algorithm and the local private dataset, and *combine*, i.e., aggregating the personalized layers of each $j \epsilon \mathcal{N}_i^{(t)}$ at the ego vehicle side, broadcasting the global personalized layers back to the neighbors, and keeping the local non-personalized layers intact:

$$Adapt: \begin{cases} \boldsymbol{K}_j^{(k)} = KalmanNet_{\theta_j}(\cdot) \\ \tilde{\mathbf{x}}_j^{(k)} = \bar{\mathbf{x}}_j^{(k)} + \boldsymbol{K}_j^{(k)}(\tilde{\mathbf{z}}_j^{(k)} - \bar{\mathbf{x}}_j^{(k)}) \\ \boldsymbol{\theta}_j^* = argmin_{\boldsymbol{\theta}_j} \ell_j(\boldsymbol{\theta}_j) \end{cases} \qquad \text{Eq. 14}$$

$$Combine = \begin{cases} pers(\boldsymbol{\theta}_g) = \frac{1}{|\mathcal{N}_i^{(t)}|} \sum_{\forall \in \mathcal{N}_i^{(t)}} pers(\boldsymbol{\theta}_j^*) \\ pers(\boldsymbol{\theta}_j^*) = pers(\boldsymbol{\theta}_g) \end{cases} \qquad \text{Eq. 15}$$

The proposed PFedKalmanNet-L approach is demonstrated in Figure *53*.

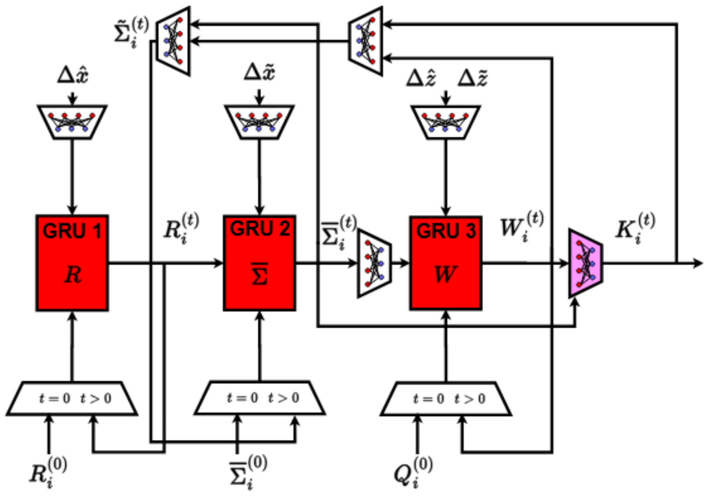


Figure 52: Original KalmaNet's architecture. Proposed SPFL strategy will focus on training in a distributed, serverless and communication efficient manner local KalmaNet's, by transmitting to ego vehicle and aggregating either the fully connected layer FC_{KG} (pink) associated explicitly to Kalman gain or GRU 1, 2 and 3 layers (red).

Decision-making stage via DNN aided Bayesian optimization

In this Section, we will describe how the efficiently trained PFedKalmaNet can be exploited by the second stage of our framework, i.e., distributed decision-making for CL. Note that our PFedKalmaNet takes as inputs the noisy 3D position, 3D velocity and time interval, so as to estimate the state of a single vehicle. As such, instead of a CL approach that addresses states, measurements, and control inputs in a unified and compact manner like [23][24] (the latter used as baseline method in the following exactly due to this feature) we will realize CL by treating target quantities dynamically and independently, via plug-and-play operation.

Ego vehicle localization

Ego vehicle initially exploits self-sensing capabilities to estimate its state, via GNSS and IMU. More specifically, measurement vector $\hat{\boldsymbol{z}}_{\boldsymbol{i\to i}}^{(\boldsymbol{t})} \in \mathbb{R}^3$ of ego vehicle i contains the GNSS position:

$$\tilde{\boldsymbol{z}}_{\boldsymbol{i\to i}}^{(\boldsymbol{t})} = \left[\, \tilde{z}_{p,i}^{(x,t)} \quad \tilde{z}_{p,i}^{(y,t)} \quad \tilde{z}_{p,i}^{(z,t)} \right]^T \qquad \text{Eq. 16}$$

In addition, it utilizes the relative-sensing capabilities of its connected neighbors. To be more detailed, each connected neighbor $j \epsilon \mathcal{N}_i^{(t)}$ transmits to ego vehicle the measurements required for the estimation of i's state from j's perception system. The corresponding measurement vector $\hat{\boldsymbol{z}}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})} \in \mathbb{R}^3$contains:

$$\tilde{\boldsymbol{z}}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})} = \begin{bmatrix} \tilde{z}_{p,j}^{(x,t)} + \tilde{z}_{d,ji}^{(t)}\cos\tilde{z}_{az,ji}^{(t)}\ \sin\tilde{z}_{in,ji}^{(t)} \\ \tilde{z}_{p,j}^{(y,t)} + \tilde{z}_{d,ji}^{(t)} sin\tilde{z}_{az,ji}^{(t)}\ \sin\tilde{z}_{in,ji}^{(t)} \\ \tilde{z}_{p,j}^{(z,t)} + \tilde{z}_{d,ji}^{(t)}\cos\tilde{z}_{in,ji}^{(t)} \end{bmatrix}, \qquad \forall\, j \in \mathcal{N}_i^{(t)} \backslash i \qquad \text{Eq. 17}$$

Note that each row of this vector actually corresponds to the (erroneous) 3D position of ego vehicle i, through simple geometric relationships. Therefore, $\hat{\boldsymbol{z}}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})}$ is an estimate about ego vehicle's state using j's sensing capabilities, i.e., j's GNSS and LiDAR.

For each one of those measurement vectors available to ego vehicle i, authors of [25] exploited a KF, to initially produce local state vectors $\widetilde{\boldsymbol{x}}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})} \in \mathbb{R}^3$ and covariances $\widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})} \in \mathbb{R}^{3x3}$, $\forall\, j \in \mathcal{N}_i^{(t)}$, which are then fused through a global fusion operation to produce the final estimation of $\widetilde{\boldsymbol{x}}_{\boldsymbol{g,i}}^{(\boldsymbol{t})} \in \mathbb{R}^3$ of ego vehicle's state. However, due to the multi-modal observations of diverse noise statistics, we will replace standard KF with the PFedKalmaNet, which has been explicitly trained to capture the uncertainty of measurements and improve the fusion. Through this plug-and-play operation, we will show in the following that increased localization accuracy can be attained. Following the methodology of [25], ATC concept of ego vehicle localization is split between a local and global fusion step:

- *Plug -and- Play Adapt*: ($\forall\, j \in \mathcal{N}_i^{(t)}$)

$$\boldsymbol{K}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})}, \overline{\boldsymbol{\Sigma}}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})} = \text{PFed}KalmanNet_{\theta_i}(\cdot) \qquad \text{Eq. 18}$$

$$\widetilde{\boldsymbol{x}}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})} = \overline{\mathbf{x}}_{\boldsymbol{g,i}}^{(\boldsymbol{t})} + \boldsymbol{K}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})}(\,\tilde{\boldsymbol{z}}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})} - \overline{\mathbf{x}}_{\boldsymbol{g,i}}^{(\boldsymbol{t})}) \qquad \text{Eq. 19}$$

$$\widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})} = \overline{\boldsymbol{\Sigma}}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})} - \overline{\boldsymbol{\Sigma}}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})}\boldsymbol{K}_{\boldsymbol{j\to i}}^{(\boldsymbol{t})} \qquad \text{Eq. 20}$$

- *Combine:*

$$\boldsymbol{A}_j^{(t)} = \widetilde{\boldsymbol{\Sigma}}_{j\to i}^{(t)-1}\left(\sum_{\forall\, l\in \mathcal{N}_i^{(t)}} \widetilde{\boldsymbol{\Sigma}}_{l\to i}^{(t)-1}\right)^{-1} \quad \text{Eq. 21}$$

$$\widetilde{\boldsymbol{x}}_{g,i}^{(t)} = \sum_{\forall\, j\in \mathcal{N}_i^{(t)}} \boldsymbol{A}_j^{(t)}\, \widetilde{\boldsymbol{x}}_{j\to i}^{(t)} \quad \text{Eq. 22}$$

Note that Eq. 18 simply states that we exploit not only the learned Kalman gain, but also the learned predicted state covariance, explicitly utilizing PFedKalmaNet's architecture. Distributed decision-making is realized by ego vehicle since the latter performs both local and global estimation, using its own data as well as the measurements transmitted by its neighbors. Furthermore, $\bar{\mathbf{x}}_{g,i}^{(t)} = f\left(\widetilde{\boldsymbol{x}}_{g,i}^{(t-1)}, \boldsymbol{u}_i^{(t)}\right) + \boldsymbol{e}_i^{(t)}$ (from Eq. 19), which indicates that global estimation depends on the global state vector of time instant $t-1$, rather than specific local estimation corresponding to individual neighbors. This important fact demonstrates once again the ability of our approach in dynamically addressing the swarm topology's modifications over time, i.e., which vehicles exit or enter the swarm, not only at the learning but also at the decision-making stage.

Neighbors localization

The same methodology is followed by ego vehicle i to localize the connected neighbors, and realize effective cooperative situational awareness. As such, for each $\{k \in \mathcal{N}_i^{(t)} \backslash i\}$, ego vehicle utilizes neighbor's GNSS, as well as its own GNSS and LiDAR, to formulate the corresponding measurement vectors $\widetilde{\boldsymbol{z}}_{j\to k}^{(t)} \in \mathbb{R}^3$. Since in practice the perception capabilities of both ego vehicle and its direct neighbor are exploited, index j for this specific formulation belongs to set $\mathcal{N}_{ik}^{(t)}$. The latter set contains only i and k, modeling in fact the individual connected pairs of the original star topology. As such, using the measurement vectors $\widetilde{\boldsymbol{z}}_{i\to k}^{(t)}$ (from i's LiDAR and GNSS) and $\widetilde{\boldsymbol{z}}_{k\to k}^{(t)}$ (from k's transmitted GNSS), where j is equal to either i or k, we end up to a similar ATC approach as previously, for estimating k's position $\widetilde{\boldsymbol{x}}_{g,k}^{(t)} \in \mathbb{R}^3$. Therefore, using PFedKalmaNet as a plug-and-play, i is capable of localizing itself as well as each one of its connected neighbors. Thus, collecting vectors $\widetilde{\boldsymbol{x}}_{g,i}^{(t)}$ and $\widetilde{\boldsymbol{x}}_{g,k}^{(t)}$, i defines the target location vector $\widehat{\boldsymbol{x}}_i^{(t)} \in \mathbb{R}^{3|\mathcal{N}_i^{(t)}|}$ of cooperative situational awareness. The proposed approach of PFedKalmaNet-D is demonstrated in Figure *53*.

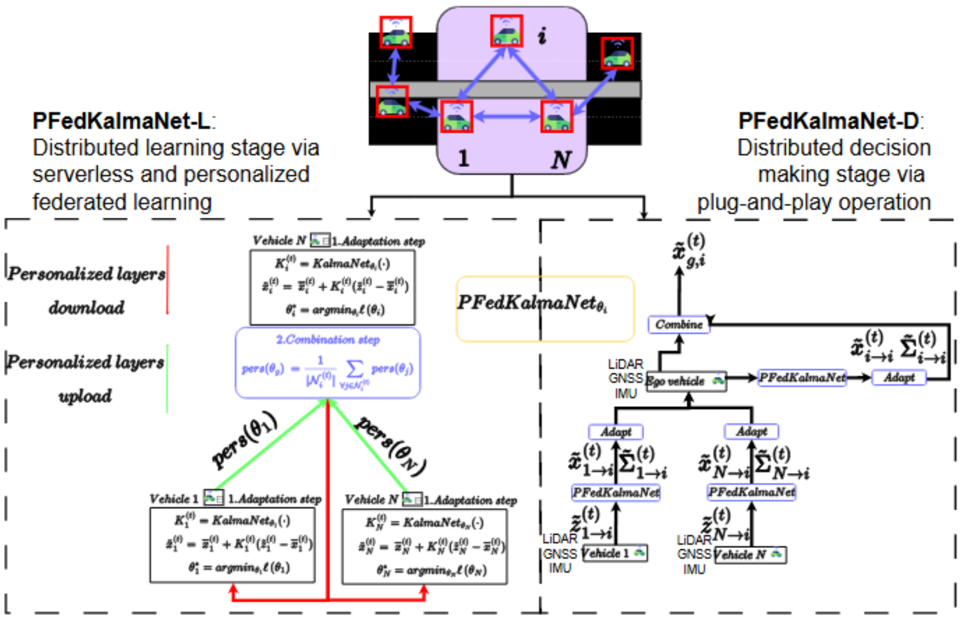
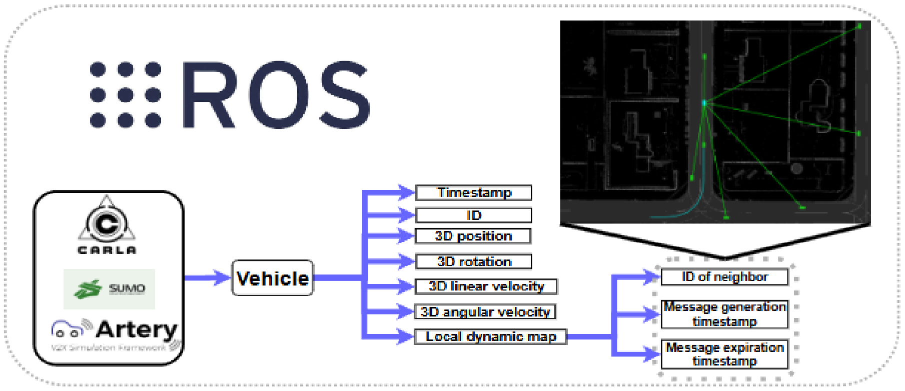

PFedKalmaNet-L:
Distributed learning stage via serverless and personalized federated learning
PFedKalmaNet-D:
Distributed decision making stage via plug-and-play operation
Personalized layers download
Personalized layers upload
1.Adaptation step
2.Combination step
PFedKalmaNet
Combine
Adapt
Ego vehicle
LiDAR
GNSS
IMU

Figure 53: PFedKalmaNet framework in adaptive swarms of CAV. Ego vehicle is acting as the fusion center in both cases, avoiding the need for a centralized server based architecture. Left: Distributed learning stage via SPFL. Right: Distributed trained PFedKalmaNet is then used at distributed decision-making stage for effective cooperative situational awareness.

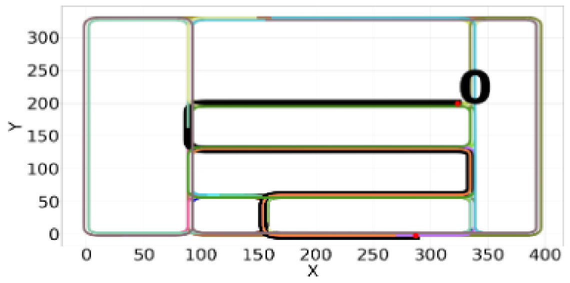
Experimental Setup

The proposed methodology will be validated using a CAV software simulation stack (Figure 54-a)), comprised of CARLA, SUMO and Artery (for implementing V2V communication based on ETSI ITS-G5 protocols), interconnected through ROS software [25], all of them running in an NVIDIA Jetson TX2 platform. Using this stack, we generated the trajectories of 60 vehicles, with length $\mathrm{T} = 3000$ time instances, shown in Figure 54-(b), where ego vehicle's trajectory corresponds to vehicle id 0. Each trajectory is split, between the first 900 time instances for testing, next 1680 time instances for training and the final 420 for validation. The input to the network is the ground truth 3D trajectory degraded by additive white Gaussian noise of zero mean and standard deviation $\mathbf{\Sigma_p} = \mathrm{diag}(1.5\mathrm{m}, 1.5\mathrm{m}, 1.5\mathrm{m})$. Individual local models are trained offline for 50 epochs and batch size is equal to 1, while learning rate and weight decay are set to 0.3. The learning stage of our approach is initiated every 100 time instances, using 2 communication rounds, and 10 epochs for training and fine tuning during SPFL. Baseline methods include cooperative and optimization based MSMV [26], LKF-SA [24], and the data-driven based single KalmaNet [21]. GNSS noise is generated through $\mathbf{\Sigma_p} = \mathrm{diag}(3.5\mathrm{m}, 3.5\mathrm{m}, 3.5\mathrm{m})$, while $\sigma_{\mathrm{d,az,in}} = 1\mathrm{m}, 4^{\mathrm{o}}, 4^{\mathrm{o}}$. Evaluation metrics include $\mathrm{RT}-\mathrm{LE}\ (\mathrm{m})$, i.e., root mean square of euclidean localization error over time for the ego vehicle, and $RT - LAE\ (m)$, i.e., root mean square of average euclidean localization error over time for the neighborhood of ego vehicle, so as to assess 4D situational awareness accuracy.

Evaluation study



(a) Simulation stack for generating vehicle info, e.g., timestamp, id, 3D position, etc.



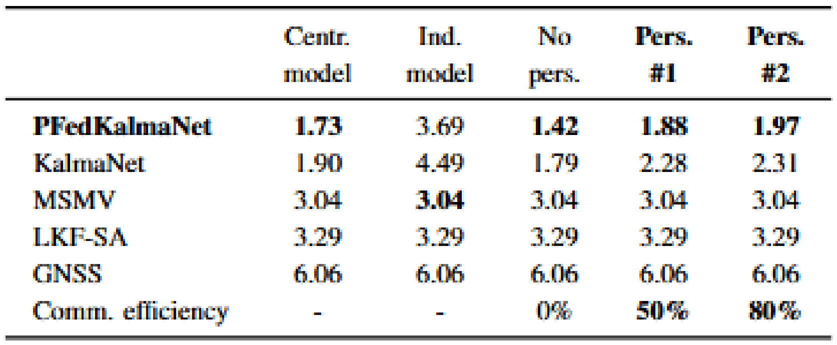
(b) Trajectories of vehicles with distinct colors

Figure 54: Simulation stack based on CARLA, SUMO and Artery, using ROS as middleware, and 60 vehicles' trajectories. Ego vehicle's is shown in black.

Impact of personalization: The first testing scenario will assess the performance with respect to the two different approaches of personalization, assuming communication range 30m. More specifically, our evaluation will consider initially the optimal case of using a centralized model trained using the data across all 60 vehicles, the case of using a model trained with the individual dataset of ego vehicle, and three additional cases of distributed training: i) serverless FL without personalization, i.e., ego vehicle receives from neighbors and aggregates the total number of parameters, ii) SPFL exploiting scenario 1, and iii) SPFL exploiting scenario 2, both from Eq. 13. The results are demonstrated in Figure 55. The first two columns clearly indicate the benefits of using a model trained with diverse datasets. Centralized model is clearly superior than its individual counterpart and the other methods, though it requires a centralized server-based processing architecture. To realize cost-efficient and practical distributed training, all three serverless based FL approaches achieved very promising results. Although serverless FL without personalization attained the greatest accuracy, it performed much less communication efficiently since it requires to exchange the total number of parameters. Most importantly, both SPFL strategies achieved superior results than all the baseline methods, but less efficiently than the previous case, as expected. Moreover, the strategy of aggregating only the layer of Kalman gain is shown to be much more effective, since it directly aims to learn this critical uncertainty matrix. The accuracy of location awareness estimation reaches $\mathbf{1.88m}$, i.e., **17%** and **69%** improvement with respect to single **KalmaNet** and GNSS. The communication efficiency reaches **50%**, since almost half of network's parameters need to be exchanged. The second personalization strategy achieves less accurate results, though outperforming once again the baseline methods, reaching greater communication efficiency of **80%**, due to the significantly smaller number of transmitted parameters. As such, **PFedKalmaNet** realizes effective joint distributed learning and decision-making for cooperative localization via SPFL, but with a trade-off in location accuracy versus communication efficiency. In the following, we will adopt the first personalization strategy due to greater accuracy.

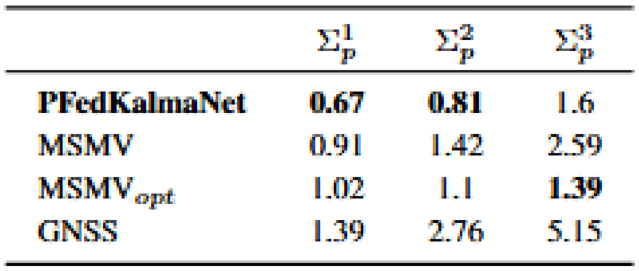
Effectiveness of fusion operation: The second scenario will assess the accuracy of fusion operation by comparing **PFedKalmaNet** with **MSMV** (upon which has been built), when the latter exploits the actual uncertainty of positioning measurements. In highly dynamic traffic conditions, it is very challenging to determine beforehand the true covariance of positioning noise, but we will assume it so as to evaluate how accurate the underlying uncertainty is utilized by our approach. We will define three different levels of positioning noise with covariances $\Sigma_{\mathrm{p}}^{1} = \mathrm{diag}(0.8^2, 0.8^2, 0.8^2)$, $\Sigma_{\mathrm{p}}^{2} = \mathrm{diag}(1.6^2, 1.6^2, 1.6^2)$ and $\Sigma_{\mathrm{p}}^{3} = \mathrm{diag}(3.0^2, 3.0^2, 3.0^2)$ that will be used by **MSMV** in order to estimate Kalman gain. The results in Figure 56 indicate that **MSMV** performance is highly improved when the uncertainty of measurements is exploited (third versus second row), but from the first two columns **PFedKalmaNet** is shown to be much more effective in capturing the uncertainty, improving single GNSS by **52%** and **70%**. However, when positioning is increased in the third case, optimal and ideal **MSMV** is shown to perform more accurate than our method, i.e., **1.39m** versus **1.6m**, respectively, indicating that addressing higher levels of measurement noise via **PFedKalmaNet** is a more challenging task. However, since our method is focused on learning and exploiting the uncertainty, location awareness accuracy is still very promising.

Impact of diverse swarm topologies and network delay: A critical aspect of our approach is how the neighborhood of the ego vehicle is formulated. In the previous scenarios, we have assumed that sensing and communication range is $30\mathrm{m}$, i.e., all vehicles within a distance of $30\mathrm{m}$ from ego vehicle can be considered as connected neighbors. To simulate now realistic V2V connections, we exploit the software simulation stack and the local dynamic maps (LDMs) that Artery simulator generates (Figure 54-(a)). The information of LDM indicate the V2V messages that ego vehicle has received, and as such, we can easily define the neighborhood at each time instant. Those messages, apart from vehicle ID, include also the generation timestamp. However, due to network congestion, limited communication resources, etc., network delay may affect the performance of cooperation, since vehicles receive outdated measurements. Therefore, if only the difference between the current timestamp and the incoming message generation timestamp is lower than a threshold of some ms, nearby vehicle can be considered as neighbor. An example of ego vehicle's neighborhood is shown Figure 54-(a), where bullets represent vehicles and green links the V2V connections. The results of our study are summarized in Figure 57. For example, with time threshold equal to $100\mathrm{ms}$, ego vehicle has only 1 neighbor but exactly due to the smaller size of neighborhood, **PFedKalmaNet** performs less efficient than single **KalmaNet**, i.e., $2.92\mathrm{m}$ with respect to $\mathbf{2.11m}$. When average neighborhood size increases to 4 with time threshold equal to $300\mathrm{ms}$, **PFedKalmaNet** optimally combines acceptable network delay with appropriate neighborhood size. It actually attains $\mathrm{RT-LE}$ equal to $\mathbf{1.91m}$, with respect to $2.28\mathrm{m}$ that **KalmaNet** has achieved, but significantly outperforming the other baseline methods. When the time threshold is equal to $500\mathrm{ms}$, neighborhood size further increases, but due to the more profound impact of receiving and fusing outdated measurements, **PFedKalmaNet** achieves now $\mathrm{RT-LE}$ of $\mathbf{2.04m}$, but still attaining greater accuracy than the other approaches. The last case actually corresponds to the ideal case of fully synchronized messages and no network delay, with the size of neighborhood now reaching 10 vehicles, and **PFedKalmaNet** achieving $1.58\mathrm{m}$, with respect to $2.32\mathrm{m}$, $2.07\mathrm{m}$, $1.71\mathrm{m}$ and $6.05\mathrm{m}$ of **KalmaNet**, **MSMV**, **LKF-SA** and GNSS, respectively. Therefore, we conclude that **PFedKalmaNet** has the potential to effectively exploit and fuse in a scalable manner the data of diverse swarm topologies, formulated by realistic network conditions and acceptable network delay.



	Centr. model	Ind. model	No pers.	Pers. #1	Pers. #2
PFedKalmaNet	**1.73**	3.69	**1.42**	**1.88**	**1.97**
KalmaNet	1.90	4.49	1.79	2.28	2.31
MSMV	3.04	**3.04**	3.04	3.04	3.04
LKF-SA	3.29	3.29	3.29	3.29	3.29
GNSS	6.06	6.06	6.06	6.06	6.06
Comm. efficiency	-	-	0%	**50%**	**80%**

Figure 55: Impact of diverse levels of personalization in location awareness (RT-LAE (m)).



	Σ_p^1	Σ_p^2	Σ_p^3
PFedKalmaNet	**0.67**	**0.81**	1.6
MSMV	0.91	1.42	2.59
MSMV$_{opt}$	1.02	1.1	**1.39**
GNSS	1.39	2.76	5.15

Figure 56: Effectiveness of fusion operation (RT-LAE (m)) is increased/decreased when positioning noise is lower/higher.

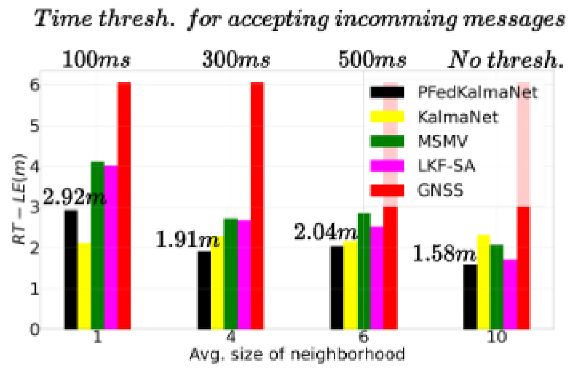


Figure 57: Impact of the number of connections RT-LE(m). No time threshold assumes that all incoming messages are fully synchronized.

3.2.3. Cooperative multi-object detection & tracking via graph signal processing optimization

System model and preliminaries

Consider a tracking-by-detection paradigm, since it is widely employed in MOT literature, as the foundation of our **early cooperative tracking** framework. Each agent i obtains M_i 3D bounding boxes at time t from the output of a 3D detector on LiDAR data. Each 3D bounding box is described by $\boldsymbol{x}_{\boldsymbol{D}}^{(\boldsymbol{i},\boldsymbol{m},\boldsymbol{t})} = \left[x_{i,m}\ y_{i,m}\ z_{(i,m)}\ \theta_{i,m}\ h_{i,m}\ w_{i,m}\ l_{i,m}\right]^T \epsilon\ \mathbb{R}^7$ with $m = 0,1,\dots,M_i$ where $x_{i,m}, y_{i,m}, z_{i,m}$ is the centroid of the bounding box, $\theta_{i,m}$ the angle around the z-axis, and $h_{i,m}, w_{i,m}, l_{i,m}$ the height, width, and length. Hence, the set of detections of agent i at time instance t is described by $\mathcal{D}_i^t = \{\boldsymbol{x}_{\boldsymbol{D}}^{(\boldsymbol{i},\mathbf{1},\boldsymbol{t})}, \boldsymbol{x}_{\boldsymbol{D}}^{(\boldsymbol{i},\mathbf{2},\boldsymbol{t})}, \dots, \boldsymbol{x}_{\boldsymbol{D}}^{(\boldsymbol{i},\boldsymbol{M_i},\boldsymbol{t})}\}\ \epsilon\ \mathbb{R}^{M_i\,x\,7}$. Furthermore, at time instant t, the state of tracked object r is defined by $\boldsymbol{x}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})} = \left[x_r\ y_r\ z_r\ \theta_r\ h_r\ w_r\ l_r\ u_{x_r}\ u_{y_r}\ u_{z_r}\right]^T \epsilon\ \mathbb{R}^{10}$, where x_r, y_r, z_r represent its centroid, θ_r the angle around z-axes, h_r, w_r, l_r the height, width, length, and $u_{x_r}, u_{y_r}, u_{z_r}$ the 3D linear velocity, respectively. Note that the tracked object is the expected outcome of the cooperative tracking, therefore it is not directly related with a specific agent. Based on that, we will employ a state transition model, as well as a measurement model, in order to perform Kalman filtering:

- State transition model:

$$\boldsymbol{x}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})} = f\left(\boldsymbol{x}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t}-\mathbf{1})}, \boldsymbol{w}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})}\right), \qquad \boldsymbol{w}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})} \sim \mathcal{G}(0, \Sigma_w) \qquad \text{Eq. 23}$$

- Measurement model:

$$\boldsymbol{z}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})} = \boldsymbol{x}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})} + \boldsymbol{n}_{\boldsymbol{p}}^{(\boldsymbol{r},\boldsymbol{t})}, \boldsymbol{n}_{\boldsymbol{p}}^{(\boldsymbol{r},\boldsymbol{t})} \sim \mathcal{G}(0, \Sigma_n) \qquad \text{Eq. 24}$$

State transition function $f(\cdot)$ can defined using constant velocity model. Both models are degraded by additive white Gaussian Noise $\boldsymbol{w}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})}$ and $\boldsymbol{n}_{\boldsymbol{p}}^{(\boldsymbol{r},\boldsymbol{t})}$, respectively. According to linear Kalman Filter, each track's state can be defined by the prediction and update equations as follows:

- State Prediction

$$\overline{\boldsymbol{x}}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})} = \boldsymbol{F}\widehat{\boldsymbol{x}}_{T}^{(\mathbf{r},\boldsymbol{t}-\mathbf{1})} + \boldsymbol{w}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})} \qquad \text{Eq. 25}$$

$$\overline{\boldsymbol{P}}^{(\boldsymbol{r},\boldsymbol{t})} = \mathbf{F}\widehat{\boldsymbol{P}}^{(\boldsymbol{r},\boldsymbol{t}-\mathbf{1})}\boldsymbol{F}^{\boldsymbol{T}} + \boldsymbol{Q}^{(\boldsymbol{r},\boldsymbol{t})} \qquad \text{Eq. 26}$$

- State Update

$$\boldsymbol{K}^{(\boldsymbol{r},\boldsymbol{t})} = \overline{\boldsymbol{P}}^{(\boldsymbol{r},\boldsymbol{t})}\boldsymbol{H}^{\boldsymbol{T}}\left[\boldsymbol{H}\overline{\boldsymbol{P}}^{(\boldsymbol{r},\boldsymbol{t})}\boldsymbol{H}^{\boldsymbol{T}} + \boldsymbol{R}^{(\boldsymbol{r},\boldsymbol{t})}\right]^{-\mathbf{1}} \qquad \text{Eq. 27}$$

$$\widehat{\boldsymbol{x}}_{T}^{(\mathbf{r},\boldsymbol{t})} = \overline{\boldsymbol{x}}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})} + \boldsymbol{K}^{(\boldsymbol{r},\boldsymbol{t})}\left[\boldsymbol{z}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})} - \boldsymbol{H}\overline{\boldsymbol{x}}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})}\right] \qquad \text{Eq. 28}$$

$$\widehat{\boldsymbol{P}}^{(\boldsymbol{r},\boldsymbol{t})} = \overline{\boldsymbol{P}}^{(\boldsymbol{r},\boldsymbol{t})} - \boldsymbol{K}^{(\boldsymbol{r},\boldsymbol{t})}\boldsymbol{H}\overline{\boldsymbol{P}}^{(\boldsymbol{r},\boldsymbol{t})} \qquad \text{Eq. 29}$$

where $\boldsymbol{F} \in \mathbb{R}^{10\ x\ 10}$ is the transition matrix, $\boldsymbol{Q}^{(\boldsymbol{r},\boldsymbol{t})} \in \mathbb{R}^{10\ x\ 10}$ the process noise covariance matrix, $\boldsymbol{R}^{(\boldsymbol{r},\boldsymbol{t})} \in \mathbb{R}^{10\ x\ 10}$ the measurement noise covariance matrix, $\boldsymbol{H} \in \mathbb{R}^{7\ x\ 10}$ the measurement model matrix, $\boldsymbol{Q}^{(\boldsymbol{r},\boldsymbol{t})} \in \mathbb{R}^{10\ x\ 7}$ the Kalman gain and $\boldsymbol{P}^{(\boldsymbol{r},\boldsymbol{t})} \in \mathbb{R}^{10\ x\ 10}$ the covariance matrix of the tracked object. Finally, set $\mathcal{T}^t = \{\widehat{\boldsymbol{x}}_{T}^{(\mathbf{1},\boldsymbol{t})}, \widehat{\boldsymbol{x}}_{T}^{(\mathbf{2},\boldsymbol{t})}, \dots, \widehat{\boldsymbol{x}}_{T}^{(\mathbf{R_T},\boldsymbol{t})}\} \in \mathbb{R}^{R_T\ x\ 10}$, contains the state of tracked objects $r = 1, 2, \dots, R_T$, after performing Kalman filtering. Those two sets of detected and tracked bounding boxes will act as the basis in order to formulate the proposed framework.

Association and Tracks Management approaches

The Association and Tracks Management approaches constitute the core components of a tracking framework handling tracks' states and their lifetimes. Specifically, the Association module employs the 3D Intersection over Union (3D IoU) as a similarity metric, and the Hungarian Algorithm (HA) as an association algorithm to correlate trajectories and detections. Upon successful association, the track's state is updated utilizing the corresponding detection $\boldsymbol{x}_{\boldsymbol{D}}^{(\boldsymbol{i},\boldsymbol{m},\boldsymbol{t})}$ according to Eq. 28, Eq. 29, where $\boldsymbol{z}_{\boldsymbol{T}}^{(\boldsymbol{r},\boldsymbol{t})} = \boldsymbol{x}_{\boldsymbol{D}}^{(\boldsymbol{i},\boldsymbol{m},\boldsymbol{t})}$. Conversely, in case of unsuccessful association, the track's state retains its previous state without update. Upon unmatched detections, new tracks are initialized with their attributes. The Tracks Management module is handling the lifetime of each track and establishes the tracking procedure after the association stages with detections [27], [28]. Specifically, a track is "confirmed", if it achieves a successful association over a predefined number of consecutive frames, denoted as $hits$. Conversely, if a track fails to establish an association for a specified number of consecutive steps, noted as age, it is considered as "dead" and is removed from further processing.

Graph Signal Processing Preliminaries

The Graph Laplacian operator is a well-known Graph Signal Processing Tool [29] which recovers the absolute coordinates of graph vertices through differential coordinates and anchor points in a least-squares sense. The inherent geometry of the connectivity graph is captured by differential coordinates of the vertices, which correspond to the barycenter of each vertex's neighboring nodes. Additionally, anchors are utilized as complementary information for each vertex. Furthermore, the Laplacian matrix captures the connections between the vertices. More specifically, consider an undirected graph with N^t nodes represented as $J^t = (\mathcal{V}^t, \mathcal{E}^t)$, where $\mathcal{V}^t$ and $\mathcal{E}^t$ denote the set of vertices and edges, respectively. The objective is to define and minimize the cost function $O(\boldsymbol{v}^t) = ||\boldsymbol{L}^t\boldsymbol{v}^t - \boldsymbol{\delta}^t||_2^2$ in each spatial attribute x, y, z, where $\boldsymbol{v}^t = \left[v^{(1,t)}\ v^{(2,t)} \ldots v^{(N^t,t)}\right]^T \epsilon\, \mathbb{R}^{N^t}$ is the vector of graph vertices, $\boldsymbol{L}^t \epsilon\, \mathbb{R}^{N^t x N^t}$ is the Laplacian matrix and $\boldsymbol{\delta}^t \epsilon\, \mathbb{R}^{N^t}$ the vector of differential coordinates. Laplacian matrix is equal to $\boldsymbol{L}^t = \boldsymbol{D}^t - \boldsymbol{A}^t$ where $\boldsymbol{D}^t$, $\boldsymbol{A}^t\ \epsilon\, \mathbb{R}^{N^t x N^t}$ are the well-known degree and adjacency matrices. Furthermore, in each spatial attribute the differential vector is equal to $\boldsymbol{\delta}^t = [\delta^{(1,t)}\ \delta^{(2,t)} \ldots \delta^{(N^t,t)}]$, where $\delta^{(i,t)} = \sum_{\boldsymbol{i}}^{N^t}(v^{(i,t)} - v^{(j,t)})$ among all connected vertices. However, due to the singular properties of Laplacian matrix, we have to extend it using the identity matrix, leading to the extended Laplacian matrix $\tilde{\boldsymbol{L}}^t \epsilon\, \mathbb{R}^{2N^t x N^t}$, and thus, $\boldsymbol{\delta}^t$ is restructured to the measurement vector $\boldsymbol{b}^t \epsilon\, \mathbb{R}^{2N^t}$ consisting of not only differential coordinates but also the so-called anchor points, which contain any knowledge we have about each vertex. Previous quantities, as well as anchors vector $\boldsymbol{a}^t \epsilon\, \mathbb{R}^{N^t}$ are defined as follows:

$$\tilde{\boldsymbol{L}}^t = \begin{bmatrix} \boldsymbol{D}^t - \boldsymbol{A}^t \\ \boldsymbol{I}^t \end{bmatrix}, \quad \boldsymbol{b}^t = [\boldsymbol{\delta}^t\ \boldsymbol{a}^t]^T \qquad \text{Eq. 30}$$

$$\boldsymbol{a}^t = \left[a^{(1,t)}\ a^{(2,t)} \ldots a^{(N^t,t)}\right]^T \qquad \text{Eq. 31}$$

Employing linear least-squares minimization, the optimal solution of vertices' locations in each spatial attribute is equal to:

$$\boldsymbol{v}_*^t = \left(\left(\tilde{\boldsymbol{L}}^t\right)^T \tilde{\boldsymbol{L}}^t\right)^{-1} \left(\tilde{\boldsymbol{L}}^t\right)^{\boldsymbol{T}} \boldsymbol{b}^t, \qquad \boldsymbol{v}_*^t \epsilon\, \mathbb{R}^{N^t} \qquad \text{Eq. 32}$$

Graph Laplacian Processing Technique

The Graph Laplacian Processing technique actually facilitates the localization of nodes of a graph by leveraging inherent geometric structure defined by the differential coordinates and anchors. Based on that fact, two methods will be derived, introducing the Graph Laplacian Cooperative MOT (CoMOT) scheme which takes advantage of a fully connected graph among all multi-agent detections as illustrated in Figure 58. The one-stage of association approach serves as the initial step for the proposed two-stage association Graph Lap CoMOT method. Both of the presented approaches reduce spatial noise of vehicle's detections by smoothing the individual error related to the bounding box's 3D centroid. Therefore, each bounding box is refined and correlated with existing tracks, enabling more accurate trajectory estimation during the Kalman Filter update step. More specifically, differential coordinates and Laplacian matrix capture the connections among all detections, while anchors are utilized for the overlapped detections that represent objects simultaneously observed from different agents. Although our methods are presented assuming two CAVs, following the standard methodology of [30], they can easily be generalized for a larger number of connected vehicles.

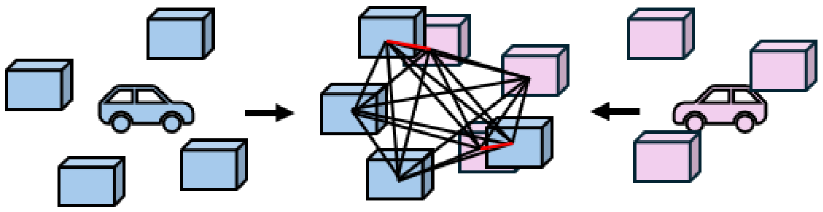


Figure 58: Fully connected graph of detections from two neighboring agents i and j at time t. Detections from agent i (blue color) are interconnected among them, and also linked to detections from agent j (pink color). Red and black edges indicate the overlapped information of the object shared between the agents and all the connections of the graph, respectively.

To be more specific, consider the undirected graph $J^t = (\mathcal{V}^t, \mathcal{E}^t)$, where the set of nodes $\mathcal{V}^t = \{\mathcal{D}_i^t, \mathcal{D}_j^t\}$ represents the detections from agent i and j. With respect to the set of edges $\mathcal{E}^t$, the detections of agent i are interconnected among them and also linked to the detections of agent j. Note that we focus only on estimating the 3D centroid of each bounding box. Therefore, in each spatial attribute x, y, z , the differential vector is equal to $\boldsymbol{\delta^t} = \left[\, \delta^{(1,t)}\ \delta^{(2,t)} \ldots\ \delta^{(N^t,t)} \right] \epsilon\, \mathbb{R}^{N^t}$, where $\delta^{(m,t)} = \sum_{n=0}^{M_j} \left(x_{i,m} - x_{j,n}\right) + \sum_{n=0}^{M_i} \left(x_{i,m} - x_{i,n}\right)$ describes the scalar differential coordinate of detection m. Additionally, the extended Laplacian matrix is defined by Eq. 30 corresponding to a fully connected graph, anchors vector $\boldsymbol{a^t} \epsilon\, \mathbb{R}^{N^t}$ will contain the scalar complimentary information of x-part of $\boldsymbol{x_D^{(i,m,t)}}$ as analytically explained in the following, while measurement vector $\boldsymbol{b^t} = [\boldsymbol{\delta^t}\ \boldsymbol{a^t}]^{\boldsymbol{T}} \epsilon\, \mathbb{R}^{2N^t}$. Similar equations are followed for y

and z attributes. Thus, the refined bounding boxes, since we actually refine their 3D centroids, will be further used by the core modules of CoMOT in order to enhance the overall accuracy.

All in One Stage (AOS) Graph Lap-CoMOT

A simple CoMOT which utilizes a fully connected graph topology on multi-agent detections is the proposed **All in One Stage (AOS) Graph Lap-CoMOT**. More specifically, this method refines and fuses multi-vehicle bounding boxes through the Graph Laplacian framework and thereafter, associates them with existing tracks. To acquire additional information at time t for forming the anchors vector $\boldsymbol{a^t}$, the 3D detections of the two agents $\mathcal{D}_i^t$ and $\mathcal{D}_j^t$ are associated through 3D IoU and HA. The set of m_i overlapped detections and u_i non-overlapped of the i are described by $m\mathcal{D}_i^t \in \mathbb{R}^{m_i\, x\, 7}$, and $u\mathcal{D}_i^t \in \mathbb{R}^{u_i\, x\, 7}$, respectively. Similarly, for the agent j, $m\mathcal{D}_j^t \in \mathbb{R}^{m_j\, x\, 7}$, and $u\mathcal{D}_j^t \in \mathbb{R}^{u_j\, x\, 7}$. The anchors vector $\boldsymbol{a^t}$ is formed for x spatial attribute as follows:

$$\boldsymbol{a^t} = \left[\boldsymbol{mx_{j,m}}\ \boldsymbol{mx_{i,m}}\ \boldsymbol{ux_{i,m}}\ \boldsymbol{ux_{j,m}}\right]^T \qquad \text{Eq. 33}$$

where $\boldsymbol{mx_{i,m}} = \left[mx_{i,1}\ mx_{i,2} \dots\ mx_{i,m_i}\right]^T \in \mathbb{R}^{m_i}$, $\boldsymbol{mx_{j,m}} = \left[mx_{j,1}\ mx_{j,2} \dots\ mx_{j,m_j}\right]^T \in \mathbb{R}^{m_j}$ are the successfully associated detections of the agent i, j and $\boldsymbol{ux_{i,m}} = \left[x_{i,1}\ x_{i,2} \dots\ x_{i,u_i}\right]^T \in \mathbb{R}^{u_i}$, $\boldsymbol{ux_{j,m}} = \left[x_{j,1}\ x_{j,2} \dots\ x_{j,u_j}\right]^T \in \mathbb{R}^{u_j}$ the unmatched bounding boxes of i and j, respectively.
Therefore, the least-squares minimization of Eq. 32 at time t is noted as $\boldsymbol{G^t} \in \mathbb{R}^{N^t}$and describes the refined and fused x attribute of the detections. Similar equations are followed for the y, z attributes. The set of refined detections which consists all the attributes on each 3D bounding box is $\mathcal{G}^t \in \mathbb{R}^{N^t\, x\, 7}$. Then, the optimized detections $\mathcal{G}^t$ are associated with the existing tracks $\mathcal{T}^t$ in terms of 3D IoU and HA and follows the Association and Tracks Management approaches

. Hence, the successfully associated trajectories are denoted as $m\mathcal{T}^t$ and update their states by Eq. 28, Eq. 29 with the successfully associated detections $m\mathcal{G}^t$. Moreover, the unsuccessfully associated tracks retain their previous state without update and are denoted as $u\mathcal{T}^t$, and the unsuccessfully associated detections $u\mathcal{G}^t$ initialize new tracks. At the next time step $t+1$, all active tracks are predicted by Eq. 25, Eq. 26. The **AOS GraphLap CoMOT** serves as the first step in deriving the proposed approach of the next subsection.

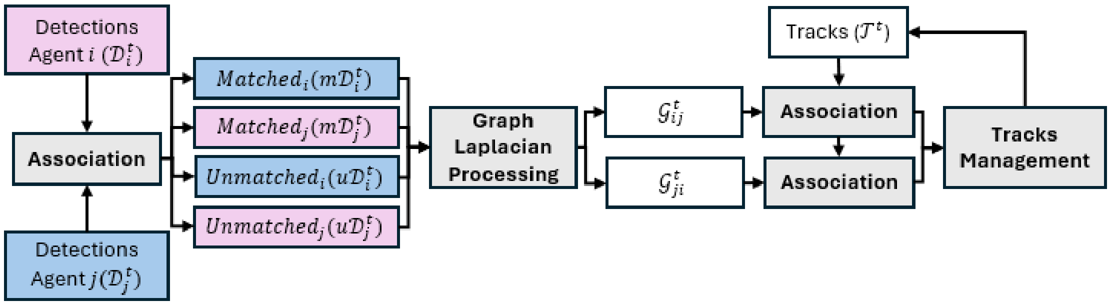

Detections Agent i ($\mathcal{D}_i^t$)
Association
Detections Agent j ($\mathcal{D}_j^t$)
$Matched_i(m\mathcal{D}_i^t)$
$Matched_j(m\mathcal{D}_j^t)$
$Unmatched_i(u\mathcal{D}_i^t)$
$Unmatched_j(u\mathcal{D}_j^t)$
Graph Laplacian Processing
$\mathcal{G}_{ij}^t$
$\mathcal{G}_{ji}^t$
Tracks ($\mathcal{T}^t$)
Association
Association
Tracks Management

Figure 59: TSA Graph Lap-CoMOT of the Graph Lap-CoMOT approach.

Two Stages Association (TSA) Graph Lap-CoMOT

The **Two Stages Association (TSA) Graph Lap-CoMOT** technique (Figure 59) is an advanced extension of the **AOS Graph Lap-CoMOT**. This method designs two stage association modules to capture unmatched trajectories from the first stage and combine obtained detections from different agents in slightly modified definition of anchors vector. Similarly to the AOS technique, the detections of the two CAVs are associated in order to provide additional information on the overlapped bounding boxes. Different anchors vectors are formulated based on the number of agents. Therefore, in case of two CAVs, two anchors vectors are defined for each spatial attribute along x, y and z, and demonstrated for x according to:

$$\boldsymbol{a_{ij}^t} = \left[\boldsymbol{mx_{j,m}}\ \boldsymbol{mx_{j,m}}\ \boldsymbol{ux_{i,m}}\ \boldsymbol{ux_{j,m}}\right]^T \quad \text{Eq. 34}$$

$$\boldsymbol{a_{ji}^t} = \left[\boldsymbol{mx_{i,m}}\ \boldsymbol{mx_{i,m}}\ \boldsymbol{ux_{i,m}}\ \boldsymbol{ux_{j,m}}\right]^T \quad \text{Eq. 35}$$

Therefore, different least-squares solutions of Eq. 32 are calculated. The fused and refined detections corresponding to the anchors vector $\boldsymbol{a_{ij}^t}$ are represented as $\boldsymbol{G_{ij}^t} \epsilon\ \mathbb{R}^{N^t}$ and similarly to the $\boldsymbol{a_{ji}^t}$ as $\boldsymbol{G_{ji}^t}$. Similar equations are followed for the y, z attributes. Hence, the detections with all detections' attributes are $\mathcal{G}_{ij}^t \epsilon\ \mathbb{R}^{N^t\ x\ 7}$ and $\mathcal{G}_{ji}^t \epsilon\ \mathbb{R}^{N^t\ x\ 7}$. Firstly, $\mathcal{G}_{ij}^t$ are associated with the existing tracks $\mathcal{T}^t$ as described previously. Upon successful associations, $m\mathcal{T}^t$ tracks update their state through the refined associated detection $m\mathcal{G}_{ij}^t$ by Eq. 28 Eq. 29. In case of unmatched refined detections $u\mathcal{G}_{ij}^t$, new tracks are initialized. Additionally, the unmatched tracks $u\mathcal{T}^t$ are not discarded by the Tracks Management Module, instead are associated with the $\mathcal{G}_{ji}^t$ refined detections. Thereafter, the Association and Tracks Management approaches

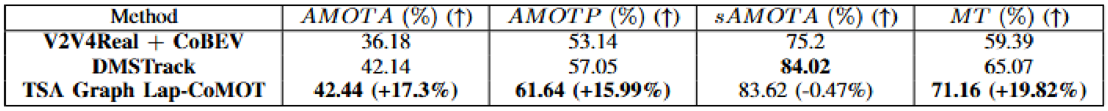
takes place with $m\mathcal{T}^t$, $m\mathcal{G}_{ji}^t$, the matched tracks, optimized detections and $u\mathcal{T}^t$ and $u\mathcal{G}_{ji}^t$ the unmatched tracks and unmatched detections. This process continues based on the number of agents. At the next time step $t+1$, all active tracks are predicted by Eq. 25-Eq. 26. Regarding computational efficiency, Graph Laplacian Processing is based on an efficient sparse least-squares optimization framework, with complexity equal or lower than $O((2N^tN^t)^2)$(where $2N^t$ and N^t the number of rows and columns of the corresponding Laplacian matrix, indicating actually the size of the topology) [31].

Experimental Setup and Metrics

Experiments have been conducted in the well-known V2V4Real [32] dataset on an NVIDIA RTX 4090 GPU. The dataset training split includes 32 driving sequences and the testing 9 driving sequences from two simultaneously driven vehicles. The frame rate is 10Hz. Our proposed method ensures real-time feasibility by sharing only the bounding box parameters among agents, and thus, agent j with M_j bounding boxes, has to transmit in total $M_j \times 7$ floats, since 7 are the parameters describing each bounding box, ensuring low latency during the communication. Additionally, we define the tracking parameters $age = 2$, and $hits = 3$ to balance robustness and adaptability. We have tested our framework in the testing sequences and we compared our method with the state-of-the-art V2V4Real and the DMSTrack [30], a deep-learning CoMOT approach. The Graph Lap-CoMOT and DMSTrack approaches utilize PointPillar detector [33] with 55% detection accuracy. DMSTrack sequentially associates the 3D detections of the ego vehicle to the tracks, and the unmatched tracks with the bounding boxes of the other. Also, induces noise covariance matrix for each detection which is calculated by a deep neural network. Furthermore, we employed the V2V4Real method using multi-agent CoBEV Detector [34] with higher detection accuracy (66.5%) than PointPillar for challenging evaluation. We employ the evaluation metrics in 3D MOT of [27] including 1) Average Multi-Object Tracking Accuracy (AMOTA), counting the errors (False Positives (FP), False Negatives (FN), Identity Switches(IDSW)) with respect to Ground Truth (GT), 2) Average Multi-Object Tracking Precision (AMOTP) calculating the overlap between the predicted tracked objects with the Ground Truth objects with respect to the True Positives, 3) scaled AMOTA (sAMOTA), a linearized AMOTA across various confident thresholds, 4) Mostly Tracked (MT), the percentage of the correct tracking of objects over 80% of their life, 6) Multi-Object Tracking Precision (MOTP) calculating the overlap between the predicted tracked objects and GT with respect to the TP with 0.25 overlapping threshold.

Evaluation Study

Figure 60 demonstrates the performance of the **TSA Graph Lap-CoMOT** method and the two baseline methods on average across all testing sequences from the V2V4Real dataset. The **TSA Graph Lap-CoMOT** consistently achieves superior performance across the average sequences, with the maximum **17.3%** improvement in AMOTA. Hence, the two stages association of tracks with refined detections captures potential occluded objects increasing the TP, reducing FN and thus, performs high tracking accuracy. Furthermore, the **TSA Graph Lap-CoMOT** reduces the noise in detections through the Graph Laplacian operator and the fully connected graph topology in least-square minimization and thus, outperforms in the tracking precision with a maximum **15.99%** improvement in AMOTP. Moreover, it enhances MT by **19.82%** by tracking most objects for over of 80% of their lifetime. However, **TSA Graph Lap-CoMOT** performs 0.47% below the DMSTrack in sAMOTA. Therefore, **TSA Graph Lap-CoMOT** achieves superior performance across most tracking metrics by fusing the centroids of detections and reducing position error in Graph Laplacian manner, while captures unmatched trajectories preventing false termination with the two stages association.

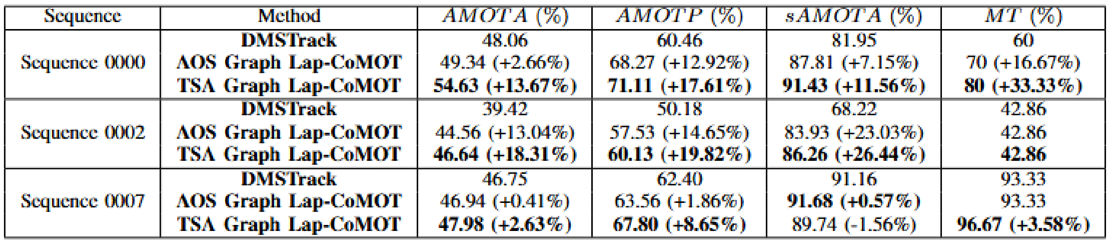


Method	$AMOTA$ (%) (↑)	$AMOTP$ (%) (↑)	$sAMOTA$ (%) (↑)	MT (%) (↑)
V2V4Real + CoBEV	36.18	53.14	75.2	59.39
DMSTrack	42.14	57.05	**84.02**	65.07
TSA Graph Lap-CoMOT	**42.44 (+17.3%)**	**61.64 (+15.99%)**	83.62 (-0.47%)	**71.16 (+19.82%)**

Figure 60: Average tracking results in V2V4Real. Plus (minus) sign indicate the rate of accuracy improvement (de-crease) with respect to the maximum deviation of the state-of-the-art CoMOT methods.

To further enrich our ablation study, Figure 61 demonstrates the tracking performance of both **Graph Lap-CoMOT** and DMSTrack methods on three indicative testing sequences of V2V4Real dataset, highlighting the benefits of two stages of association. The **TSA Graph-Lap-CoMOT** achieves superior performance with maximum improvements of **18.31%, 19.82%, 26.44%, 33.33%** in AMOTA, AMOTP, sAMOTA, MT respectively over DMSTrack. Additionally, **TSA Graph Lap-CoMOT** outperforms with maximum improvements up to **10.72%, 6.67%, 4.12%,14.29%** in AMOTA, AMOTP, sAMOTA, MT respectively over the one stage of association, **AOS Graph Lap-CoMOT**, in all evaluated metrics. This demonstrates its effectiveness in addressing unmatched trajectories from the first stage by preventing termination in cases of failed association leveraging two association stages. Moreover, the **AOS Graph Lap-CoMOT** accomplishes significant improvements across various metrics despite being a simplified and single stage association method of the **TSA Graph Lap-CoMOT**. Specifically, it achieves a **13.04%** improvement in AMOTA and **23.03%** in sAMOTA for Sequence 0002, with more TP and consequently fewer FN tracked objects revealing and exploiting spatial geometric structures on the fully connected graph instead of DMSTrack. Additionally, a **14.65%** in AMOTP and a **16.67%** in MT improvement for Sequences 0002 and 0000 demonstrate the reduced spatial error of multi-agent detections through the Graph Laplacian Operator. Therefore, **TSA Graph Lap-CoMOT** exploits spatial coherences between multi-agent detections formulating a fully connected graph topology while addresses potential occluded objects with the two stages of association.

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI
KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology



Sequence	Method	$AMOTA$ (%)	$AMOTP$ (%)	$sAMOTA$ (%)	MT (%)
Sequence 0000	**DMSTrack**	48.06	60.46	81.95	60
	AOS Graph Lap-CoMOT	49.34 (+2.66%)	68.27 (+12.92%)	87.81 (+7.15%)	70 (+16.67%)
	TSA Graph Lap-CoMOT	**54.63 (+13.67%)**	**71.11 (+17.61%)**	**91.43 (+11.56%)**	**80 (+33.33%)**
Sequence 0002	**DMSTrack**	39.42	50.18	68.22	42.86
	AOS Graph Lap-CoMOT	44.56 (+13.04%)	57.53 (+14.65%)	83.93 (+23.03%)	42.86
	TSA Graph Lap-CoMOT	**46.64 (+18.31%)**	**60.13 (+19.82%)**	**86.26 (+26.44%)**	**42.86**
Sequence 0007	**DMSTrack**	46.75	62.40	91.16	93.33
	AOS Graph Lap-CoMOT	46.94 (+0.41%)	63.56 (+1.86%)	**91.68 (+0.57%)**	93.33
	TSA Graph Lap-CoMOT	**47.98 (+2.63%)**	**67.80 (+8.65%)**	89.74 (-1.56%)	**96.67 (+3.58%)**

Figure 61: Ablation study on three indicative V2V4Real sequences. Plus (minus) sign indicate the rate of accuracy improvement (decrease) with respect to the DMSTrack

Figure 62 highlights the impact on tracking precision when the size of graph topology increases, demonstrating in fact the scalability of **TSA Graph Lap-CoMOT**. Higher localization accuracy (MOTP) is achieved on average through **TSA Graph Lap CoMOT** when 6 TP objects appear across the testing sequences, despite the higher frequency of appearance in DMSTrack. Moreover, a maximum MOTP up to **64.55%** is achieved once again by our approach, when 19 TP objects exist across the testing sequences. Therefore, this fact emphasizes the significance of Graph Laplacian Processing in fusing diverse and growing number of bounding boxes.

Finally, Figure 63 demonstrates GT, **TSA Graph Lap-CoMOT** and DMSTrack trajectories at frames 91 and 130 in Sequences 0000 and 0007, with green, red and yellow colors. In all cases, we can clearly observe that proposed **TSA Graph Lap-CoMOT** significantly improved the location of each bounding box with respect to DMSTrack, as well as captured an object that DMSTrack failed to track succesfully. Therefore, **TSA Graph Lap-CoMOT** achieved competitive, accurate and precise tracking results in the real-world V2V4Real dataset.

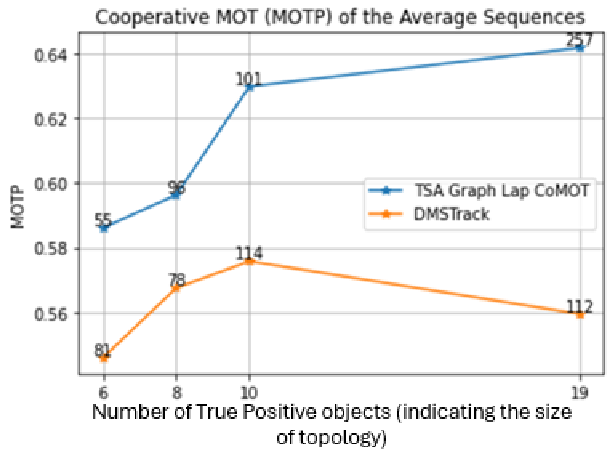
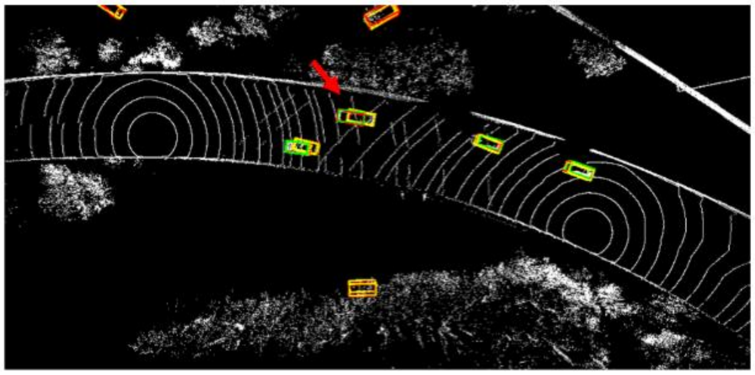


Figure 62: Impact of increasing graph topology size on MOTP on average. The x and $y-$axis demonstrate the number of True Positive (TP) objects per frame and average MOTP, respectively. Bullet points indicate the frequency of each TP count per frame. TSA Graph Lap-CoMOT enhances tracking performance as the graph topology size increases.


(a) Precise localization of the object by the TSA Graph Lap-CoMOT, as indicated by the red arrow, Frame 91

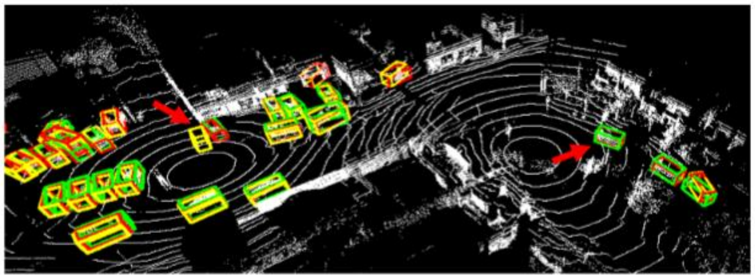

(b) Object capturing and precise localization, while the baseline fails, Frame 74

Figure 63: Qualitative results of CoMOT based on assessing tracking accuracy and precision with GT (green), TSA Graph Lap-CoMOT (red) and DMSTrack (yellow).

3.3. Concluding remarks

This section has detailed the design, development, and evaluation of the AutoTRUST external sensing and monitoring system aimed at enabling 4D cooperative situational awareness for connected and automated vehicles. By combining robust single-agent perception modules with advanced multi-agent cooperative frameworks, the system significantly enhances environmental understanding, road safety, and context-aware mobility. In more detail, the single-agent modules—focusing on road condition and traffic sign detection—demonstrated efficient, real-time performance on embedded platforms using lightweight yet accurate deep learning models such as YOLOv8. These modules provide essential data streams for interpreting external scenes and contribute directly to driver assistance and autonomous decision-making. On the multi-agent side, the section introduced a comprehensive framework for cooperative localization and object tracking using federated learning and distributed inference strategies. Techniques, such as PFedKalmaNet and Graph Laplacian-based CoMOT, represent state-of-the-art methods for decentralized learning and decision-making, capable of operating under real-world constraints such as network delays, communication efficiency, and heterogeneous sensor inputs. In the following section, we move away from the internal and external monitoring system introduced by AutoTRUST and focus on approaches that enhance the security, privacy and trustworthiness of the entailed AI-powered solutions.

4. Privacy and data trustworthiness of sensing system

As highlighted in the previous chapters of the deliverable, modern AVs operate as mobile data collection platforms, continuously gathering vast amounts of information through various sensing modalities (LiDARs, cameras, radar systems, etc.), and environmental monitoring devices. This data encompasses not only vehicle operational parameters but also detailed information about passengers, surrounding environments, traffic patterns, and infrastructure conditions. The sensitive nature of this information, combined with the need for real-time processing and inter-vehicle communication, creates complex privacy and security challenges that traditional data protection approaches are insufficient to address. Furthermore, the distributed nature of AV networks, where multiple vehicles must collaborate and share information to achieve optimal performance, introduces additional vulnerabilities and trust considerations that require innovative technical solutions.

This section addresses the critical dimensions of data privacy, security, and trustworthiness in advanced sensing systems for autonomous and connected vehicles. To this end, the presented work explores both foundational and cutting-edge privacy-preserving strategies. Section 4.1 outlines the ethical and legal frameworks guiding data handling, while Section 4.2 delves into federated learning methodologies and their security implications within AV contexts. Section 4.3 focuses on the enhancement of the privacy and resolution of LiDAR-based external sensing, utilizing techniques such as deep unrolling as well as federated approaches, while, finally, our concluding remarks are presented in Section 4.4.

4.1. Security and privacy of data, confidentiality and ethical compliance

All technical activities will be guided by principles that prioritize safety, transparency, and inclusiveness, particularly for vulnerable and underrepresented user groups. The project adopts a proactive approach to identifying and mitigating risks associated with the collection, processing, and deployment of data. This ensures that development is ethically grounded and technologically robust. In doing so, AutoTRUST seeks to anticipate and address societal concerns.

This includes adherence to the GDPR, AI Act (where applicable), and other sector-specific legal frameworks relevant to autonomous mobility and data processing (aligned with national and international requirements). All partners will receive guidance and support in implementing these practices through internal protocols and continuous training.

The project also fosters a culture of accountability and shared responsibility across institutions. By promoting regulatory alignment from the outset, AutoTRUST ensures legal soundness and ethical integrity in its research and innovation efforts.

The project places a strong emphasis on designing adaptive systems that not only respond to environmental and operational uncertainties but also protect user rights and autonomy. This involves integrating privacy-by-design and security-by-design principles in all system layers, from data sensing to decision-making algorithms. Users' trust will be enhanced through transparent and co-creative processes, consent mechanisms, and secure data handling.

In the scope of WP1 (D1.2 and D1.5), the ethical and data management plan was established. Ethical principles and data governance strategies that guide the entire project were established, ensuring coherence across all technical and research activities. By being embedded from the early stages of the project, the plan supports proactive ethical reflection and regulatory alignment, while each partner ensures compliance and a homogenous approach at the local level. Its transversal application guarantees that all work packages contribute consistently to the project's overall integrity and responsibility framework.

4.2. Privacy preservation techniques in autonomous vehicles systems

4.2.1. Federated Learning

Autonomous vehicles function as essential building blocks which form the basis of future Intelligent Transportation Systems (ITS). The development of Big Data, Internet of Things (IoT), edge computing, and Artificial Intelligence (AI) has expanded potential improvements for transportation systems through AVs, particularly by improving driving safety, reducing traffic congestion, and lowering pollution levels.

The operation of AVs depends on V2X communication. This type of communication enables AVs to establish communication links not only with peer vehicles, but also with traffic infrastructure, satellites, pedestrians, and different ITS components. Additionally, under specified operational conditions and regulatory rules, AVs execute autonomous driving operations through sensor data analysis and Machine Learning (ML) algorithms.

Currently, AVs generate vast volumes of raw data, which stems from three main sources which include electronic control units (ECUs), internal and external sensors, and V2X communications. It is transmitted either continuously or at intervals to multiple destinations, including other vehicles, roadside infrastructure, and cloud platforms. These data transmission supports multiple purposes which include traffic monitoring, congestion control, Quality of Service (QoS) and Quality of Experience (QoE) enhancement, and vehicle diagnostics.

However, these enormous amounts of data needed to train ML models have generated substantial concerns about data protection alongside unapproved data exploitation and privacy violations. In some regions, the collection of sensor data by AVs must comply with privacy regulations. For instance, the General Data Protection Regulation (GDPR) presents a legal framework that European countries have established to enforce personal data processing compliance with privacy standards [35].

Despite significant progress in ML techniques, the development of secure and scalable centralized ML models for all vehicles has proven to be unfeasible. However, a new ML technique called FL has emerged as a promising solution to the problems mentioned. Figure 64 shows the topology of FL for AVs. In FL, edge devices, including AVs, do not transmit raw local data to a central server. Instead, as shown in Figure 64, they share only gradients or learnable parameters (W1, W2, W3, ..., Wn) derived from local AI model training. A central server, called federated server, collects all gradients from the AVs to improve the global model which gets redistributed back to the AVs. Multiple communication rounds exist to achieve global model convergence at its optimal performance level. Using the mentioned approach, FL delivers two major advantages which decrease network data transfer while protecting the privacy of data stored on local devices [36].

In AVs, and in each vehicle in a FL setup, multiple types of local Artificial Neural Network (ANN) models could be utilized, such as Convolutional Neural Network (CNN), Multilayer Perceptron (MLP), Recurrent Neural Network (RNN), Transformers, and Autoencoders. These models will be discussed in more detail below.

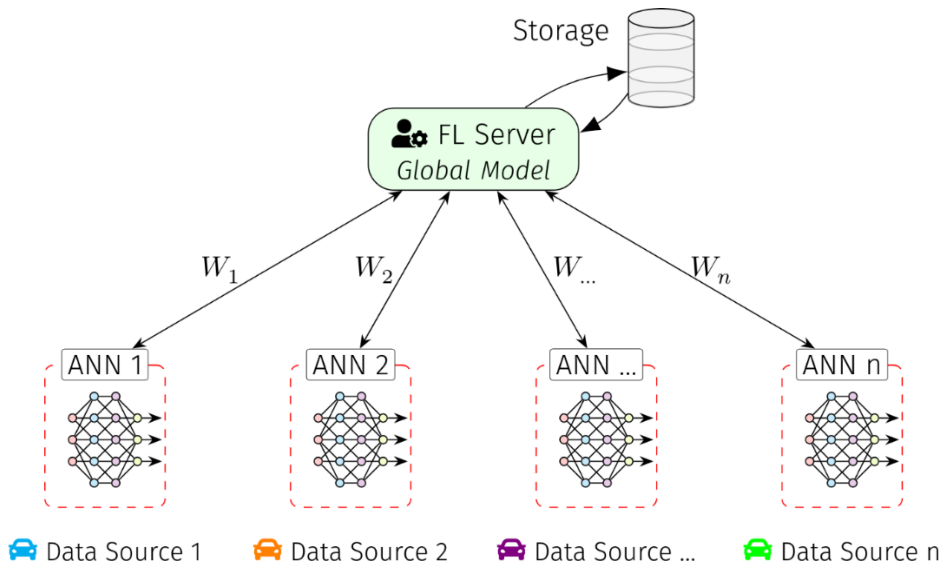


Figure 64: Federated Learning topology for autonomous vehicles.

Convolutional Neural Network: Due to its exceptional performance in visual perception tasks, CNNs have emerged as a key technology in the development of AVs. CNNs are excellent at automatically deriving hierarchical features from unprocessed pixel data, which allows them to analyse images and videos taken by cameras mounted on vehicles. Critical AV functions including object detection, lane detection, and scene understanding, all necessary for safe navigation in challenging driving environments, require this capacity. Furthermore, CNN architecture developments like Faster R-CNN [37] and other real-time hybrid frameworks [38] guarantee that the models continue to be effective and responsive, which are essential qualities required for obstacle avoidance and real-time navigation.

Multilayer Perceptron: MLPs are a fundamental neural network architecture that finds broad application in various machine learning applications, including those in the domain of AVs. As feed-forward neural networks, MLPs consist of multiple layers of neurons, including input, hidden, and output layers, that are connected through weighted links. It is this structure that enables MLPs to learn complex, non-linear relationships in data, making them useful for applications like trajectory [39] and motion prediction [40].

Recurrent Neural Network: RNNs specifically designed to deal with sequential data by maintaining a hidden state that preserves temporal dependencies, which are very important in tasks such as lane change prediction [41], [42] and path prediction [43]. The RNN structure enables them to capitalize on previous information from input sequences, thus supporting the modeling of time-series data, which is a common characteristic in autonomous driving scenarios.

However, standard RNNs do not do well with long-term dependencies due to issues like the vanishing gradient problem. As a specific type of RNNs, Long Short-Term Memory (LSTM) solves such constraints by possessing a sophisticated gating mechanism to retain and forget information over long periods of time. This makes LSTMs perfectly suitable for tasks based on memory spanning long sequences of information. A couple of works [44], [45] have applied LSTM to AV-related challenges.

Transformers: Transformer [46] architecture and its variant, Vision Transformer (ViT) [47] have emerged as powerful alternatives to traditional RNNs and CNNs. One of the most important advantages of using Transformers in FL settings is their attention mechanism, which allows the model to dynamically focus on the most relevant parts of the input sequences. This functionality is particularly helpful in AV use cases where some elements of the environment surrounding us, captured with various sensors, can vary extensively in importance depending on the context (e.g., pedestrian detection in a city vs. road sign detection on the highway).

Transformers have been applied to the field of AVs, for tasks such as sensor fusion [48], computer vision [49], decision making [50], and anomaly detection [51].

Autoencoders: Autoencoders [52] and Variational Autoencoders (VAEs) [53] are utilized primarily on tasks such as dimensionality reduction, feature extraction, and data generation. Autoencoders consist of two main elements: an encoder that transforms the input into a lower-dimensional latent space, and a decoder that reconstructs the input from the latent space. This architecture enables the model to learn low-dimensional representations of the input data such that AVs are able to extract salient features from high-dimensional data and sensor observations. Autoencoders have been used for tasks like anomaly detection [54], [55], and attack detection [56] in AVs.

On the other hand, VAEs extend the basic autoencoder model by incorporating probabilistic elements into the latent space. VAEs facilitate learning a distribution over the latent representations, which allows for more flexible generation of new data samples related to the training sample. VAEs have been used for path [57], [58] and trajectory [59], [60] prediction, and decision-making [61] tasks within AV environments.

Despite the promising potential of ANN models in FL settings, there are certain challenges to be addressed if FL is to be applied in AV environments. One of the most severe challenges is data heterogeneity and trustworthiness.

Data heterogeneity refers to variations in the structure, format, and distribution of data across diverse AVs. Such heterogeneity would significantly hinder the deployment as well as operation of collaborative model training. One major issue is the non-uniformity of data volume and quality [62], [63] which can cause uneven client contribution and participation in the FL process. This non-uniformity will then undermine the fairness and representativeness of the resulting global model. Moreover, heterogeneous data attributes (e.g. varying feature distributions, data modalities, and domain-specific features) may introduce compatibility problems within the process of model aggregating and combining local model updates. Such problems may impede the uniform integration of heterogeneous data sources into the global model, impacting its overall accuracy and generalization capability over heterogeneous AV environments adversely. Aside from heterogeneity, data reliability is yet another factor of major concern in the proper deployment of FL [64]. Since FL is based on multiple parties collaboratively training a global model without sharing local data, the trustworthiness of such local data inputs is important. If the information being delivered by cooperating devices is compromised or not trustworthy, whether due to attacks such as spoofing or abnormal sensor or communication channel behavior, it can contribute to the convergence of incorrect models, undermining the performance of the FL framework in AV environments. Additionally, issues of trust in both the shared model and local training data, especially across heterogeneous and potentially untrusted AV devices, compound privacy and security concerns. Therefore, it becomes essential to develop effective mechanisms for authenticating and ensuring the reliability of data originating from various sources. Only then can the data become reliable, when FL is able to develop its full potential towards enhancing the intelligence, security, and efficiency of AVs.

4.2.2. Federated Learning Security and Privacy

Although FL has received significant attention in recent years for its ability to preserve data privacy, FL alone cannot guarantee mitigation of privacy concerns due to the risk that model updates exchanged during the learning process might leak sensitive information. Furthermore, FL communications are susceptible to poisoning, gradient inversion, and spoofing attacks, to name a few [65]. Review of some FL privacy and security threats are available in [66], and [67]. Therefore, additional layers of security are required in order to further safeguard FL systems so that the integrity of the learning process is preserved, as well as the confidentiality of the data of the participants. To satisfy the mentioned goals, Homomorphic Encryption (HE) is a privacy method that is often applied in FL to prevent information leakage while sharing parameters between clients. This method encrypts parameters prior to performing an addition or multiplication operation, and the results of the FL model (e.g. accuracy, precision, recall) are the same as for a non-encrypted function [68]. In HE-supported FL, key-pair is synchronized on all end-nodes using a secure channel [69], [70]. Initially, all the end-nodes encrypt their local model gradients and send the ciphertexts to a central FL server. Then the server performs the model update using the encrypted gradients and then provides the aggregated model to the end-nodes. The benefit of using HE is that there is no need for decryption at the FL server.

HE is divided into three general categories namely partially HE, somewhat HE, and fully HE [71]. These types of HE offers flexibility of performing mathematical operations depending on the type used. For instance, partially HE accommodates only one type of mathematical operation, whereas full HE accommodates many different types of mathematical operations [72]. A couple of papers [73], [74], [75] have applied HE to secure FL models for AV environments. However, HE usually results in the use of extra computation and communication resources [72]. In AV environments, sensors like radar, Lidar, camera, GNSS, as well as ECU, typically operate with limited computational resources, and bounded storage capacity. Executing resource-consuming ML models and repeated training iterations can strain these scarce resources significantly, greatly compromising FL's efficiency and usability [76]. This difficulty is further exacerbated by the inclusion of privacy-preservation mechanisms such as HE, which, while a required measure for safeguarding personal data in FL, has large computational and communication overhead. HE-based FL systems need many times more computational resources and memory for computing encrypted data. These computations not only contribute to the local processing burden but also increase the size of model updates, leading to more bandwidth consumption. For the AVs, this additional overhead might result in longer training time, less complex and accurate ML models, and higher energy consumption, degrading the sustainability and scalability of FL within AV environments.

In addition, the local storage space of such devices may restrict the volume and intensity of data that can be made accessible to the local training process, reducing the effectiveness of the global model. In highly heterogeneous AV environments, device settings with very different capabilities, the construction of an FL framework in which the lowest-capability members are served without degrading overall system performance while preserving the security and data privacy of the FL framework is a challenging task.

4.2.3. Future Contributions

Although there is a huge potential in ANN models for FL settings, their actual achievement in the complete potential in AV environments requires a shift from hypothetical security layers to actual, efficient, and operationally viable FL frameworks. The following are the key factors for the development of an actually secure and privacy-preserving FL model for AVs. Ignoring these renders the very reason for FL collapse, and any other security approach becomes irrelevant.

Computation is constrained by design in AV environments. Radar, LiDAR, cameras, and GNSS sensors feed inputs to already heavily loaded ECUs with real-time processing needs. Traditional FL systems, especially HE-based ones, are too computationally intensive. Future work must cater to lightweight FL algorithms which are not only efficient computationally but also privacy-oriented in a way that they can be executed without taxing local resources. Without efficiency, model training becomes either futile or excessively delayed, making the effort of ensuring the process a waste of time. UNIGE has been exploring different lightweight ANN and FL architectures to solve this issue, and the results will be presented in the next version of this deliverable.

Optimization of Heterogeneous data

As mentioned earlier, data heterogeneity caused by differences in sensor settings, environments, driving styles, and hardware configurations is not a phenomenon, but the nature of AVs. Such heterogeneity results in asymmetric participation and varying data quality, threatening fairness, accuracy, and convergence of global FL models. Therefore, FL algorithms must be specifically designed for heterogeneous setups with provisions for fair contribution and robust model fusion. Again, acquiring an FL method that cannot handle heterogeneity is a waste because its outcomes will always be faulty.

Communication-Efficient Techniques

Communication bandwidth is another finite shared resource for AV networks. Ongoing transmission of large encrypted model updates, especially with HE, invites high communication cost and energy usage. Without optimization, it will create latency and resource limitations, reducing the useability of FL. Thus, developing communication-efficient techniques, such as update compression, and attention-based models, is crucial. These techniques ensure timely, reliable, and secure communication while maintaining the integrity of the FL process without exhausting bandwidth.

Robust and Secure Aggregation Mechanisms

As mentioned before, FL depends on the process of aggregating a series of AV updates, and therefore strong and secure aggregation is necessary. The process must be robust to poisoned or abnormal updates and furthermore resistant to attacks like model inversion and leakage of gradients. However, secure aggregation is only possible when the received model updates are representative and legitimate. Future research must focus on aggregation methods that can handle client-side trustworthiness, data disparity, and hostile action, with a focus on scalability and privacy. This includes exploring Differential Privacy (DP), and verifiable computation techniques.

Attack and Anomaly Management on Client-Reported Data

The performance of FL models depends entirely on the trustworthiness of client inputs. With an AV network, the risk of compromised or manipulated AVs, spoofed sensors, false updates, or even malfunctioning hardware can all threaten model accuracy. An active response, therefore, to managing anomalies (including forecasting, detection, and mitigation), and assessing trust is critically important. Future work must incorporate techniques such as behavioral modeling, detection of outliers, and adversarial testing in order to monitor and verify the validity of local updates on an ongoing basis. Otherwise, FL models will find themselves in suboptimal or even dangerous states, rendering any security or privacy guarantee void.

In the following Section, we will actually demonstrate how privacy preservation, and more specifically HE, impact directly on the external monitoring system of an AV.

4.3. Enhancing the security and privacy of external sensing system

4.3.1. Federated Deep Unrolling for LiDAR Super-Resolution Using Homomorphic Encryption

With the continuous progress made over the years, a lidar-centric perception system is expected to mature in terms of model-based processing algorithms while satisfying at the same time requirements for the majority of AVs such as precise localization and accurate mapping of unknown surroundings. AVs frequently operate in environments characterized by constant changes, posing challenges to the creation of consistent maps. For instance, self-driving cars must possess the capability to consistently locate legal parking spots and identify safe passenger exit points, even in previously unexplored locations that lack accurate mapping data. The emergence of adaptive federated optimization in the field of CAVs has the capacity to transform such Lidar Based Simultaneous Localization and Mapping (SLAM) solutions [77], [78], [79].

Building on this line of thought, federated learning [80] can serve as a continual learning methodology enabling collaboration between trusted agents and respecting at the same time privacy concerns. Here, we aim to combine federated learning methods with analytical and well-justified optimization-based methods, similar to the methods resented in Deliverable 3.1. This novel combination offers the advantages of both worlds: high performance due to the data offered by a number of cooperating agents as well as low computational complexity and explainable model architectures.

Federated Learning in Automotive Domain

Although the Federated Learning framework has been extensively explored in numerous disciplines, e.g., signal processing, medical processing, its application in the autonomous driving domain remains under-investigated [80], [81], [82]. The current body of literature contains a limited number of works that investigate the benefits of FL in autonomous driving. For instance, study [83] used the FL scheme to examine the object detection problem in autonomous driving scenes. Additionally works in [84], [85] proposed methods for predicting the wheel steering angle in AVs under the FL scenario. Our study differentiates from the existing literature in two important aspects. Firstly, we explore the novel problem of deep unrolling-based lidar super-resolution from a federated learning perspective, which has not been previously examined. By capitalizing on the distributed nature of federated learning, our approach enables the utilization of private lidar data gathered from diverse AVs operating in different environmental conditions in order to improve the lidar slam solutions. Secondly, and more importantly, we propose a novel federated learning scheme based on the proposed deep unrolling formulation. We argue that the well-justified structure of the deep unrolling model can be fully utilized by the Federated learning strategy.

4.3.2. Federated Deep Unrolling Method

In this Section, we formulate a novel Federated Deep Unrolling methodology (i.e., FL-DU). Inspired by the distributed parameter estimation approaches [86], we argue that the proposed FL framework can be expressed as an ATC strategy. More specifically:

Federated Deep Unrolling Framework

To mathematically establish the Federated Deep Unrolling (FL-DU) framework, we consider a network of N edge devices (or agents) participating in the learning process. Each device is identified by an index n within the set $N = 1,2, \dots, N$, and holds a local dataset $D_n = \{\boldsymbol{X}_n, \boldsymbol{Y}_n\}$. In this setup, $\boldsymbol{X}_n$ denotes the high-resolution range images, while $\boldsymbol{Y}_n$ represents the corresponding low-resolution range images. To simplify the notations, we assume that each local dataset is composed of one pair of high- and low-resolution range images.

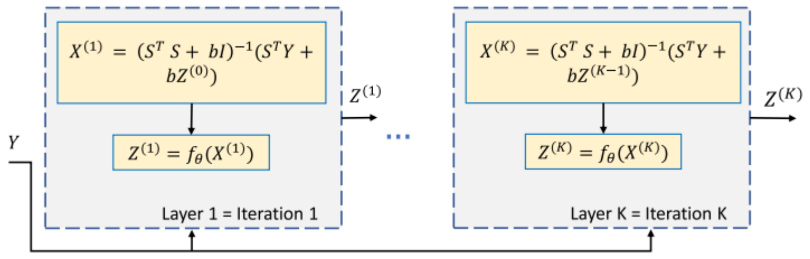

$X^{(1)} = (S^T S + bI)^{-1}(S^T Y + bZ^{(0)})$
$Z^{(1)} = f_\theta(X^{(1)})$
Layer 1 = Iteration 1
$Z^{(1)}$
Y
...
$X^{(K)} = (S^T S + bI)^{-1}(S^T Y + bZ^{(K-1)})$
$Z^{(K)} = f_\theta(X^{(K)})$
Layer K = Iteration K
$Z^{(K)}$

Figure 65: Proposed Deep unrolling model. In particular, a small number of iterations, say K, of the a local HQS solver in Eq. 40 are unrolled and treated as a deep learning architecture. Each iteration of the iterative solver is considered a unique layer of the proposed model, resulting in a K-layer deep learning architecture.

Each agent aims to solve a local optimization problem utilizing the proposed problem formulation for the Lidar super-resolution, defined as follows:

$$\underset{\boldsymbol{X}_n}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{Y}_n - \boldsymbol{S}\boldsymbol{X}_n\|_F^2 + \mu_n R_n(\boldsymbol{X}_n) \qquad \text{Eq. 36}$$

where $R_n(\cdot)$ denotes to the learnable regularizer corresponding to the n-th agent.

The fact that the local agents utilize only their local data to address the proposed optimization problem may produce a local regularizer (prior) that is not able to generalize well in various environmental conditions. Thus, this limitation may result in a local learnable regularizer ($R_n(\cdot)$) that is only limited to capture dependencies of the range images generated from the local distribution. To efficiently overcome this, the proposed federated deep unrolling framework allows agents to collaborate under the orchestration of a central server. Through this collaboration, they are able to learn a more robust regularizer, or prior, which exhibits greater generalization capabilities across diverse conditions.

The goal of the server in this context is to solve the sum of the local optimization problems, i.e.,

$$\sum_{n=1}^{N}\left(\frac{1}{2}\|\boldsymbol{Y}_n-\boldsymbol{S}\boldsymbol{X}_n\|_F^2+\mu_n R_n(\boldsymbol{X}_n)\right) = \sum_{n=1}^{N}\left(\frac{1}{2}\|\boldsymbol{Y}_n-\boldsymbol{S}\boldsymbol{X}_n\|_F^2\right)+\sum_{n=1}^{N}(\mu_n R_n(\boldsymbol{X}_n)) \qquad \text{Eq. 37}$$

The optimization problem presented in Eq. 37 consists of two terms. The first term, known as the data consistency term, requires access to each agent's private local datasets. This part is crucial in maintaining the accuracy of the optimization. However, direct access to this local private data may raise privacy concerns. The second term represents the sum of the learnable regularizers corresponding to each agent. These regularizers are expressed as neural networks that capture the underlying structure in the data. Importantly, while the regularizers are learned using local data, they don't expose sensitive information. This makes them suitable for sharing with the server, which facilitates global optimization without compromising privacy.

In light of this, the proposed federated deep unrolling framework can be expressed as an ATC strategy [87], taking into account the above optimization formulation. Given the interpretable structure of the deep unrolling model, the proposed framework consists of two steps: adaptation and combination.

In the adaptation step, each agent aims to solve the optimization problem defined in Eq. 36. This step is designed to solve the proposed local optimization problem using the deep unrolling strategy and adapting the respective DU models to the specific characteristics of the local datasets from each AV (agent).

In the combination step, the focus is on merging the local learnable regularizers obtained from the deep unrolling models. This process results in a more powerful and robust global regularizer that effectively incorporates the information gathered from a diverse range of Avs operating in different environmental conditions. By combining the local regularizers, the overall performance and generalization capabilities of the federated learning framework are enhanced.

The proposed approach provides the deep unrolling-based federated learning a clear and interpretable structure. The role of FL is to facilitate the merging of local learnable regularizers without compromising the privacy of individual datasets.

Adaptation Step

In the proposed approach, the adaptation step takes place within the local devices. Focusing on the devices' side, at the t-th communication round each device n aims to solve the following optimization scheme (see, also Eq. 36), i.e.,

$$\underset{\boldsymbol{X}_n}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{Y}_n - \boldsymbol{S}\boldsymbol{X}_n\|_F^2 + \mu_n R_n(\boldsymbol{X}_n) \quad \text{Eq. 38}$$

where $R_n(\cdot)$ denotes to the learnable regularizer corresponding to the n-th agent.

Each agent n employs the Half quadratic splitting (HQS) methodology to tackle the local optimization problem in Eq. 38, thus forming the following local objective function:

$$L_n = \frac{1}{2} \|\boldsymbol{Y}_n - \boldsymbol{S}\boldsymbol{X}_n\|_F^2 + \mu_n R_n(\boldsymbol{Z}_n) + \frac{b_n}{2} \|\boldsymbol{Z}_n - \boldsymbol{X}_n\|_F^2 \quad \text{Eq. 39}$$

The solution of this optimization problem consists of two interpretable modules that is that is the data consistency solution for estimating the high-resolution range image Eq. 40 and the denoising step in Eq. 41. Thus, at each communication round t, the local device n solves the following iteration map

$$\boldsymbol{X}_n{}^{(k+1)} = (\boldsymbol{S}^T\boldsymbol{S} + b\boldsymbol{I})^{-1}\left(\boldsymbol{S}^T\boldsymbol{Y}_n + b\boldsymbol{Z}_n{}^{(k)}\right) \quad \text{Eq. 40}$$

$$\boldsymbol{Z}_n{}^{(k+1)} = f_{\theta_n}\left(\boldsymbol{X}_n{}^{(k+1)}\right) \quad \text{Eq. 41}$$

Local Deep Unrolling model: However, instead of solving the above iterative map for a large number of iterations, each device employs the deep unrolling strategy, thus unrolling a small number of K iterations and creating a K-layer deep architecture, as depicted in Figure 65. Having formed the local deep-unrolling model the device n employs some version of the stochastic gradient descent to train it end-to-end using some loss function.

Combination Step

After all participating edge devices $n \in N$ have updated their local deep unrolling models, the next step is the combination step. The objective of this step is to learn an appropriate regularizer (prior) that captures the unique characteristics of the range images by utilizing local information from the agents. Due to the structure of the local DU models, the devices only upload to the server the neural network $f_{\theta_n}(\cdot)$ responsible for the denoising process in Eq. 41. Subsequently, the server combines all the local denoisers using a fusion rule, as follows:

$$f_{\theta_g} = \sum_{n=1}^{N} a_n f_{\theta_n} \quad \text{Eq. 42}$$

where f_{θ_g} denotes the global denoiser (regularizer) and a_n stand for combination weights. Consequently, the server transmits the global denoiser back to the local devices. These devices initialize the denoisers of the local deep unrolling models (i.e., Eq. 41) with the received global denoiser. This procedure is repeated for T communication rounds. Hence, the FL-DU algorithm can be written as an agent Adaptation step, which involves the local data consistency term (Eq. 43) and the local denoiser (Eq. 44) (solved by unrolling these equations using the proposed deep unrolling strategy) and a Combination step (Eq. 45):

$$\boldsymbol{X}_n = (\boldsymbol{S}^T\boldsymbol{S} + b_n\boldsymbol{I})^{-1}\left(\boldsymbol{S}^T\boldsymbol{Y}_n + b\boldsymbol{Z}_n{}^{(K)}\right) \qquad \text{Eq. 43}$$

$$\boldsymbol{Z}_n = f_{\theta_n}(\boldsymbol{X}_n) \qquad \text{Eq. 44}$$

$$f_{\theta_n} = \sum_{n=1}^{N} a_n f_{\theta_n} \qquad \text{Eq. 45}$$

Figure 66 illustrates the proposed FL-DU framework.

4.3.3. Experiments and Evaluation Study

To evaluate the effectiveness of the proposed federated deep unrolling framework, a series of experiments were carried out in the context of LiDAR super-resolution. The aim was to upscale data from a 16-channel LiDAR to a 64-channel LiDAR by a factor of 4. Furthermore, we assessed the benefits of our proposed method on a LiDAR SLAM system based on the LeGO-LOAM algorithm [87]. The LiDAR SLAM experiments were conducted on a developed simulation framework.

Dataset

Regarding the training, we employed the same dataset presented in study [88]. A 64-channel lidar, OS-1-64, was simulated in the CARLA Town 1 and Town 2 scenes, matching the Ouster dataset field of view (33.2°). For the same scene, a 16-channel lidar, OS-1-16, was simulated to generate low-resolution point clouds. Both high- and low-resolution point clouds were projected onto range images [89], resulting in 7000 pairs of 64x1024 and 16x1024 images. The images were then normalized to a range of 0-1 for training. Testing Data: To validate the performance of the proposed FL architecture, the real-world Ouster lidar dataset was utilized. This dataset comprises 8825 scans collected over a 15-minute drive in San Francisco using an OS-1-64 3D lidar sensor.

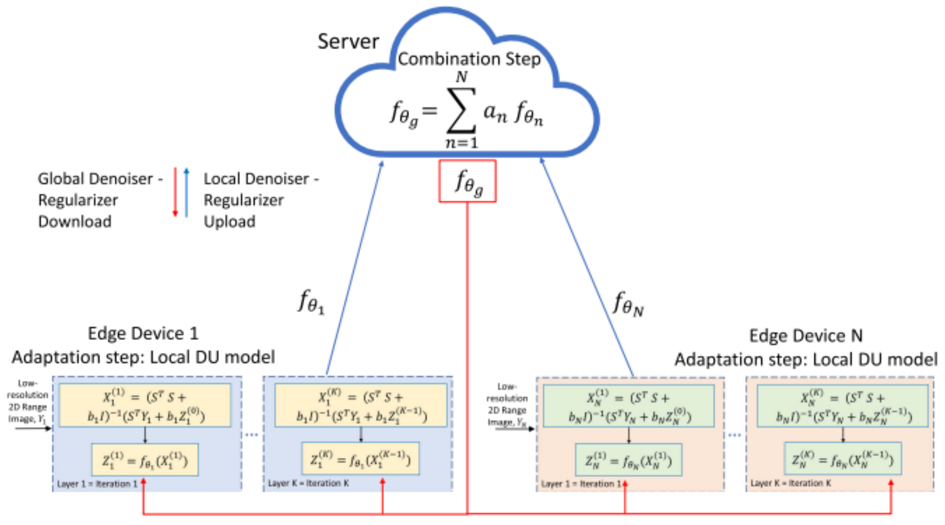

Server
Combination Step
$f_{\theta_g} = \sum_{n=1}^{N} a_n f_{\theta_n}$
f_{θ_g}
Global Denoiser - Regularizer Download
Local Denoiser - Regularizer Upload
f_{θ_1}
f_{θ_N}
Edge Device 1
Adaptation step: Local DU model
Edge Device N
Adaptation step: Local DU model
Low-resolution 2D Range Image, Y_1
Low-resolution 2D Range Image, Y_N
Layer 1 = Iteration 1
Layer K = Iteration K

Figure 66: Proposed end-to-end Deep unrolling-based Federated Learning approach. This strategy contains two key parts i.e., the adaptation and the combination step. In the adaptation step, each agent updates its local deep unrolling model using the local data consistency term and the local denoiser. This process ensures adaptation to the specific characteristics of the agent's dataset. In the Combination part, the server combines the outputs of all local denoisers. This creates a global denoiser (regularizer) that captures knowledge from diverse local datasets.

Implementation Details

Federated Learning Scenario: In our experiment, we examined a network made up of AVs, each functioning as a distinct agent. The training data previously mentioned was partitioned into 5 unique blocks. Each block was sourced from different locations within the CARLA simulator, thereby representing varied environmental conditions.

During the local training on the edge devices, we utilized 5 epochs with a learning rate of 1e − 04 and a batch size of 6. Additionally, we determined the number of communication rounds between the central server and the edge devices to be T = 50. The local models were trained using Adam Optimizer.

3) Secured Federated Learning: In literature, several studies have explored the use of privacy-preserving techniques such as Homomorphic Encryption [90], [91], [92] within the realm of classical federated learning. Our goal is to demonstrate how security mechanisms, such as Homomorphic encryption, can be easily integrated to enhance the security of the proposed Federated deep unrolling method, without negatively affecting the models performance. To this end, we used homomorphic encryption based on the Tenseal library [93]. In the context of the proposed federated deep unrolling system, multiple vehicles collaborate to improve a global model during the combination step, while keeping their training data local. However, sharing information between these agents or with a central server can lead to potential privacy

breaches. Herein lies the importance of using HE. It enables each client to encrypt their trained denoiser (prior) parameters before sending them to a central server for aggregation. Thus, the agents need to encrypt only the denoising step in Eq. 44 from their local deep unrolling models. Due to the special properties of HE, the server can perform computations directly on these encrypted parameters to generate an encrypted global model. This method ensures that the server, while able to aggregate the model updates and further distribute them, never has access to the raw data or individual model parameters, maintaining the privacy of each participant in the federated deep unrolling process. Finally, even though the aggregated encrypted model is then decrypted, the privacy is still preserved since vehicles have access only to the aggregated model not the individual ones.

LiDAR Super-Resolution Performance on Raw Data: Federated Learning Approach

To thoroughly evaluate the advantages and possibilities of our proposed federated deep unrolling framework, called FL-DU with and without the homomorphic encryption part, we compare it with the following approaches:

- Centralized-DU: This represents our deep unrolling model used in a centralized context, where a central server gathers data from distributed edge devices to train the lidar super-resolution model.
- Centralized-SRAE: Since the SRAE method [88] was found to be the top-performing competitor, we include it in our comparison.
- FL-SRAE: Additionally for completeness purposes, we also consider a straightforward federated learning scenario [94], where edge devices utilized the deep neural network proposed in study [88].

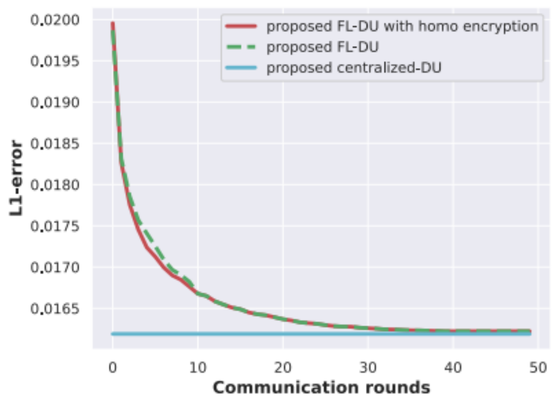


Figure 67: L1 loss of the derived global model from the proposed deep unrolling FL scheme with and without homomorphic encryption vs communication rounds along with the best accuracy achieved by the centralized deep unrolling model.

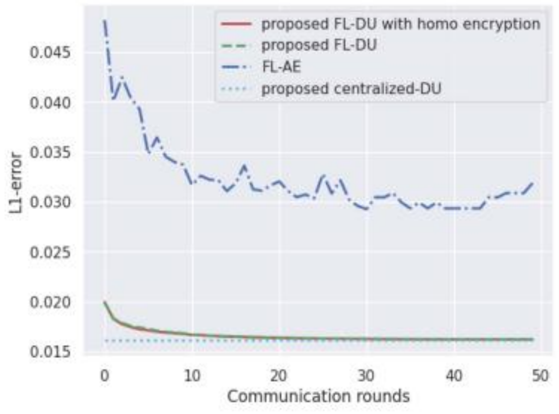


Figure 68: Loss of the derived global model from the proposed deep unrolling FL scheme with and without homomorphic encryption vs communication rounds against the classical federated learning framework with the SRAE model denoted as FL-SRAE.

Table 11: Quantitative Results- Federated Learning

Dataset	Method	Data training size	L1 loss
	proposed FL-DU with encryption	700 per client	0.0211
	proposed FL-DU	700 per client	0.0210
	FL-SRAE	700 per client	0.0357
Ouster	proposed centralized-DU	7000	0.0208
	centralized-SRAE	7000	0.0214

Comparison with the centralized methods: As we can see from Figure 67 and Table 11 the proposed FL-DU method is able to achieve similar performance against the centralized solution. Crucially, the proposed method necessitates only a limited number of communication rounds between the server and local agents, along with a mere five epochs of local training per round, to converge effectively to the centralized solution. Another important aspect that stems from the proposed federated unrolling strategy is the fact that we can incorporate any privacy preserving strategy. Interestingly, the FL-DU with the homomorphic encryption achieves the same convergence behavior as compared to the FL-DU without the encryption part.

Comparison proposed FL-DU with the federated learning methods: As illustrated in Figure 68 and Table 11, the proposed FL-DU method considerably outperforms the comparative federated learning approach that uses a state-of-the-art deep learning model. This superiority is observed in both accuracy and convergence rate. Notably, our method achieves similar results to the centralized solution while requiring fewer communication rounds.

Impact of LiDAR Super-Resolution on LiDAR-based SLAM Approaches

In order to thoroughly assess the effectiveness of our proposed deep unrolling model, along with the federated learning approach, we examined its applicability in a real-world automotive scenario. To do this, we utilized the LeGO-LOAM [87] system, which is a Lidar based SLAM mechanism that offers real-time six-degree-of-freedom pose estimation and a generated 3D map. We tested it on two sequences from the Ouster dataset:

- The first sequence consists of 2600 consecutive scans, representing a relatively simple trajectory followed by the vehicle.
- The second sequence is composed of 6000 scans that correspond to a more challenging trajectory with short, closely spaced loops.

Table 12: LiDAR SLAM: Absolute Pose Error w.r.t Translation Part (m).

	Ouster: 2600 scans					**Ouster: 6000 scans**				
Metrics	Lidar-16	centralized SRAE	proposed centralized-DU	proposed FL-DU	FL-SRAE	Lidar-16	centralized SRAE	proposed centralized-DU	proposed FL-DU	FL-SRAE
Mean	6.84	15.97	**3.86**	4.39	39.2	299.90	48.29	26.06	31.54	110.47
Mse	8.52	18.52	**4.42**	4.74	35.12	316.82	61.73	27.85	33.20	135.78
max	23.61	43.04	15.69	**10.02**	71.60	479.02	183.42	51.33	52.15	389.45

To investigate the influence of the Super Resolution (SR) approach on such a SLAM system, we conducted a performance comparison of the LeGO-LOAM algorithm using for distinct inputs:

- High-resolution 3D point clouds reconstructed with the proposed centralized-DU method
- High-resolution 3D point clouds reconstructed using our proposed FL-DU approach. In this case, we solely utilized the federated learning scenario with homomorphic encryption, as our findings demonstrated that it achieved practically the same performance as the corresponding federated learning scenario without the encryption part. In more detail, the point clouds were reconstructed using the derived model from the 50-th communication round of the FL-DU framework.
- low-resolution 3D point clouds generated by a 16-channel lidar sensor
- High-resolution 3D point clouds reconstructed using the centralized SRAE method [88].
- High-resolution 3D point clouds reconstructed using the Federated Learning method that uses the deep learning model of the SRAE method [94].

For our analysis, we employed error metrics from earlier studies [95], [96]. The results, including the output metrics, the cumulative distribution function (CDF) of Absolute Pose Error (APE) translation, and trajectory heatmaps, are presented in Table 12. The reference pose (trajectory) for these results is derived from a 64-channel lidar. In our analysis, we observe that both the proposed centralized-DU and FL-DU methods consistently outperform the 16-channel lidar and the SRAE [94]. across all the examined trajectories. This highlights the superior accuracy of our reconstructed 64-channel lidar data compared to the state-of-the-art SRAE method [88] and the FL-SRAE approach. Additionally, although the 16-channel lidar provides satisfactory results during the simple trajectory (2600 scans), in the more challenging trajectory (6000 scans), which contains close loops, the 16-channel lidar is not able to follow the reference trajectory. This can be justified by the fact that the LeGO-LOAM method relies on the availability of the edge and planar features to estimate the vehicle transformation, and thus it fails to generate robust features from the sparse point cloud derived from the 16-channel lidar.

In comparison to the traditional federated learning approach (i.e., FL-SRAE), our proposed Federated Deep Unrolling framework consistently demonstrates superior results across both trajectories. It’s important to note that the local models used in the FL-SRAE approach consist of more than 30 million parameters. This not only imposes significant communication constraints but also necessitates an extensive and diverse set of training examples for these models, which local agents often lack, resulting in subpar performance.

4.4. Concluding remarks

This section has outlined the critical role of privacy and data trustworthiness in the development and deployment of AV systems, considered in the frame of AutoTRUST. As AVs increasingly rely on high-resolution sensor data and distributed machine learning techniques, such as Federated Learning, ensuring secure, ethical, and efficient data handling becomes imperative. To this end, it has been examined how FL can address privacy concerns by enabling collaborative model training without sharing raw data, while also highlighting the limitations posed by data heterogeneity, trustworthiness, and computational constraints. The integration of Homomorphic Encryption further strengthens privacy but introduces significant resource demands that must be carefully managed, especially in resource-limited AV environments. Moreover, the section has introduced a novel Federated Deep Unrolling framework, combining optimization-based modelling with deep learning in a privacy-preserving, communication-efficient manner. By aligning technical innovation with ethical and legal safeguards, AutoTRUST contributes to a future where autonomous systems are not only intelligent but also trusted and accountable.

Co-funded by the European Union
Project funded by
Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra
Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI
KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

5. Conclusion

This deliverable has presented a comprehensive overview of the design, implementation, and initial validation of the advanced internal and external sensing systems developed within the AutoTRUST project. The solutions detailed herein demonstrate significant progress in realizing safe, user-centric, and resilient autonomous mobility, addressing both technical innovation and practical deployment challenges.

The internal sensing system integrates advanced multi-modal perception for real-time monitoring of occupant state, behavior, and comfort, including robust approaches to driver distraction detection, emotion recognition, and in-cabin environmental quality. The adoption of privacy-preserving methodologies and efficient edge computing ensures that user data is protected without compromising performance or responsiveness.

The external sensing system introduces reliable and interpretable scene understanding, with capabilities for precise road condition assessment, hazard detection, and cooperative awareness through V2X communication. The use of distributed and federated learning frameworks further enhances system robustness and data privacy, enabling vehicles to collaboratively improve perception and localization without sharing raw data.

A major strength of the work presented is the holistic integration of privacy, security, and ethical considerations throughout all system layers. Advanced encryption, secure aggregation, and local data processing are combined with ongoing risk assessment to support regulatory compliance and user trust. The validated architectures and methodologies described in this deliverable provide a solid foundation for the next stages of the project, including large-scale pilot deployments, further integration with user feedback, and continued optimization for real-world conditions. Moving forward, the focus will be on expanding evaluation in diverse operational scenarios, strengthening the adaptability of the sensing modules, and ensuring alignment with the evolving needs of all road users.

In summary, D3.2 sets a new benchmark for advanced sensing in connected and automated mobility, delivering modular, scalable, and trustworthy solutions that underpin the AutoTRUST project's vision of inclusive and reliable autonomous transport systems. Finally, the second and final version of the sensing technologies, along with associated data privacy and trustworthiness aspects, will be described in D3.4 "Advanced internal and external sensing system.v2", due to M29.

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI
KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

References

[1] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment Using Multi-task Cascaded Convolutional Networks," *IEEE Signal Process Lett*, vol. 23, no. 10, pp. 1499–1503, 2016.

[2] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, MA, USA, 2015.

[3] A. K. Roy, H. K. Kathania, A. Sharma, A. Dey, and M. S. A. Ansari, "ResEmoteNet: Bridging Accuracy and Loss Reduction in Facial Emotion Recognition," *ArXiv*, 2024.

[4] A. Mollahosseini, B. Hasani, and M. H. Mahoor, "AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild," *IEEE Trans Affect Comput*, vol. 10, no. 1, pp. 18–31, 2019.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016.

[6] J. Chen, X. Qiu, C. Ding, and Y. Wu, "SAR image classification based on spiking neural network through spike-time dependent plasticity and gradient descent," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 188, pp. 109–124, 2022, doi: https://doi.org/10.1016/j.isprsjprs.2022.03.021.

[7] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Reliable detection of audio events in highly noisy environments," *Pattern Recognit Lett*, vol. 65, pp. 22–28, Nov. 2015, doi: 10.1016/j.patrec.2015.06.026.

[8] M. M. Rashid, G. Li, and C. Du, "Nonspeech7k dataset: Classification and analysis of human non-speech sound," *IET Signal Processing*, vol. 17, no. 6, Jun. 2023, doi: 10.1049/sil2.12233.

[9] T. Khan, "A Deep Learning Model for Snoring Detection and Vibration Notification Using a Smart Wearable Gadget," *Electronics (Basel)*, vol. 8, no. 9, p. 987, Sep. 2019, doi: 10.3390/electronics8090987.

[10] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics,*

Speech and Signal Processing (ICASSP), IEEE, Apr. 2015, pp. 5206–5210. doi: 10.1109/ICASSP.2015.7178964.

[11] J. Salamon, C. Jacoby, and J. P. Bello, “A Dataset and Taxonomy for Urban Sound Research,” in *Proceedings of the 22nd ACM international conference on Multimedia*, New York, NY, USA: ACM, Nov. 2014, pp. 1041–1044. doi: 10.1145/2647868.2655045.

[12] C. A. Dim, N. C. S. Neto, and J. M. de Morais, “HornBase: An audio dataset of car horns in different scenarios and positions,” *Data Brief*, vol. 55, p. 110678, Aug. 2024, doi: 10.1016/j.dib.2024.110678.

[13] J. F. Gemmeke *et al.*, “Audio Set: An ontology and human-labeled dataset for audio events,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Mar. 2017, pp. 776–780. doi: 10.1109/ICASSP.2017.7952261.

[14] A. Grattafiori *et al.*, “The Llama 3 Herd of Models,” Nov. 2024.

[15] T. B. Chang, J. J. Sheu, J. W. Huang, Y. S. Lin, and C. C. Chang, “Development of a CFD model for simulating vehicle cabin indoor air quality,” *Transp Res D Transp Environ*, vol. 62, pp. 433–440, 2018.

[16] M. das Neves Almeida, A. A. de Paula Xavier, A. O. Michaloski, and A. L. Soares, “Thermal comfort in bus cabins: A review of parameters and numerical investigation,” in *Occupational and Environmental Safety and Health II*, 2020, pp. 499–506.

[17] B. Pirouz, D. Mazzeo, S. A. Palermo, S. N. Naghib, M. Turco, and P. Piro, “CFD investigation of vehicle’s ventilation systems and analysis of ACH in typical airplanes, cars, and buses,” *Sustainability*, vol. 13, no. 12, p. 6799, 2021.

[18] P. Nandi, A. Mishra, P. Kedia, and M. Rao, “Design of a real-time autonomous in-cabin sensory system to detect passenger anomaly,” in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, Oct. 2020, pp. 202–206.

[19] A. Singh, M. Islam, and N. Dinh, “Forecasting Indoor Air Quality Using Machine Learning Models,” in *2024 IEEE International Conference on Consumer Electronics (ICCE)*, IEEE, Jan. 2024, pp. 1–6. doi: 10.1109/ICCE59016.2024.10444387.

[20] N. Hao et al., “Decentralized Cooperative Localization: A Communication-Efficient Dual-Fusion Consistent Approach,” *IEEE Robot Autom Lett*, vol. 10, no. 1, pp. 636–643, 2025, doi: 10.1109/LRA.2024.3511413.

[21] G. Revach et al., “KalmanNet: Neural Network Aided Kalman Filtering for Partially Known Dynamics,” *IEEE Trans. on Signal Processing*, vol. 70, pp. 1532–1547, 2022, doi: 10.1109/TSP.2022.3158588.

[22] N. Piperigkos, A. Gkillas, C. Anagnostopoulos, and A. S. Lalos, “Federated Data-Driven Kalman Filtering for State Estimation,” in *2024 IEEE 26th International Workshop on Multimedia Signal Processing*, 2024, pp. 1–6. doi: 10.1109/MMSP61759.2024.10743751.

[23] R. He, Y. Shan, and K. Huang, “Robust Cooperative Localization With Failed Communication and Biased Measurements,” *IEEE Robot Autom Lett*, vol. 9, no. 3, pp. 2997–3004, 2024, doi: 10.1109/LRA.2024.3362682.

[24] N. Piperigkos, C. Anagnostopoulos, A. S. Lalos, and K. Berberidis, “Extending Online 4D Situational Awareness in Connected and Automated Vehicles,” *IEEE Trans. on Intelligent Vehicles*, vol. 9, no. 8, pp. 5316–5335, 2024, doi: 10.1109/TIV.2023.3335605.

[25] C. Anagnostopoulos, C. Koulamas, A. Lalos, and C. Stylios, “Open-Source Integrated Simulation Framework for Cooperative Autonomous Vehicles,” in *2022 11th Mediterranean Conference on Embedded Computing*, 2022, pp. 1–4. doi: 10.1109/MECO55406.2022.9797115.

[26] P. Yang et al., “Multi-Sensor Multi-Vehicle (MSMV) Localization and Mobility Tracking for Autonomous Driving,” *IEEE Trans. on Vehicular Technology*, vol. 69, no. 12, pp. 14355–14364, 2020, doi: 10.1109/TVT.2020.3031900.

[27] X. Weng et al., “3D Multi-Object Tracking: A Baseline and New Evaluation Metrics,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10359–10366. doi: 10.1109/IROS45743.2020.9341164.

[28] C. Luo et al., “Exploring Simple 3D Multi-Object Tracking for Autonomous Driving,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 10468–10477. doi: 10.1109/ICCV48922.2021.01032.

[29] O. Sorkine, “Laplacian mesh processing,” *Eurographics (State of the Art Reports)*, vol. 4, no. 4, p. 1, 2005.

[30] H.-K. Chiu et al., “Probabilistic 3d multi-object cooperative tracking for autonomous driving via differentiable multi-sensor kalman filter,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 18458–18464.

[31] N. Piperigkos et al., “Graph Laplacian Processing based multi-modal localization backend for robots and autonomous Systems,” *IEEE Trans Cogn Dev Syst*, 2024.

Co-funded by the European Union

Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI

KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

[32] R. Xu et al., “V2V4Real: A Real-World Large-Scale Dataset for Vehicle-to-Vehicle Cooperative Perception,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 13712–13722. doi: 10.1109/CVPR52729.2023.01318.

[33] A. Lang et al., “PointPillars: Fast Encoders for Object Detection From Point Clouds,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12689–12697. doi: 10.1109/CVPR.2019.01298.

[34] R. Xu et al., “CoBEVT: Cooperative bird’s eye view semantic segmentation with sparse transformers,” *arXiv preprint arXiv:2207.02202*, 2022.

[35] P. Voigt and A. von dem Bussche, *The EU General Data Protection Regulation (GDPR)*. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-57959-7.

[36] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, “A survey on federated learning: challenges and applications,” *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 2, pp. 513–535, 2023.

[37] R. Bin Issa *et al.*, “Double deep Q-learning and faster R-Cnn-based autonomous vehicle navigation and obstacle avoidance in dynamic environment,” *Sensors*, vol. 21, no. 4, p. 1468, 2021.

[38] J. Fan, B. Gao, Q. Ge, Y. Ran, J. Zhang, and H. Chu, “SegTransConv: Transformer and CNN hybrid method for real-time semantic segmentation of autonomous vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 2, pp. 1586–1601, 2023.

[39] Z. Meng, J. Wu, S. Zhang, R. He, and B. Ge, “Interaction-aware trajectory prediction for autonomous vehicle based on LSTM-MLP model,” in *Proceedings of KES-STS International Symposium*, Singapore: Springer Nature Singapore, 2023, pp. 91–99.

[40] S. Yoon and D. Kum, “The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles,” in *2016 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2016, pp. 1307–1312.

[41] Y. Jeong, “Interactive lane keeping system for autonomous vehicles using LSTM-RNN considering driving environments,” *Sensors*, vol. 22, no. 24, p. 9889, 2022.

[42] L. Li, W. Zhao, C. Xu, C. Wang, Q. Chen, and S. Dai, “Lane-change intention inference based on RNN for autonomous driving on highways,” *IEEE Trans Veh Technol*, vol. 70, no. 6, pp. 5499–5510, 2021.

[43] K. Min, D. Kim, J. Park, and K. Huh, "RNN-based path prediction of obstacle vehicles with deep ensemble," *IEEE Trans Veh Technol*, vol. 68, no. 10, pp. 10252–10256, 2019.

[44] Z. Bai, B. Cai, W. ShangGuan, and L. Chai, "Deep learning based motion planning for autonomous vehicle using spatiotemporal LSTM network," in *2018 Chinese Automation Congress (CAC)*, IEEE, 2018, pp. 1610–1614.

[45] Z. Zhong, R. Li, J. Chai, and J. Wang, "Autonomous Vehicle Trajectory Combined Prediction model based on C-LSTM," in *2021 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*, IEEE, Oct. 2021, pp. 1–6. doi: 10.1109/iFUZZY53132.2021.9605087.

[46] A. Vaswani *et al.*, "Attention is all you need," *Adv Neural Inf Process Syst*, vol. 30, 2017.

[47] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[48] A. Abdulmaksoud and R. Ahmed, "Transformer-based sensor fusion for autonomous vehicles: A comprehensive review," *IEEE Access*, 2025.

[49] Q. V Lai-Dang, "A survey of vision transformers in autonomous driving: Current trends and future directions," *arXiv preprint arXiv:2403.07542*, 2024.

[50] H. Gao *et al.*, "A spatial–temporal predictive transformer network for level-3 autonomous vehicle decision-making," *IEEE Trans Neural Netw Learn Syst*, 2024.

[51] H. Min *et al.*, "Toward interpretable anomaly detection for autonomous vehicles with denoising variational transformer," *Eng Appl Artif Intell*, vol. 129, p. 107601, 2024.

[52] P. Li, Y. Pei, and J. Li, "A comprehensive survey on design and application of autoencoder in deep learning," *Appl Soft Comput*, vol. 138, p. 110176, 2023.

[53] L. Pinheiro Cinelli, M. Araújo Marins, E. A. da Silva, and S. Lima Netto, "Variational autoencoder," in *Variational Methods for Machine Learning with Applications to Deep Networks*, Springer International Publishing, 2021, pp. 111–149.

[54] A. Abdulmaksoud and R. Ahmed, "Transformer-Based Sensor Fusion for Autonomous Vehicles: A Comprehensive Review," *IEEE Access*, vol. 13, pp. 41822–41838, 2025, doi: 10.1109/ACCESS.2025.3545032.

[55] B. Wang, W. Li, and Z. H. Khattak, "Anomaly detection in connected and autonomous vehicle trajectories using LSTM autoencoder and Gaussian mixture model," *Electronics (Basel)*, vol. 13, no. 7, p. 1251, 2024.

Co-funded by the European Union

Project funded by

[56] F. W. Alsaade and M. H. Al-Adhaileh, “Cyber attack detection for self-driving vehicle networks using deep autoencoder algorithms,” *Sensors*, vol. 23, no. 8, p. 4086, 2023.

[57] H. Min *et al.*, “Toward interpretable anomaly detection for autonomous vehicles with denoising variational transformer,” *Eng Appl Artif Intell*, vol. 129, p. 107601, Mar. 2024, doi: 10.1016/j.engappai.2023.107601.

[58] D. N. Jagadish, A. Chauhan, and L. Mahto, “Conditional variational autoencoder networks for autonomous vehicle path prediction,” *Neural Process Lett*, vol. 54, no. 5, pp. 3965–3978, 2022.

[59] L. P. Cinelli, M. A. Marins, E. A. Barros da Silva, and S. L. Netto, *Variational Methods for Machine Learning with Applications to Deep Networks*. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-70679-1.

[60] G. Oh and H. Peng, “CVAE-H: Conditionalizing Variational Autoencoders via Hypernetworks and Trajectory Forecasting for Autonomous Driving,” Jan. 2022, Accessed: Jul. 31, 2025. [Online]. Available: https://arxiv.org/pdf/2201.09874

[61] S. Wang, Z. Wang, X. Wang, Q. Liang, and L. Meng, “Intelligent vehicle driving decision-making model based on variational AutoEncoder network and deep reinforcement learning,” *Expert Syst Appl*, vol. 268, p. 126319, 2025.

[62] X. Yuan *et al.*, “Fedstn: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 8, pp. 8738–8748, 2022.

[63] M. Xia, D. Jin, and J. Chen, “Short-term traffic flow prediction based on graph convolutional networks and federated learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 1191–1203, 2022.

[64] L. Xing, K. Wang, H. Wu, H. Ma, and X. Zhang, “FL-MAAE: An intrusion detection method for the Internet of Vehicles based on federated learning and memory-augmented autoencoder,” *Electronics (Basel)*, vol. 12, no. 10, p. 2284, 2023.

[65] N. Bouacida and P. Mohapatra, “Vulnerabilities in federated learning,” *IEEE Access*, vol. 9, pp. 63229–63249, 2021.

[66] Z. Ju, H. Zhang, X. Li, X. Chen, J. Han, and M. Yang, “A survey on attack detection and resilience for connected and automated vehicles: From vehicle dynamics and control perspective,” *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 4, pp. 815–837, 2022.

Co-funded by the European Union
Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI
KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

[67] M. Alazab, S. P. Rm, P. K. R. Maddikunta, T. R. Gadekallu, and Q. V Pham, "Federated learning for cybersecurity: Concepts, challenges, and future directions," *IEEE Trans Industr Inform*, vol. 18, no. 5, pp. 3501–3509, 2021.

[68] V. P. Chellapandi, L. Yuan, C. G. Brinton, S. H. Żak, and Z. Wang, "Federated learning for connected and automated vehicles: A survey of existing approaches and challenges," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 119–137, 2023.

[69] C. Zhang *et al.*, *{BatchCrypt}: Efficient Homomorphic Encryption for {Cross-Silo} Federated Learning*. {USENIX} Association, 2020. Accessed: Jul. 31, 2025. [Online]. Available: https://www.usenix.org/conference/atc20

[70] S. Hardy *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," Nov. 2017, Accessed: Jul. 31, 2025. [Online]. Available: https://arxiv.org/pdf/1711.10677

[71] R. Shrestha and S. Kim, "Integration of IoT with blockchain and homomorphic encryption: Challenging issues and opportunities," in *Advances in Computers*, vol. 115, Elsevier, 2019, pp. 293–331.

[72] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, "Federated Learning for the Internet of Things: Applications, Challenges, and Opportunities," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 24–29, 2022, doi: 10.1109/IOTM.004.2100182.

[73] M. Han, K. Xu, S. Ma, A. Li, and H. Jiang, "Federated learning-based trajectory prediction model with privacy preserving for intelligent vehicle," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 10861–10879, Dec. 2022, doi: 10.1002/INT.22987.

[74] Y. Li, X. Tao, X. Zhang, J. Liu, and J. Xu, "Privacy-Preserved Federated Learning for Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8423–8434, 2022, doi: 10.1109/TITS.2021.3081560.

[75] M. A. Mohammed *et al.*, "Homomorphic federated learning schemes enabled pedestrian and vehicle detection system," *Internet of Things*, vol. 23, p. 100903, Oct. 2023, doi: 10.1016/J.IOT.2023.100903.

[76] K. Xie *et al.*, "Efficient federated learning with spike neural networks for traffic sign recognition," *IEEE Trans Veh Technol*, vol. 71, no. 9, pp. 9980–9992, 2022.

[77] S. Wang *et al.*, "Federated Deep Learning Meets Autonomous Vehicle Perception: Design and Verification," *IEEE Netw*, pp. 1–10, 2022, doi: 10.1109/MNET.104.2100403.

Co-funded by the European Union

Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI

KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology

[78] A. Eskandarian, C. Wu, and C. Sun, “Research Advances and Challenges of Autonomous and Connected Ground Vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 683–711, 2021, doi: 10.1109/TITS.2019.2958352.

[79] L. Luo, S.-Y. Cao, Z. Sheng, and H.-L. Shen, “LiDAR-Based Global Localization Using Histogram of Orientations of Principal Normals,” *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 771–782, 2022, doi: 10.1109/TIV.2022.3169153.

[80] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, “A survey on federated learning,” *Knowl Based Syst*, vol. 216, p. 106775, 2021, doi: https://doi.org/10.1016/j.knosys.2021.106775.

[81] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, “Federated Learning for the Internet of Things: Applications, Challenges, and Opportunities,” *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 24–29, 2022, doi: 10.1109/IOTM.004.2100182.

[82] Y. Tian, J. Wang, Y. Wang, C. Zhao, F. Yao, and X. Wang, “Federated Vehicular Transformers and Their Federations: Privacy-Preserving Computing and Cooperation for Autonomous Driving,” *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 456–465, 2022, doi: 10.1109/TIV.2022.3197815.

[83] D. Jallepalli, N. C. Ravikumar, P. V. Badarinath, S. Uchil, and M. A. Suresh, “Federated Learning for Object Detection in Autonomous Vehicles,” in *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*, 2021, pp. 107–114. doi: 10.1109/BigDataService52369.2021.00018.

[84] H. Zhang, J. Bosch, and H. H. Olsson, “End-to-End Federated Learning for Autonomous Driving Vehicles,” in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8. doi: 10.1109/IJCNN52387.2021.9533808.

[85] A. Nguyen *et al.*, “Deep Federated Learning for Autonomous Driving,” in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1824–1830. doi: 10.1109/IV51971.2022.9827020.

[86] J. Chen and A. H. Sayed, “Diffusion Adaptation Strategies for Distributed Optimization and Learning Over Networks,” *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, 2012, doi: 10.1109/TSP.2012.2198470.

[87] T. Shan and B. Englot, “LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain,” in *2018 IEEE/RSJ International Conference on*

Intelligent Robots and Systems (IROS), 2018, pp. 4758–4765. doi: 10.1109/IROS.2018.8594299.

[88] T. Shan, J. Wang, F. Chen, P. Szenher, and B. Englot, “Simulation-based lidar super-resolution for ground vehicles,” *Rob Auton Syst*, vol. 134, p. 103647, 2020, doi: https://doi.org/10.1016/j.robot.2020.103647.

[89] L. Fan, X. Xiong, F. Wang, N. Wang, and Z. Zhang, “RangeDet: In Defense of Range View for LiDAR-Based 3D Object Detection,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 2918–2927.

[90] Y. Aono, T. Hayashi, L. Wang, S. Moriai, and others, “Privacy-preserving deep learning via additively homomorphic encryption,” *IEEE transactions on information forensics and security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[91] J.-W. Lee *et al.*, “Privacy-preserving machine learning with fully homomorphic encryption for deep neural network,” *IEEE Access*, vol. 10, pp. 30039–30054, 2022.

[92] B. Pulido-Gaytan *et al.*, “Privacy-preserving neural networks with homomorphic encryption: C hallenges and opportunities,” *Peer Peer Netw Appl*, vol. 14, no. 3, pp. 1666–1691, 2021.

[93] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, “TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption,” 2021.

[94] A. Gkillas, G. Arvanitis, A. S. Lalos, and K. Moustakas, “Federated Learning for Lidar Super Resolution on Automotive Scenes,” in *2023 24th International Conference on Digital Signal Processing (DSP)*, 2023, pp. 1–5. doi: 10.1109/DSP58604.2023.10167942.

[95] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580. doi: 10.1109/IROS.2012.6385773.

[96] N. Piperigkos, A. S. Lalos, and K. Berberidis, “Graph Laplacian Diffusion Localization of Connected and Automated Vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12176–12190, 2022, doi: 10.1109/TITS.2021.3110650.

Co-funded by the European Union

Project funded by Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER State Secretariat for Education, Research and Innovation SERI

KIAT 한국산업기술진흥원 Korea Institute for Advancement of Technology