Efficient Sensor Fusion of LiDAR and Radar for autonomous vehicles

DOCTORAL THESIS

Javier Mendez Gomez

Doctoral Programme in Information and Communication Technologies
Department of Electronic and Computer Tecnology
University of Granada

March 2022

Editor: Universidad de Granada. Tesis Doctorales
Autor: Javier Méndez Gómez
ISBN: 978-84-1117-350-6
URI: http://hdl.handle.net/10481/75425

Este documento está preparado para ser impreso a doble cara.

Efficient Sensor Fusion of LiDAR and Radar for autonomous vehicles

Directed by:

Prof. Diego Pedro Morales Santos
Prof. Manuel Pegalajar Cuellar

Developed at:

Infineon Technologies AG
BEX RDE RDF EET
Munich, Germany

Doctoral Programme in Information and Communication Technologies
Department of Electronic and Computer Tecnology
University of Granada

March 2022

Acknowledgements

We know what we are, but know not
what we may be

William Shakespeare

When starting a PhD it seems you will be following a research individually but during the years of the PhD you get i touch with a lot of people who help to smooth the path as well as make it more entertaining, which is also important since it takes 3 years of your life. Because of this, it is important to thank all those people who were there.

First, I want to thank Diego P. Morales and Manuel Pegalajar, my thesis directors, for all the help during these years. You both helped me not only to learn about how to plan a research (from researching the state of the art to evaluate the results correctly) and writing (maybe this was even harder than the research itself) but also supporting me in those moments when I really thought I would not be able to finish this path. I am extremely thankful for all those moments, talks and messages to encourage me to keep trying. Thanks to that I learnt that things can be hard but the key point is never giving up.

I also want to thank the IFAG RDE RDF team for giving me this opportunity when I was a fresh graduate without much experience, specially to my supervisor Kay Bierzynski, who also taught me not only about AI but also about project management and, maybe even more important for the future, investments. Thanks to the PhD position at Infineon I got to know not only some experts from who to learn but also some close friends who not only helped me with technical stuff but also made me enjoy the days in Munich organizing cool plans like "merienda y juegos de mesa.ºr just a simple conversation. All of you made me feel I have a family here who can always support me be there when I need you. Specially I want to thank some close friends (that doesn't mean the rest were not important) who taught me a lot and were always there: Miguel Molina, Antonio Cabrera, Juan Cruz y Borja Saez.

I also want to thank my family (this time biological family) because they always were much sure than me about the fact that I would be able to finish

this path and, even when for some of them it was specially hard that I wasn't in Spain (not pointing at anyone mom) they always supported me and gave me strength to continue. Even with the distance, through calls and a lot of visits I never felt I was left out. Thanks for always calling me, telling me about daily things, pictures, videos, etc. and always being able to meet me when I went to Spain even when sometimes it was just a weekend and now with covid the situation was hard.

Lastly, I specially want to thank my partner, Uyen, you are the one who helped me the most. You spent a lot of hours correcting my papers (and also scolding me) because I keep making stupid mistakes or trying to translate directly from Spanish. Apart from that, you helped me by supporting me in those moments when I got really down thinking I was not able to finish my tasks on time. You gave me the strength and motivation to be better. All I can say is thank you, without you I would have not been able to do it.

For sure I would like to thank a lot of other people who also helped me during these years but in that case the acknowledgement would be even longer than the main thesis so, to everyone else who was also there helping me, thank you.

Resumen

La conducción autónoma cada vez es más relevante gracias a los avances conseguidos por compañías como Tesla o Google. Esto se debe al avance en el campo de la inteligencia artificial al mismo tiempo que se desarrollan y/u optimizan sensores como el LiDAR, especialmente relevante para esta tarea. Sin embargo, ya que se estima que cada vez habrá más vehículos autónomos, se debe asegurar su correcto funcionamiento en todos los escenarios comunes y no solo en condiciones de laboratorio. Esto implica la necesidad de introducir técnicas de fusion sensorial para la detección de objetos relevates para la conducción en condiciones adversas.

Esta tesis doctoral es el resultado de la investigación de tecnologías para la percepción basadas en técnicas de fusión sensorial así como deep learning para la detección de objetos relevantes en aplicaciones para vehículos autónomos. Dichos resultados se presentan en forma de compendio de publicaciones, recopilando los artículos científicos publicados durante el periodo doctoral.

La investigación se ha realizado teniendo en cuenta las limitaciones impuestas en el network edge respecto a memoria de los dispositivos y latencia esperada para aplicaciones en tiempo real. El objetivo ha sido la implementación en un edge device el proceso completo requerido para la detección de objetos, incluyendo el preprocesamiento de los datos antes de su evaluacion con técnicas de inteligencia artificial. Para ello, se han investigado los distintos sensores que pueden ser relevantes para la conducción autónoma (cámara, LiDAR y radar) así como técnicas para preprocesar estos datos con el objetivo de mantener la información mientras que se reduce su tamaño. Posteriormente se han investigado algoritmos de deep learning que puedan usarse para la detección de objetos siguiendo un esquema de fusión sensorial. Por último, estos algoritmos se han optimizado para poder ejecutarse en un edge device, como el Google Coral TPU usado en esta investigación.

La investigación se ha realizado bajo un contrato doctoral en las instalaciones de Infineon Technologies AG, en su sede principal de Múnich, Alemania.

Abstract

Autonomous driving is more relevant recently thanks to the advances achieved by companies such as Tesla or Google. This is a result of the technological advances in the field of artificial intelligence at the same time as new automotive sensors are been developed or further optimized, such as LiDAR sensors, which are highly relevant for autonomous driving tasks. However, since it is assumed in the future there will be a large number of autonomous vehicles in the streets, it must be ensured that these vehicles can perform as expected in all normal scenarios present in the real world and not only in laboratory conditions. Therefore, it is required to integrate sensor fusion techniques in the target detection pipeline for autonomous vehicles.

This doctoral thesis is the result of a research about technologies for computer perception based on sensor fusion techniques as well as deep learning algorithms for target detection for autonomous vehicle applications. These results are presented in the form of a compendium of publications, compiling the scientific articles published during the doctoral period.

The research has been planned taking into consideration the constrains related to the network edge regarding device memory and latency expected for real time applications. The general objective has been the implementation of the full target detection pipeline in an edge device, including the data preprocessing as well as the evaluation of the data using artificial intelligence techniques. For this purpose, relevant automotive sensors for perception that can be used for autonomous driving have been researched (such as camera, LiDAR and radar sensors) as well as techniques to preprocess the data provided by these sensors in order to maintain relevant features while reducing their memory size and complexity. After this, deep learning algorithms that can be used for target detection following a sensor fusion paradigm have also been researched. Lastly, these algorithms have been optimized to fit in edge devices, such as the Google Coral TPU used in this research.

The research has been carried out under a doctoral contract at the facilities of Infineon Technologies AG, at its headquarters in Munich, Germany.

Contents

List of Figures

List of Tables

Part I

PhD Dissertation

Chapter 1

Introduction

The best way to predict the future is to invent it

Alan Kay

The society has always have the goal of reducing the workload in people and make the technology advances more accessible to the general public in order to improve the people´s life conditions. Consequently, since the beginning of the human history, we have tried to develop tools or systems to help us with our tasks, for example the invention of the wheel to transport materials and people, or to directly execute the tasks in our place by using automatic systems such as the printing machine to replace people having to copy each book manually. This tendency is a characteristic of our society and because of that there are always new tasks to automatize.

In recent years, following this tendency to develop autonomous systems, artificial intelligent (AI) algorithms have increased their relevance in the market due to their promising results in all industry fields, from computer vision tasks (1; 2; 3) to prediction (4; 5) or data classification (6; 7). These results has has led numerous authors as well as companies to work on developing autonomous vehicles using AI techniques (8; 9). These vehicles may be classified due to their level of capabilities to be autonomous. This classification, according to the Society of Automotive Engineers (SAE International) (10), divides the autonomous driving paradigm in 5 levels where the lowest ones the vehicle may assists sometimes the driver with longitudinal and lateral driving tasks and this help increases until level 5, where the vehicle can run autonomously under any circumstances. However, this topic can be divided in multiple subtasks due to the complexity of the autonomous driving. These subtasks include topics such as environment detection (11), route planning (12) and adversarial attack detection (13).

1.1. Motivation

Most of the current algorithm for target detection in the autonomous driving paradigm are based on a single sensor, i.e. camera or LiDAR sensors (14; 15; 16; 17). However, nowadays there are numerous sensors in the market that could support in this task rather than always depending on the same sensor. An example of these sensors are ultrasound sensors, radar sensors and LiDAR sensors apart from the traditional camera sensor. These sensors are used with different purposes and, at the same time, they have different limitations.

- Camera sensor: this sensor has been used since long time ago due to its capabilities to not only return information about shapes but also colors. This may be highly relevant for autonomous vehicles since some of the most common traffic signals are color based, i.e. traffic lights. As a result, this sensor provide a wide range of features that can be relevant for the environment study (target detection, scenarios conditions study, etc.) (16). However, the main problem of this sensor is the fact it may gather private information due to the large volume of data acquired, such as pedestrian faces. At the same time, the data gathered by this sensor can be highly affected by hazard situations such as rain, fog and low light among others. As a result of this, a number of companies and researchers are studying the possibility of achieving similar results in autonomous vehicle tasks without including this sensor.

- Ultrasound sensor: this sensor transmits a pulse/signal to later calculate the time until the echo is received. By repeating this with multiple receiver sensors it is possible to locate targets in 3D positions (18). However, even when the precision of the detection may be high, the effective range of this sensor is highly reduced in contrast to the previous sensor (in the range of centimeters or a meter). Nevertheless, this sensor is still relevant for the discussed topic, specifically in scenarios where accurate maneuvers are required, such as when parking.

- Radar sensor: this sensor follows a similar approach to the ultrasound sensor since it also transmits a signal (in this case a variable frequency signal) and measures the frequency difference between the echo and the current signal. This provides information regarding the distance, angle of arrival and speed of the detected target respect the sensor. As a result of this, this is one of the most popular sensors integrated in vehicles to detect other vehicles in the scenario (19). However, this sensor also has limitations such as the maximum range, which is highly shorter than LiDAR sensor data. This can lead to a late detection when a target is not located until the last moment when it can be dangerous for the ego vehicle.

- LiDAR sensor: this sensor transmits laser pulses and calculate the time to receive the reflection of each of these pulses at the same time it knows the horizontal and vertical angle used when transmitting each pulse. Knowing the angles, time to receive the echo and the speed of the light in the air it generates a 3D map of shapes in the scenario. These sensors usually have a range of hundreds of meters in autonomous vehicle applications. As a result, they can gather information from all relevant targets, such as vehicles, walls, signals, etc from a large enough distance to take them into consideration when planning the route (20). However, the transmitted laser pulses may be absorbed by some materials such as water. This may generate problems in hazard weather conditions such as rain or fog, where the number of reflections may be highly reduced in comparison with a sunny scenario. At the same time, due to the hardware limitations of this sensors, they have a blind area in the near field, being this area usually of 1-1.5 meters around the sensor, which may be a highly relevant problem for accurate maneuvers.

Not only these sensors but also the algorithms to study their output data have been improved in recent years to extract more information from the data as well as improve the efficiency of these processes (21; 22; 23; 24). Some of the most relevant algorithms for this topic are the target detection models.

These models focus on detecting targets belonging to one or multiple relevant classes decided by the user who trained the model. These models return the location in the frame of these targets as well as their predicted classification (21; 22). Depending on the input data, this target detection algorithms may be adapted to 2D (23) or 3D targets (25). These target detection models can be divided into two classes:

- Two-stage detectors: these algorithms execute two processed for the target detection. The first process is the localization of areas in the 2D-3D data that can contain relevant targets. In a second step, the algorithm evaluates all the proposed regions to determine in a relevant target is present in each region and, in case there is a target, what class it is apart from refining the coordinates of the target. As a result, these models usually achieve high accuracy results due to their deep search of regions of interest at the cost of a larger latency, such as the Region Convolutional Neural Network (RCNN) (26), Fast RCNN (27) or Faster RCNN (22) models.

- One-stage detectors: differently from the previous algorithms, it performs the detection and classification without having to extract a first set of regions of interest. Consequently, a reduction in the latency is

achieved due to the reduction of operations in the pipeline but usually the final accuracy performance doesn't achieve two-stage models accuarcy results. An example of one-stage detectors are the You Only Look Once (YOLO) (24) and Single Shot Detector (SSD) (21).

However, as previously observed when explaining the sensors, these sensors have limitations and strengths that may complement each other. Therefore, they may benefit from using Sensor Fusion techniques to merge their data in order to improve the overall performance of the system (28). An example of this collaboration may be the combination of LiDAR and radar sensors to integrate the high range of the LiDAR data while radar may provide data of the near area that LiDAR may not detect. At the same time radar may support in hazard weather conditions since radio signals are affected in a lesser degree by water in contrast to laser pulses. As a result, the final system would have the strengths of both sensors and it would improve their limitations.

Nevertheless, the platform where these algorithms will be deployed must be taken into consideration when designing the overall pipeline. Since the information gathered for autonomous vehicles applications may include private data of the vehicle, such as speed and destination, a Cloud approach where the information is transmitted to an external server may not be suitable due to the information leak risk. A different approach, Edge Computing, is taking the lead in these applications (29). Edge Computing is referred when the data is processed near the data source/sensor. Consequently, since data is not transmitted to any external device, the possibility of data leak and problems during the data transmission are reduced. At the same time, the overall latency may be reduced since the time to transmit and receive the data is removed from the pipeline. However, this approach has limitation regarding the resources at the network edge, which are highly reduced in comparison to Cloud servers.

Consequently, different preprocessing techniques for automotive sensor data must be researched as well as Sensor Fusion pipelines to fit low-resource edge devices and hardware accelerators while maintaining high performance results.

For all of this, the research on Efficient Sensor Fusion techniques based on Deep Learning algorithms at the Network Edge appears as an attractive innovation and relevant topic in the current autonomous vehicle paradigm.

1.2. Objectives

The objectives of this doctoral thesis are the ones derives from the development of all the steps to design and deploy a Target Detection model based on Sensor Fusion techniques at the Network Edge. The intended con-

tribution to the state of the art includes the study of the emerging Edge Devices as well as frameworks to deploy AI capabilities at the network edge, the data preprocessing and labeling techniques and techniques to fuse the information from multiple sensors at the Network Edge. These contributions are summarized as follows:

1. **Edge Devices research**
    - Research about emerging Edge Devices where the Sensor Fusion pipeline can be implemented.
    - Research about frameworks to deploy and adapt state-of-the-art Target Detection models as well as Sensor Fusion techniques for Edge Devices.
2. **Data preprocessing and labeling research**
    - Research techniques to efficiently label gathered data in order to use it in a later step to train AI models.
    - Research techniques to preprocess sensor data to reduce their memory consumption while maintaining their relevant features.
3. **Sensor Fusion research**
    - Research new Sensor Fusion algorithms that may be suitable for the Network Edge while still achieving state-of-the-art accuracy results.
    - Implementation of the complete solution in Edge Devices.

1.3. Outline

The presented document constitutes a thesis by compendium of publications. This means, this thesis is formed by the most relevant publications achieves as result of the research carried out during the doctoral program. The work includes three indexed articles in multiple scientific journal and one article currently under review by a scientific journal. We list them following:

- J. Mendez, K. Bierzynski, M. P. Cuellar and D. P. Morales, *Edge Intelligence: Concepts, architectures, applications and future directions,* in ACM Transactions on Embedded Computing Systems, 2021. (Q4)
- J. Mendez, S. Schoenfeldt, X. Tang, J. Valtl, M. P. Cuellar and D. P. Morales, *Automatic Label Creation Framework for FMCW Radar Images Using Camera Data,* in IEEE Access, vol. 9, pp. 83329-83339, 2021, doi: 10.1109/ACCESS.2021.3087207, 2021. (Q2)

- J. Mendez, M. Molina, N. Rodriguez, M.P. Cuellar and D. P. Morales, *Camera-LiDAR Multi-Level Sensor Fusion for Target Detection at the Network Edge* in Sensors, 21(12), 3992, doi: 10.3390/s21123992, 2021. (Q1)
- J. Mendez, M.P. Cuellar and D. P. Morales, *Lidar-Radar Robust Multi-Level Sensor Fusion for Target Detection at the Network Edge* UNDER REVIEW by Elsevier Measurements, 2021 (Q1).

This thesis is divided in two main parts:

- **PhD Dissertation:** is devoted to describe the problems we have addressed and discuss the research we have performed. In particular, Section 2 presents the research and the main results, and Section 3 summarizes them.
- **Publications:** collects the journal papers related with the doctoral research shown in this thesis. These papers are devoted to the research the objectives previously mentioned.

Chapter 2

Methodology

In this section, the methodologies carried out in pursuance of the objectives we present in Section 1.2. Since these methodologies may highly differ from each other, a specific subsection focus on each one.

2.1. Research on Edge Devices

A research about emerging Edge Devices has been carried out thanks to numerous surveys such as (30; 31) among others that can be consulted in Chapter 2 of this thesis. In these works, it is possible to compare the technical characteristics of the most relevant edge devices (including, internal memory and hardware acceleration among others). At the same time, some of these emerging devices, such as the Google Coral TPU board and the Jetson nano board were evaluated in our dependencies to evaluate the same parameters as previously commented.

This evaluation led to the identification of the approach to deploy AI models in each of these devices as well as their capabilities. Ultimately, the viability of these devices was measured by the capabilities of each device as well as the performance results of common AI models in these devices.

Further information as well as results gathered from this research are included in the publication *Edge Intelligence: Concepts, architectures, applications and future directions* (32).

2.2. Study on data labeling techniques

In our research, large quantities of data were used to train the AI models to improve their performance in a large set of scenarios while including at the same time a large number of target classes. The labeling process, when executed manually may consume long times while it can result in human errors during the labeling do to the repetition of this task. Consequently, a

more efficient way to label the data was research.

As a result of this, an automatic labeling tool was developed focused on camera and radar sensors. The performance of this tool was evaluated in multiple datasets from different locations that include scenarios in Singapore and Munich. The metrics used to evaluate the performance of the proposed system was the detection accuracy and the latency since these parameters show if the proposed system is better and faster than a traditional manually labelling approach. The sensors used in these scenarios were different to also evaluate the suitability of this tool for different radar and camera configurations.

Results are gathered in the publication *Automatic Label Creation Framework for FMCW Radar Images Using Camera data* (33).

2.3. Research on preprocessing techniques for sensor data

We investigate different techniques to preprocess sensor data in order to reduce the data size and dimensionality while maintaining their relevant features. To do this, different preprocessing techniques were applied to radar and LiDAR data.

For radar data, numerous representations were evaluated, including range-Doppler maps, range-angle maps, occupancy grids and depth maps. These representations proved to provide some of the features but none of them included all the possible features that can be extracted from radar data. Consequently, the preprocessing technique applied to radar data was selected depending on each specific application to extract the desired features. The results from these preprocessing techniques are gathered in the publications *Automatic Label Creation Framework for FMCW Radar Images Using Camera data* (33) and *Lidar-Radar Robust Multi-Level Sensor Fusion for Target Detection at the Network Edge* (34).

Similarly, LiDAR data preprocessing techniques were also evaluated. The most relevant preprocessing techniques, voxel approach and depth maps, were evaluated. Results are gathered in the publication *Camera-LiDAR Multi-Level Sensor Fusion for Target Detection at the Network Edge* (35) and *Lidar-Radar Robust Multi-Level Sensor Fusion for Target Detection at the Network Edge* (34).

The metrics used for the evaluation of the preprocessing techniques are the reduction of the data size achieved with each preprocessing technique as well as the accuracy of the maintained features.

2.4. Research on Sensor Fusion techniques

In order to improve the performance of the Target Detection algorithms, multiple Sensor Fusion approaches were researched based on the state-of-the-art techniques. As a result, a novel layer and network structure for ANN was developed.

This Sensor Fusion technique was evaluated with camera and LiDAR data from multiple scenarios to prove its accuracy performance in normal and hazard environments where camera data may not be reliable. Results from this technique are gathered in publications *Camera-LiDAR Multi-Level Sensor Fusion for Target Detection at the Network Edge* (35) and *Lidar-Radar Robust Multi-Level Sensor Fusion for Target Detection at the Network Edge* (34). In these publications, the metrics used to evaluate the performance of the proposed sensor fusion techniques in comparison with single-sensor models were the latency (to evaluate in increasing the number of sensors leads to a higher latency) as well as the detection accuracy in normal and hazard scenarios.

Other Sensor Fusion techniques were applied for the automatic labeling process based on radar and camera data. Results from this technique are gathered in the publication *Automatic Label Creation Framework for FMCW Radar Images Using Camera data* (33).

2.5. Research on AI models deployment frameworks

We realized an analysis of the state-of-the-art frameworks for AI models at the Network Edge such as TensorFlow Lite and MXNet among others. The comparison of these frameworks was based on their capabilities to deploy generic ANN models in generic Edge devices while ensuring the memory and latency optimization.

We identified the most relevant frameworks that fit the Edge devices that would be used during the doctoral program at the same time the ensure the optimization of the ANN models.

Results from this research are gathered in the publication *Edge Intelligence: Concepts, architectures, applications and future directions* (32).

2.6. Research on AI models deployment

In order to reduce the memory size of AI models, multiple techniques, including pruning and quantization, were evaluated. The performance of these techniques to adapt AI models for the Network Edge was evaluated based on the memory size reduction achieved, latency reduction achieved and decrease of the accuracy of the models.

At the same time, due to constrains of the used frameworks and Edge devices, some specific limitations were established for the AI models deployment at the Network Edge. These limitations are the quantization required for the data and models to fit in the Google Coral TPU board as well as the ANN layers supported by this device. Further information as well as results are gathered in the publication *Camera-LiDAR Multi-Level Sensor Fusion for Target Detection at the Network Edge* (35) and *Lidar-Radar Robust Multi-Level Sensor Fusion for Target Detection at the Network Edge* (34).

Chapter 3

Achievements

If there is no struggle, there is no progress

Frederick Douglass

In this chapter, we describe the conclusive results accomplished in several areas during the doctoral program that led to the publications attached to this doctoral thesis. We have organized this section in several subsections corresponding each one to one of the publications included in this doctoral thesis. Consequently, the results from each publication will be presented as well as how these publications are related with the previously commented research objectives.

3.1. Research on architectures and applications of Edge Intelligence

As described in Section 1.3, there are numerous objectives in this doctoral research. However, the first step in a research should always be the research of the state of the art in order to understand recent improvements of the technology. Because of this, the first step when we started this research was the evaluation of the current edge computing techniques as well as edge devices.

This field includes numerous topics, since when studying the edge computing, or the edge intelligence specifically in this PhD research, not only algorithms but also the devices must be researched. It is important to understand the constrains imposed by the edge computing paradigm, where data is meant to be preprocessed near the data origin. As a result, the devices where data must be preprocessed do not have large resources in contrast to cloud servers. However, since data is not transmitted to external or far servers, the risk of data leakage is reduce. This is an important point sin-

ce numerous emerging applications use private data (such as autonomous vehicles where information of destination, origin, etc. must not be share for security reasons) or traditional applications where personal information is used (such as medical applications where patient data is required).

However, due to these constrains, not all AI algorithms are suitable for the network edge. Therefore, algorithms that will be deployed at the network edge must be optimized to ensure a low memory consumption as well as reducing the complexity of operations required when possible. These optimizations also aim to reduce the power required by the device, since these devices are often not connected to the electricity network but working with battery (29).

At the same time, some new algorithms and techniques have been proposed for the network edge in order to face those constrains while maintaining a state-of-the-art performance. An example of these techniques is the distributed learning, where the learning and inference phases are not performed by an individual node but partially executed in each of the nodes of an edge computing network. Consequently, the workload is distributed, enabling the implementation of complex techniques that a single node would not have capabilities to execute (36; 37).

This approach, apart from the previously described advantages, can also be used as an abstraction layer between the sensor data and the final application since edge computing nodes can also be used to preprocess the data to filter only relevant features/information while sharing a common final data structure. Therefore, output data from these nodes would share the structure no matter the initial sensor model or manufactures (37).

At the same time, not only the algorithms but also the frameworks and devices for edge intelligence must be researched since they may add constrains specific from each device/framework. An example of this problem is the Google Coral TPU, which requires the TensorFlow Lite framework and it has limitations regarding supported neural network layers (38).

All these key points are evaluated in our survey (32), where a comprehensive analysis of edge computing and edge devices is presented. In this survey, we focus not only on general application edge computing but also on artificial intelligence algorithms at the network edge, often referred as edge AI. At the same time, we also discussed the emerging frameworks that can be used to deploy AI algorithms at the network edge, since it is not only important the algorithm but also how it can be integrated in the edge device.

Therefore, recent advances in edge computing have been explained in this publication, such as the new hardware and software that have emerged during the last few years to face the problems of cloud computing in the current situation. The relevance of this technology has been demonstrated with diverse examples of applications including autonomous driving, security solutions, IoT applications, location awareness applications, and network

management. The hardware advances follow the line of incorporating new modules/layers which enable the parallelization of the processes. Meanwhile, the software research lines focus on improving the data processing speed as well as improving the security of the systems. These features have been compared in diverse tables for a deeper understanding.

Later, new Edge AI devices have been commented to explain what advantages they can bring to the current state of the art in diverse research lines such as Natural Language Processing (NLP), Computer Vision, Internet of Things (IoT) and Virtual Reality (VR). Following the line of the research made about Edge Computing, the current Edge AI devices and frameworks have been explained based on their relevant features for a posterior comparison.

As a conclusion from this publication, we could observe how numerous companies are developing their own frameworks and devices, such as Google that developed TensorFlow Lite and the Google Coral TPU. This will lead in the future to a large variety of devices in the market, giving the user the possibility to select the most suitable one for each application regarding required memory, programming language and/or AI framework used and application purpose among others. This is a result of the relevance of IoT, where people want to include each time more sensors in daily life applications to ease them or to improve their efficiency.

3.2. Research on data preprocessing and data labeling

After the previously explained research regarding the AI frameworks and devices suitable for the network edge, we started researching about relevant algorithms that could be use for target detection using radar and/or LiDAR data (23; 21; 22). The first step for this was to understand these two sensors in order to understand what features we can extract from each one and when their data may be relevant.

At this point, in order to fully understand the sensors, data was required in order to evaluate the different data preprocessing techniques.

In the case of radar data, usually a double Fourier transformation is performed in the initial time domain raw data in order to generate a range Doppler image. This image provide information regarding the range and speed of moving targets, represented as clusters in these images. Similarly, these images can be further preprocessed to extract the angle of arrival from each of these clusters using techniques such as MUSIC (39) or beamforming (40). Once the angle of arrival information is extracted, range angle maps can be generated, where similar to an occupancy grid, the image represents in what distance and angle a possible moving target may be located.

In the case of LiDAR data, since it does not have a fixed structure where 3D points have an specific order, it is usually preprocessed to make it invariant to order. This is a requirement when using LiDAR data as an input for deep learning models. The most relevant preprocessing techniques are the voxel approach and the depth maps. The voxel approach generates a three dimensional matrix where each of the cells represents a cell of the real space studied with the LiDAR. Therefore, LiDAR data can be organized in these ordered cells that can be later studied using 3D convolutional layers or fully connected layers after flattening the matrix. On the other hand, the depth map approach generated a 2D image where distance of each point is codified in the color, obtaining a final data similar to a camera image (35).

However, most of the public datasets contain raw camera data and raw LiDAR data but not so much information regarding raw radar data in automotive applications. This problem triggered a question, if we don't have large quantities of raw radar data, for sure we don't have a labeled dataset so how will we train our AI models? Because of this we proposed a automatic labeling tool for radar data to solve this problem in the publication (33).

In this publication, an efficient Sensor Fusion framework to automatically generate labels for range Doppler maps has been developed. As a result, large number of labeled range Doppler maps can be generated efficiently while data is gathered. Therefore, it is possible to speed up the dataset creation that can later be used to train AI models.

The proposed technique is based on multiple state-of-the-art sensor fusion algorithms, extracting the advantages from each of them to further improve the system. The general pipeline is based on the extraction of labels from camera data that later can be used to generate labels for radar data. This pipeline is proposed due to the large number of deep learning models to detect targets in camera image as well as the fact that usually there is camera data present when recording sensor data to later use it for manual labeling tasks. Consequently, we can use the camera data, assuming the relevant targets of the scenario are in the shared field of view of the camera and radar sensors. At the same time, since single-camera sensors do not provide depth-range information, we calibrated the system based on some initial known data to extrapolate the target distance in the rest of the camera frames. Therefore, required features for the label creation in radar data (range, angle respect the sensor and classification) can be efficiently extracted from camera data.

However, this tool also has limitations since in case it is a moving scenario the camera calibration approach will not be suitable and, therefore, it will no be possible to match detected targets in camera data with radar clusters based on distance. Similarly, in conditions where camera data may not be reliable because of hazard conditions the labelling tool may not achieve the same accuracy as is standard scenarios. However, it has been presented that in normal weather and light conditions in an outdoor scenario the proposed

approach can achieve high accuracy results for the label creation task while highly reducing the required time to generate these labels in comparison with manually labeling the radar data.

This approach proposed for this tool could also be used for other automotive sensors, such as LiDAR sensors. LiDAR sensor could be used instead of camera sensors, or as an addition to camera sensor, to label the data at the same time LiDAR labels can be generated. This could overcome traditional camera/vision sensor approach regarding adverse environmental conditions (i.e. lighting, reflections, etc.) while providing relevant depth information of the scenario.

At the same time, it has been shown that the fusion of radar and camera data does not require complex structures to achieve high accuracy results while maintaining low latency. This lends justification to a variety of new sensor fusion algorithms where the algorithms are optimized for radar and camera sensor.

3.3. Research on Sensor Fusion to improve camera detections

After the first task related to develop sensor fusion pipelines, to further learn about sensor fusion techniques for target detection as well as data preprocessing algorithms, we started researching about a sensor fusion algorithm for camera and LiDAR data, following the research line of (41).

This idea started due to the fact that LiDAR data can perfectly complement camera data. Camera data provides information regarding color and details but it lacks of information regarding surfaces and distance. As a result, camera data may not be reliable in hazard conditions, i.e. low-light conditions, where the data from the camera would be completely black. Similar problems will appear when targets are not real but they are images, since camera does not include depth information it cannot distinguish between a real object and a picture of it. However, if we include the data from LiDAR we can have a more complete understanding of the environment where we have not only the details but also the shapes and distances. Therefore, adding the features extracted from LiDAR data to the camera data could solve the previously commented problems, achieving a more robust system.

Nevertheless, LiDAR data requires large memory volumes to be stored since each frame may contain thousands of 3D points. Therefore, before starting the research regarding camera and LiDAR sensor fusion, LiDAR data preprocessing algorithms had to be further evaluated to reduce their memory size while maintaining their key features such as distance and surfaces information.

Therefore, the previously LiDAR preprocessing techniques commented

in Section 3.2 (voxel and depth map approaches). The first one (the voxel approach) is highly used in the literature (42; 43; 44) due to its high accuracy results for target detection since data regarding possible target detection is not lost with this technique. At the same time, it is possible to access all data points gathered by the sensor. However, this technique only solves the problem of the data structure and order but it does not reduce its memory size enough for the network edge. At the same time, to efficiently study this data, neural networks should include 3D convolutional layers to study the 3D occupancy matrix but this layers are not supported by most of the edge devices. As a result of this, we started the evaluation of the second approach, the depth map generation. This technique aims to convert the 3D LiDAR data into a 2D image. By doing this, it is possible to represent LiDAR data in a 2D plane, similar to a camera image. In this case, to maintain the information regarding point distance, the distance is codified in the color of each pixel of the depth map. When using this data there may be occasions of target overlapping since we are projecting the 3D points into a 2D plane. Because of this transformation, in case the information of a specific point is later required the opposite transformations have to be applied to extract the real X, Y and Z coordinates, which could lead to high latency. However, this approach generates LiDAR data that can be easily used as an input for deep learning models and, due to its similarity with camera data, can be combined with camera data as desired. Therefore, since in the desired camera-lidar target detection application we dont need the specific X, Y, Z values of the LiDAR points in the final stage, we decided to follow the depth map generation approach in the PhD research.

Once the data preprocessing techniques were evaluated, we moved to the development of an algorithm for target detection. At this moment, due to the similarities between LiDAR depth maps and camera data, it was possible to execute the data fusion in an early stage by adding the LiDAR depth map as an extra channel to the RGB camera images or after the target detection, as in a voting algorithm. This large number of possibilities led to a question: *what if rather than me I let the neural network decide when it is better to fuse the data?*.

As a result of this question, we started researching neural network structures where data were fused not only in one layer but at multiple levels, so the network can set during the training phase the relevance of the fusion at each of the proposed levels. Consequently, we proposed a neural network algorithm where data from camera and LiDAR were combine at multiple levels, creating a new layer we called *Fusion Layer* (35). This approach of combining the data was executed at multiple levels, letting the network combine the data after different levels of feature extraction. As a result of this, the network may combine features from small targets during the initial layers before those features are lost due to the effect of further convolutional

and pooling layers. Features from larger targets will still be present in later stages, because of this the network will fuse the data at the relevant stage depending of each specific target.

The idea for this approach came from the Single Shot Detector (21), where multiple feature maps of different sizes are used to maintain features from small targets as well as larger targets rather than using only one feature map at a time.

Since the goal was still to deploy the developed algorithm in an edge device, the algorithm had to be later optimized following techniques of pruning and quantization to remove non relevant connections/neurons in the model as well as reducing the memory required to store its parameters. At the same time, the proposed model was designed following a one-stage detector approach to reduce the latency imposed by the model structure in contrast with two-stage approaches where two different models are required (one for the region proposal and one for the classification and refinement). As a result of this, the proposed model was compressed to fit the network edge constrains.

To evaluate this algorithm, we selected a popular public dataset that contains LiDAR and camera data, the KITTI dataset (45). The accuracy results on this dataset do not surpass the state-of-the-art accuracy shown by other sensor fusion models evaluated on this dataset such as Deep Gated Information Fusion Network (DGFN) (46). Nevertheless, the proposed model still achieves a 90.92 %, 87.81 % and 79.63 % accuracy results for easy, medium and hard to detect targets in the challenging KITTI dataset while requiring a 92.1 % less memory than the DGFN model. And, as a result of the optimization applied to the proposed model, the latency of the model has also been reduced to 0.057 seconds.

At the same time, the advantages of using a Sensor Fusion approach in contrast to a single-sensor model were discussed in this publication by comparing our proposed algorithm with other single-sensor target detection algorithms. It was possible to observe in the comparison how some of the single-sensor models achieve better accuracy than our proposed algorithm. However, we studied the effect of using only camera data for the target detection task and we showed how the accuracy drops to 73.81 %–65.41 %–39.63 % (easy, medium and hard to detect targets) in hazard environments such as cases where the light is not good. Since the LiDAR data does not depend on the light of the scenario, using a camera-LiDAR Sensor Fusion approach helps to maintain the initial accuracy or reduce the accuracy drop to 83.18 %–80.02 %–47.83 %. This proves that even if high accuracy can be achieved by a single sensor model, these algorithms are not robust when facing changes in the environment in comparison with a sensor fusion algorithm.

This lends justification to further research the sensor fusion of automotive sensors for autonomous vehicle applications, even without including sensors

that may not be suitable for all scenarios, such as camera. Camera may not be suitable sometimes due to the fact that it gathers private data such as vehicle plate numbers or faces in public scenarios. Consequently, after the good results of the camera-LiDAR sensor fusion, we moved to designing a sensor fusion algorithm based on the two main sensors of this doctoral research, radar and LiDAR sensors.

3.4. Research on LiDAR and radar Sensor Fusion for Target Detection

As previously commented, after the good results achieved with the camera-LiDAR sensor fusion for target detection, we moved to design a similar technique but this time based on radar and LiDAR sensors. These sensor were selected due to the fact that they do not gather private information so they can be used in all scenarios, in contrast to camera data, while still gather relevant data for the target detection that can complement each other. At the same time, these sensors are not affected by light conditions, making a sensor fusion system based on them more robust in hazard situations.

Regarding the information gathered by each of the selected sensors, radar sensor gathers relevant data regarding moving targets from a short range (in the order of centimeters) to medium ranges (in the order of tens of meters) with a high resolution that can be usually configured at the cost of less range. Therefore, a trade-off between maximum range and resolution must be selected depending on each specific application. In the case of this doctoral research, main relevant targets are vehicles and pedestrians so we do not need a resolution under 5-10 centimeters with the radar data but we would prefer a higher maximum range. At the same time, radar data is not highly affected by weather conditions or light as previously commented in this thesis. However, it is not possible to detect specific static targets in radar data in normal conditions due to the fact that all static targets will be aggregated in the same pixel of the range Doppler map.

On the other hand, LiDAR data can provide information regarding further targets (in the order of hundreds of meters as maximum range) as well as detailed information about their shapes, which can be relevant for the target classification, and distances. Nevertheless, LiDAR sensor also has constrains regarding distance, since it has a blind area in the area of approximately 2 meters around the sensor due to the fact that the receiver sensor in the LiDAR is not active during the light pulse transmission phase, so if there is a target closer than 2 meters to the LiDAR sensor, the pulse reflection will be received before the sensor receiver is enable, resulting in a blind area in short field of the sensor. At the same time that LiDAR data can be highly affected by rain and fog weather conditions among others since translucent materials such as water do no reflect a high intensity signal. Similar problems appear

with other translucent materials such as glass. Therefore, LiDAR data can provide accurate data regarding shape of targets but it has a blind area and the quality of the data may drop in hazard situations.

However, the combination of these two sensor may provide reliable data since radar data can be used as an attention mechanism for the LiDAR data in hazard situations where the quality of the 3D LiDAR data drops. At the same time, radar may provide information regarding near moving targets that LiDAR sensor cannot detect, even when information regarding their shapes would not be gathered.

Once the suitability of these two sensors and how they can complement each other was researched, we started to design a sensor fusion algorithm for target detection based on radar and LiDAR sensors. The first challenge when using these sensors is the high difference in their data features and structures. To face this problem, we decided LiDAR data should be preprocessed to generated depth maps due to the hgh performance results achieved in the camera-lidar sensor fusion when using this preprocessing technique as well as the fact that, as previously discussed, raw LiDAR data cannot be easily used an input for deep learning models. Consequently, the question was how to preprocess radar data to reduce the structure differences with LiDAR depth maps. Therefore, raw radar data was preprocessed to generate range Doppler maps and range angle maps following the techniques described in Section 3.2. However, this time we didnt use these maps directly but we used them to create a 2D occupancy grip (in the horizontal plane) based on radar data. At this point, radar data could also be understood as 3D points similar to LiDAR points where the Z component was always the same for all targets. Because of this, it was possible to generate depth maps from the radar occupancy grids by assuming all radar targets had a fixed height from the ground (0 meters) to 1.5 meters. This height was selected as a representative height of the relevant targets in this doctoral thesis (pedestrian, motorbike, bicycle, car, etc.).

Once both sensor data were represented in a shared structure (depth maps), we followed the same approach we followed when implementing the camera-lidar sensor fusion. Consequently, we designed a deep learning model based on the previously commented *Layer Fusions* to fuse the data at multiple levels to maintain information regarding small and big targets.

This algorithm, since it required radar and LiDAR data, was evaluated not only in the NuScenes dataset (which provide information of 3D LiDAR detections and 2D detections from radar that can be used to generated the previously explained occupancy grid) but also in a custom dataset where all raw data was available. The accuracy results achieved with this model do not surpass the state-of-the-art accuracy results in the NuScenes dataset shown by other models such as CenterPoint v2 (47) and FusionPainting (48). However, as a result of the optimization applied to our model, it achieved a

reduced latency and memory requirements in contrast to the rest of compared models in (34).

At the same time, the advantages of using a sensor fusion approach rather than a single-sensor approach were studied by comparing the proposed algorithm, an state-of-the-art LiDAR model (MEGVII (49)) and a single LiDAR sensor SSD model (50). These models were evaluated on a custom dataset and a synthetic dataset that emulates hazard weather conditions such as rain or fog. In this test it was possible to observe the robustness of the proposed model to data corruption in comparison with single-sensor techniques.

Similarly to the camera-lidar sensor fusion model, the goal with this application was also its deployment at the network edge, specifically in the Google Coral TPU board, so compression techniques such as quantization and pruning were applied.

Chapter 4

Conclusions

This doctoral research has focused on the design and optimization of sensor fusion algorithms for target detection at the network edge based on deep learning models. Due to the focus to deploy these algorithms at the network edge, an edge computing approach has been followed during the whole research. The main conclusions and contributions obtained on this research are listed below:

- We have researched the current state of the art regarding edge computing techniques to deploy AI algorithms at the network edge in our publication (32). To do so, techniques to compress deep learning models have been discussed, such as quantization and pruning. The goal of this compression is the reduction of the memory size of the models as well as the computational complexity so they can be executed in edge devices, which have limitations regarding memory and computational capabilities to reduce the energy consumption. Apart from these compression techniques, new emerging techniques such as model distillation or distributed learning have been discussed to achieve state-of-the-art performance results at the network edge. At the same time, emerging edge computing devices have been compared to understand their main features and what capabilities they do not include at the moment but would be desirable for the future. Consequently, we have achieved the objectives 1-a and 1-b commented in Section 1.2.

- There are not large raw-data dataset for automotive applications with a large number of sensors. Therefore, it is not easy to have access to resources to evaluate and design new techniques/algorithms for autonomous vehicles. Because of this, we proposed an automatic label creation tool for radar data to speed up the dataset creation while reducing the human errors during the data labeling (33). As a result, by having rage Doppler map data, other preprocessing techniques to further preprocess the data such as occupancy grid or range angle maps can be evaluated

in contrast with public datasets such as NuScenes where radar data is provided as 3D points so no further preprocessing techniques can be applied. As a result of this publication, we can observe objective 2-a was achieved.

- There are numerous techniques to preprocess sensor data. The aim of all these techniques is to maintain some relevant features from the data while reducing its complexity or/and memory size. However, depending on the specific sensor data, for example LIDAR data or radar data, different techniques can be applied. These techniques fit the specific features and structure of the sensor data. In the case of LiDAR data, the main preprocessing techniques to reduce the data complexity while adding an ordered structure (required for deep learning models) are the voxel approach and the depth map generation. Each of these techniques have different advantages and limitations but due to the final 3D matrix structure achieved with the voxel approach, this technique may not be suitable for most of the current edge devices. Regarding radar data preprocessing, we have observe how some of the techniques build upon previous ones, such as range angle map and range Doppler map, where you need the range Doppler information to generate the range angle map. At the same time, we have been able to extract the knowledge from these preprocessing techniques to extrapolate them to other sensors, such is the case of depth maps fro radar data. Consequently, we have achieved objective 2-b regarding data preprocessing techniques described in Section 1.2. Examples where these techniques have been applied have been published in the publications attached to this doctoral thesis (33; 35; 34).

- Sensor fusion approaches may not improve the target detection performance of the state-of-the-art deep learning models but these approaches are highly beneficial in hazard conditions where single-sensor systems may run into data corruption or quality drop. Some examples of this have been included in this thesis, in publications (35; 34) where we discussed the accuracy drop when using only camera sensors in low light conditions or only LiDAR sensors in hazard weather conditions. Consequently, sensor fusion approached have been proven to be robust to scenery and conditions change, which is highly relevant in autonomous vehicle applications since they must perform well under all conditions to ensure the safety of passengers. After the research regarding sensor fusion techniques, we can conclude we have achieved objective 3-a presented in Section 1.2.

- We have evaluated our algorithms using edge devices to prove how state-of-the-art performance results for target detection can be achieved at the network edge. To do so, we have applied multiple techniques

to compress the deep learning models at the same time we have considered the limitations of the network edge while designing the network structure (i.e., using a one-stage detector rather than a two-stage detector) and the data preprocessing techniques. Further details about the deployment of the developed models on edge devices can be found in the publications attached to this thesis (35; 34). Because of the deployment of our models at the network edge, we can conclude we have achieved the objective 3-b regarding the deployment of the target detection system at the network edge.

- Along the doctorate duration, we have contributed to the state-of-the-art edge intelligence, achieving remarkable results that were published in scientific journals. Among these journals, the most relevant ones are IEEE Access and MDPI Sensors journals apart from the ACM Transaction of Embedded Comouting Systems and the publication that is still currently under review by the journal Elsevier Measurement.

To conclude, based on our research we forecast a clear improve in the near future of the Edge Intelligence due to its relevance in emerging topics such as autonomous vehicles. This is a result of the improvement regarding Edge Devices, which keep improving to integrate more capabilities like Tensor Processing Units, and specific frameworks for Deep Learning at the network edge.

4.1. Future trends

After the presented research, based on the experience gather through the doctoral period, we can foresee some main trends in the field of autonomous vehicles that could be further researched:

- **Emerging deep learning algorithms:** during the last year, new algorithms that highly improved the state of the art were presented such as the transformer networks that study the relationship of the data. These networks have been already applied to computer vision tasks (51; 52) so in the future it could be researched how to apply them to study camera or LiDAR data in order to further improve the accuracy and efficiency of the models. These networks, since they study the internal relationship of the data could be used to study LiDAR data efficiently by studying the relationship among the 3D points. It should also be researched the viability of deploying these networks at the network edge to ensure even if their performance overpower the presented ones in this thesis, they should still respect the network edge constrains. Similarly, new algorithms for edge AI will appear due to the high relevance of the IoT and edge computing during last years, and

example of these algorithms is the distributed learning. As a result of designing the algorithms for the network edge rather than compressing standard models, they will fit better the network edge constrains and take advantage of its capabilities, such as the communication between nodes.

- **Integration of other relevant sensors:** the sensors selected for this research have been selected due to their relevance in the automotive industry but other sensors that provide similar data, such as the Time of Flight (ToF) cameras could be research as a substitute of LiDAR data since LiDAR sensors are costly and require large computational power due to the quantity of data produced. However, current ToF cameras have a maximum range of few meters. Once these sensors are further optimized to increase their maximum range, they will be able to provide data similar to LiDAR (since it include depth information) but at a much reduced price.
- **Data labeling :** in this thesis, an automatic labeling tool for radar data is proposed in one of the articles. However, due to the integration of multiple sensors apart from radar and camera sensors in autonomous vehicles, a research about a flexible and general automatic labeling tool for automotive sensors would be highly useful to reduce the time required to create autonomous vehicle labeled datasets. An example would be the integration of LiDAR data to use LiDAR depth information rather than a static camera calibration to estimate the distance. As a result of this, this tool could achieve high performance results in general scenarios. Tools like this one will become more relevant in the near future, specially for autonomous vehicles where large quantity of data is required but since they use emerging sensors there are no public datasets to use.

References

[1] Thomas M Ward, Pietro Mascagni, Yutong Ban, Guy Rosman, Nicolas Padoy, Ozanan Meireles, and Daniel A Hashimoto. Computer vision in surgery. *Surgery*, 169(5):1253–1256, 2021.

[2] Vijay Kakani, Van Huan Nguyen, Basivi Praveen Kumar, Hakil Kim, and Visweswara Rao Pasupuleti. A critical review on computer vision and artificial intelligence in food industry. *Journal of Agriculture and Food Research*, 2:100033, 2020.

[3] Moran Ju, Jiangning Luo, Panpan Zhang, Miao He, and Haibo Luo. A simple and efficient network for small target detection. *IEEE Access*, 7:85771–85781, 2019.

[4] Mahdi Panahi, Nitheshnirmal Sadhasivam, Hamid Reza Pourghasemi, Fatemeh Rezaie, and Saro Lee. Spatial prediction of groundwater potential mapping based on convolutional neural network (cnn) and support vector regression (svr). *Journal of Hydrology*, 588:125033, 2020.

[5] AKRMLPJ Khosravi, RNN Koury, L Machado, and JJG Pabon. Prediction of wind speed and wind direction using artificial neural network, support vector regression and adaptive neuro-fuzzy inference system. *Sustainable Energy Technologies and Assessments*, 25:146–160, 2018.

[6] Wei Wang, Yujing Yang, Xin Wang, Weizheng Wang, and Ji Li. Development of convolutional neural network and its application in image classification: a survey. *Optical Engineering*, 58(4):040901, 2019.

[7] Samir S Yadav and Shivajirao M Jadhav. Deep convolutional neural network based medical image classification for disease diagnosis. *Journal of Big Data*, 6(1):1–18, 2019.

[8] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):712–733, 2020.

[9] Jamil Fayyad, Mohammad A Jaradat, Dominique Gruyer, and Homayoun Najjaran. Deep learning sensor fusion for autonomous vehicle perception and localization: A review. *Sensors*, 20(15):4220, 2020.

[10] SAE On-Road Automated Vehicle Standards Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *SAE Standard J*, 3016:1–16, 2014.

[11] Zhangjing Wang, Yu Wu, and Qingqing Niu. Multi-sensor fusion in automated driving: A survey. *Ieee Access*, 8:2847–2868, 2019.

[12] Ben Beklisi Kwame Ayawli, Ryad Chellali, Albert Yaw Appiah, and Frimpong Kyeremeh. An overview of nature-inspired, conventional, and hybrid methods of autonomous vehicle path planning. *Journal of Advanced Transportation*, 2018, 2018.

[13] Yao Deng, Xi Zheng, Tianyi Zhang, Chen Chen, Guannan Lou, and Miryung Kim. An analysis of adversarial attacks and defenses on autonomous driving models. In *2020 IEEE international conference on pervasive computing and communications (PerCom)*, pages 1–10. IEEE, 2020.

[14] Ziyu Li, Yuncong Yao, Zhibin Quan, Wankou Yang, and Jin Xie. Sienet: spatial information enhancement network for 3d object detection from point cloud. *arXiv preprint arXiv:2103.15396*, 2021.

[15] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3164–3173, 2021.

[16] Tai Wang, ZHU Xinge, Jiangmiao Pang, and Dahua Lin. Probabilistic and geometric depth: Detecting objects in perspective. In *Conference on Robot Learning*, pages 1475–1485. PMLR, 2022.

[17] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11873–11882, 2020.

[18] Borja Saez-Mingorance, Antonio Escobar-Molero, Javier Mendez-Gomez, Encarnacion Castillo-Morales, and Diego P Morales-Santos. Object positioning algorithm based on multidimensional scaling and optimization for synthetic gesture data generation. *Sensors*, 21(17):5923, 2021.

[19] Igal Bilik, Oren Longman, Shahar Villeval, and Joseph Tabrikian. The rise of radar for autonomous vehicles: Signal processing solutions and future research directions. *IEEE signal processing Magazine*, 36(5):20–31, 2019.

[20] Raúl Dominguez, Enrique Onieva, Javier Alonso, Jorge Villagra, and Carlos Gonzalez. Lidar based perception solution for autonomous vehicles. In *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 790–795. IEEE, 2011.

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[23] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6569–6578, 2019.

[24] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[25] Yunxiang Liu and Jinpeng Ren. Laser point cloud road 3d target detection based on deep learning. In *2021 2nd International Conference on Big Data and Informatization Education (ICBDIE)*, pages 78–81. IEEE, 2021.

[26] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[27] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[28] Jelena Kocić, Nenad Jovičić, and Vujo Drndarević. Sensors and sensor fusion in autonomous vehicles. In *2018 26th Telecommunications Forum (TELFOR)*, pages 420–425. IEEE, 2018.

[29] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

[30] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219 – 235, 2019.

[31] Wenbin Li and Matthieu Liewig. A survey of ai accelerators for edge environment. In *World Conference on Information Systems and Technologies*, pages 35–44. Springer, 2020.

[32] Javier Mendez, Kay Bierzynski, Manuel Cuéllar, and Diego Morales. Edge intelligence: Concepts, architectures, applications and future directions. *ACM Transactions on Embedded Computing Systems*, 01 2022.

[33] Javier Mendez, Stephan Schoenfeldt, Xinyi Tang, Jakob Valtl, MP Cuellar, and Diego P Morales. Automatic label creation framework for fmcw radar images using camera data. *IEEE Access*, 2021.

[34] Javier Mendez, Manuel Cuéllar, and Diego Morales. Lidar-radar robust multi-level sensor fusion for target detection at the network edge. *Elsevier Measurements - Under review*, 012 2021.

[35] Javier Mendez, Miguel Molina, Noel Rodriguez, Manuel P Cuellar, and Diego P Morales. Camera-lidar multi-level sensor fusion for target detection at the network edge. *Sensors*, 21(12):3992, 2021.

[36] Zhuoran Zhao, Kamyar Mirzazad Barijough, and Andreas Gerstlauer. Deepthings: Distributed adaptive deep learning inference on resource-constrained iot edge clusters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2348–2359, 2018.

[37] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11, 2013.

[38] Google I/O. Google tpu. Accessed: 2022-02-07.

[39] R. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34(3):276–280, 1986.

[40] Simon Haykin, John Litva, Terence J Shepherd, et al. *Radar array processing*. Springer, 1993.

[41] Felix Nobis, Maximilian Geisslinger, Markus Weber, Johannes Betz, and Markus Lienkamp. A deep learning-based radar and camera sensor fusion architecture for object detection. In *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–7. IEEE, 2019.

[42] Sorin C Popescu and Kaiguang Zhao. A voxel-based lidar method for estimating crown base height for deciduous and pine trees. *Remote sensing of environment*, 112(3):767–781, 2008.

[43] Miao Wang and Yi-Hsing Tseng. Incremental segmentation of lidar point clouds with an octree-structured voxel space. *The Photogrammetric Record*, 26(133):32–57, 2011.

[44] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1631–1640, 2020.

[45] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[46] J. Kim, J. Choi, Y. Kim, J. Koh, C. C. Chung, and J. W. Choi. Robust camera lidar sensor fusion via deep gated information fusion network. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1620–1625, 2018.

[47] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021.

[48] Shaoqing Xu, Dingfu Zhou, Jin Fang, Junbo Yin, Bin Zhou, and Liangjun Zhang. Fusionpainting: Multimodal fusion with adaptive attention for 3d object detection. *ArXiv*, abs/2106.12449, 2021.

[49] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019.

[50] Xinge Zhu, Yuexin Ma, Tai Wang, Yan Xu, Jianping Shi, and Dahua Lin. Ssn: Shape signature networks for multi-class object detection from point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 581–597. Springer, 2020.

[51] Mathieu De Coster, Mieke Van Herreweghe, and Joni Dambre. Sign language recognition with transformer networks. In *12th International Conference on Language Resources and Evaluation*, pages 6018–6024. European Language Resources Association (ELRA), 2020.

[52] Yehao Li, Ting Yao, Yingwei Pan, and Tao Mei. Contextual transformer networks for visual recognition. *arXiv preprint arXiv:2107.12292*, 2021.

Part II

Publications

Chapter 5

Edge Intelligence: Concepts, architectures, applications and future directions

Javier Mendez[1,3], Kay Bierzynski[1], M.P. Cuellar[2], Diego P. Morales[3].

1. Infineon Technologies AG, Am Campeon 1-15, 85579 Neubiberg, Germany

2. Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain

3. Department of Electronics and Computer Technology, University of Granada, 18071 Granada, Spain

ACM Transactions on Embedded Computing Systems

- Received April 2021, Accepted September 2021, Published January 2022
- DOI: 10.1145/3486674
- Impact factor: 1.193
- JCR Rank: 83/108 in category Computer science and software engineering (Q4)

Abstract: The name *Edge Intelligence*, also known as *Edge AI*, is a recent term used in the last few years to refer to the confluence of Machine Learning, or broadly speaking Artificial Intelligence, with Edge Computing. In this manuscript, we revise the concepts regarding Edge Intelligence, such as Cloud, Edge and Fog Computing, the motivation to use Edge Intelligence, and compare current approaches and analyze application scenarios. To provide a complete review of this technology, previous frameworks and platforms for Edge Computing have been discusses in this manuscript in order to provide the general view of the basis for Edge AI. Similarly, the emerging techniques to deploy Deep Learning (DL) models at the network edge, as well as specialized platforms and frameworks to do so, are review in this manuscript. These devices, techniques and frameworks are analyzed based on relevant criteria at the network edge such as latency, energy consumption and accuracy of the models to determine the current state of the art as well as current limitations of the proposed technologies. Because of this, it is possible to understand what are the current possibilities to efficiently deploy state-of-the-art DL models at the network edge based on technologies such as AI accelerators, Tensor Processing Units and techniques that include Federated Learning and Gossip Training. Finally, the challenges of Edge AI are discusses in the manuscript as well as the Future directions that can be extracted from the evolution of the Edge Computing and Internet of Things (IoT) approaches.

keywords: Edge Intelligence, Edge AI, Edge Computing, Machine Learning, Deep Learning, Artificial Intelligence.

5.1. Introduction

During the first decade of the XXI century, Cloud Computing arose as a service-based, large scale, and distributed computing paradigm (1). A Cloud platform uses a service model that can offer different services to the end user, such as: Hardware infrastructure, software, platforms (hardware+software), workspaces, data, and security solutions. With the increase in the number of devices connected to a Cloud system, the volume of data to be processed, and the growing of Internet of Things (IoT), Cloud Computing has limitations regarding the high bandwidth requirements to transmit data to the centralized Cloud architecture, high computational power to process the data, and therefore high latency of data processing (2). Different solutions have been devised to solve these problems. The main approaches are Cloudlets, Fog Computing, and Edge Computing:

- Cloudlets (3; 4) are computers, servers or clusters located in the network nodes near the end devices, that connect the devices to the Cloud. A Cloudlet is equipped with services from the Cloud in order to reduce

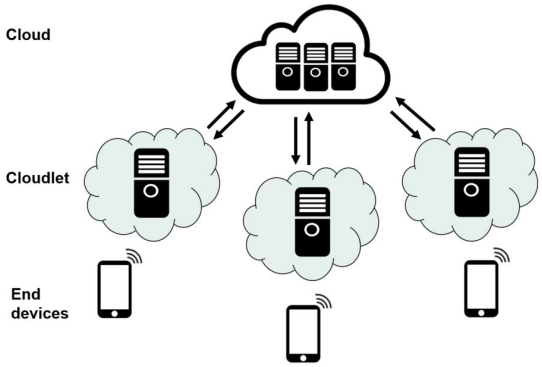

Figure 5.1: Example of Cloudlet architecture

the latency in the access to each service. As a result of this, Cloudlets may be understood as a Cloud service at the network edge. The basic structure of a Cloudlet is organized in three layers: Cloud, Cloudlet and Edge, as shown in Figure 5.1. Cloudlet has been used with two different objectives: Reducing the data transfer between end devices and Cloud, and reducing the computing workload in the end device. Examples of solutions pursuing the first objective are data caching of useful information for end devices from the Cloud in the Cloudlet, as for instance video data (5), or to improve data storage (6). On the other hand, examples of applications to reduce the computing workload of end devices are Computer Vision (image recognition, pose identification, etc.), Speech Processing (text-to-speech, speech recognition, etc.) (3), virtualization (7), healthcare (8; 9; 10) or smart grids (11).

- Cloud-Fog Computing systems (12; 13) are similar to Cloudlets in the sense that both approaches attempt to reduce the latency problems of Cloud Computing by integrating in the network some devices to process the information closer to the network edge. However, Fog computing is targeted at scenarios where multiple things are interconnected (such as IoT), while Cloudlets are more specific for network or service-oriented systems. Fog computing include capabilities to compute, store and manage data apart from networking on network nodes within the close vicinity of the Edge Device or End Device. Consequently, the main difference between Cloudlets and Fog Computing is not the difference in their network structure, since they are similar, but the final goal or task to perform as well as the computing power available. Therefore, Cloudlets should be understood as a sub-category of Fog Computing. Thus, hardware and software technologies, and specific architectures, are designed to develop systems over Cloudlet or Fog paradigms. The basic components of a Fog Architecture are Fog

devices, Fog servers, Cloud platform, and gateways, as it is shown in Figure 5.2. A Fog computing architecture is organized in layers (12) (Application, security, resource management, storage, data preprocessing, monitoring, gateways, and physical layer), and the main feature of Fog computing systems is the virtualization of the complete platform to enable different and heterogeneous devices (mainly sensors and IoT devices) to connect to the platform. Thus, Fog devices and servers (which are in changer of the networking among the End or Edge Devices) enable communication between devices, and cooperate to perform computing and storage tasks. Example applications of Fog computing are mainly related to Industry 4.0 and IoT, such as transportation systems (14), enhanced virtual reality and human-computer interaction systems (15), healthcare (10), or Smart cities (16), among others.

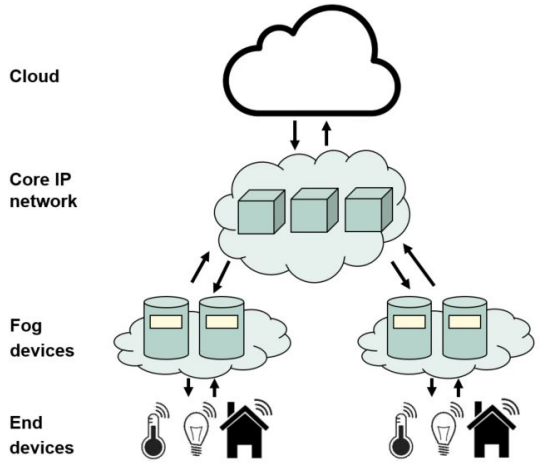

Figure 5.2: Example of a general Fog Computing architecture

- Edge Computing (17) is aimed at reducing Cloud workload to process device data, by means of performing some preprocessing and/or computing tasks at the network edge. Thus, Edge Computing is suitable to support Big Data analytics and scenarios where a real-time response is required for the user, or where the end device application has time criticality constraints. According to Figure 5.3, the general idea of an Edge Computing architecture, in the broad sense, is not much different from Fog and Cloudlet architectures. In fact, some authors classify Cloudlets and Fog computing as types of Edge Computing (18), or also as necessary technologies to enable Edge Computing (19) at different levels. However, one difference relies on the fact that Edge servers need a closer proximity to edge devices in Edge Computing, to fultill the real-time or time criticality constraints required by users and applications. Also, some of the required preprocessing can be performed at the end

device. A deeper analyze of this technology is presented in Section 5.3.

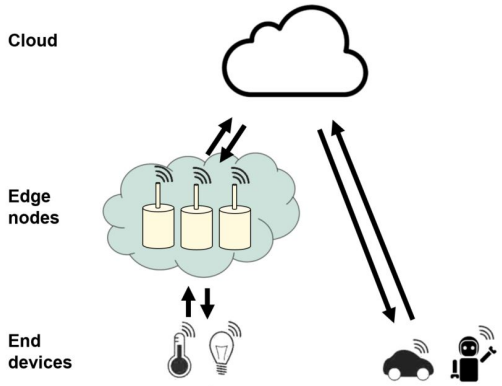

Figure 5.3: Example of a general Edge Computing architecture

- Ad Hoc Cloud computing clouds is another approach to execute process that require high computational capabilities at the network edge. This technique is based on harvesting resources from existing sporadically available, non-exclusive and unreliable infrastructures at the network edge (20; 21). However, this tasks executed in Ad Hoc clouds does not interfere with executing host processes, making it suitable for applications where large quantity of data needs to be preprocessed while the host supervise the task or execute the data gathering process. Consequently, it is possible to observe the similarities with the Fog Computing technique but in this case the nodes are specific nodes planned for the task and it is unpredictable when they will be available.
- Device-enhanced Multi-access edge computing (MEC) which is based on improving the capabilities of the MEC devices based on increasing computation and storage capacities of mobile edge devices. This technique emerged as a consequence of the increasing demand of computationally demand applications in mobile edge devices while taking into account the constrains of these devices regarding battery and computational capabilities (22). The device-enhanced MEC mechanisms can be classified into different categories based on the aim of the device (computation offloading and caching), as proposed in (23), while maintaining low energy consumption and latency results.

In this paper, we focus first on the Edge Computing paradigm, since it provides the necessary background technologies that enable Edge Intelligence. Edge Intelligence is a new term that has been used in the last few years to refer to the confluence of Edge Computing with Machine Learning (ML) (24), or Artificial Intelligence in the broad sense. Different names in the literature refer to the same concept, such as *Edge AI* (25), *Artificial Intelligence*

at the Network Edge (26), or *Intelligent Edge Computing* (27), and we use these names in the remaining of the article indistinguishably. In Edge AI, we assume that end devices are also data producers, and ML algorithms are used to process, acquire, summarize or transform this data, in the network edge nodes, and sometimes in the end devices themselves.

In addition to the benefits of Edge Computing, including ML capabilities at the network edge provides several advantages: (a) Executing the raw data preprocessing at the network edge can reduce data dimensionality to extract contextual knowledge that can be transferred to the Cloud for higher-level analyses (28) to reduce bandwidth requirements; (b) it also enables the possibility to perform sensor data fusion at the network edge, as in surveillance scenarios (29), reducing the workload in Cloud platforms; and (c) using the computing capabilities of devices and edge nodes helps to accelerate the response of distributed ML algorithms as in Big Data scenarios (30). However, some limitations arise under the Edge Intelligence ecosystem. To mention a few (31): (a) The need to adapt ML algorithms for distributed computing and distributed storage systems; (b) the management of computing and memory resources to train ML models, and to perform inferences of data using the trained models; (c) the adaptation of ML algorithms to be used at multiple time scales or data with different granularity; and (d) the design of structured solutions and standards to integrate ML in Edge Computing. Although Edge Intelligence is still an emergent paradigm, different solutions have been proposed in the literature to address these problems, and we describe the existing approaches in the following sections of this manuscript.

This article is structured as follows: Section 5.3 introduces the state of the art in Edge Computing, as a ground set of technologies to support Edge Intelligence. Section 5.4 summarizes ML approaches that have been used in Edge Computing, leading to Edge Artificial Intelligence and its applications. Section 5.5 describes Edge AI architectures and frameworks. Finally, section 5.6 focuses on challenges, future directions and conclusions of this survey.

5.2. Related Works

Multiple authors have researched this topic with the goal of summarizing the most relevant trends of the Edge Intelligence to ease the understanding of the reader such as (32), who focus on the software frameworks for the Edge Intelligence explaining at what level each of the techniques and frameworks should be applied. By doing so, the reader can get a deep understanding on what software should be used in each case as well as the reasons why. However, this author does not include in his research the edge devices where the system will be implemented. As a result, the reader can understand what technique to use but not where. Therefore, the paper does not explain the limitations of these tools that fall on the hardware, as presented in our paper.

Following this research line, (33) also research this topic focused on the algorithms as well as the software frameworks, as the previous author. Nevertheless, (33) gives some details about the hardware developed for this topic. However, the information provided in this paper is not enough to fully understand the current trends and future directions of the hardware for Edge Intelligence. This is due to the fact that, even when some specific hardware for Edge AI is discussed in the paper, most of the emerging technologies and techniques are not included. An example of this are the edge devices compared in our survey.

In contrast with this software-algorithm level survey, other works, such as (34), focus on the hardware platform to enable Edge AI. However, this approach is too specific and does not provide information regarding relevant emerging frameworks and algorithms that as highly relevant to optimize the AI models at the network edge.

On the other hand, (35) researched the overview of this technique by adding information of the hardware architecture, frameworks as well as algorithms in his paper. The current algorithms for Edge Intelligence are explained in depth with numerous examples to distinguish details between similar techniques in order to make the reader aware of when each of them should be used. (35) also researches in detail the current edge devices that can be used for the Edge Intelligence separating them according to the architecture. However, the limitations of these devices regards to the frameworks that must be used to deploy the AI models on the device are not researched. Therefore, the result is the opposite as the achieved with the previous authors, a deep understanding of the hardware without taking into account the limitations of the devices regarding the software.

Other authors have focused on some of the specific Edge Computing techniques, such as Quoc-Viet P. et al. (36) who gathered the information regarding Multi-Access Edge Computing in 5G. However, this does not provide a general overview since the manuscript aims to provide a deep understanding on Multi-Access Edge Computing rather than general Edge Computing. At the same time, this work does not include all the possible frameworks and platforms that could be used to deploy AI model at the network edge but provide general information about what requirements would need to be satisfied in order to execute efficiently ML tasks at the network edge.

Since the literature is extensive for this topic, our article aims to envelope the goals of the previously commented papers among others in order to give a deep and general overview of the current status of the Edge Intelligence. As a result, the connection between the software and hardware for this technique can be understood as well as its advantages and limitations. The current techniques to adapt the DL models to the network edge are also researched in order to understand the differences between the frameworks, since this is one of the key points to fully understand the research lines and

future trends. At the same time, the techniques that originated the Edge Intelligence are explained due to the fact they can provide information regards the future challenges/trends as well as a good background for the Edge Intelligence. Lastly, emerging techniques as well as frameworks for the Edge AI are included in this survey to give an updated overall view to the readers.

5.3. Background on Edge Computing

Edge Computing attempts to perform some data preprocessing and/or computing tasks at the network edge, instead of relying on external service providers as in Cloud computing. As a result, it enables the possibility of real-time service response at the same time the integrity and confidentiality of the information are ensured (37). This technique also reduces the energy consumption and the time required to process the information since the moment the data is generated (38; 39; 40). An example of this is studied in (40) where the energy-consumption was reduced a 40 % approximately due to the reduction on the communications. At the same time, the insecure channel to transmit the information (WiFi, Bluetooth, etc.) can be removed from the hierarchical structure of the data processing (38) in a Cloud system. During the preprocessing of the raw data, the confidential information which is not relevant for the final task can be masked/deleted before being shared with an external device. This reduces the risk of confidential or sensible information leakages.

An indirect benefit of the Edge Computing approach is the abstraction of the end device. When heterogeneous end devices are used to gather information that complements each other, an adaptation layer is required to transform all the data into a common structure. In Edge Computing, this abstraction layer is included inside each end device where relevant information is shared exclusively for the fusing process (41). This strategy alleviates the computational power required to fuse the raw data in Cloud servers.

The limitations of this technology fall on the hardware and computing capabilities of the end devices at the network edge. Usually, these devices do not have high computational power as well as large battery due to their size restrictions, making it difficult to integrate advanced algorithms to process the data. A trade off between processing speed and energy consumption must be achieved (42), and this is a current challenge of Edge Computing.

Using Edge Computing does not imply excluding the Cloud Computing or other similar technologies like Fog Computing. The Edge computing could be a layer in a hierarchical processing structure where the data is preprocessed or/and processed at a closer level to the network edge which has capabilities to execute the task (43; 30; 44; 45).

Due to its relevance, it is worth mentioning a special case of Edge Computing applied for telecommunications, i.e. *Mobile Edge Computing* (MEC)

(46; 47; 48), since a large amount of literature can be found specifically for this application area. While traditional telecommunication operators perform control of the traffic flow in Cloud, MEC deploys edge servers in base stations following the Edge Computing structure. The first MEC system was developed by Nokia Networks in 2012 (49). In Nokia's concept, MEC servers are standard telecommunication equipment where storage and processing capabilities are integrated to collect real-time network data. This technology reduces the latency by using the same principle as the Edge Computing and Fog Computing, moving the processing center near the network edge. At the same time, due to its location awareness, like Fog Computing, it enables developers content providers and users to perform tasks where the location data is a key factor.

5.3.1. Application scenarios of Edge Computing

The Edge Computing, as well as Fog Computing and Mobile Edge Computing, could be applied where latency is an essential factor and the Cloud Computing is not adequate. Some examples of these applications are:

- **Autonomous driving.** In autonomous driving (50; 51), time is crucial to decide the actions made by the car for safe driving following the correct route. The system needs to execute the task with a short time constraint. It is not efficient for the raw data to be sent to another device where it will be processed to execute a task. Instead, the raw data should be processed in the vehicle to reduce the latency, for example, latency of 0.1 seconds at 100km/h implies that the vehicle would roam for, approximately, 3 meters before executing the computed result. In the survey (50), this restriction is studied to prove the relevance on the fast data processing for autonomous driving since otherwise it could lead to problems such as the previously commented. At the same time, this survey presents the requirements for the Central Processing Unit (CPU) and Graphic Processing Unit (GPU) devices that would be required for the Edge Computing in autonomous vehicles as well as V2X applications in order to enable the information sharing between vehicles and computation offloading. Following these restrictions, the paper of (52) proposes an Edge Computing based framework (F-Cooper) to fuse the information. In this case, the data collected by each vehicle is preprocessed to obtain point cloud features, which are later fused to achieve a better object detection performance. Another example is the pedestrian detection method developed by (53) where data collected by a LiDAR sensor is propsed to obtain relevant features such as the movement and the dimensions/shape of the objects. The paper (54) provides a complete survey of techniques used in this field.
- **Security solutions.** Another application for this technology is secu-

rity solutions, such as surveillance systems where the cameras or a device at the edge of the network could process the raw data to infer a result or fuse the information (55; 29). It could be used to recognize specific users, for example, known criminals, before any dangerous action could be started. In the paper of (56), this technique is used to control vehicle speed on highways by cameras, which obtains successful results on its implementation. Following the security line of work, (57) review some security measures (Edge-based Authentication and Authorization Mechanisms and Edge-ISP collaborative architecture to detect and isolate IoT security attacks among other techniques) which can be implemented in distributed networks such as IoT to ensure the privacy of the data while maintaining the functionality of the system. In the vehicle context, Edge Computing in collaboration with security techniques such as Blockchain can be implemented to ensure data privacy as well as efficient energy interactions as proposed by (58). This paper proposes a framework for efficient implementation of these technologies while maintaining low energy consumption and real-time data transmission.

- **IoT applications.** In the IoT scenario, where there are numerous sensors, this technology can be beneficial (59; 60; 61; 62). In this case, it would be more efficient to preprocess the data before sending it to the cloud servers. In case the devices have enough computational power for the task, the process could be executed in the devices themselves or within a distributed end devices network. In the paper (59), an Edge Computing framework is developed, called WuKong, to fuse the information collected by heterogeneous sensors. The authors applied this framework for online activity recognition, where the results obtained support the smart home paradigm. Following the same line of work, in the paper (60) the concept of smart city is studied in collaboration with Edge Computing as a technique to process the data generated by distributed and numerous sensors of the smart city concept.

  When all the data from the IoT sensors have been gathered, security is a key factor due to the nature of the data. Therefore, in the paper of (57) some security solutions for IoT based on Edge Computing are reviewed. These solutions include User-centric edge-based IoT security architecture, to establish a trusted domain at the edge layer, Device-centric edge-based design for IoT security, which is based on security frameworks integrated in the Edge Devices to increase their security using advanced security algorithms, and End-to-end security for IoT, which is the most challenging due to the heterogeneity of the Edge Devices and frameworks.

- **Location services.** Edge Computing can be applied to location servi-

ces such as smart-parking explained in the paper (63) where a network based on Fog Computing in collaboration with the network edge and the cloud is used to determine the location of available parking slots. This not only helps users look for a parking slot but also distributes vehicles parking slots efficiently, avoiding unused resources in the city. In this context, the location of the end device is relevant for the task it is executing. The edge device can fuse the data with the location to provide information of the context before transmitting this information. (64) also researched a similar application where Edge Computing was used for license plate number detection, smart parking meter and vision-based parking spot detection based on a distributed camera system and Deep Learning algorithms with low latency. Other services which can be carried out at the network edge taking into account the location by the implementation of Edge Computing are indoor localization and navigation system as proposed by (65) which can be used for patient tracking in hospitals based on Bluetooth Low Energy (BLE) and IEEE 802.15.4a compliant Ultra-wideband (UWB) RF time-of-flight based positioning at the network edge.

- **Network functions.** Although network solutions have been traditionally implemented in hardware, it is possible to turn these devices into software that can be integrated into edge devices by using Network Function Virtualization (46). Moving this service to the network edge enables the privacy of the systems to be improved, at the same time, it is possible to reduce the latency to achieve real-time responses. An example of this application is the paper (46), where some techniques to implement communication protocols at the network edge are explained in depth. Following this line of work, (66) proposed using Edge Computing techniques for tasks such as performance monitoring, physical impairment evaluation, and alarm message filter among others. The alarm prediction scenario was implemented and tested, reaching a 99 % accuracy. As well as before, (67) proposed using Edge Computing technologies as a gateway to support efficient deployment of ML models based on containers to reduce bandwidth consumption, increase the security of the systems, ensure efficient resource usage, and conduct life-cycle management among other applications.

Although Edge Computing provides the capabilities required for heterogeneous applications, specific software and hardware are needed maintain the pace in order to incorporate all these advantages. In the next section, current hardware architectures and software platform for Edge Computing will be explained for a deeper understanding of the main relevant features.

5.3.2. Edge Computing Architectures and software

Edge Computing attempts to perform the data processing at the source device, whenever it is possible. However, CPUs do not have enough computational power to execute most of complex tasks under real-time constraints, so specific process units are required in Edge Computing end devices. For instance, Facebook's researchers have already explained how most of the inference process of their applications are carried out by the end device where it is used, like smartphones or tablets (68). There are other options to improve edge computational capabilities, such as adding more CPUs to the device to parallelize the process, or integrating other co-processors such as Graphic Processing Units (GPU) (69; 70), among the most popular approaches.

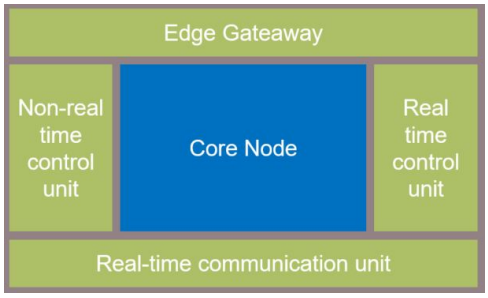

Figure 5.4: Edge Computing device architecture proposed by (71)

Some researches propose a modular architecture for Edge Computing devices (71; 72) where real-time is ensured at the same time non-real-time and real-time control units perform a control of communication protocols as well as memory allowance, as shown in the figure 5.4. A central core would run an operating system and track the resources of the device to decide where to execute each task of the global process. Besides this, the device should be accessible for updating purposes in order to enable new functionalities and improve current ones. Scalability is also a key factor in Edge Computing. Therefore, devices should be aware of neighbor devices as well as their functionalities to establish efficient network resource management. The communication protocols of the device must enable communications not only with other edge devices but also with other devices at the network edge such as Fog nodes or MEC nodes in order to transfer information between them efficiently (72; 73).

Another proposed architecture is the Lambda-CoAP Architecture (74), which starts from the Lambda Architecture (75) deployed in the cloud to abstract heterogeneous devices at the network edge while working on real time. This architecture is composed of different modules that provide the edge devices with functionalities for processing, analyzing and consuming data. It splits the input data into three different layers in order to reduce the latency:

- **Real time layer.** This layer processes data that needs to be computed on real-time.
- **Batch layer.** This layer preprocesses data to generate batches using the processing of historical data.
- **Serving layer.** This layer displays and communicate the computed results. At the same time, by using this layer it is also possible to access all generated data.

This architecture must be complemented with an adequate communication protocol or gateway to connect the devices of the network as in the previous architecture.

As we can see, both architectures follow the same general structure with different implementations. This common idea is the processing pipeline which is shared among multiple architectures for Edge Computing (76; 77). Thus, the processing pipeline consists of the following elements: data gathering, data preprocessing (homogenization, filtering, masking private information, etc.), result computation, result storage, and system interface.

After establishing the hardware architecture, an adequate software framework is necessary in order to establish efficient communication protocols at the network edge, as well as configuring each node. Therefore, current existing software frameworks for Edge Computing are:

- **Apache Kafka**. This framework was developed by Apache to be a distributed streaming platform on real time. It is based on the idea of multiple devices checking the same information simultaneously. By establishing which nodes can consume and/or produce data, this framework enables a multi-tenancy approach to control the information distribution. Due to its characteristics, it can be used as a communication protocol for a distributed network as well as data storage for end devices and applications where historical data is required. The combination of these functionalities is not common but it enables this platform to work as a streaming platform (78).
- **FAR-Edge RA**. It is another Edge Computing framework developed for the H2020 FAR-Edge project (79) to adopt decentralized automation architectures. It allows the collaboration with external frameworks or systems such as the Cloud. This framework is notable for guaranteeing the security of the data through techniques such as Blockchain. By including the functionalities of the Blockchain, this framework can be used as a reference for automation cases, analysis, and simulations in industrial environments (80).
- **F-Cooper**. This framework was developed with special capabilities for vehicles in the paper (52). It proposes an edge computing-based

approach to fuse the information from point cloud features to achieve a better object detection performance.The data was collected by different LiDAR sensors (Velodyne HDL-64E LIDAR) integrated in the vehicles in a collaborative smart vehicle scenario.

- **Macchina.io**. It is a new IoT platform for the network edge that integrates a software tool for developing purposes as well as tools for resource managing of the system (81; 82). It hosts the application server, based on java-script, to manage the communication protocols. When using this software, edge devices can be managed through the Macchina platform, which can be accessed through common web browsers and SSH among other communication protocols.

- **OGEMA**. It is an Edge Computing platform developed by (83). This framework is meant to work in a heterogeneous context where there are multiple end devices connected together, such as IoT or industry 4.0. It allows cloud connection to transmit the information as well as relying on the cloud for some computation when required. In order to ensure the interoperability between end devices at the network edge, this framework maps the collected data to standardized data types. This framework integrates a variety of security levels to fit the application.

- **CRESCO**. It is a free and open-source distributed-agent framework (84). This framework implementation is designed for large quantities of heterogeneous end devices geographically distributed. It supports real-time processing at the network edge for IoT applications among others. In order to ease the implementation and development, an application description language was developed by the same company.

- **Edge-computing-embedded-platform (ECE platform)**. This framework was proposed by (85) as an edge computing-oriented platform to remotely control geographically distributed IoT end devices, independently of their network configuration. It reduces the bandwidth consumption by using asynchronous communications, which enables faster communications and real-time capabilities. A lightweight virtual space is obtained in this framework as result of integrating the Docker technology, which reduces the memory footprint relevant for low-resource devices at the network edge.

- **SA Framework**. The purpose of this framework is to detect the current situation of the environment where it is implemented (60). By an efficient understanding of the context, it is possible to perform actions efficiently. This framework is designed for IoT applications specifically due to the fact it takes into account the information propagation in IoT networks.

- **Cisco Fog Director**. This framework integrates capabilities which are beneficial in situations such as smart cities to manage large quantities of nodes. It integrates APIs for an easy human-machine interface (HMI) when it comes to developing applications and debugging support. This framework provides improvement over operational effectiveness over business agility (86).

- **Crosser**. This framework enables the collaboration between heterogeneous systems through its built-in orchestration capabilities, including edge-cloud collaborations. This lightweight IoT-edge platform can collect data on real time from the sensor of the network as well as preprocessing them. It is good at obtaining low cost Machine-to-Machine communication, lowering bandwidth and establishing cloud-independent networks (87).

- **EdgeX Foundry**. It is a linux-based open-source framework for IoT-edge applications. It includes an operative system-agnostic plug-and-play-based software for easy development of systems based on it. This framework is deployed for controlling the evolution of key values by data orchestration and edge system management (88).

- **Edgent**. It is an Apache open-source framework to develop edge applications. The edge devices allowed in this framework are those based on java7/8, Android or Raspberry Pi B. It works as a back-end system for edge devices and applications. Due to the limited devices which can be implemented with this framework, Edgent is used as an interface media among similar devices (89).

- **Edge Computing RA 2.0 (EC RA)**. This software architecture was proposed by the collaboration of Edge Computing Consortium and the Industrial Internet Alliance, based on international standards such as ISO/IEC/IEEE 42010:2011. This framework includes the following services: management of the end devices, data life cycle, and security. It can be implemented in collaboration with other systems due to its horizontal structure as well as provide real-time response (90).

- **Industrial Internet Consortium RA (IIC RA)**. The IIC developed its reference architecture using some international IoT standards (ISO/IEC/IEEE 42010:2011). This structure is based on three layers: (a) Edge, where the data from the end devices is collected, (b) Platform, where the data is processed and sent to the third layer (c) Enterprise, where specific applications are hosted such as support systems or user interfaces among others. This layer is also responsible for controlling the execution of the previous layers (91).

- **PiCasso**. It is a platform for lightweight service orchestration at the network edge (92). The key factor for an efficient performance of this framework is the discovery of the critical parameters of the use-case for specific applications. This framework includes strategies such as local replication and remote replication to fit the application after the critical parameters are known.

In order to compare the previously commented frameworks, some relevant features for the Edge Computing have been selected based on their relevance in this field. These selected features are:

- Real-time processing. This feature describes if the framework is able to be executed on real-time or with time constraints.

- Heterogeneous devices. Edge Computing paradigm includes heterogeneous devices due to the diversity on information that can be extracted at the network edge, leading to numerous sensors that may not share the output data format or communication protocols. This is a key factor in numerous scenarios so the capability of the framework to be implemented in diverse and heterogeneous devices is a key feature to be compared.

- Cloud connection. This connection can expand the resource capabilities of the system when the application requires it due to computational or storage requirements.

- Device management capabilities. This feature must be understood as the capability to manage the connections among multiple edge devices as well as the network.

- Scalability. This feature must be understood as the capability to increment the size of the edge network easily.

- Security. This is a key feature as well as one of the main reasons which led to the creation of Edge Computing. This feature will be used to describe if the framework includes any security technique by default in order to ensure the reliability of the data and its privacy.

- Resource Optimization. The capability of the framework to optimize the resources of the system such as bandwidth and energy.

- Open Source. If the framework is Open Source, the scientific community can work on new capabilities for the system as well as fixing problems leading to an improvement in the efficiency and performance of the framework.

Devices	Real-time	Heterogeneous nodes	Cloud connection	Device management	Scalability	Security	Resource optimization	Opensource
Apache Kafka (78)	✓	✓	✓	✓	✓	✓	✓	x
FAR-Edge RA (79)	✓	✓	✓	✓	✓	✓	✓	✓
F-Cooper RA (52)	✓	x	x	x	✓	✓	x	✓
Macchina.io RA (81; 82)	✓	✓	✓	✓	✓	✓	x	x
OGEMA RA (83)	✓	✓	✓	x	✓	✓	✓	x
CRESCO RA (84)	✓	✓	x	✓	✓	✓	x	✓
ECE (85)	✓	✓	x	✓	✓	✓	✓	✓
SA (60)	x	✓	x	✓	✓	x	x	x
Fog Director (86)	x	✓	✓	✓	✓	✓	x	x
Crosser (87)	✓	✓	✓	✓	✓	x	✓	x
EdgeX (88)	x	✓	✓	✓	✓	✓	✓	✓
Edgent (89)	x	x	x	✓	✓	✓	✓	✓
EC RA (90)	✓	✓	✓	✓	✓	✓	✓	x
IIC RA (91)	x	✓	x	✓	✓	✓	✓	x
PiCasso (92)	✓	✓	x	✓	✓	x	✓	x

Table 5.1: Comparison of Edge Computing frameworks

Table 5.1 compares all the described frameworks according to these functionalities.

As it is shown in Table 5.1, all these frameworks include capabilities for scalability due to its importance for Edge Computing where the number of nodes is not pre-defined in some applications. On the other hand, features such as Cloud connection or Opensource are present only in half of the frameworks. Real-time capabilities are included in the majority of the studied frameworks because of the relevance of this feature for Edge Computing. Security techniques are also present in all the frameworks except for the SA and Crosser, where programmers must include the security in case it is necessary, leading to a more complex implementation.

As concluded from this table, scalability, heterogeneous nodes, security and device management are considered to be the most popular features for Edge Computing frameworks while Opensource, resource optimization and Cloud connection are not so frequently included in the frameworks.

Recently, the increasing demand for AI oriented services leads to an in-

crease in the global bandwidth. As a result, new lines of work are emerging to study the movement of the computation of the AI algorithms from the Cloud to the Edge, consequently providing a confluence of AI algorithms with the Edge Computing techniques under the term *Edge Intelligence* which will be discussed in the following section.

5.4. Edge Intelligence

Edge Artificial Intelligence (Edge AI) or Edge Intelligence can be understood as the confluence of Artificial Intelligence and Edge Computing (93). The goal of this technology is to bring AI capabilities to the network edge. Therefore, Edge AI end devices must have enough computational power to run AI inference algorithms that process data with time constraints, and, in specific applications, learning algorithms as well. Hence, Edge AI performance indicators differ from AI systems running in high computational resource systems (94). Besides accuracy, there are more key factors which must be studied such as (59):

- Energy consumption requirements, since edge devices use to work as individual nodes of a distributed network without connection to the electrical network.
- Real-time, or soft real-time constraints to give a response to the required AI service.
- Frequency of communication with other nodes, to establish the dependency of the nodes with the central network to process the data.

Taking into consideration the resource limitations of the Edge AI devices, the AI algorithms need to be adapted for optimal execution in a distributed environment with low-power devices. To accelerate the AI algorithms, the memory access must also be carried out in an efficient way to avoid redundant accesses, which decreases the memory access bottleneck at the same time that the energy of the device is not used for redundant tasks (95; 93).

In the paper of (32), different levels of the implementation of Edge Intelligence were proposed, as shown in Figure 5.5, understanding these levels as the level of migration of the ML algorithms from the Cloud to the network edge. The levels are summarized as follows:

1. Cloud Intelligence: where all the processes of the ML rely on Cloud Computing.
2. Cloud-Edge Co-inference and Cloud training: the training of the ML algorithms rely on the Cloud Computing due to its high computing capabilities, while the inference process is executed as a collaboration between the cloud and the edge by data offloading.

3. In-Edge Co-inference and Cloud training: as in the previous level, the training relies on the cloud but this time the inference process is carried out within the network edge, by fully or partially offloading of the data.

4. On-Device Inference and Cloud-Edge co-training: training process in the cloud and edge while inferring the result in a fully local on-device approach.

5. Cloud-Edge co-training and on-device inference: both processes are carried out in a collaborative approach between the Cloud and the Edge.

6. All In-Edge: the training and inference processes are carried out at the network edge, understanding the network edge as a collaborative network of edge devices.

7. All On-Device: training and inference processes are carried out within the edge device.

Figure 5.5: Levels for the Edge Intelligence application developed by (32).

Although Edge Intelligence has been devised to run Artificial Intelligence algorithms in general at the network edge, current trends and applications focus on Machine Learning, and more specifically Deep Learning. For this reason, Subsections 5.4.1 and 5.4.2 analyze emerging and relevant AI techniques to reduce the workload in the edge devices in depth. Subsection 5.4.1 focus on general Machine Learning techniques to provide a general overview while Subsection 5.4.2 focus on emerging Deep Learning techniques.

5.4.1. Machine Learning algorithms at the Edge

ML algorithms have been already adapted and deployed at the network edge using the classic AI concept of Intelligent Agent (96). An Intelligent Agent can be understood as an autonomous, rational agent that perceives the environment, and selects an action or sequence of actions to perform a

task according to its internal representation, current (and sometimes past) observations, and a set of rules or AI model inference engine.

The execution of AI algorithms at the network edge requires reducing the size of current AI models while maintaining their performance in low-resource devices. Decision Jungle (97) is an example of this approach to implement decision trees. Decision Jungle is the term to define ensembles of rooted decision directed acyclic graphs that are trained level by level while optimizing an objective function taking into account for this process the structure of the graphs. Following this technique, successful compressing results have been obtained with different datasets like Kinect (98), where the final Decision Jungle size was of 9 MB from the initial 80 MB, or Faces dataset (99), where the initial size of 7.17 MB was reduced to 1.72 MB, obtaining state-of-the-art accuracy (97). Similar proposals have been devised with Random Forests (RF) (100). The compression process starts with the training phase of the RF using followed by a pruning phase where each decision tree is pruned while attempting to meet a global resource constrain and performance (101). As a result, a global optimal pruning of the RF to minimize the error is learned. Empirically, the results of this pruned RF outperforms state-of-the-art compressed algorithms (100). In the paper of (59), the RF technique was implemented for human activity recognition at the network edge after a first feature extraction from the data. The paper obtains successful results, with an accuracy between 94 % and 98 % depending on the volume of the data studied. This technique, as well as the Jungle Decision, is scalable to large datasets and obtains high accuracy.

Another popular algorithm used in Edge AI is the *k-Nearest Neighbours* (kNN) classifier (102). There are multiple variations of its adaptation to Edge Intelligence, such as *ProtoNN* (103) where the clusters are compressed in order to fit the limitations of low-resource devices. This technique makes a projection of the entire dataset into lower dimensional data, by means of dimensionality reduction and feature selection. This approach can be deployed in devices with small memory capability in the order of approximately 16KB, while producing start-of-the-art accurate results in numerous benchmarks.

Some application-specific ML techniques such as computer vision have been implemented at the network edge, for example, feature extraction from images with techniques including eins-gradient and histogram of gradient orientation followed by a classification method. This technique was used in the paper of (104) for human detection in collaboration with SVM for classification on the MIT pedestrian database (105), where it achieves state-of-the-art results. Following this line of work, (106) proposed using HOC and linear SVM classification for preceding vehicle detection in the context of smart vehicles based on monocular vision. In the paper (106), this technique was tested under different scenarios such as simply structured highways, complex urban environments, and local occlusion conditions, where it provi-

ded good results. ML algorithms can also be implemented for Natural Speech Recognition such as Dragon Naturally Speaking developed by Nuance (107).

Apart from the previously commented techniques, other ML algorithms for tasks such as prediction or regression have been implemented at the network edge. (108) introduced a technique to train ML models at the network edge based on gradient descends (i.e. SVM, K-means and linear regressions) without relying on external devices such as cloud servers. The approach consists in using multiple edge devices to execute the ML task, thus solving the problem of the restricted computational capabilities by using a distributed network, followed by a final edge device, which aggregates the results generated by the other edge devices. This final edge device must also process or preprocess the collected data for adaptation purposes and transmit the final data for further processing. To test this technique, (108) used three Raspberry Pi 3 and a laptop as the edge devices of their system and another laptop as the aggregator edge device obtaining close to the state-of-the-art results when evaluating SVM, K-means and linear regressions.

The aforementioned algorithms share the same limitation: the time complexity increases with the quantity of data. When facing this problem, there are other algorithms/techniques that can be more suitable when large volumes of data must be processed, such as Deep Learning (DL) models.

5.4.2. Deep Learning at the Network Edge

Deep Learning is based on artificial neural networks (ANN) and consists of a number of layers with artificial neurons in those layers to compute a result. The data is computed in each layer, where matrix multiplications are applied to compute the input of the subsequent layer until the final layer, where a final classification or feature is obtained. When the structure includes a large number of layers, it is known as Deep Neural Network (70; 94; 109). This technique includes different DNN structures developed for specific applications like the case of Convolutional Neural Networks (CNN), designed to fit image and video processing, where convolutional filters operations are applied apart from matrix multiplications. For time-series processing, there are DNNs called Recurrent Neuronal Networks (RNN), which include loops in the internal connections of the neurons by means of memory to store past inputs (109).

As a consequence of the high computational power required for the training phase in DL, Cloud Computing was the default approach for the training and inference phases before the current constraints of time and security in applications such as autonomous cars or health care, based on Edge Computing (95; 110). To fit these constraints, new approaches for DL at the network edge are emerging (82; 94): (a) training the DNN using a traditional approach (Cloud Computing) and moving the inference process to the network edge, shown in Figure 5.6(b), or (b) perform the training and inference in the Edge

device, as shown in Figure 5.6(c).

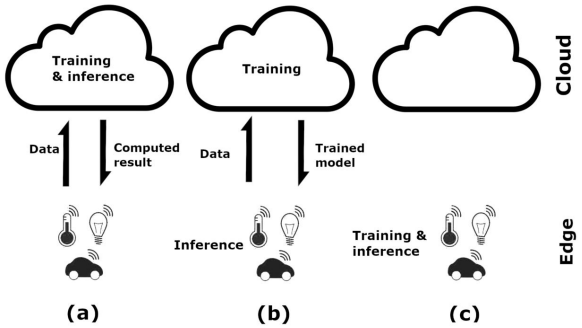

Figure 5.6: Different approaches for training and inference. (a) Traditional training and inference in Cloud servers, (b) training on Cloud server and inference at the Edge and (c) training and inference at the Edge.

Depending on the approach, the resources for the end device must fit different requirements (82). When the training process is carried out at the network edge, the device must include a computing unit that enables this process by integrating high computational capabilities as well as a sufficiently large memory to store the data necessary for this task. When only the inference is carried out on the device, the memory necessity becomes more relaxed, since it only needs to store the trained model.

Besides specific hardware for the training process of the DL models at the network edge, researchers have developed new techniques for the training process which reduce the memory footprint in the edge device as well as increase the training speed in low-resource devices. These techniques are listed below:

- **Pruning.** Based on the synaptic pruning of biological brains and their connections during the life cycle, this process consists in removing connections between neurons that are not relevant for the application, reducing the number of operations computed when studying new data. As well as removing connections, it can also remove neurons that are classified as no relevant when most of its weights have low values in comparison with the global context of the DNN. This technique enables faster, smaller and more memory efficient DNN which can be implemented in low-resources devices like Edge devices (111; 112).

- **Weights quantization.** Besides removing unnecessary connections and/or neurons, all the weights are stored as individual values. This configuration is not memory efficient due to redundant values. The weights quantization technique intends to cluster similar weight values in a single value and reduce these values to integers or numbers which

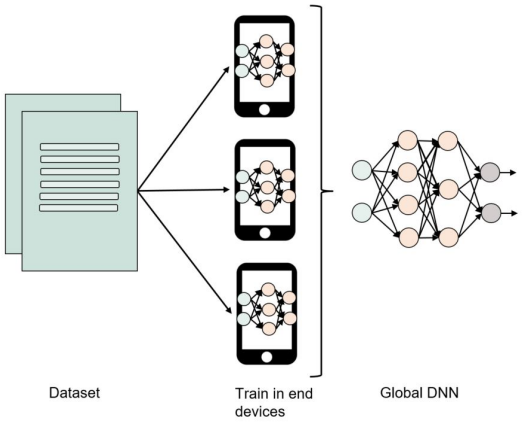

Figure 5.7: Example structure of DNN model using Federated Learning.

occupy the least bits as possible. Therefore, the weights will be re-adjusted, which implies the accuracy will be modified as well. This leads to a recursive implementation where, after each training, the weights are quantified (113).

- **Federated Learning and Model Partition.** When facing complex tasks or a training phase with a large volume of data, the approach for the training could be a distributed learning or Federated Learning (114). The data would be split into smaller groups which would be distributed amongst the nodes of the edge network, as it is shown in Figure 5.7. Each node would train based on the data it received, training this way part of the final DNN to achieve active learning capabilities at the network edge. The same approach can be followed for the inferring phase, where the technique is known as Model Partition. In the Model Partition each layer of the DNN would be computed by a different node to distribute the workload (94). This approach would also allow easy scalability.

- **Transfer Learning.** This technique is based on the existence of similar already trained models and the possibility to transfer the knowledge between DNN (115). It consists on maintaining the weights of the first layers of a trained DNN on a source domain similar to the target domain. These layers are trained to extract the main features so it is possible to extrapolate its application to extract the features of the new data. These features will be used for a later classification based on the last layers of the DNN which will be specifically trained for the application. By doing this, it is possible to obtain a well trained DNN on a large dataset with a structure that has been proven to provide good results. This technique is recommended when there is not enough

information in the dataset for adequate training at the same time as the complexity of the training phase is reduced, leading to a faster and less resource consuming process.

- **Knowledge Distillation.** Knowledge Distillation emerges as a novel form of Transfer Learning (116; 117). This technique can extract knowledge from a large and well-trained DNN, called teacher in this context, into a reduce DNN, called student. By doing this, the student network can learn to obtain the same results as the teacher network but increasing the computing speed by reducing the size (using techniques like pruning previously commented).

- **Gossip Training.** This technique aims to reduce the time required to train a model at the network edge based on randomized gossip algorithms. This algorithm has the advantage of full asynchronization and total decentralization. Each of the nodes of the network executes the training of a fraction of the whole data and during these random communications, the different nodes compare their results until a consensus is achieved (118; 119). By using this technique, it is possible to implement active learning at the network edge since the workload during the training would be distributed among multiple devices rather than computing the whole task in a single device.

5.4.3. Application scenarios of Edge Intelligence

As new emerging technologies like IoT or Big Data become mature, more systems start integrating sensors for gathering data in real time, which also require to be processed on real time or with time criticality to detect emergencies or relevant events. The quantity of sensors for data gathering will keep increasing to improve the vision of the context, which means a larger volume of data (69). Processing the data with the traditional technique of Cloud Computing will not catch up with the speed the data is generated due to the bottleneck to upload and download the information from all the sensors. This situation is when Edge Computing provides an efficient solution by computing the data at the edge of the network (59; 43; 30).

The common theme across the applications where Edge AI can be implemented is that they are complex tasks where ML has been shown to provide good results, and these tasks need to run in real time at the same time the privacy and security of the data are ensured.

- **Computer vision.** Deep Learning is considered the state-of-the-art technique for image processing (classification and detection) which is a fundamental task for computer vision. The data for these tasks is generated at the network edge by surveillance cameras, time-of-flight

cameras and motion detectors among other sensors that already integrate DL capabilities in some occasions. Uploading the data from these sensors to the cloud to be processed implies reducing the frames per second the system computes. At the same time, it may incur privacy concerns, especially if the data contains sensitive information. The bottleneck due to the large quantity of data being uploaded from all these sensors is another reason to implement Edge Computing for this task. There are some examples of Edge AI implemented for computer vision like "Vigil"(120). This system consists of a network of wireless cameras that process the information at the edge of the network. These edge nodes provide information about relevant frames to be studied further. By preprocessing the images at the edge, Vigil reduces the bandwidth consumption in comparison with the traditional approach of uploading the raw data to be studied. Another example is Video-Edge (121), where a hierarchical structure is deployed, enabling Edge and Cloud to work in collaboration in order to balance the workload while maintaining the low bandwidth requirement.

- **Natural language processing.** ML, specifically DL, has become one of the most popular approaches for Natural Language Processing (NLP) tasks (122) such as speech recognition (123), machine translation (124) and entity recognition (125). Using ML enables the reduction of latency down to hundreds of milliseconds, following the constraints of Edge Computing. Voice assistants of different companies are examples of this application, such as Echo Dot from Amazon, .ok Google"from Google and Siri from Apple. While part of the processing of the data collected by these devices is carried out in the cloud, some significant tasks have to be executed at the network edge. An example of this is the keyword detection for the activation of the system. The rest of the message is sent to the cloud to be computed and the result is returned only if this keyword is detected. In the case of Siri, the keyword detection is carried out by two ANNs (126). The first one, a reduced size DNN with low-power mode, studies all the data the system is collecting to determine if the second one must be initialized. Once the second DNN, a deeper DNN than the previous one, is triggered, the main processor determines the keyword has been detected. Keyword detection techniques need to be further improved in order to be adapted to low-resource devices at the network edge. Some techniques like pruning or quantization can be integrated into these DNNs to compact them for the edge devices. In order to detect keywords, researchers from Microsoft developed a reduced size RNN which fits in 1 kB of memory following these ideas among others (127). For other NLP tasks, latency remains a significant issue because of its complexity to be reduced.

- **Internet of Things (IoT).** Data collected by some IoT sensors requires an automatic understanding in real time for use cases such as smart grids, smart cities, smart cars or healthcare. DL has been proven to obtain successful results in some researches about these topics like activity recognition (128; 59) and pedestrian detection (129). Sometimes in these tasks it is necessary to fuse the information collected by different sensors which may need to be preprocessed before being fused in order to obtain accurate results. There are different frameworks such as DeepSense (128), WuKong edge framework (59) or NeuroPilot (130) which support this by leveraging the spatio-temporal relationship of the data from different sensors. Compressing the DL models to fit into IoT devices, which have limited resources for computation and battery, is another approach that must be researched. Some examples of techniques to compress DL models have been discussed in section 5.4.2.

  At the same time, by using Edge Computing for IoT applications, it is possible to solve the problem of data privacy concerns due to the fact that it can contain sensitive information or public information from street, cities, etc. where the owner of the device does not hold the right to share the information (131). Therefore, Edge AI may solve the problem of privacy while reducing the bandwidth consumption (which is high when cameras or complex sensors are involved). An example of this application is the system proposed by (132), where Edge Intelligence was used in the context of smart homes, obtaining successful results that open new lines of work about its application in other scenarios like smart industries or smart cities.

- **Virtual Reality (VR) and Augmented reality (AR).** Researchers have proposed DL as a successful technique to predict the field of view of the user when using virtual reality (VR) (133; 134). The goal of the DL in this application is to determine the region of the 360° video or images that must be fetched on real time to minimize the frame drops, maximizing that way the user experience. DL can also be used in Augmented Reality (AR) to detect objects in the field of view which can be key objects to apply virtual overlay on top of them or to start an activity (135; 136). Since depending on the cloud for the processing of these tasks can incur into high latency, Edge AI is studied as a solution due to its low latency. This reduced latency is needed to provide a successful performance, as proved in the Gabriel system of Google for Google Glass (137)

5.4.4. Analysis and discussion

Different algorithms have been implemented at the network edge following this approach such as Intelligent Agents, Decision Jungles, Computer Vision or Speech recognition algorithms to match different applications. Due to a large amount of data generated at the network edge and the necessity of fast processing, these techniques have been devalued in favor of Deep Learning models. Deep Learning is a technique specialized in working with a large quantity of data for regression and classification among other tasks. However, Deep Learning was traditionally implemented in the Cloud due to its higher computational power than the network edge. This approach was suitable for the training and inference in the past but due to the low latency required for emerging applications, Edge Computing (specifically Edge Intelligence) has become the most efficient approach.

Current DL algorithms are developed to be executed in high-resource devices but not in Edge devices, therefore, new techniques to adapt these DL models to the network edge have been researched. This has led to emerging techniques such as quantization, federated learning, knowledge distillation, and pruning. These techniques reduce considerably the size of the DL models and enable their implementation at the network edge. They may not be suitable for all applications, for instance when the accuracy is fundamental, pruning and quantization should not be used alone due to the accuracy reduction they may imply. As the iterative method to design a DNN, these techniques must be tested to establish which one matches the specific application under test.

Table 5.2 compares the previously explained techniques to be able to understand their main differences, which can be an initial step to decide which one to use. The features to compare are: (1) whether this technique reduces the memory footprint of the DL model, (2) whether the technique can maintain the original accuracy, (3) whether a pretrained DL model is necessary for the technique and (4) whether multiple devices are required to implement the technique.

As shown in Table 5.2, most of these techniques enable memory reduction of DL models to fit in low-resource devices at the network edge. These techniques lead to smaller memory requirements but simultaneously, the accuracy may fall due to this connection/parameter being modified. On the other hand, Federated Learning bases its performance on distributing the DL model across different devices while maintaining the initial features of the DL model in the general vision. The key parameter all of them share is the requirement of a pretrained DL model which fits for the application like a role model for the reduced DL model. Lastly, techniques such as Transfer Learning enable the device to re-train a model to adapt it to the specific scenario using a reduced dataset and re-training only some of the layers, leading to a less resource-demanding process.

Technique	Memory reduction	Maintain accuracy	Original DNN needed	Multiple devices needed	Reduced datasets
Pruning	✓	x	✓	x	x
Quantization	✓	x	✓	x	x
Federated Learning	✓	✓	✓	✓	x
Transfer Learning	x	✓	✓	x	✓
Knowledge Distillation	✓	✓	✓	x	✓
Gossip Learning	✓	✓	x	✓	x

Table 5.2: Comparison of techniques to adapt Deep Learning models to Edge Intelligence devices

Most of the cited papers who implement an Edge Intelligence approach focus on the middle levels of Figure 5.5, where the training is still carried out at the Cloud and the inference starts to move to the network edge or uses an edge-cloud collaborative approach. This fact is the result of numerous researches to migrate the training and inference processes to the network edge but this process/research has not been completed yet due to the restrictions at the network edge such as low power consumption and reduced computing capabilities.

Table 5.3 compares the papers that have been cited in section 5.4 of this survey using the following comparing features: (1) the ML task, (2) the Edge AI level, (3) whether the inference is carried out at the network edge, (4) whether the training is carried out at the network edge and (5) whether the inference is executed using a collaborative approach between the Cloud and the Edge.

Table 5.3 shows the relation between the approaches described in this section and the level of AI implementation at the network edge, according to the categorization of Figure 5.5. As it can be observed, most of the current researches can be classified as level 3 of the Edge AI categorization, since they perform AI inference at the network edge while the training is performed at the Cloud level. However, it is noteworthy that recent research lines are emerging in attempt to increase the Edge AI level by trying to optimize the hardware and software for the network edge which enables more complex DL models to be executed or even trained at the network edge. The next section focuses on the current hardware and software architectures for Edge AI that

Reference	ML task	Edge AI level	Inference at the edge	Training at the edge	Cloud-Edge co-inference
Shotton, J. (97)	Classification	3	✓	x	x
Li, L. (100)	Prediction	3	✓	x	x
Huang, Z. (59)	Classification	3	✓	x	x
Gupta, C. (103)	Prediction	3	✓	x	x
Mao, L. (106)	Object Detection	3	✓	–	–
Yu, Q. (108)	Classification	5	✓	✓	x
Chang, M. (95)	General ML tasks	3-5	✓	✓	x
CHen, J. (94)	General ML tasks	3-5	✓	✓	x
Lin, J. (112)	Classification	4	✓	x	✓
Chiliang, Z. (111)	Classification	3	✓	x	x
Pan, S.J. (115)	General ML tasks	3	✓	x	x
Chen, G. (116)	Object Detection	3	✓	x	x
Zhang, T. (120)	Computer Vision	3	✓	x	x
Hung, C.C. (121)	Computer Vision	3	✓	x	x
Lample, G. (125)	Natural Language Processing	1	x	x	✓
Apple (123)	Speech recognition	1	x	x	✓
Yao, S. (128)	Classification-Regression-Feature extraction	3	✓	x	x
Chen, T.C. (130)	Classification - Authentication	1-3	–	x	–
Hou, X. (133)	Object Detection	3	✓	x	x
Liu, L. (136)	Object Detection	1	x	x	✓
Ha, K.(137)	Object Detection	1	x	x	✓

Table 5.3: Comparison of Edge Intelligence proposals described in this section

cover these research lines.

5.5. Hardware and Software Architectures for Edge Intelligence

In order to adequate AI models to be executed at the network edge, the user has multiple options. One of these options is to adapt the algorithms to the new paradigm of the network edge, explained in the previous section, however there are also other options: Hardware and Software Architectures

for Edge Intelligence.

These options can be implemented in a collaborative approach or individually. An example of this is the implementation of DL models in the Google Coral TPU, which will be further commented in Subsection 5.5.1.

Next subsections will study in depth the current hardware and software architectures currently in the market or under research in order to help the reader to understand the current research lines in this topic.

5.5.1. Hardware for Edge Intelligence development

New research lines about architectures to perform efficiently Edge AI have been carried out recently. The goal of these researches is to develop new architectures where it is possible to implement ML algorithms without relying on the Cloud, or at least, to perform the maximum number of processes at the network edge. These devices are referred by the term *Edge AI Processors* (95; 110).

Using these architectures, some studies have proved the success of taking into account the initialization process for the DNN weights as well as new activation functions to ease and accelerate the training process at the network edge (138). Speeding up this process enables the implementation of edge devices based on the levels 5 and 6 of Figure 5.5, where the training and the inference are carried out at the network edge. In case the devices are unable to perform the training at the network edge, the devices would be at levels 3 to 4, where the training is executed under collaboration between the Cloud and the Edge but the inference is carried out at the network edge.

A different approach for the same problem is trying to implement DNNs in hardware, *Neuromorphical Hardware* (NH) (139; 95), instead of relying on software tools for this task. This approach is possible due to recent advances of CMOS and memristor technologies (140; 138), which enable the integration of fire neurons in a reduced size while maintaining a power efficient consumption.

About the processing unit for these devices, there are numerous papers comparing architectures for Edge AI Processors. Although some researchers support the idea of using GPU, TPU, NPU or GPGPU to follow the architecture implemented in the current servers to run ML algorithms (141; 142), others support the CPU as the main processing unit (69).

As (69) defends: *"...the CPU will remain the workhorse for ML workloads because it benefits from common software and programming frameworks that have been built up over many years during the mobile compute revolution. Additionally, it plays a vital role as mission control in more complex systems that leverage various accelerators via common and open software interfaces..."*. When it comes to the diversity of programming languages, the flexibility of the CPU can not be easily achieved by other architectures.

Contrarily, GPUs are more complex processor units specialized in processes where high parallelization helps to reduce training and/or inference time significantly, which are relevant features in DL algorithms. This architecture splits the data in reduced batches that are processed in parallel, which can only be done when we have all the data a priori. Otherwise, the data can not be split in the necessary partitions to improve the performance of the system based on a parallelization approach (141). However, this structure also improve the performance of the devices when executing traditional algorithms and processes where a single stream of operation is repeated on data due to its Single Instruction Multiple Data (SIMD) architecture.

Another approach is the use of neural processing units (NPU). These processors are specialized and hyper-efficient designed for specific DL tasks (142). These specialized capabilities help to improve the performance of the devices but come at the cost of losing flexibility, and obstructing the communication among the nodes at the network edge.

Other architectures highly relevant for Edge AI are the Systolic Arrays (SAs), which are used to accelerate the computation of matrix multiplications that are required in most ML algorithms. As a result of their improvement regarding the performance of AI models at the network edge, numerous authors research their integration in Edge Devices such as (143; 144; 145).

Tensor Processing Units (TPU), developed by Google I/O in 2016 (146), are other processing units designed to work with Tensorflow and TensorFlow Lite frameworks. They are AI accelerators for automatic learning tasks, such as DL, which are tailored for larger volumes of calculations of reduced precision based on the previously commented SAs. These devices can run state-of-the-art DNNs such as MobileNet V2 at almost 400 FPS while maintaining a reduced power consumption. Other AI accelerators are emerging in the market as a response to this device (147).

Following these lines of work, there are some Edge AI devices currently in the market which accelerate the DL processed by hardware.

- **Neuroshield** (148). This device is a platform/shield to add DL capabilities to low-resource devices such as Arduino or Raspberry Pi. This shield provides a software tool to configure the training and inferring processes at the network edge by using its 576 neurons based on a radial basic function for classification. As explained before, the limitations of this device fall on flexibility.

- **Google Coral Edge TPU** (149). This device is based on the integration of a TPU co-processor to execute the computations with tensors. Apart from this TPU specific to process tensor-based operations, the device include a CPU and GPU to execute operations that are not supported by the TPU. Thanks to the programming language used for this device, TensorFlow Lite, it is possible to configure the structure

of the network as well as its parameters. This enables high flexibility while maintaining a successful performance.

- **Nvidia Jetson Nano** (147). This device, as the Google TPU, is based on the integration of a co-processor, in this case, a GPU that enables fast computation for ML algorithms due to its parallelization capabilities.
- **Intel Movidius** (150). It is a hardware accelerator for DNN inference at the network edge to achieve real-time responses without relying on external devices or services like Cloud Computing. This device is based on a very long instruction words (VLIW) architecture that uses multiple register files in order to operate on a relatively big set of values at once. As a result of this, the latency of the system may be highly reduced.
- **SparkFun Edge** (151). This device, developed by Google, is designed for real-time audio processing. Its goal is to detect keywords in the speech by inference processes. It includes two built-in microphones, an accelerometer and a camera connector among other connectors. The software tool *Apollo3* can be used in collaboration with this device to ease the AI application building.
- **BeagleBone AI**. It is a board to develop ML and computer vision applications powered by an SoC - TI AM5729 dual-core Cortex-A15 processor. It integrates four embedded vision engine cores supported by a Texas Instrument ML module (152). It can used for segmentation, classification and detection tasks.
- **SmartEdge Agile and Branium**. The SmartEdge Agile (153) in collaboration with the Branium (154) software enables the deployment of AI models in low-resource devices at the network edge.
- **ECM3531** (155). It is a high-efficiency ASIC based on ARM Cortex-M3 and NXP Coolflux DSP processors designed for ML applications. The processor of this ASIC is called Tensai and it can execute TensorFLow and Caffe frameworks.
- **Smart-Edge-CoCaCo**. This new algorithm, CoCaCo algorithm, in collaboration with the edge device proposed by (45) can be used as a system for the optimization of the computation, data gathering and communication processes at the network edge. It is based on the collaboration of the Cloud and the Edge using a collaborative filter catching model which obtains low latency. In the paper (45), this technique is also tested and obtains lower latency than relying on the traditional Cloud Computing approach with increasing computations and concurrent users.

- **MediaTek Solution**. This architecture, proposed by (156), is based on an AI processing unit (APU), which is 55 times more power-efficient than average CPU and GPU. It includes a global buffer to share data among the different processors integrated into this architecture while a direct link is implemented to connect the peripherals with the system. This architecture has been proven to reduce the DRAM bandwidth as well as the energy consumption while improving the performance.

In contrast with the previously studied devices, a different approach is a software-defined chip. These devices, which can be based on Field-programmable gate array (FPGA) (157) or System on Chip (PSOC) (95; 158), can modify their internal connections to reconfigure themselves in order to fit the code.

Therefore, these devices can provide more flexibility than hardware defined static structures, allowing them to switch among different tasks during the execution process of ML algorithms. Apart from flexibility, these devices include low-power consumption and high performance capabilities to fit the network edge (157; 158). Multiple authors are researching the integration of these devices at the network edge by codifying the AI algorithms structure in the hardware (159; 160), leading to a fast execution of the algorithms that can achieve a latency of 390-433 giga operations per second, as the Mutilayer Perceptron (MLP) model proposed by Michaela B. e al. (159). At the same time, these devices efficiently manage the energy consumption by removing non relevant processes. Consequently, low energy is required to executed a model, as shown in (159) where the required energy per frame was 2.5 - 11 W to achieve an accuracy between 75 % and 99 % depending on the platform. The complexity of this hardware approach falls on the difficulty to codify in hardware all the emerging AI algorithms.

An example of this technology is the *Thinker AI* chip of Tsinghua University's Microelectronics Institute (157; 158). This chip can support numerous AI algorithms by its reconfigurable architecture via software. It has been tested with algorithms like CNNs and RNNs algorithms, obtaining successful results.

The techniques this device uses for being reconfigurable are (157; 158):

- **Memory bandwidth reconfigurable**: the distribution of the memory will be done according to the bandwidth to improve the energy efficiency and latency.

- **Bit-width reconfigurable**: it supports 16 and 8 bit-width, where computing accuracy is improved in 16 bit-width and speed is improved in 8 bit-width. Switching between these working methods, it can adapt to the problem it has in each moment.

- **Computing array reconfiguration**: depending on the function or basic operators needed in each moment, its interconnected parallel computing units can adapt their functions to highly performance the executed task.

FPGA can also be used for this purpose as shown by (161). They use an FPGA which has high performance, low-power consumption and small size for the implementation of a DL predictor. Data access was optimized as well as the pipeline structure. After the development of the system, they compared it with a Core 2 CPU 2.3 GHz and the results proved that the FPGA could achieve 30 times its speed (on average). FPGAs are an alternative to GPUs due to their capabilities for pipeline parallelism and reduced energy consumption, giving them a unique advantage over other processing units for DL (162). In the paper (163), another DL system based on FPGA is presented for CNN acceleration for Image-Net large-scale image classification. Some of the previously commented techniques, such as quantization, were used in this system and only 0.4 % accuracy loss was introduced for the very deep neural network models when using 8/4-bit quantization at the same time the results of this DNN significantly outperformed previous approaches.

In Table 5.4, there is a comparison of the previous Edge Intelligence devices where the parameters to compare are (1) the RAM memory, (2) the processing unit integrated, (3) whether there is a specific software for DNN development, (4) whether the system is for general purpose and (5) whether the device includes hardware programmability capabilities.

As shown in Table 5.4, there are heterogeneous devices for Edge AI and some of the emerging ones do not provide all the data to fulfill this table. Nevertheless, it is possible to compare their most relevant features.

There are two main groups among the previous Edge AI devices, the devices with large RAM memory, such as Coral Edge TPU and Jetson nano, and the devices which use techniques to optimize the memory consumption based on specific software that has been designed for the devices, as the SmartEdge Agile and Branium. Another architectures, such as FPGAs, try to optimize the memory consumption by sharing parameters to reduce the quantity of values to store in the memory of the device. Therefore, they do not require a large amount of memory. The goal of both approaches is the same, fit the DNN to the resources of the devices, however one approach is based on increasing the resources while the other reduces the required resources.

The majority of these devices are for general purpose tasks and includes specific software constraints for DL. The software constraints must be understood as a limitation of DL libraries that can be implemented in the system, such as TensorFlow Lite for Coral Edge TPU. This is understandable since each provider develops a specific framework to optimized the

Device	RAM	Processing unit	Software	General purpose	Hardware programable
Neuroshield (148)	–	NM500	Yes	No	No
Coral Edge TPU (149)	1 GB	NXP iMXM+GC7000 GPU+TPU	Yes	Yes	No
Jetson Nano (147)	4 GB	Quad core Cortex-A57+GPU	Yes	Yes	No
Intel Movidius (150)	1 GB	Myriad 2 VPU	Yes	No	No
SparkFun Edge (151)	384KB	ARM Cortex-M4F	Yes	Yes	No
BeagleBone (152)	1 GB	Cortex-A15+Sitara AM5729	Yes	Yes	No
Agile-Branium (153; 154)	–	Smartedge Agile	Yes	Yes	No
ECM3531 (155)	–	ARM Cortex-M3 NXP Coolflux DSP	Yes	Yes	No
SE-CoCaCo (45)	–	Hao et al. (45) architecture	Yes	Yes	No
MediaTek (156)	–	APu+CPU+GPU	Yes	Yes	No
Thinker AI (157; 158)	–	2x16x16 reconfigurable heterogeneous PE arrays	No	Yes	Yes
FPGA (161)	–	FPGA	No	Yes	Yes
CNN-FPGA (163)	–	FPGA	No	No	Yes

Table 5.4: Comparison of Edge Intelligence devices

implementation of ML models on their platforms. However, this is only possible when a company/research center works on both, hardware and software frameworks for ML. As a result, it is also possible to find hardware frameworks that accept multiple software frames, such as the platform developed by (161) and the CNN-FPGA by (163).

The processing units used in the Edge AI platforms are heterogeneous as well as the existence of a co-processor unit in the device. A relevant feature only a few of them incorporate is the capability to reconfigure the hardware to adapt to new applications, such as Thinker AI or the FPGA developed by (161).

5.5.2. Software for Edge Intelligence development

While the selection of the processing unit architecture is one of the key factors, the software tools for developing Edge AI must also be taken into account to control the resource accesses as well as the optimization of the

ML algorithms. Following this line of work, some ML frameworks have been adapted to the limitations at the network edge to create new versions of those frameworks and/or develop completely new frameworks (35):

- **CMSIS-NN kernels**. ARM Cortex-M microcontrollers are being researched to integrate them in edge devices by being used in collaboration with the CMSIS-NN kernels, which are efficient neural network kernels designed for these microcontrollers to minimize the memory footprint (164). Therefore, it is possible to implement ML in low-resource devices at the network edge while maintaining a state-of-the-art performance with low-energy consumption (82).
- **DeepMon**. It is a software tool that optimizes the processing of CNN layers by studying the similarity among consecutive frames in the data to reduce the inferring time. This approach inspires the idea of storing the results of the CNN layers to optimize future computations which accelerates the inferring process (165).
- **TensorFlow Lite**. TensorFlow Lite is a lightweight implementation of the Google's framework for DL, TensorFlow (166; 167). This framework is designed to be implemented at the network edge in devices with low resources without having to rely on Cloud computing. This framework includes neural networks APIs to support hardware acceleration in order to improve its performance. Due to its configurations, it enables parallel processes that fit in the context of a distributed network at the network edge. Some researches have been carried out to test its performance, obtaining successful results in edge devices like smartphones (168)
- **MXNet**. This framework for DNN was developed by Apache (169). It is a scalable, lean, resource-efficient and open-source framework for edge devices, which supports a distributed network implementation and Cloud collaboration.
- **Caffe2Go**. Caffe2Go is a variation of the Caffe framework developed by Facebook to improve its flexibility and speed (170). A modular approach was used in this framework to enable its use in collaboration with the Caffe2 framework. It provides a direct way to implement DNN in mobile devices at the network edge in order to compute data in real time. It can run on Android as well as iOS without having to modify the code depending on the OS. This framework has been integrated into the Python IDLE PyTorch.
- **CoreML3**. This framework was developed by Apple, thus can only be implemented in iOS. It enables the addition of ML capabilities to iOS applications and makes the use of ML algorithms possible in end

devices without dedicated resources for these tasks such as dedicated GPU for DL (171).

- **DeepCham**.This framework was developed to deploy DL models on mobile devices to recognize objects captured by the cameras of these devices. It is designed to work in collaboration with edge devices in the same network (172).

- **DeepThings**. It is a framework to adapt existing DL models, specifically CNN models, for inference process. This framework implements Fused Tile Partitioning in order to reduce the memory footprint of the DL models, enabling its execution on edge devices. The developers of this framework tested it by deploying the DL model *Yolov2* on a low-resource device such as Raspberry Pi. In this experiment, the DeepThings software in collaboration with a Raspberry Pi 3 Model B devices provided scalable CNN inference speedups of 1.7×–3.5× on 2–6 edge devices with less than 23 MB memory each (173) .

- **DeepIoT**. This framework, developed by (174), reduces DNN into smaller dense matrices while maintaining the performance almost the same. It reduces the redundant elements by finding the minimum number of filters and dimensions required for the application. In the paper (174), it was proven this framework could reduce a DNN by a factor of 90 % leading to a time response decrease of 70 % as well as an energy consumption decrease by a factor of 72.2 % to 95.7 %.

- **SparseSep**. This framework, developed by (175), optimizes large DL models for low-resource devices with reduced impact in the accuracy. In the paper (175), it was tested on different devices such as ARM Cortex, NVidia Tegra K1, and Qualcomm Snapdragon processors, reporting an inference speed 13.3 times faster than the original DL models and consuming 11.3 times less memory.

- **ML Kit**. It is a mobile SDK framework developed by Google. It uses Google's APIs such as TensorFlow Lite to accomplish different ML tasks (text recognition, image studying, etc.) (176). This framework was tested on an iPhone smartphone, as a representation of an edge device, for classification tasks among other DL common tasks, where the inception v3 model achieved an accuracy of 78 % while occupying 95 MB with a latency of 90 ms and the mobilenet model achieved a accuracy of 71 % with a memory consumption of 17 MB and latency of 32 ms among other models (177).

- **AI2GO**. This framework tunes DL models to be implemented on edge devices with low-resource specifications (178). Custom DL models

generated with this framework for inference maintain state-of-the-art performance obtained via traditional approaches (179).

- **AWS Greengrass**. This framework (180), developed by Amazon, for Edge Computing focuses on three key points: (a) using lambda functions to response as fast as possible to IoT device requests, (b) using ML models trained in Amazon cloud sites for machine learning inference and (c) using IoT core authentication engines to control the data flow among nodes of the network.

- **Foghorn**. The high performance, near real-time response and heterogeneous applications of this framework fit better for a small range edge environment. It reduces the memory consumption, enabling its implementation in Edge AI devices with low-resource limitations. It can store and process data to apply ML algorithms to current or historical data collected by the sensors of the network. Simultaneously, it includes managing capabilities to configure and control the end devices of the network (181).

- **Azure IoT Edge**. It is a platform of Microsoft that can be used to offload large amounts of work from the cloud to the network edge. This migration reduces the latency and adds reliability to work in offline systems (182).

- **OpenEI**. This framework, developed by (183), is an open-source lightweight software platform to add intelligent processing and data sharing capabilities to the network edge. This framework enables low-resource devices from raspberry Pi to advanced AI devices to become an Edge AI device while maintaining the accuracy of the state-of-the-art ML models. This framework takes into account the restrictions of edge devices such as energy consumption and reduced memory among others.

- **VideoEdge**.This framework was developed to study video data (121). This framework is design to enable to collaboration of multiple levels, such as cloud and edge, during the inference process of the execution of an AI model. The limitation of this framework falls in the fact it does not include training capabilities at the same time it requires a collaborative approach of the cloud and the edge for the inference.

- **AdaDeep**. This framework combines multiple techniques in order to optimize the DL models with the goal adapt them to the network edge while maintaining their accuracy results as well as low latency (184). To do this, this framework using Deep Reinforcement Learning to find how to combine multiple optimization techniques for each specific DL model taking into account the latency, energy consumption, accuracy and memory consumption as parameters to optimize.

- **Minerva**. Similar to AdaDeep, this framework combines multiple optimization techniques in order to optimize a DL model based on the final accuracy of the model, latency and energy consumption (185). The techniques this framework uses are quantization and pruning.
- **Neurosurgeon**. This framework is based on one the the previously commented techniques to implement DNN at the network edge, the Model Partition (186). This framework divides the DL model between the edge device and a server. By doing this, the resource-requiring layers of the model can be executed in a server with high resources. This leads to an increase of the speed to calculate a result from a model. At the same time, the energy consumption is taken into account in this framework as a parameter to optimize. However, because of the partition and its goal to implement part of the model on a server, this technique cannot be used for all-on-device approaches.
- **JALAD**. This framework optimized the DL models using the same technique as Neurosurgeon, the model partition (187). However, this framework study this optimization as an integer linear programming (ILP) problem where it tries to minimize the latency during the inference process taking the accuracy of the model as a constrain during the optimization. As in Neurosurgeon, because of the partition, this framework is not suitable for all-on-device solutions since it requires to distribute the model across multiple devices to speed up the inference process.
- **FedAvg**. This framework was designed to implement Federated Learning in order to train DL models at the network edge (188). It is based on a model averaging, meaning each of the nodes of the network updates their results for the weights to a server where these weights are averaged and implemented as the final weight for the DL model. The training carried out by each node is based on Stochastic Gradient Descent (SGD).
- **GossipGrad**. To solve the problem of the scalability of the GoSGD (119), (189) developed GossipGraD . This technique reduces the complexity of the communications between the nodes of the network and execute the communications between nodes after a fixed number of iterations to update the calculated weights. It also takes into account the rotation of the nodes to communicate with to ease the communication process. By applying these techniques, these framework prevent the over-fitting during the training of the models.
- **PipeDream**. This framework is based on the partition of the DL model during the training phase to distribute the process across multiple nodes taking into account the resource of each node. By using this

framework, the communication among nodes is reduced as well as the resource are use efficiently (190).

These frameworks are compared in Table 5.5 based on the relevant features that are required or desirable in an Edge AI framework. The criteria used for the comparison regarding technical aspects from the developer's point of view are: (1) whether it is possible to develop general purpose DNN with the framework or if it only supports specific applications such as computer vision or natural language processing, (2) whether they require a specific edge device to be implemented in, (3) whether they are able to train a DNN or they are meant exclusively for DNN deployment and (4) whether the framework optimizes the DNN to be implemented in low-resource devices at the network edge.

Framework	General purpose	Specific edge device	Training	Optimization of DNN
CMSIS-NN kernels (164)	✓	✓	x	✓
Deep Mon (165)	x	x	x	✓
TensorFlow Lite (166)	✓	x	✓	✓
MXNet (169)	✓	x	✓	✓
Caffee2Go (170)	✓	x	x	✓
CoreML3 (171)	✓	✓	–	✓
DeepCharm (172)	x	x	x	✓
DeepThings (173)	x	x	x	✓
DeepIoT (174)	✓	x	x	✓
SparseSep (175)	✓	x	x	✓
ML Kit (176)	✓	x	–	✓
AI2GO (178)	✓	x	x	✓
AWS Greengrass (180)	✓	x	x	✓
Foghorn (181)	✓	x	✓	✓
Azure IoT Edge (182)	✓	x	✓	✓
OpenEI (183)	✓	x	x	✓
VideoEdge (121)	x	x	x	x
AdaDeep (184)	✓	x	x	✓
Minerva (185)	✓	x	x	✓
Neurosurgeon (186)	✓	x	x	✓
JALAD (187)	✓	x	x	✓
FedAvg (188)	✓	x	✓	✓
GossipGrad (119)	✓	x	✓	x
PipeDream (190)	✓	x	✓	x

Table 5.5: Comparison of Edge Intelligence Frameworks

Table 5.5 shows how most of the current Edge AI frameworks are de-

veloped for inference at the network edge but only a few can execute the training process (there was no information regarding this capability for ML Kit and CoreML3 so a "symbol was included in the table to represent the lack of information), such as MXNet and TensorFlow Lite. This reveals the current status of Edge AI technology where the migration of the complete DL process is still being researched. Therefore, most of the available frameworks have not integrated training capabilities for the network edge yet due to the complexity of execute this process with low resources in a time-critial environment. At the same time, most of the frameworks can be implemented for heterogeneous applications as well as devices to fit the heterogeneous environment found at the network edge, only two of the studied frameworks require a specific hardware decive to be implemented in, such as CMISIS-NN Kernels, where due to its development for an specific device lead to its low compatibility with other architectures.

Therefore, even when numerous frameworks, as well as hardware architectures, for Edge Intelligence are being researched, this study shows how most of them are designed for the low levels of the pyramid of the figure 5.5. As a result, the training process of the AI models at the network edge is still a research line that needs to mature in order to lead to devices that would be able to be implemented autonomously at the network edge without depending on another system to carry out the training of the models.

Nevertheless, the abundance of ML frameworks for edge devices shows how this is a relevant topic being research for numerous companies and research centers. As a result, these frameworks are more completed each time, adding new capabilities to fit the emerging techniques for ML models at the network edge as it can be the distributed learning.

5.6. Challenges and Future directions

As stated in the previous sections of this survey, Edge-based ML is an emerging research line that plays a relevant role in applications where low latency as well as data privacy is required in collaboration with AI capabilities for tasks such as classification or regression. This technique has proven to improve current limitations of the Cloud Computing approach while providing the benefits of the Edge Computing.

Besides the advances in Edge AI, there are a number of challenges that must be attended in order to fully merge ML algorithms with the Edge paradigm:

- **Heterogeneous data.** As a result of integrating heterogeneous devices at the network edge (due to the increasing number of IoT sensors and distributors), Edge AI works in a heterogeneous environment since numerous architectures, type of sensors and companies are deploying

their sensors at the network edge. This leads to a situation where the data at the network edge, as well as the preprocessing techniques required for each sensor, may highly vary from application to application (191; 94). In order to work with the data provided by these devices, ML algorithms need to learn to fuse data with different features/structures such as image, sound, and text among others.*Multimodal deep learning* is an emerging research line used to extract relevant features from heterogeneous data (192; 193). This technique is one of the approaches to face this limitation based on adapting the DL model to the problem. Another approach to solve this problem may be preprocessing the data at the data source when possible to follow a standard structure. If this challenge is not faced, even when the devices can execute accurate AI models at the network edge, a specific device would be required to study each data type. This would lead to new problems such as the fast increase of devices required for a single task, which would not be as efficient as studying all the possible data in a single device. At the same time, this would lead to having to communicate the outputs of all these models for a later fusion, increasing the overall latency of the system and moving part of the process far away from the network edge.

- **Distributed ML algorithms** is a current research line to accelerate the training process of ML algorithms as well as to reduce the memory required for this process. This challenge is originated in the increasing complexity of the ML models developed for new applications as well as the trend of designing models that will be retrained at the user end during the life cycle of the system in order to improve the system accuracy even after modifying the scenario/conditions. Researchers are studying and testing some distributed learning algorithms for edge applications to face this challenge. An example of these research lines is the *Federated Learning* (114) where multiple DL models are deployed at the network edge for the same task. However, the impact of the heterogeneity of the data on the accuracy is still an open research field (191; 94; 114). The impact of the model distribution on the security-privacy of the data is also a topic that lead to new secure communication protocols to ensure that even with this communication between devices the data is still reliable and has not been access by third parties during its transmission. Some protocols to achieve this have been proposed in (194) or in (195) by using BlockChain.

- **Training DL models.** Due to the resource requirements for the training process, the cloud approach is used for this phase in most of the cases as it has been indicated in Table 5.3, where only the two top level execute the training at the network edge. Some attempts to migrate both processes, the training and the inference processes, to the network

edge have been made by using techniques such as pruning and quantization to reduce the DL model size. These approaches, consequently, accelerate the training and inference processes by reducing the number of required calculations. However, the results often provide lower accuracy than cloud training DL models. Training at the network edge is a complex task due to the memory and computational constraints of the edge but the interest to achieve this goal leads to emerging research lines such as Edge AI frameworks (178; 176; 173; 172; 171; 170; 169; 166; 164) and algorithms to accelerate the training. At the same time, this challenge is leading to new AI model structures such as Spiking Neural Network (196), which promises a reduction of the energy consumed during the training and inference processes as well as an increase in the speed. However, as other DL model architectures, this DNN cannot be generalized for all applications so this research line is not the only one that must be contemplated in the Edge AI paradigm. Another approach to face this challenge is the neuromorphic hardware, where the connections between the neurons of the DL models are implemented in hardware reduces the latency in, at least, one order of magnitude and three orders of magnitude more energy-efficient (139). This leads to a low latency during the training and inference processed at the network edge.

- **Models accuracy.** High accuracy is a requirement in most of the ML applications, specially in some edge applications such as autonomous driving or health care. Consequently, emerging research lines are working on this problem by developing new algorithms following the path set by DL model pruning and quantization to implement state-of-the-art models at the network edge maintaining their accurated results. Another research line for this problem is the development of new DL algorithms or structures which fit better the network edge. At the same time, to achieve high accuracy, large specific datasets for new applications are required when using supervised learning techniques. Some research lines are proposing solutions for this topic such as synthetic dataset creation (197; 198). On the other hand, to fit the low-labeled-data edge applications, new approaches for the training of the ML models are emerging. One of these approaches to achieve high accuracy in applications where the quantity of labeled data is highly reduced is using Meta-learning approaches (199), where the model is trained in a large dataset that shares the metadata or, at least, on a set of really similar applications, defined as tasks. This technique enables a fast retraining in a later step for the specific application while still maintaining its accuracy.

- **Hierarchical structure.** Some researchers are studying the possibi-

lity of splitting ML processes into sub-processes which could be executed by different devices or even using varied approaches. A hierarchical structure combining Edge, Fog and Cloud could be used to design a structure with different levels where the computing power would increase as the distance to the cloud decreases. Therefore, the structure would include different *computing power layers* to process the data as the closest level to the network edge. This paradigm, which is a current open research line, maintains the advantages of Cloud Computing while adding the benefits of the Edge Computing (43; 30; 44; 45).

- **Higher-level abstract decision making.** Due to the fast changing environment at the network edge due to emerging platforms, frameworks and techniques, an approach where assemble itself given high level instructions would be a stunning advance for the Edge Intelligence. This approach would follow the researches presented in (200; 201; 202; 203). As a result of applying this technique, the Edge Devices for AI could be more flexible regarding data structure as well as application since the user could configure the device based on some high-level instructions without the need of specialized knowledge to design the AI model. However, this technique creates challenges to achieving a robust and trustworthy designs. Previous attempts to achieve this have face numerous problems but (200) states that one possible reason is that previous projects have not been able to find the correct high-level abstractions, therefore more researches are required to achieve the desired abstraction level.

After all the information gathered in this survey, it is highly appreciable that emerging research lines are working to deploy ML models at the network edge using multiple approaches from reducing the size of the state-of-the-art models to create new models based of emerging techniques. However, most of the current edge AI systems have not achieved the high levels of Edge AI, as shown in Table 5.3, where most of the current researchers are working on the level 3 of the Edge AI according to Figure 5.5. Nevertheless, emerging techniques such as Spiking Neural Networks or neuromorphic hardware promise to change this as explained previously. As a result, numerous authors are working on these research lines as well as processes to transfer the knowledge from complex DNN into simpler ones rather than using techniques such as pruning or quantization. This may be due to the fact that removing low-relevance parameters has reached its limit to reduce the size of the model without incurring into high accuracy decreases.

As a result, it is possible to observe how the future trends for the Edge AI are the deployment on hardware of complex systems, where the hardware implementation is the fact that improve the performance of the system, and using emerging techniques to transfer the knowledge between DNN effi-

ciently. These trends need to be further researched in order to keep climbing the levels of the Edge AI.

5.7. Conclusion

This survey provides a comprehensive overview of enabling technologies for the Edge AI, focusing on new significant Edge AI frameworks and devices.

Recent advances in Edge Computing have been explained such as the new hardware and software which have emerged during the last few years to face the problems of Cloud Computing in the current situation. The relevance of this technology has been demonstrated with diverse examples of applications including autonomous driving, security solutions, IoT applications, location awareness applications, and network management. The hardware advances follow the line of incorporating new modules/layers which enable the parallelization of the processes. Meanwhile, the software research lines focus on improving the data processing speed as well as improving the security of the systems. These features have been compared in diverse tables for a deeper understanding.

Later, new Edge AI devices have been commented to explain what advantages they can bring to the current situation in diverse research lines such as NLP, Computer Vision, IoT and VR. Following the line of the research made about Edge Computing, the current Edge AI devices and frameworks have been explained based on their relevant features for a posterior comparison.

References

[1] Kusumlata Jain and Smaranika Mohapatra. *Taxonomy of Edge Computing: Challenges, Opportunities, and Data Reduction Methods*, pages 51–69. Springer International Publishing, Cham, 2019.

[2] Edoardo Cavalieri d'Oro, Simone Colombo, Marco Gribaudo, Mauro Iacono, Davide Manca, and Pietro Piazzolla. Modeling and evaluating a complex edge computing based systems: An emergency management support system case study. *Internet of Things*, 6:100054, 2019.

[3] Z. Pang, L. Sun, Z. Wang, E. Tian, and S. Yang. A survey of cloudlet based mobile computing. In *2015 International Conference on Cloud Computing and Big Data (CCBD)*, pages 268–275, Nov 2015.

[4] Julien Gedeon, Jeff Krisztinkovics, Christian Meurisch, Michael Stein, Lin Wang, and Max Mühlhäuser. A multi-cloudlet infrastructure for future smart cities: An empirical study. In *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, EdgeSys'18, pages 19–24, New York, NY, USA, 2018. ACM.

[5] Shogo Ando and Akihiro Nakao. In-network cache simulations based on a youtube traffic analysis at the edge network. In *Proceedings of The Ninth International Conference on Future Internet Technologies*, CFI '14, pages 10:1–10:6, New York, NY, USA, 2014. ACM.

[6] Zhi Yang, Ben Y. Zhao, Yuanjian Xing, Song Ding, Feng Xiao, and Yafei Dai. Amazingstore: Available, low-cost online storage service using cloudlets. In *Proceedings of the 9th International Conference on Peer-to-peer Systems*, IPTPS'10, pages 2–2, Berkeley, CA, USA, 2010. USENIX Association.

[7] Kiryong Ha, Padmanabhan Pillai, Wolfgang Richter, Yoshihisa Abe, and Mahadev Satyanarayanan. Just-in-time provisioning for cyber foraging. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 153–166, New York, NY, USA, 2013. ACM.

[8] Vladimir Stantchev, Ahmed Barnawi, Sarfaraz Ghulam, Johannes Schubert, and Gerrit Tamm. Smart items, fog and cloud computing as enablers of servitization in healthcare. *Sensors & Transducers*, 185(2):121, 2015.

[9] Yu Cao, Songqing Chen, Peng Hou, and Donald Brown. Fast: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation. *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 2–11, 2015.

[10] Amir M. Rahmani, Tuan Nguyen Gia, Behailu Negash, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, and Pasi Liljeberg. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, 78:641 – 658, 2018.

[11] Rasel Mahmud, Ranganath Vallakati, Anupam Mukherjee, Prakash Ranganathan, and Arash Nejadpak. A survey on smart grid metering infrastructures: Threats and solutions. In *2015 IEEE International Conference on Electro/Information Technology (EIT)*, pages 386–391. IEEE, 2015.

[12] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan. Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access*, 6:47980–48009, 2018.

[13] C. Shi, Z. Ren, K. Yang, C. Chen, H. Zhang, Y. Xiao, and X. Hou. Ultra-low latency cloud-fog computing for industrial internet of things.

In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, April 2018.

[14] Nam Ky Giang, Victor C.M. Leung, and Rodger Lea. On developing smart transportation applications in fog computing paradigm. In *Proceedings of the 6th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications*, DIVANet '16, pages 91–98, New York, NY, USA, 2016. ACM.

[15] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Méndez, C. E. Chung, Y. T. Wang, T. Mullen, and T. P. Jung. Augmented brain computer interaction based on fog computing and linked data. In *2014 International Conference on Intelligent Environments*, pages 374–377, June 2014.

[16] Andrea Giordano, Giandomenico Spezzano, and Andrea Vinci. Smart agents and fog computing for smart city applications. In *Proceedings of the First International Conference on Smart Cities - Volume 9704*, Smart-CT 2016, pages 137–146, Berlin, Heidelberg, 2016. Springer-Verlag.

[17] Partha Pratim Ray, Dinesh Dash, and Debashis De. Edge computing for internet of things: A survey, e-healthcare case study and future direction. *Journal of Network and Computer Applications*, 140:1 – 22, 2019.

[18] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219 – 235, 2019.

[19] Jitender Grover and Ram Murthy Garimella. *Optimization in Edge Computing and Small-Cell Networks*, pages 17–31. Springer International Publishing, Cham, 2019.

[20] Gary A McGilvary, Adam Barker, and Malcolm Atkinson. Ad hoc cloud computing. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 1063–1068. IEEE, 2015.

[21] Ana Juan Ferrer, Joan Manuel Marquès, and Josep Jorba. Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing. *ACM Computing Surveys (CSUR)*, 51(6):1–36, 2019.

[22] Syed Danial Ali Shah, Mark A Gregory, and Shuo Li. Cloud-native network slicing using software defined networking based multi-access edge computing: a survey. *IEEE Access*, 9:10903–10924, 2021.

[23] Mahshid Mehrabi, Dongho You, Vincent Latzko, Hani Salah, Martin Reisslein, and Frank HP Fitzek. Device-enhanced mec: Multi-access edge computing (mec) aided by end device computation and caching: A survey. *IEEE Access*, 7:166079–166108, 2019.

[24] G. Plastiras, M. Terzi, C. Kyrkou, and T. Theocharidcs. Edge intelligence: Challenges and opportunities of near-sensor machine learning applications. In *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 1–7, July 2018.

[25] E. Li, L. Zeng, Z. Zhou, and X. Chen. Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, pages 1–1, 2019.

[26] A. H. Sodhro, S. Pirbhulal, and V. H. C. de Albuquerque. Artificial intelligence-driven mechanism for edge computing-based industrial applications. *IEEE Transactions on Industrial Informatics*, 15(7):4235–4243, July 2019.

[27] Yi Liu, Chao Yang, Li Jiang, Shengli Xie, and Yan Zhang. Intelligent edge computing for iot-based energy management in smart cities. *IEEE Network*, 33, 03 2019.

[28] E. Aleksandrova, C. Anagnostopoulos, and K. Kolomvatsos. Machine learning model updates in edge computing: An optimal stopping theory approach. In *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 1–8, June 2019.

[29] Khan Muhammad, Salman Khan, Vasile Palade, Irfan Mehmood, and Victor Hugo C De Albuquerque. Edge intelligence-assisted smoke detection in foggy surveillance environments. *IEEE Transactions on Industrial Informatics*, 16(2):1067–1075, 2019.

[30] Seraphin B Calo, Maroun Touna, Dinesh C Verma, and Alan Cullen. Edge computing architecture for applying ai to iot. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 3012–3016. IEEE, 2017.

[31] M. Wolf. Machine learning + distributed iot = edge intelligence. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1715–1719, July 2019.

[32] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, Aug 2019.

[33] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys Tutorials*, 20(4):2923–2960, 2018.

[34] Wenbin Li and Matthieu Liewig. A survey of ai accelerators for edge environment. In *World Conference on Information Systems and Technologies*, pages 35–44. Springer, 2020.

[35] M. G. Sarwar Murshed, Christopher Murphy, Daqing Hou, Nazar Khan, Ganesh Ananthanarayanan, and Faraz Hussain. Machine learning at the network edge: A survey. *ArXiv*, July 2019.

[36] Quoc-Viet Pham, Fang Fang, Vu Nguyen Ha, Md Jalil Piran, Mai Le, Long Bao Le, Won-Joo Hwang, and Zhiguo Ding. A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art. *IEEE Access*, 8:116974–117017, 2020.

[37] Inés Sittón-Candanedo, Ricardo S Alonso, Juan M Corchado, Sara Rodríguez-González, and Roberto Casado-Vara. A review of edge computing reference architectures and a new global edge proposal. *Future Generation Computer Systems*, 99:278–294, 2019.

[38] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[39] Tseng Mitch, Edmunds Todd, and Canaran Lalit. Introduction to edge computing in iiot. 2018.

[40] Inés Sittón-Candanedo, Ricardo S Alonso, Óscar García, Lilia Muñoz, and Sara Rodríguez-González. Edge computing, iot and social computing in smart energy scenarios. *Sensors*, 19(15):3353, 2019.

[41] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. A survey on the edge computing for the internet of things. *IEEE access*, 6:6900–6919, 2017.

[42] Yen-Lin Lee, Pei-Kuei Tsung, and Max Wu. Techology trend of edge ai. In *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 1–2. IEEE, 2018.

[43] Hasibur Rahman, Rahim Rahmani, and Theo Kanter. The role of mobile edge computing towards assisting iot with distributed intelligence: a smartliving perspective. In *Mobile Solutions and Their Usefulness in Everyday Life*, pages 33–45. Springer, 2019.

[44] Mabrook Al-Rakhami, Mohammed Alsahli, Mohammad Mehedi Hassan, Atif Alamri, Antonio Guerrieri, and Giancarlo Fortino. Cost efficient edge intelligence framework using docker containers. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 800–807. IEEE, 2018.

[45] Y. Hao, Y. Miao, L. Hu, M. S. Hossain, G. Muhammad, and S. U. Amin. Smart-edge-cocaco: Ai-enabled smart edge with joint computation, caching, and communication in heterogeneous iot. *IEEE Network*, 33(2):58–64, 2019.

[46] A. Ahmed and E. Ahmed. A survey on mobile edge computing. In *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, pages 1–8, Jan 2016.

[47] Hongxing Li, Guochu Shou, Yihong Hu, and Zhigang Guo. Mobile edge computing: Progress and challenges. In *2016 4th IEEE international conference on mobile cloud computing, services, and engineering (MobileCloud)*, pages 83–84. IEEE, 2016.

[48] D. Sabella, Alessandro Vaillant, Pekka Kuure, U. Rauschenbach, and F. Giust. Mobile-edge computing architecture: The role of mec in the internet of things. *IEEE Consumer Electronics Magazine*, 5:84–91, 2016.

[49] Intel and Nokia Siemens Networks. Increasing mobile operators' value proposition with edge computing. 2013.

[50] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

[51] Mingyue Cui, Shipeng Zhong, Boyang Li, Xu Chen, and Kai Huang. Offloading autonomous driving services via edge computing. *IEEE Internet of Things Journal*, 7(10):10535–10547, 2020.

[52] Qi Chen, Xu Ma, Sihai Tang, Jingda Guo, Qing Yang, and Song Fu. F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 88–100, 2019.

[53] Pedro J Navarro, Carlos Fernandez, Raul Borraz, and Diego Alonso. A machine learning approach to pedestrian detection for autonomous vehicles using high-definition 3d range data. *Sensors*, 17(1):18, 2017.

[54] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. Vehicular edge computing and networking: A survey. *Mobile Networks and Applications*, pages 1–24, 2020.

[55] Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodík, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath, and Sudipta Sinha. Real-time video analytics: The killer app for edge computing. *computer*, 50(10):58–67, 2017.

[56] Ning Chen, Yu Chen, Sejun Song, Chin-Tser Huang, and Xinyue Ye. Smart urban surveillance using fog computing. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 95–96. IEEE, 2016.

[57] Kewei Sha, T Andrew Yang, Wei Wei, and Sadegh Davari. A survey of edge computing-based designs for iot security. *Digital Communications and Networks*, 6(2):195–202, 2020.

[58] Hong Liu, Yan Zhang, and Tao Yang. Blockchain-enabled security in electric vehicles cloud and edge computing. *IEEE Network*, 32(3):78–83, 2018.

[59] Zhenqiu Huang, Kwei-Jay Lin, Bo-Lung Tsai, Surong Yan, and Chi-Sheng Shih. Building edge intelligence for online activity recognition in service-oriented iot systems. *Future Generation Computer Systems*, 87:557–567, 2018.

[60] SK Alamgir Hossain, Md Anisur Rahman, and M Anwar Hossain. Edge computing framework for enabling situation awareness in iot based smart city. *Journal of Parallel and Distributed Computing*, 122:226–237, 2018.

[61] Kay Bierzynski, Antonio Escobar, and Matthias Eberl. Cloud, fog and edge: Cooperation for the future? In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 62–67. IEEE, 2017.

[62] Thavavel Vaiyapuri, Velmurugan Subbiah Parvathy, V Manikandan, N Krishnaraj, Deepak Gupta, and K Shankar. A novel hybrid optimization for cluster-based routing protocol in information-centric wireless sensor networks for iot based mobile edge computing. *Wireless Personal Communications*, pages 1–24, 2021.

[63] Oanh Tran Thi Kim, Nguyen Dang Tri, Nguyen H Tran, Choong Seon Hong, et al. A shared parking model in vehicular network using fog and cloud environment. In *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 321–326. IEEE, 2015.

[64] Harshitha Bura, Nathan Lin, Naveen Kumar, Sangram Malekar, Sushma Nagaraj, and Kaikai Liu. An edge based smart parking solution using camera networks and deep learning. In *2018 IEEE International Conference on Cognitive Computing (ICCC)*, pages 17–24. IEEE, 2018.

[65] Shweta Prabhat Khare, Janos Sallai, Abhishek Dubey, and Aniruddha Gokhale. Short paper: Towards low-cost indoor localization using edge computing resources. In *2017 IEEE 20th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 28–31. IEEE, 2017.

[66] Yongli Zhao, Boyuan Yan, Wei Wang, Yi Lin, and Jie Zhang. Onboard artificial intelligence based on edge computing in optical transport networks. In *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3. IEEE, 2019.

[67] David Sarabia-Jácome, Ignacio Lacalle, Carlos E Palau, and Manuel Esteve. Efficient deployment of predictive analytics in edge gateways: Fall detection scenario. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 41–46. IEEE, 2019.

[68] Carole-Jean Wu, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, Marat Dukhan, Kim Hazelwood, Eldad Isaac, Yangqing Jia, Bill Jia, et al. Machine learning at facebook: Understanding inference at the edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 331–344. IEEE, 2019.

[69] Rene Hass et al. What's powering artificial intelligence? *ARM*, 2019.

[70] Gaurav Batra, Zach Jacobson, Siddarth Madhav, Andrea Queirolo, and Nick Santhanam. Artificial-intelligence hardware: New opportunities for semiconductor companies. *McKinsey & Company, New York, NY, USA, Tech. Rep*, 2018.

[71] Volkan Gezer, Jumyung Um, and Martin Ruskowski. An extensible edge computing architecture: Definition, requirements and enablers. *Proceedings of the UBICOMM*, 2017.

[72] Ju Ren, Hui Guo, Chugui Xu, and Yaoxue Zhang. Serving at the edge: A scalable iot architecture based on transparent computing. *IEEE Network*, 31(5):96–105, 2017.

[73] Partha Pratim Ray, Dinesh Dash, and Debashis De. Edge computing for internet of things: A survey, e-healthcare case study and future direction. *Journal of Network and Computer Applications*, 140:1–22, 2019.

[74] Cristian Martín Fernández, Manuel Díaz Rodríguez, and Bartolomé Rubio Muñoz. An edge computing architecture in the internet of things. In *2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)*, pages 99–102. IEEE, 2018.

[75] James Warren and Nathan Marz. *Big Data: Principles and best practices of scalable realtime data systems*. Simon and Schuster, 2015.

[76] Semir Salkic, Baris Can Ustundag, Tarik Uzunovic, and Edin Golubovic. Edge computing framework for wearable sensor-based human activity recognition. In *International Symposium on Innovative and Interdisciplinary Applications of Advanced Technologies*, pages 376–387. Springer, 2019.

[77] Chetan Sharma Consulting. Edge computing framework: Understanding the opportunity roadmap. 2019.

[78] Apache. Apache kafka. Accessed: 2020-12-11.

[79] P. FAR-EDGE. Far-edge project h2020. *Available online: http://faredge.eu//.*, 2017.

[80] Mauro Isaja, John Soldatos, and Volkan Gezer. Combining edge computing and blockchains for flexibility and performance in industrial automation. In *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*, 2017.

[81] Macchina. Macchina platform for the network edge. Accessed: 2020-11-21.

[82] Xiaofei Wang, Yiwen Han, Victor CM Leung, Dusit Niyato, Xueqiang Yan, and Xu Chen. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(2):869–904, 2020.

[83] Fraunhofer Institute for Integrated Circuit IIS. Ogema. Accessed: 2020-12-03.

[84] Cresco. Cresco. Accessed: 2020-12-15.

[85] Kaikai Liu, Abhishek Gurudutt, Tejeshwar Kamaal, Chinmayi Divakara, and Praveen Prabhakaran. Edge computing framework for distributed smart applications. In *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–8. IEEE, 2017.

[86] Cisco. Cisco fog director. Accessed: 2020-12-17.

[87] Crosser. Crosser. Accessed: 2020-12-17.

[88] The linux foundation projects. Edgex. Accessed: 2020-12-15.

[89] Apache. Edgent. Accessed: 2020-12-17.

[90] Edge Computing Consortium and the Industrial Internet Alliance. Edge computing reference architecture 2.0.

[91] Tseng Mitch, Edmunds Todd, and Canaran Lalit. Introduction to edge computing in iiot, 2018.

[92] Adisorn Lertsinsrubtavee, Anwaar Ali, Carlos Molina-Jimenez, Arjuna Sathiaseelan, and Jon Crowcroft. Picasso: A lightweight edge computing platform. In *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, pages 1–7. IEEE, 2017.

[93] Shuiguang Deng, Hailiang Zhao, Weijia Fang, Jianwei Yin, Schahram Dustdar, and Albert Y Zomaya. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, 7(8):7457–7469, 2020.

[94] J. Chen and X. Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, August 2019.

[95] Meng-Fan Chang et al. White paper on ai chip technologies. 2018.

[96] Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, and Min Chen. In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165, 2019.

[97] Jamie Shotton, Toby Sharp, Pushmeet Kohli, Sebastian Nowozin, John Winn, and Antonio Criminisi. Decision jungles: Compact and rich models for classification. 2016.

[98] ONR MURI Intel and NSF. Kinect dataset. Accessed: 2020-11-21.

[99] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[100] Feng Nan, Joseph Wang, and Venkatesh Saligrama. Pruning random forests for prediction on a budget. In *Advances in neural information processing systems*, pages 2334–2342, 2016.

[101] Li-Jia Li, Hao Su, Fei-Fei Li, and Eric P Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. 2010.

[102] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[103] Chirag Gupta, Arun Sai Suggala, Ankit Goyal, Harsha Vardhan Simhadri, Bhargavi Paranjape, Ashish Kumar, Saurabh Goyal, Raghavendra Udupa, Manik Varma, and Prateek Jain. Protonn: Compressed and accurate knn for resource-scarce devices. In *International Conference on Machine Learning*, pages 1331–1340. PMLR, 2017.

[104] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection, 2005.

[105] MIT. Mit pedestrian database. Accessed: 2020-12-22.

[106] Ling Mao, Mei Xie, Yi Huang, and Yuefei Zhang. Preceding vehicle detection using histograms of oriented gradients. In *2010 International Conference on Communications, Circuits and Systems (ICCCAS)*, pages 354–358. IEEE, 2010.

[107] Nuance. Dragon naturallyspeaking. Accessed: 2021-01-24.

[108] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin Leung, Christian Makaya, Ting He, and Kevin Chan. When edge meets learning: Adaptive control for resource-constrained distributed machine learning. *In Proceedings of the Conference IEEE INFOCOM 2018*, April 2018.

[109] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of Computational Methods in Engineering*, pages 1–22, 2019.

[110] Byron Reese. Ai at the edge: A gigaom research byte. 2019.

[111] Zhang Chiliang, Hu Tao, Guan Yingda, and Ye Zuochang. Accelerating convolutional neural networks with dynamic channel pruning. In *2019 Data Compression Conference (DCC)*, pages 563–563. IEEE, 2019.

[112] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2178–2188, 2017.

[113] Taylor Simons and Dah-Jye Lee. A review of binarized neural networks. *Electronics*, 8(661), June 2019.

[114] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated

learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[115] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[116] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. *Advances in neural information processing systems*, 30, 2017.

[117] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, March 2015.

[118] Michael Blot, David Picard, Nicolas Thome, and Matthieu Cord. Distributed optimization for deep learning with gossip exchange. *Neurocomputing*, 330:287–296, 2019.

[119] Michael Blot, David Picard, Matthieu Cord, and Nicolas Thome. Gossip training for deep learning. *arXiv preprint arXiv:1611.09726*, 2016.

[120] Tan Zhang, Aakanksha Chowdhery, Paramvir Bahl, Kyle Jamieson, and Suman Banerjee. The design and implementation of a wireless video surveillance system. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 426–438, 2015.

[121] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodik, Leana Golubchik, Minlan Yu, Paramvir Bahl, and Matthai Philipose. Videoedge: Processing camera streams using hierarchical clusters. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 115–131. IEEE, 2018.

[122] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[123] Apple. Deep learning for siri's voice: On-device deep mixture density networks for hybrid unit selection synthesis. *[Online]. Available: https://machinelearning.apple.com/2017/08/06/sirivoices.html*, 2017.

[124] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[125] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[126] Apple. Hey siri: An on-device dnn-powered voice trigger for apple's personal assistant. *[Online]. Available: https://machinelearning.apple.com/2017/10/01/heysiri.html*, 2017.

[127] Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. *arXiv preprint arXiv:1901.02358*, 2019.

[128] Y. Zhao A. Zhang S. Yao, S. Hu and T. Abdelzaher. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. *Proc. 26th Int. Conf. World Wide Web*, pages 351–360, 2017.

[129] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. *Proc. IEEE Int. Conf. Comput. Vis.*, pages 2056–2063, December 2013.

[130] Tung-Chien Chen, Wei-Ting Wang, Kloze Kao, Chia-Lin Yu, Code Lin, Shu-Hsin Chang, and Pei-Kuei Tsung. Neuropilot: A cross-platform framework for edge-ai. In *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 167–170. IEEE, 2019.

[131] David Jeans. Related's hudson yards: Smart city or surveillance city? *Online]. Available: https://therealdeal.com/2019/03/15/hudsonyards-smart-city-or-surveillance-city/*, March 2019.

[132] Muhammad Sharjeel Zareen, Shahzaib Tahir, Monis Akhlaq, and Baber Aslam. Artificial intelligence/ machine learning in iot for authentication and authorization of edge devices. In *2019 International Conference on Applied and Engineering Mathematics (ICAEM)*, pages 220–224, 2019.

[133] Xueshi Hou, Sujit Dey, Jianzhong Zhang, and Madhukar Budagavi. Predictive view generation to enable mobile 360-degree and vr experiences. In *Proceedings of the 2018 Morning Workshop on Virtual Reality and Augmented Reality Network*, pages 20–26, 2018.

[134] Shahryar Afzal, Jiasi Chen, and KK Ramakrishnan. Characterization of 360-degree videos. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, pages 1–6, 2017.

[135] Amit Jindal, Andrew Tulloch, Ben Sharma, Bram Wasti, Fei Yang, Georgia Gkioxari, Jaeyoun Kim, Jason Harrison, Jerry Zhang, Kaiming He, et al. Enabling full body ar with mask r-cnn2go, 2018.

[136] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[137] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. Towards wearable cognitive assistance. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 68–81, 2014.

[138] Hyongsuk Kim, Maheshwar Pd Sah, Changju Yang, Tamás Roska, and Leon O Chua. Neural synaptic weighting with a pulse-based memristor circuit. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(1):148–158, 2011.

[139] Timo Wunderlich, Akos F Kungl, Eric Müller, Andreas Hartel, Yannik Stradmann, Syed Ahmed Aamir, Andreas Grübl, Arthur Heimbrecht, Korbinian Schreiber, David Stöckel, et al. Demonstrating advantages of neuromorphic computation: a pilot study. *Frontiers in neuroscience*, 13:260, 2019.

[140] Myonglae Chu, Byoungho Kim, Sangsu Park, Hyunsang Hwang, Moongu Jeon, Byoung Hun Lee, and Byung-Geun Lee. Neuromorphic hardware system for visual pattern recognition with memristor array and cmos neuron. *IEEE Transactions on Industrial Electronics*, 62(4):2410–2419, 2014.

[141] Firas Al-Ali, Thilina Doremure Gamage, Hewa WTS Nanayakkara, Farhad Mehdipour, and Sayan Kumar Ray. Novel casestudy and benchmarking of alexnet for edge ai: From cpu and gpu to fpga. In *2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–4. IEEE, 2020.

[142] Yujeong Choi and Minsoo Rhu. Prema: A predictive multi-task scheduling algorithm for preemptible neural processing units. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 220–233. IEEE, 2020.

[143] Xuechao Wei, Cody Hao Yu, Peng Zhang, Youxiang Chen, Yuxin Wang, Han Hu, Yun Liang, and Jason Cong. Automated systolic array architecture synthesis for high throughput cnn inference on fpgas. In *Proceedings of the 54th Annual Design Automation Conference 2017*, pages 1–6, 2017.

[144] Zhijie Yang, Lei Wang, Dong Ding, Xiangyu Zhang, Yu Deng, Shiming Li, and Qiang Dou. Systolic array based accelerator and algorithm mapping for deep learning algorithms. In *IFIP International Conference on Network and Parallel Computing*, pages 153–158. Springer, 2018.

[145] Shamik Kundu, Kanad Basu, Mehdi Sadi, Twisha Titirsha, Shihao Song, Anup Das, and Ujjwal Guin. Special session: Reliability analysis for ml/ai hardware. *arXiv preprint arXiv:2103.12166*, 2021.

[146] Google I/O. Google tpu. Accessed: 2021-01-24.

[147] Nvidia. Nvidia jetson nano. Accessed: 2020-11-21.

[148] General Vision. Neuroshield. Accessed: 2020-11-21.

[149] Google. Google coral. Accessed: 2020-11-21.

[150] Intel. ntel movidius. Accessed: 2020-12-03.

[151] Google. Sparkfun edge. Accessed: 2020-12-03.

[152] Christine Long. Beaglebone ai makes a sneak preview, 2019. Accessed: 2020-12-17.

[153] Gopinath Sridhar, Ghanathe Nikhil, Seshadri Vivek, and Rahul Sharma. Machine learning models to tiny iot devices. *In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 79–95, 2019.

[154] Alasdair Allan. Hands-on with the smart-edge agile. Accessed: 2020-12-17.

[155] Ahmed Ghoneim, Ghulam Muhammad, Syed Umar Amin, and Brij Gupta. Medical image forgery detection for smart healthcare. *IEEE Communications Magazine*, 56(4):33–37, 2018.

[156] Pei-Kuei Tsung, Tung-Chien Chen, Chien-Hung Lin, Chih-Yu Chang, and Jih-Ming Hsu. Heterogeneous computing for edge ai. In *2019 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 1–2. IEEE, 2019.

[157] Shouyi Yin, Peng Ouyang, Shibin Tang, Fengbin Tu, Xiudong Li, Leibo Liu, and Shaojun Wei. A 1.06-to-5.09 tops/w reconfigurable hybrid-neural-network processor for deep learning applications. In *2017 Symposium on VLSI Circuits*, pages C26–C27. IEEE, 2017.

[158] Shouyi Yin, Peng Ouyang, Shibin Tang, Fengbin Tu, Xiudong Li, Shixuan Zheng, Tianyi Lu, Jiangyuan Gu, Leibo Liu, and Shaojun Wei. A high energy efficient reconfigurable hybrid neural network processor for deep learning applications. *IEEE Journal of Solid-State Circuits*, 53(4):968–982, 2017.

[159] Michaela Blott, Thomas B Preußer, Nicholas J Fraser, Giulio Gambardella, Kenneth O'brien, Yaman Umuroglu, Miriam Leeser, and Kees Vissers. Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 11(3):1–23, 2018.

[160] Quentin Ducasse, Pascal Cotret, Loïc Lagadec, and Robert Stewart. Benchmarking quantized neural networks on fpgas with finn. *arXiv preprint arXiv:2102.01341*, 2021.

[161] Qi Yu, Chao Wang, Xiang Ma, Xi Li, and Xuehai Zhou. A deep learning prediction process accelerator based fpga. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 1159–1162. IEEE, 2015.

[162] Griffin Lacey, Graham W Taylor, and Shawki Areibi. Deep learning on fpgas: Past, present, and future. *arXiv preprint arXiv:1602.04283*, 2016.

[163] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, and et al. Going deeper with embedded fpga platform for convolutional neural network. *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 26–35, 2016.

[164] Li Du, Yuan Du, Yilei Li, Junjie Su, Yen-Cheng Kuan, Chun-Chen Liu, and Mau-Chung Frank Chang. A reconfigurable streaming deep convolutional neural network accelerator for internet of things. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(1):198–208, 2017.

[165] Loc N Huynh, Youngki Lee, and Rajesh Krishna Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 82–95, 2017.

[166] Google. Tensorflow lite. Accessed: 2020-11-21.

[167] Google. Tensorflow lite performance. Accessed: 2020-11-21.

[168] Xingzhou Zhang, Yifan Wang, and Weisong Shi. pcamp: Performance comparison of machine learning packages on the edges. In {*USENIX*} *Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[169] Apache. Apache mxnet. Accessed: 2020-12-03.

[170] Facebook. Caffe2go. Accessed: 2020-12-03.

[171] Apple. Coreml3 from apple. Accessed: 2020-12-03.

[172] Dawei Li, Theodoros Salonidis, N. Desai, and M. Chuah. Deepcham: Collaborative edge-mediated adaptive deep learning for mobile object recognition. *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 64–76, 2016.

[173] Zhuoran Zhao, Kamyar Mirzazad Barijough, and Andreas Gerstlauer. Deepthings: Distributed adaptive deep learning inference on resource-constrained iot edge clusters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2348–2359, 2018.

[174] Shuochao Yao, Yiran Zhao, Aston Zhang, Lu Su, and Tarek Abdelzaher. Deepiot: Compressing deep neural network structures for sensing systems with a compressor-critic framework. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pages 1–14, 2017.

[175] Sourav Bhattacharya and Nicholas Lane. Sparsification and separation of deep learning layers for constrained resource inference on wearables. *In Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM, SenSys*, pages 176–189, November 2016.

[176] Google. Ml kit. Accessed: 2020-12-03.

[177] Yunbin Deng. Deep learning on mobile devices - a review. *arXiv*, 2019.

[178] Xnor. Ai2go. Accessed: 2020-12-03.

[179] Alasdair Allan. Benchmarking the xnor ai2go platform on the raspberry pi. Accessed: 2021-01-17.

[180] Amazon. Aws greengrass. Accessed: 2020-12-17.

[181] Foghorn. Foghorn framework. Accessed: 2020-12-17.

[182] Microsoft. Azure iot edge. Accessed: 2020-12-17.

[183] Xingzhou Zhang, Yifan Wang, Sidi Lu, Liangkai Liu, Weisong Shi, et al. Openei: An open framework for edge intelligence. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1840–1851. IEEE, 2019.

[184] Sicong Liu, Junzhao Du, Kaiming Nan, Zimu Zhou, Hui Liu, Zhangyang Wang, and Yingyan Lin. Adadeep: a usage-driven, automated deep model compression framework for enabling ubiquitous intelligent mobiles. *IEEE Transactions on Mobile Computing*, 2020.

[185] Brandon Reagen, Paul Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David Brooks. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 267–278. IEEE, 2016.

[186] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.

[187] Hongshan Li, Chenghao Hu, Jingyan Jiang, Zhi Wang, Yonggang Wen, and Wenwu Zhu. Jalad: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 671–678. IEEE, 2018.

[188] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[189] Jeff Daily, Abhinav Vishnu, Charles Siegel, Thomas Warfel, and Vinay Amatya. Gossipgrad: Scalable deep learning using gossip communication based asynchronous gradient descent. *arXiv preprint arXiv:1803.05880*, 2018.

[190] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. Pipedream: generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 1–15, 2019.

[191] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11, 2013.

[192] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *ICML*, 2011.

[193] Garrett B Goh, Khushmeen Sakloth, Charles Siegel, Abhinav Vishnu, and Jim Pfaendtner. Multimodal deep neural networks using both engineered and learned representations for biodegradability prediction. *arXiv preprint arXiv:1808.04456*, 2018.

[194] Bakkiam David Deebak and Fadi Al-Turjman. A hybrid secure routing and monitoring mechanism in iot-based wireless sensor networks. *Ad Hoc Networks*, 97:102022, 2020.

[195] Rekha Goyat, Gulshan Kumar, Mritunjay Kumar Rai, Rahul Saha, Reji Thomas, and Tai Hoon Kim. Blockchain powered secure range-free localization in wireless sensor networks. *Arabian Journal for Science and Engineering*, 45(8):6139–6155, 2020.

[196] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. Spiking neural networks hardware implementations and challenges. *ACM Journal on Emerging Technologies in Computing Systems*, 15(2):1–35, July 2019.

[197] Jörg Drechsler and Jerome P.Reiter. An empirical evaluation of easily implemented, non parametric methods for generating synthetic datasets. *Computational Statistics and Data Analysis*, 55(12):3232–3243, December 2011.

[198] Georgia Albuquerque, Thomas Lowe, and Marcus Magnor. Synthetic generation of high-dimensional datasets. *IEEE transactions on visualization and computer graphics*, 17(12):2317–2324, 2011.

[199] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv*, pages 1–23, 2020.

[200] David D Clark, Craig Partridge, J Christopher Ramming, and John T Wroclawski. A knowledge plane for the internet. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–10, 2003.

[201] Albert Mestres, Alberto Rodriguez-Natal, Josep Carner, Pere Barlet-Ros, Eduard Alarcón, Marc Solé, Victor Muntés-Mulero, David Meyer, Sharon Barkai, Mike J Hibbett, et al. Knowledge-defined networking. *ACM SIGCOMM Computer Communication Review*, 47(3):2–10, 2017.

[202] Wolfgang Kellerer, Patrick Kalmbach, Andreas Blenk, Arsany Basta, Martin Reisslein, and Stefan Schmid. Adaptable and data-driven softwarized networks: Review, opportunities, and challenges. *Proceedings of the IEEE*, 107(4):711–731, 2019.

[203] Yupin Huang, Liping Qian, Anqi Feng, Ningning Yu, and Yuan Wu. Short-term traffic prediction by two-level data driven model in 5g-enabled edge computing networks. *IEEE Access*, 7:123981–123991, 2019.

Chapter 6

Automatic Label Creation Framework for FMCW Radar Images Using Camera data

Javier Mendez[1,3], Stephan Schoenfeldt[1], Xinyi Tang[1], Jakob Valtl[1], M.P. Cuellar[2], Diego P. Morales[3].

1. Infineon Technologies AG, Am Campeon 1-15, 85579 Neubiberg, Germany

2. Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain

3. Department of Electronics and Computer Technology, University of Granada, 18071 Granada, Spain

IEEE Access

- Received April 2021, Accepted June 2021, Published June 2021
- 10.1109/ACCESS.2021.3087207
- Impact factor: 3.367
- JCR Rank: 94/273 in category Engineering, Electrical Electronic (Q2)

Abstract: Data acquisition and treatment are key issues for any Deep Learning (DL) technique, especially in computer vision tasks. A big effort must be done for the creation of labeled datasets, due to the time this task requires and its complexity in cases where different sensors must be used. This is the case of radar imaging applications, where radar data are difficult to analyze and must be labeled manually. In this paper, a semi-automatic framework to generate labels for range Doppler maps (radar images) is proposed. This technique is based on a sensor fusion approach with radar and camera sensors. The proposed scheme operates in two steps: The first step is the environment features extraction, in which the radar data is preprocessed and filtered to remove ghost targets and detect clusters, and camera data are used to extract the information of the targets. In the second step, a rule-based system that considers the extracted features fuses the information to generate labels for the radar data. By using the proposed framework, the experimentation performed suggests that the time required to label the data is reduced as well as the possibility of human error during the labeling task. Our results show that the proposed technique can improve the final model accuracy with regards the traditional labeling method, carried out by human experts.

Keywords: Sensor Fusion, Machine Learning algorithms, Deep Learning, radar, auto-labeling system.

6.1. Introduction

In recent years, radar imaging techniques have been proven to provide high performance results when used for classification tasks in autonomous driving (1)(2), object detection (3)(4) and activity recognition (5)(6). Researchers have also studied the integration of Machine Learning (ML) techniques for the radar signal preprocessing (5) as well as the previously commented tasks (3)(2)(6) to achieve high performance results. However, large application-specific datasets are required when training Deep Learning (DL) models for these purposes using a supervised approach.

The creation of new datasets is a current problem due to the required time to gather the data and, especially, to correctly label the data. In order to solve this problem, new training approaches are been researched to avoid the labeling step of the dataset. An example of this is the use of Reinforcement Learning (7)(8), where the DL model is trained without a dataset itself but in a simulated environment. Even when these techniques are being researched, a large set of applications still use supervised training (9)(10). Therefore, labeled datasets are still required for new applications.

In recent approaches, some authors have studied the implementation of ML techniques to label the datasets automatically. These techniques aim

to reduce the human errors in the labeling process as well as the time required for this task. Some examples of these techniques are proposed in (11)(12)(13)(14)(15), which are deeply analyzed in Section 2. Some of these techniques are based on a Sensor Fusion approach such as (1). These Sensor Fusion techniques can be divided into Early Fusion and Late Fusion pipelines. The Early Fusion approach combines the data with a low level of preprocessing to generate new raw data that can be later studied as a single input. The Early Fusion can also be used to extract the final information from the initial raw data from multiple sensors. This technique is beneficial when the data can be merged easily due to similarities in the format or the features. The issues of this technique fall into the restriction of using data with similar formats or implementing complex algorithms to overcome these differences as in (1). On the other hand, the Late Fusion combines the data at a later step after a deeper preprocessing of the data. This approach can be used to efficiently study each data separately to determine what features can be useful before the merging step. This leads to a further preprocessing that can be optimized for each data type. At the same time, the complexity of the fusing algorithm can be reduced.

Conventionally, manually labeling radar data is performed frame by frame. The efficiency of this process is limited due to the fact that a ground truth must be provided for each radar image. At the same time, it is possible to incur in human errors during the execution of this task, mainly as misinterpretation of the data. This is a result of the complexity and non-intuitive visualization of the radar data.

The developed framework in this paper aims to help during the dataset creation using a Late Sensor Fusion pipeline. This pipeline, based on DL models as well as traditional approaches, merges relevant features from input data after an individual preprocessing. These specific preprocessing techniques are optimized to extract accurate and relevant information from the camera and radar sensors. The developed framework aims to generate labels for the studied dataset. By applying this pipeline, the possibility of incurring in human error is reduced, and also the required time to study new data, as shown in the experiments in Section 4.

As a secondary objective, we study the implementation of the resulting DL models in edge devices with low hardware resources. For this reason, different Deep Learning frameworks have been researched as well as the memory allocation of the data. The experimental section discusses performance both in accuracy and time, respectively.

The remaining of the manuscript is organized as follows: Section 2 focuses on related works to the addressed problem. In Section 3, the Sensor Fusion pipeline is described for a deep understanding of the label creation process. After that, Section 4 will focus on the experiment where this framework has been applied, and finally Section 5 concludes.

6.2. Related Works

The use of radar sensors for classification tasks is increasing its popularity. This enables a system to recognize its environment (1)(3)(4)(2) without having to deal with privacy concerns as it happens when using camera data. However, this leads to a requirement of large specific labeled datasets for each application (when using AI models that required a supervised training). The creation of labeled datasets is a bottleneck in the ML model creation due to the time required for its creation as well as the possibility to incur human errors during the labeling process. As a result, numerous researchers are working on this field.

An example of this approach is the object detection based on the fusion of these sensors researched by F. Nobis et al. (1). This Early Fusion approach is based on the data fusion using an Artificial Neural Network (ANN). The raw data from the camera sensor and the low-preprocessed radar data are used as inputs of the ANN that execute the fusion and classification process. At the same time the system studies at what point the fusing process should be executed to obtain the best classification results. In this technique, the radar data is used as reinforcement, as 2D points, for the camera image in the ANN. This technique proved its utility in environments where the camera data are corrupted or they do not provide enough information, for example in dark environments or with extreme weather conditions such as rain. The accuracy provided with this approach exceeds the state-of-the-art results obtained only with camera data in the NuScenes dataset as well as the Technical University of Munich (TUM) dataset. The limitation of this technique relies in the complex structure of the Deep Neural Network (DNN) designed for the fusion tasks. However, due to the study of multiple frames, this technique can achieve state-of-the-art results. The accuracy of this algorithm is further compared with the proposed framework in Subsection 4.4. where it is presented how our tool achieves a 24.556 % higher accuracy when studying single frames.

Teck-Yian Lim et al. (16) proposed a similar pipeline of (1)to the previously commented. Their proposed pipeline of Early Fusion to combine radar and camera information for target detection is based on an ANN. This ANN has an input for the radar data and another one for the camera data. Therefore, the data can be studied independently in a first step to extract high level features before merging them. The main difference of this technique respect (1) is the data type used. They use range-azimuth radar images instead of 2D points. This approach allows the system to employ feature pyramid network structures. Since there are no public datasets with this radar data, they built their own dataset to evaluate their technique, achieving a 73.5 %. Their low accuracy in comparison with other techniques may be due to the Early Fusion approach followed. This limits the evaluation of the

preprocessing of the data before the fusion step, what can lead to efficiency problems when extracting the features. At the same time, the proposed pipeline contains 2 Single Shot Detectors. As a result, the complexity of the algorithm is higher than our proposed pipeline.

Z. Ji et al. (17) proposed a method to locate and classify objects based on radar and camera sensors. This method can be divided in two steps: In the first step, the data are preprocessed to extract the possible target location from the radar data using a Kalman filter and, in the second step, these possible points are projected into the camera plane to locate the relevant areas of the camera image to study. Later, these areas of the camera images are analyzed using a Multilayer In-place Learning Network to classify the objects into a set of known categories. The overall accuracy obtained with this method is 96.8 % in the dataset studied in the paper which contains 400 images. Because of the pipeline of this method, the time required to classify all the objects depends on the number of objects due to the fact that each target is fed into the DNN individually in a loop approach. This loop approach may lead to high latency in environments with high density of targets in contrast with other techniques such as Single Shoot Detection (SSD) or Faster R-CNN where the whole image is studied at the same time by the DNN.

J. Kocic et al. (18) shows a pipeline for sensor fusion in the field of autonomous driving. The presented pipeline has three steps. In the first step, the data collected by multiple sensors is preprocessed to represent the same environment: LiDAR and radar generate 3D point clouds and the camera provides RGB images. These two types of data are fed into two different DNNs, one for the 3D points and one for the RGB image to generate labels for the objects in the environment separately. These labels are later fed into another DNN to execute a high-level fusion. Apart from localization, the results from the high-level fusion can be used to generate an occupancy grid surround the ego vehicle. The benefits of this structure lie on the fact it avoids lossy input predictions while using simple DNNs already used in other applications. At the same time, these factors are also the limitations for this method. Because of integrating multiple DNN in the same system, the resource requirements of this approach are higher than other techniques developed in previously commented papers as well as the technique explain in our paper. These multiple DNNs may also lead to high latency even when the paper does not research the output frame rate this pipeline can generate.

X. Zhang et al. (19) follows a similar approach to the previous authors. They propose a Late Sensor Fusion pipeline where a millimeter-wave radar is used to extract the position and the speed of the obstacles. The data of the location of the obstacles is used to generate a region of interest for a deeper study in the data collected by a camera sensor, where Machine Learning techniques are applied to extract reliable 3D bounding boxes for the objects

and track them. The results obtained with this technique are the 91.6 % of accuracy on dataset used for their experiment. Because of its similarities, the limitations of this technique lie in the same facts as the techniques presented by J. Kocic et al. (18) and Z. Ji et al. (17).

The system proposed in (2) also uses radar data to classify and locate objects. The technique applied to study the data in this paper is based on the intensity and decayed spectrum. The approach used in that paper is based on a DNN following a traditional approach to create an Object Detector where first the clusters, based on range-azimuth intensity map, are located before been fed into a Convolutional Neural Network (CNN) to classify them. The paper proves how this approach can achieve state-of-the-art accuracy for the object classification without depending on any other sensor due to the fact that they achieve an accuracy of 65.30 % at an initial frame, being later further improved with accumulated frames. Our technique can provide results on real time with higher accuracy due to the use of Range Doppler Maps (RDM) images instead of the range-azimuth intensity map followed by the authors of this paper.

Focus on the autolabeling process, as in our paper, T. Winterling et al. (11) proposed a technique based on a CNN to automatically generate labels for occupancy grid maps generated from radar data. In (11), the initial data is preprocessed to obtain an occupancy grid where each cell contains the information of the probability of an object being at that position. These maps are fed into a CNN where the author had to manually label a set of data in each iteration to improve the accuracy achieved. Therefore, this is an iterative process that still requires human interaction during the labeling process to manually label the unsure data of the dataset, in contrast with the proposed framework in our paper.

The problem presented in the previous paper was also researched by M. Di Cicco et al. (13). In their manuscript, the authors explained how the creation of labeled datasets for Deep Learning is extremely time consuming due to the volume of data require as well as the complexity to label correctly each frame. As a result, they proposed a technique to create datasets for new applications in the agriculture field based on synthetic data creation. The dataset created with this technique proved its quality by training a DNN using this dataset and studying its final accuracy over a manually labeled dataset. This technique, even when it reduces the human effort and time to create the dataset, does not target the problem researched in our paper due to the fact that it use synthetic data instead of automatizing the labeling process as we have researched in our paper.

M. Suchi et al. (14) also proposed a technique to create a semi-automatic labeled datasets of RGB images at pixel level. During the dataset creation, their tool requires data from a depth sensor apart from the RGB images. The followed approach is based on comparing the distance of each 3D point

with the neighbors points by exploiting spatial shifts in the depth data to determine if it belong to an object. To overcome the problems due to the variation of the distance of the objects respect the recording system, they include a preprocessing to adapt the system to the range of the target. This approach is similar to our technique in the sense it is based on a sensor fusion paradigm for the label creation. However, since we are targeting radar data, this technique cannot be used due to the fact that we are missing the data that a depth sensor could provide to accurately differentiate targets that are near each other.

Following the idea of combining data from multiple sources as in the previous technique, Gabriel M. et al. (15) developed an approach to automatically create datasets of audio, lyrics and notes. Their technique is based on karaoke user data that contains annotations of time-aligned lyrics and notes. In a later step, this data is compared with audio candidates from the web to select the best candidate using a CNN. In the last step, a teacher-student paradigm is applied to improve the results obtained with this DL model.

Pursuing an implementation without using Deep Learning techniques, J. Tang et al. (12) researched the auto-labeling of images for image classification. They compared different algorithms for the labeling process on multiple datasets in order to establish the benefits from each technique. The techniques researched include techniques such as CSD-Prop, SvdCos, and CSD-SVM. Among the researched techniques, the CSD-SVM provided the best results taking into account the quantity of prior information required by the system. This technique labeled correctly 767 images out of the 4500 images in the Corel dataset (20) while the CSD-Prop and SvdCos labeled 577 and 349 images respectively. However, the number of correct labels does not achieve a large enough volume to be implemented as an automatic labeling system for the creation of labeled datasets.

After showing the most relevant research papers that justify our research, we can see that some of these papers explain how the use of radar in collaboration with ML techniques for object detection is an emerging research line. The accuracy as well as latency results of these techniques can still be further improved. This has led us to develop the semi-automatic labeling framework. In order to improve the previously explained techniques, some of the stages of the pipeline of our framework have been optimized to improve the latency as well as the memory consumption. At the same time, it is possible to see how previous automatic labeling techniques have not being able to create an accurate and efficient process in contrast with our proposed framework since their accuracy is not high enough (12) or they do not directly face the problem of labeling the data (13).

The approach proposed in our paper is based in an DNN to extract relevant features for the camera data. The later data fusion, at object level,

is based on rule-based system approach using the cluster Angle of Arrival (AoA) and distance of the targets from each sensor to match the targets from both sensors. As a result, high accuracy is achieved in our dataset while maintaining low latency.

The proposed preprocessing method, as well as the Sensor Fusion approach, will be explained in detail in the following section.

6.3. Sensor Fusion pipeline

This section describes the preprocessing techniques applied in the proposed framework to reduce the noise and reconstruct missing data. At the same time, this preprocessing extract high-level features to be fused in a later step. We remark that our final target application is self-autonomous driving and, more specifically, object recognition and localization.

The Sensor Fusion pipeline for the proposed autolabeling framework in this paper consists of two main modules: data preprocessing and data fusion. These modules can be subdivided depending on the source data sensor, as shown in the Figure 6.1.

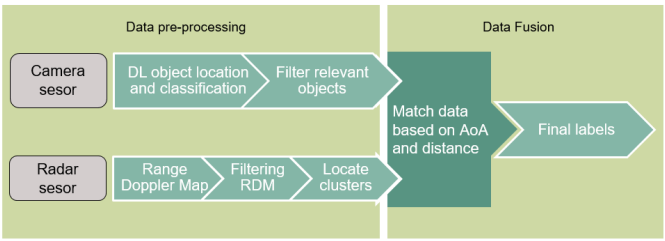

Figure 6.1: Sensor Fusion pipeline.

This pipeline follows a natural information flow from raw data to feature extraction and a final step where these high-level features are fused to obtain the final data. The final data, extracted using this pipeline, is the information about the classification of the clusters from the RDM image, a bounding box estimation, the AoA and the presence of multiple objects in the cluster.

In order to extract relevant features, the data from both sensors is preprocessed separately due to the differences in their features/structures. As a result of this, a specific preprocessing pipeline is designed for each data type. The preprocessing of the radar data is based on a traditional signal preprocessing pipeline. The raw radar data is converted into a new format, RDM, where the relevant features can be extracted more efficiently. This is followed by a filtering step to reduce the noise. On the other hand, the camera data preprocessing is based on computer vision techniques, specifically DNN. DNN were chosen due to their high accuracy results achieved for computer vision in the literature. These preprocessing techniques are further

explained in the next two subsections for a deeper understanding.

6.3.1. Radar data preprocessing

A Multiple-input-multiple-output (MIMO) frequency modulated continuous wave (FMCW) radar with four channels is used. The number of channels can be further increased by applying virtual array concept with a cost of lower frame rate. Calibration was applied to correct the AoA estimation errors after adding a radar random. The sampling rate, frame size, chirp bandwidth and chirp time are properly selected to ensure a good resolution within our region of interest.

In order to fuse the information from the radar data and the camera data, the object is first detected and located in the radar RDM data to reduce the data dimension. This technique transforms the time domain ADC signals to RDM images, where the information about the distance and speed of the targets is maintained. A Constant False Alarm Rate (CFAR) filter is then applied to the RDM image for target detection (21)(22). This pipeline is presented in Figure 6.2.

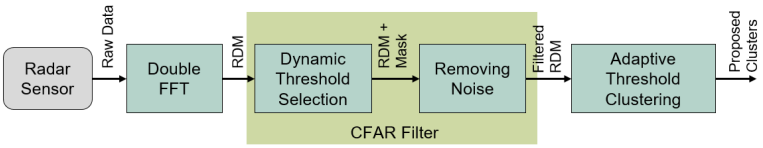

Figure 6.2: Radar data processing pipeline.

The CFAR technique compares each bin of the RDM with its surrounding ones, setting a maximum range called training cells without including a subset of this set called guard cells. This filter is used to estimate the presence of a real target within the bin under test. There are training bins near the bin under test to compute the noise floor. Immediately adjacent bins to the bin under test are considered as guard bins and are ignored so that the possible leaked signals do not contribute to the noise floor computation. A bin is declared to contain an object when its value is greater than the scaled noise floor. With CFAR, the computed noise floor acts as a dynamic threshold instead of a fixed threshold value.

In Figure 6.2, the CFAR filtering has been divided into the two subprocessed required for its implementation: dynamic threshold selection and filtering the points whose value is under the threshold. This dynamic threshold is calculated using (6.1), where m and n are the horizontal and vertical number of training cells and $\hat{m}$ and $\hat{n}$ are the guard cells range horizontally and vertically.

$$
\begin{aligned}
T_i = (\sum_i \sum_j c_{ij})/(4(mn - \hat{m}\hat{n})) \\
i \in [-m, -\hat{m}] \cup [\hat{m}, m] \\
j \in [-n, -\hat{n}] \cup [\hat{n}, n]
\end{aligned}
\tag{6.1}
$$

At this point, it is possible to extract clusters of real targets from the RDM images with high accuracy. An adaptive threshold technique has been used to determine the limits of the clusters. For each of the proposed clusters, a proportional threshold to its maximum value is generated. This threshold can be divided into speed threshold and range threshold to study the vertical and horizontal bins respectively. A later algorithm will use these thresholds to study the surrounding points of each cluster to determine the associated bounding box of each object. Therefore, the detection of a target is independent of target size (due to the precision of the radar sensor, real target size or distance to the object) in the RDM image.

It is important to understand that the white horizontal central line in the RDM images, as shown in Figure 6.3, represents the static objects captured by the radar sensor. In this representation, when multiple targets are static, they generate a line in the no speed area (the center line) that must be ignored since they cannot be separated for a proper classification.

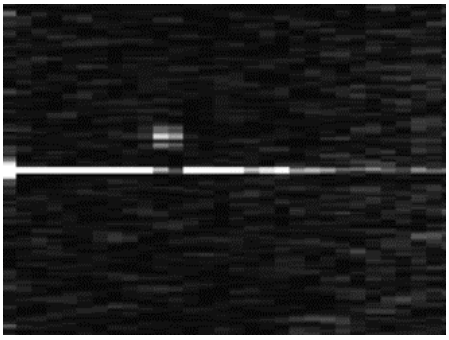

Figure 6.3: RDM frame example.

These detected clusters will be later used in the fusing process with the targets located in the camera data to extract the label data expected from this framework.

6.3.2. Camera data preprocessing

The data recorded from the camera will be fed into a DL object detector to extract the classification, AoA and location of the targets in each frame as it is presented in Figure 6.4.

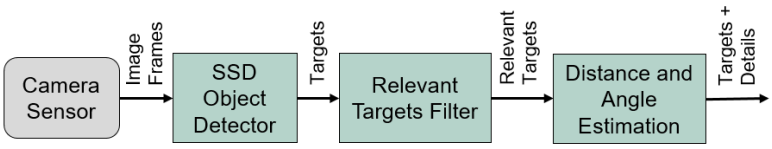

Figure 6.4: Camera data processing pipeline.

The SSD structure (23) has been chosen due to its capabilities to enable real-time detection in comparison with other state-of-the-art structures such as Faster R-CNN (24). SSD structure speeds up the processing of the data by removing the region proposal network present in other structures. To recover from the accuracy drop due to this, SSD applies techniques such as multi-scale features and default boxes. These improvements increase its accuracy to match the Faster R-CNN's accuracy while using lower resolution images. As a result, the size of this model as well as the resource requirements to execute it are reduced in comparison with other DL models for object detection.

The loss function of the model consists of two terms, the localization loss and the confidence loss. The first loss is the loss related to the position of the bounding box of the detected targets, penalizing only predictions from positive matches. The equations of this loss metric are (6.2), (6.3), (6.4), (6.5) and (6.6). Localization loss is the loss during the prediction of the target classification, which is calculated using (6.7) and (6.8). These two loss functions are later combined in a general one by (6.9) to communicate the loss of the DNN in a single parameter.

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in (cx,cy,w,h)} x_{ij}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m) \tag{6.2}$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \tag{6.3}$$

$$\hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \tag{6.4}$$

$$\hat{g}_j^{w} = log(\frac{g_j^w}{d_i^w}) \tag{6.5}$$

$$\hat{g}_j^{h} = log(\frac{g_j^h}{d_i^h}) \tag{6.6}$$

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p - \sum_{i \in Neg} log(\hat{c}_i^0) \tag{6.7}$$

$$\hat{c}_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)} \tag{6.8}$$

$$L(x, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{6.9}$$

Where N indicates the number of matches default boxes, l represents the predicted boxes, g the ground truth boxes, x the coordinates of the bounding boxes and c the classes confidences. These parameters also include the offset for the center points (cx, cy), the width of the box (w) and its height (h).

The object detector was built based on public pre-trained models available in multiple DL libraries such as MXNet (25) or TensorFlow Lite (26). These DL framework were designed by different companies but they share the same goal, to reduce the resource requirements during the training and inference phases. In this paper we will not aim to train the DL model at the network edge. Nevertheless, these frameworks increase the efficiency of the inference process and reduce the model size.

The pre-trained DL model was used to extract a first iteration of the labels from a reduced number of frames from the camera data. These labels needed to be filtered in order to remove not relevant classes as well as adding new classes. After this step, the new labels are used to train a new DL model for object detection using transfer learning. This technique was used due to the reduced dimension of the new dataset. This new model will be used to generate labels from more frames, increasing the number of samples in our specific dataset as shown in Figure 6.5. This iterative process can lead to exponential errors if the labels are not correct. Therefore, an inspection of the new data will be executed after the first re-training phases, which are the most critical. This ensures the correct application of this approach. The iterative process will finish once the dimensions of the dataset are large enough to ensure an accuracy of the model over 90 % in the data of our experiment.

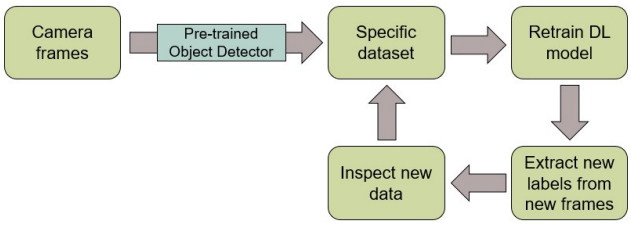

Figure 6.5: Dataset creation.

After training our object detector, labels from the camera data can be generated automatically. These labels may not be relevant for the sensor fusion, due to the fact that the camera may see objects which are not in the range of the radar or they are not relevant for the sensor fusion. As a result, these labels need to be preprocessed to remove non relevant targets before the data fusion stage.

Once the localization of the targets is extracted, it is possible to estimate the angle of arrival of those objects based on their coordinates. The resolution of the camera images plays a key role in this process since it is directly proportional to the precision of the AoA estimation in the camera data. This process is based on assuming an AoA of 0° for targets located in the center of the image and it increases as the targets moves to the right side of the camera frame, as the AoA extracted from the radar data.

In order to ensure the correct measurement of this parameter, these values have been calibrated with the AoA from the radar data. At the same time, the distance of the targets to the camera sensor can be extracted after a calibration based on the specific scenario. As a result of this, the X and Y coordinates of the targets in the camera data can be transformed into distance and AoA measurements. These features will be used for the data fusion process further explained in the next subsection.

6.3.3. Data Fusion

The previous preprocessing techniques provide relevant features from the raw camera and radar data. These features are shown in Table 6.1.

	Features			
Radar sensor	Cluster	AoA in radar	Speed	Distance
Camera sensor	Classification	AoA in camera	Localization	Target groups

Table 6.1: Features extracted from each sensor data.

Before fusing the data, a correct synchronization of the data from both sensors is crucial. To enable this, timestamps have being added to the data during the recording phase. However, the frame rate of both sensors may not be the same. Therefore, the sensor with the highest latency will be used as the standard latency of the system during the synchronization. A possible delay in the timestamps between the sensors has been taking into account during the synchronization step. Therefore, the data will be matched with the nearest data in the time domain.

The labeling process is triggered when a target is detected in the camera data. Therefore targets that are not detected by this sensor, such as highly covered targets or partially visible targets may not be labeled. At this point, the targets localized in the camera are compared with the clusters from the RDM image based on the distance of the targets respect the recording system. The distance between the camera and radar sensors of the system is reduced enough to be able to be ignored in comparison with the distance of the studied targets.

One more limitation of the framework is the difference in the of View (FoV) of both sensors. Since our approach is based on comparing the targets

found in each sensor´s FoV, if the overlapping of the FoV of the studied sensors is not total, there may be targets that are not found by all the sensors. In order to overcome this, only the intersection FoV of these sensors has been studied for the label creation. Another implemented approach to reduce the impact of this limitation is the use of labels from previous frames. By knowing where a target was in a previous frame as well as its speed (extracted from the RDM), it is possible to estimate where it will be located in the current frame. The estimation of the location of the known targets is used to locate relevant areas in the RDM image as well as clusters what may have not being detected initially or when the reflection power of the radar signal was not powerful enough in that frame.

At the same time, to distinguish targets at the same distance, the measured AoA from both sensors is used as auxiliary parameter to ensure the correct labeling. Because of the difference in the resolution of both sensors, an error range of 5° has been included in the algorithm. As a result, the AoA from both sensors does not have to be exactly the same. This approach solves multiple problems due to angle offset between the sensors as well as noise in the data.

There are multiple algorithms to extract the AoA from the radar data in the literature. Among all the methods, the beamforming method (27) was chosen for the experiment. In this technique, the angle is calculated by coherently summing the signals from different receiving channels.

Since this tool should be understood as a first approach for the dataset labeling process, it also generates a text file with relevant information after its execution. This file contains information such as the path for the data and the number of the frames that need to be reviewed. The revision may be necessary because of targets in the camera data that did not match with any cluster of the RDM. This will only be reported when these targets are located in an area where the radar should be able to detect them. Similarly, it will also report cases where there are unmatched clusters in the RDM. As a result of this, the reviewing process can be executed efficiently rather than having to study all the labeled frames.

In order to evaluate the performance of the proposed framework, an experiment has been executed and explained in the next section.

6.4. Experiment

In this section, we evaluate the designed framework on a custom proprietary dataset created by Infineon specifically for this application. Our goal is to compare the accuracy of the labeling process and its latency with a traditional approach of labeling the data manually as well as some state-of-the-art techniques. The techniques used for the comparison are described in (1)(2)(28)(11). These baseline techniques were selected due to the state-of-

the-art results obtained in similar fields as the one research in our paper. Next, we describe the dataset and how it was built.

6.4.1. Dataset

The previously explained framework was tested to label data from radar sensors in a vehicle context where all the studied frames include relevant targets for the application. This main dataset was generated with data from two different locations: Singapore Polytechnic Campus, shown in Figure 6, and Infineon Singapore Campus, shown in Figure 7. The height of the sensors in these locations was different to fit the location as well as gather data from different conditions to create a general dataset. The height of the sensors was 5 and 2.3 meters for the chosen locations respectively.

The relevant categories that have been used for the classification of the labels are: pedestrian, car, truck, bicycle, motorbike and personal mobility device (PMD). The last category was not included in any public-pretrained object detector leading to the test of the addition of new categories in the pipeline. The distribution of the classes has been studied to ensure there are no imbalanced classes during the training phase of the DL model.

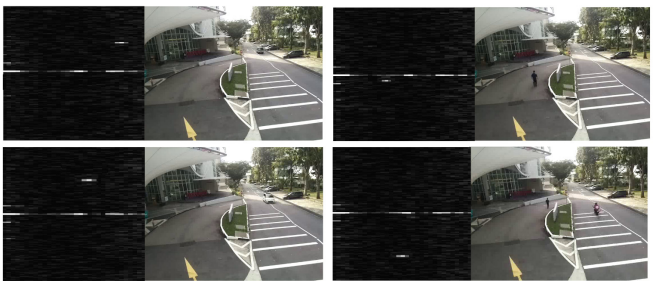

Figure 6.6: Samples of pairs of images used to label the RDM images.

The Figure 6.6 shows some examples of the pair of images from both sensors included in the mentioned dataset. The initial labels for our dataset have been created manually including target classification and 2D bounding boxes.

An additional dataset was generated using different radar and camera sensors as well as a different scenario to test the tool developed with the previous dataset. This reduced dataset contains 50 frames with 2 target classes (vehicle and obstacle). In this case, the sensors were integrated in a reduced scale vehicle where the height of the sensors was 30 centimeters to the floor. An example of camera and radar RDM input data from the second dataset is shown in Figure 6.8.

Size of the model	TP	FP	FN	Accuracy
132.18 KB	961	27	12	92.1 %

Table 6.2: Results and specifications of the Object Detector.

6.4.2. Hardware architecture

Both sensors researched in our paper, in the main dataset, for the Sensor Fusion were controlled by a Rasberry Pi 3 device. This device was used to ensure the synchronization of the data during the data gathering.

As previously explained, the radar sensor used in the main dataset is a Multiple-input-multiple-output (MIMO) frequency modulated continuous wave (FMCW) radar with four channels developed by Infineon Technologies AG. This radar sensor can detect targets until a distance of 40 meters with a resolution of 1.25 meters because of its configuration, what make suitable for this application. Depending on the manufacturer, the initial parameters of the radar sensor may variate from the one used in this experiment. However, RDM images can be generated with other configurations resulting in the RDM format studied in this paper. In our case, the configuration of the radar sensor used in the main dataset is 24 GHz for the center frequency, 200 MHz for the bandwidth, 64 samples per chirp and 256 chirps per frame.

The camera sensor used in the main dataset is an optical camera for Raspberry Pi model IMX219PQ. This camera sensor can record data at a maximum frame rate of 60 frame/second. However, the frame rate used in this project is 6 frames/sec in order to reduce the memory consumption of the data. Multiple recording sessions in two different scenarios were executed to collect enough data for the training and evaluation of the algorithms.

6.4.3. Deep Learning model

An iterative process has been executed in order to train an object detector for the camera data as explained in Subsection 3.2. This approach has been followed due to the fact that new classes, which were not present in previous object detectors, have been included for this experiment. The pre-trained model used was a SSD structure based on ResNet50 backbone.

This pre-trained model was used to extract 774 labels from 100 camera images. These labels were filtered to correct labels corresponding to new classes, in this case PMD, as well as removing wrong labels. These labels were used to re-train the model in order to fit our specific application achieving an accuracy of 46.72 % in the first iteration (measured as the correct predictions over the wrong predictions and false negative results). The iterative process was executed 10 times adding 100 new camera pictures in each iteration. The information about the last iteration of the re-training phase of the model is shown in Table 6.2.

The DL library used for the training of this model was MXNet because of its focus on low-resource devices. This leads to optimize the DNN during the design phase, leading to a faster inference process.

6.4.4. Evaluation

Once the labels for the camera data are generated with this new DL model, the distance of the targets has been estimated based on some camera calibrations. These calibrations consists on the comparison of manually labeled targets in the RDM as well as the camera frames to determine the relationship between the coordinates in the camera image and the real distance. At the same time, the synchronization of the data was ensure by the comparison of the timestamps of the data.

The Intersection over Union (IOU) technique has been used to compare the predicted bounding box labels of the proposed framework and the ground truth labels. IOU is an evaluation metric commonly used to measure the accuracy of object detector DNNs. This technique can be applied to any system that predicts bounding boxes in scenarios where the ground truth is known. An IOU result above 0.5 is normally considered as a good prediction. The algorithm itself is explained in (6.10) where A means the ground truth bounding box and B the predicted bounding box.

$$IOU = \frac{A \cap B}{A \cup B} \tag{6.10}$$

Using the features previously explained as well as the rule system explained in Subsection 3.3, the data from both sensors was fused to generate labels for the RDM data, as shown in Figure 6.7. The accuracy results obtained using the proposed framework are shown in Table 6.3. The mean average precision (mAP) is also shown in this table. The global system works as an object detector for the RDM images, locating relevant targets as well as classifying them. Therefore, in Table 6.3 only the true positive, false positive and false negative are studied due to the fact that there is no true negative results in this approach.

Figure 6.7: (a) Image extracted from the Object detector using the camera data. (b) Result of the sensor fusion approach proposed in this paper.

CPU and GPU time gain cannot be accurately compared with objective

measures since a CPU and a GPU have a very different hardware architecture. However, these times can be used as an orientation of the time the tool requires for the labeling process. The CPU and GPU used are the E5-2643 v3 and a NVIDIA Tesla P4 GPU respectively. The speed of the tool in these platforms, including all the required preprocessing of the data before the labeling process, has been shown in Table 6.3. These times can also be used to test the possibility of using small Edge AI devices equipped with GPU capability to achieve a good performance in time complexity being compared even with a desktop CPU.

Average speed of process	TP	FP	FN	mAP
3 frames/sec (GPU)	2474	541	2079	82.056 %
0.667 frames/sec (CPU)	2474	541	2079	82.056 %

Table 6.3: Results of the automatic label creation process.

These results can be further analyzed by studying the accuracy achieved by the proposed tool in each of the scenarios of the dataset, as shown in Table 6.4. This table shows how the accuracy in both scenarios is similar but the results in the Singapore Polytechnic Campus are higher. This might be due to the fact this scenario has less objects near the sensors that could difficult the target location as well as provide a more general view of the scenario.

Location	mAP
Singapore Polytechnic Campus	85.260 %
Infineon Singapore Campus	80.730 %

Table 6.4: Results of the automatic label creation process in each of the scenarios.

The second dataset was used to test the developed approach. Consequently, the tool was also evaluated in the 50 frames of the second dataset which was recorded using a different radar sensor from the main dataset. This second dataset´s radar is the Infineon's BGT60TR13C 60 GHz radar sensor (29). The configuration of this sensor was 60.7 GHz for the center frequency, 1 GHz for the studied bandwidth, 128 samples per chirp and 64 chirps per frame. The camera sensor used for the data gathering of this dataset was a 5 MP Raspberry Pi camera, with a viewing angle of 160°. An example of the data contained in this dataset is shown in Figure 6.8.

The final accuracy achieved in this dataset was 87.61 %. This result should be understood as a proof of the suitability of the proposed framework for multiple sensors and scenarios.

In order to compare the time reduction achieved by using this automatic labeling tool, the same dataset has been labeled manually and using the

Figure 6.8: Camera and radar range Doppler map image from the second dataset.

Technique	initial mAP	Average speed
Our approach	82.056 %	3 frames/sec
K. Patel et al. (2)	65.30 %	2 frames/sec
F. Nobis et al. (1)	57.50 %	–
T.Y. Lim et al. (16)	73.5 %	40 frames/sec
T. Winterling et al. (11)	94.93 %	–

Table 6.5: Comparison of results achieves with other techniques.

proposed tool. The required time to label 400 frames manually was 5 hours. On the other hand, when using the proposed autolabeling framework, the required time for the same task was reduced to 6 minutes (using a GPU platform). Therefore, the time reduction achieve in this dataset was 96.76 % respect to the manual labeling process. When using a CPU platform, the time required to use the tool increased, leading to an inferior time reduction of 85.01 % respect to the manual labeling process.

Other emerging techniques for object detection based on radar and camera, such as the technique proposed in (2), were able to achieve higher accuracy than our technique. However, these results are based on the study of multiple frames for the target tracking to enable high accuracy results. If the results are compared only when the system study a single frame, the initial accuracy of (2) is reduced considerably to the point where the accuracy of our technique outperforms it. The time required to process the data has also been improved in our technique in comparison with the K. Patel et al. technique (2).

As shown in Table 6.5, multiple authors do not include the speed of their proposed algorithms in their papers. Therefore, the latency of our system can only be compared with the technique proposed in (2) and (16).

The results obtained in (11), as well as the results from (2) (if we include the tracking system to study multiple frames), achieved a higher accuracy than our proposed framework. However, (11) is able to achieve these results through an iterative approach where human interaction is still required to manually label part of the dataset in each iteration. At the same time, the

technique was not tested with multi-class data, in contrast with our technique which is able to detect multiple targets in the same frame. Due to the fact that our goal is to create a fully automatic process, the human interaction in our technique is minimal but this is a trade off with the final accuracy achieved.

On the other hand, in (2), the high performance results were obtained after applying a tracking technique to correct wrong classification as well as unclassified objects. This tracking system also increases the complexity as well as the memory consumption of this technique. In a first stage, the average accuracy results obtained are 62.95 %, where our technique outperforms achieving an 82.06 % accuracy. It is important to remark the techniques compared with our approach were tested under a different dataset since NuScenes dataset does not provide range Doppler maps, which are required for our framework.

The proposed pipeline by F. Nobis in (1) is based on extracting 2D points from the radar data before the fusing process, what enable its direct compatibility with other sensors such as LiDAR sensors. This generalization of the input data format as well as their Early Fusion approach may be some of the reasons of its low accuracy results (57.50 %) in comparison with out tool (82.056 %) and the rest of algorithms compared. However, it is important to mention this approach may not provide high accuracy as other techniques but enables its implementation in a wider sensor scenario.

The same conclusion can be extracted from the proposed algorithm by T.Y Lim et al. (16). Since this author also designed an Early Fusion technique based on DNNs, the limitations of their technique is the same as (1). However, this author achieves a higher accuracy than (1) because of the different data format enabling a pyramid network structure. Nevertheless, this approach still achieves lower accuracy than the rest of the compared algorithms due to the lack of control over the preprocessing of the data and the evaluation of these preprocessing techniques. On the other hand, due to the fact that this algorithm does not execute a previous deep preprocessing, the system´s speed is higher than other techniques, achieving 40 frames/sec. Therefore it is possible to see this as a trade-off between accuracy and frame rate. Nevertheless, we focus on achieving high accuracy results (8.556 % higher than the T.Y Lim et al. technique (16)) since accurate labels are required, otherwise the usefulness of the framework would be decreased.

It is also important to remark all the previously commented approaches predict bounding boxes for targets in camera data. Therefore the radar data is used as support/reinforcement data. On our framework, the goal is to generate bounding boxes for the RDM images using the camera data to extract relevant features for this task. As a consequence of this, the previous comparisons are used as an orientation of the accuracy in comparison with the state of the art.

6.5. Conclusions

An efficient Sensor Fusion framework to automatically generate labels for range Doppler maps has been described in this paper. An experiment where this framework has been evaluated has been explained as an use-case of this framework for the industry. The tool provided high accuracy results while maintaining low-resource requirement and low latency in this experiment.

The proposed technique is based on multiple state-of-the-art sensor fusion algorithms, extracting the advantages from each of them to further improve the system. Difficulties and solutions to process the required data in our algorithm have been discussed in this paper. We show that the fusion of radar and camera data does not require complex structures to achieve high accuracy results while maintaining low latency. This lends justification to a variety of new sensor fusion algorithms where the algorithms are optimized for radar and camera sensor.

The proposed pipeline approach in our paper is flexible enough to be applied with other sensors and environment conditions. As an example, LiDAR sensors could be used instead of camera sensors to label the data to train the stand-alone radar-based target detection system. This could overcome traditional camera/vision sensor approach regarding adverse environmental conditions (i.e. lighting, reflections, etc.). In future works, we will attempt to fuse data coming from different heterogeneous sensors, such as LiDAR, radar and camera, to overcome the sensor limitations in environment data gathering.

All the experiments to test the proposed labeling tool were executed in multiple computer platforms as previously described in Section 4. However, the final goal of this tool is to be implemented at the network edge. For this reason, the AI model integrated in the proposed pipeline was developed taking into account the restrictions of memory available in Edge devices. Therefore, it would be possible to move its execution to the network edge where it could execute the data gathering and labeling simultaneously if the Edge device where it is implemented has enough resources.

References

[1] F. Nobis, M. Geisslinger, M. Weber, J. Betz, and J. Lienkamp. A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection. *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF), Bonn, Germany*, pages 1–7, 2019.

[2] Kanil Patel, Kilian Rambach, Tristan Visentin, Daniel Rusev, Michael Pfeiffer, and Bin Yang. Deep Learning-based Object Classification on

Automotive Radar Spectra. *2019 IEEE Radar Conference (RadarConf), Boston, MA, USA*, pages 1–6, 2019.

[3] H. Han, J. Kim, J. Park, Y. Lee, H. Jo, Y. Park, E. T. Matson, and S. Park. Object classification on raw radar data using convolutional neural networks. *2019 IEEE Sensors Applications Symposium (SAS), Sophia Antipolis, France*, pages 1–6, 2019.

[4] J. Lombacher, M. Hahn, J. Dickmann, and C. Wöhler. Object classification in radar using ensemble methods. *2017 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), Nagoya*, pages 87–90, 2017.

[5] N. del-Rey-Maestre, M. Jarabo-Amores, D. Mata-Moya, J. Bárcena-Humanes, and P. G. del Hoyo. Machine learning techniques for coherent cfar detection based on statistical modeling of uhf passive ground clutter. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):104–118, 2018.

[6] S. Zhu, J. Xu, H. Guo, Q. Liu, S. Wu, and H. Wang. Indoor Human Activity Recognition Based on Ambient Radar with Signal Processing and Machine Learning. *IEEE International Conference on Communications (ICC), Kansas City, MO*, pages 1–6, 2018.

[7] L. Li, Y. Lv, and F. Wang. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254, 2016.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *[Online] in https://arxiv.org/abs/1312.5602*, 2013.

[9] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-Task Weakly Supervised Learning From Instructional Videos. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[10] V. Havlíček, A.D. Córcoles, and K. et al Temme. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567:209–212, 2019.

[11] T. Winterling, J. Lombacher, M. Hahn, J. Dickmann, and C. Wöhler. Optimizing labelling on radar-based grid maps using active learning. In *2017 18th International Radar Symposium (IRS)*, pages 1–6, 2017.

[12] J. Tang and P. H. Lewis. A study of quality issues for image auto-annotation with the corel dataset. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(3):384–389, 2007.

[13] Maurilio Di Cicco, Ciro Potena, Giorgio Grisetti, and Alberto Pretto. Automatic model based dataset generation for fast and accurate crop and weeds detection. 12 2016.

[14] Markus Suchi, Timothy Patten, David Fischinger, and Markus Vincze. Easylabel: A semi-automatic pixel-wise object annotation tool for creating robotic rgb-d datasets. 05 2019.

[15] Gabriel Meseguer-Brocal, Alice Cohen-Hadria, and Geoffroy Peeters. Dali: a large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm. 06 2019.

[16] Teck-Yian Lim, Amin Ansari, Bence Major, Daniel Fontijne, Michael Hamilton, Radhika Gowaikar, and Sundar Subramanian. Radar and camera early fusion for vehicle detection in advanced driver assistance systems. pages 1–11, 2019.

[17] Z. Ji and D.V. Prokhorove. Radar-vision fusion for object classification. *11th International Conference on Information Fusion*, 2008.

[18] J. Kocic, N. Jovicic, and V. Drndarevic. Sensors and sensor fusion in autonomous vehicles. *in TELFOR 2018. Belgrade: Telecommunications Society and Academic Mind*, pages 420–425, 2018.

[19] X. Zhang, M. Zhou, P. Qiu, Y. Huang, and J. Li. Radar and vision fusion for the real-time obstacle detection and identification. *Industrial Robot: the international journal of robotics research and application*, 46(3):391–395, 2019.

[20] Pinar Duygulu, Kobus Barnard, João Freitas, and David Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. volume 2353, pages 349–354, 03 2002.

[21] H. Rohling. Radar CFAR Thresholding in Clutter and Multiple Target Situations. *IEEE Transactions on Aerospace and Electronic Systems*, AES-19(4):608–621, 1983.

[22] F. C. Robey, D. R. Fuhrmann, E. J. Kelly, and R. Nitzberg. A CFAR adaptive matched filter detector. *IEEE Transactions on Aerospace and Electronic Systems*, 28(1):208–216, 1992.

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. *Lecture Notes in Computer Science*, pages 21–37, 2016.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*, 28:91–99, 2015.

[25] Apache. Mxnet deep learning framework. Accessed: 20 of April 2020.

[26] Google. Tensorflow lite deep learning framework. Accessed: 20 of April 2020.

[27] S. Chandran. Smart antennas for wireless communications (with matlab) (gross, f.; 2005) [reviews and abstracts]. *IEEE Antennas and Propagation Magazine*, 51(3):134–134, 2009.

[28] K. Kim, C. Lee, D. Pae, and M. Lim. Sensor fusion for vehicle tracking with camera and radar sensor. *17th International Conference on Control, Automation and Systems (ICCAS), Jeju*, pages 1075–1077, 2017.

[29] Thomas Stadelmayer, Avik Santra, Robert Weigel, and Fabian Lurz. Parametric convolutional neural network for radar-based human activity classification using raw adc data. *TechRxiv, Sep*, 2, 2020.

Chapter 7

Camera-LiDAR Multi-Level Sensor Fusion for Target Detection at the Network Edge

It is during our darkest moments that we must focus to see the light

Aristoteles

Javier Mendez[1,2], Miguel Molina[1,2], Noel Rodriguez[2], Manuel P. Cuellar[3], Diego P. Morales[2].

1. Infineon Technologies AG, Am Campeon 1-15, 85579 Neubiberg, Germany

2. Department of Electronic and Computer Technology, University of Granada, Avenida de Fuente Nueva s/n, Granada, 18071, Spain

3. Department of Computer science and AI, University of Granada, Avenida de Fuente Nueva s/n, Granada, 18071, Spain

MDPI Sensors

- Received May 2021, Accepted June 2021, Published June 2021
- 10.3390/s21123992
- Impact factor: 3.367
- JCR Rank: 14/64 in category Instruments and instrumentations (Q1)

Abstract: There have been significant advances regarding target detection in the autonomous vehicle context. To develop more robust systems that can overcome weather hazards as well as sensor problems, the sensor fusion approach is taking the lead in this context. Laser Imaging Detection and Ranging (LiDAR) and camera sensors are two of the most used sensors for this task since they can accurately provide important features such as target´s depth and shape. However, most of the current state-of-the-art target detection algorithms for autonomous cars do not take into consideration the hardware limitations of the vehicle such as the reduced computing power in comparison with Cloud servers as well as the reduced latency. In this work, we propose Edge Computing Tensor Processing Unit (TPU) devices as hardware support due to their computing capabilities for machine learning algorithms as well as their reduced power consumption. We developed an accurate and small target detection model for these devices. Our proposed Multi-Level Sensor Fusion model has been optimized for the network edge, specifically for the Google Coral TPU. As a result, high accuracy results are obtained while reducing the memory consumption as well as the latency of the system using the challenging KITTI dataset.

Keywords: Sensor fusion; Deep Learning; Edge Computing; Camera sensor; LiDAR sensor; Target detection.

7.1. Introduction

The interest in autonomous vehicles has increased in recent years due to the advances in multiple engineering fields such as machine learning, robotic systems and sensor fusion (1). The progress of these techniques leads to more robust and trustworthy computer vision algorithms. Using sensors such as Laser Imaging Detection and Ranging (LiDAR), radar, camera or ultrasonic sensors with these techniques enables the system to detect relevant targets in highly dynamic surrounding scenarios. These targets may include pedestrians, cyclists, cars or motorbikes among others, as discussed in public autonomous car datasets (2; 3).

Computer vision algorithms such as You Only Look Once (YOLO) (4), Region based Convolutional Neural Network (R-CNN) (5), Fast R-CNN (6) or Single Shot Detector (SSD) (7) have been designed upon the previously mentioned sensors leading to numerous models for target detection. These models calculate the score of the bounding box location for each detected target as well as its classification based on high-level features generated by using Convolutional Neural Networks (CNN). Nevertheless, these models are not robust since the use of only one sensor may result in target detection problems in hazard situations. There are machine learning (ML) attacks that can affect single sensor models such as models only based on camera

data. These attacks slightly modify target data in a scene to get a different classification result (8; 9; 10). In addition, sensor data information can be degraded due to weather conditions (11; 12; 13). Other sensors can also be attacked with similar results, as is the case of the LiDAR (14; 15) or radar (16; 17).

One viable option to improve the reliability of these systems as well as to improve the accuracy of the results is the sensor fusion. This technique implements a system with multiple data sources to complement the inputs. This approach results in a more complete knowledge of the scenario for a better computer vision. The sensor fusion approach can be divided into multiple techniques: *Early Fusion*, *Late Fusion* and *Intermediate Fusion* (18). In the *Early Fusion* the raw data or low-level preprocessed data is combined to generate a more complete raw data while in *Late Fusion* high-level features such as target location are merged for a better final result. *Intermediate Fusion* can be understood as a combination of both previous techniques in which the data is merged at multiple levels to effectively find the merged representation of multiple input data (19). Recently, some authors have proposed computer vision models based on these approaches (19; 11; 20).

Most of these studies focus on improving the detection algorithms without taking into account the constraints imposed by the autonomous vehicle industry of latency and privacy to ensure the safety of the passengers (1). At the same time, the limitations of the final system in which the models should be integrated must be studied to ensure the suitability of the models for the system regarding memory and computing power requirements due to the specifications of the processing units deployed at the network edge (21; 22). Because of this, in our paper we focused on researching a sensor fusion algorithm that can be deployed in edge devices. These devices are meant to work without a high frequency communication with external devices to process the data at the network edge. One of the advantages of this is the increase of the security of the raw data since it is not broadcast to an external device. However, they have constraints regarding memory in the device and computing power due to their size and energy consumption.

Our proposed algorithm is based on the fusion of LiDAR and camera data. LiDAR data provides reliable information of depth and target presence which can complement the high-quality camera image data of the surroundings for the target classification. Consequently, the LiDAR provides further information about the relevant areas of the camera image. The combination of these sensors can be used to ensure the presence of the targets and to provide a solution for scenarios where one of the sensors does not provide data. Our algorithm, called Multi-Level Sensor Fusion (MLSF), executes the data fusion at multiple levels using the *Intermediate Fusion* as a bidirectional reinforcement approach for both input data. A new layer structure, *fusion layer*, has been integrated in the proposed deep learning model. This

layer generates a shared feature map which is later used as a mask for the feature map of the LiDAR and camera to further extract relevant features before the final target localization and classification. Because of this, the target detection results significantly improve. The latency and memory consumption have been used as constraints during the design of the model to ensure its suitability for the network edge, specifically the Google Coral TPU edge device.

This model has been evaluated using the challenging KITTI dataset (2), where it achieves a final latency of 0.057 seconds and accuracy of 90.92 %, 87.81 % and 79.63 % for easy, medium and hard to detect targets.

The manuscript is organized as follows: Section 7.2 focuses on related works to the addressed problem. In Section 7.3, the proposed Deep Fusion Network is described for a deep understanding of its structure as well as the used input data. After that, Section 7.4 will focus on the experiment where this algorithm has been applied, and finally Section 7.5 is the conclusion.

7.2. Related Works

Since autonomous vehicles are a current trend, numerous authors are researching the sensor fusion applied to this topic in order to improve the state-of-the-art accuracy results. As previously mentioned, some of them research single-sensor scenarios for target detection in the autonomous vehicle paradigm such as Deep Manta (23). This model uses only the data from a monocular camera to generate 2D and 3D bounding boxes as well as classification for targets. At the same time, this model provides information regarding the visibility of each of the targets´ parts, making it suitable for annotation tasks. However, due to its numerous algorithms to extract the location information of each part of the target, its visibility and classification, this model has high execution times in comparison with other techniques such as our proposed model.

Nevertheless, numerous authors are starting to research the use of LiDAR sensors for mapping and detection due to the accuracy of this sensor to generate point clouds based on the surfaces in the environment. This sensor has been applied to other research topics apart from autonomous vehicles; it is used in the agricultural industry to generate accurate maps. M.P. Christiansen et al. (24) developed a UAV Mapping System for Agricultural Field Surveying where LiDAR data was gathered using an Unmanned Aerial Vehicle (UAV) device. Global Navigation Satellite System (GNSS) and Inertial Measurement Unit (IMU) sensors were also integrated in the device in order for the point cloud reconstruction based on multiple frames to be later used for tasks such as estimation of the soil surface and total plant volume. Following the same research line, A. Patil et al. (25) proposed a framework to align LiDAR and video data using a point cloud registration algorithm. By

using this framework in five different experiments, they proved how it can help to reduce the time to complete retrofitting tasks by 20 % on average.

LiDAR sensors have also been used for other topics such as autonomous cars. One of the authors who researched this topic is J.Zarzar. This author proposed a target detection algorithm, PointRGCN, based on a single LiDAR sensor (26). This algorithm is based on Graph Convolutional Networks integrated in a multiple 3D object detection pipeline. By doing so, the bounding boxes can be refined multiple times to achieve state-of-the-art accuracy results. Nevertheless, even if the model provides high accuracy results, it faces the same problems previously stated with the Deep Manta model regarding adversarial attacks or lack of information.

Similar to our proposed algorithm, other researchers are studying the implementation of a sensor fusion technique for target detection. One of the proposed algorithms following this research line is the Camera-LiDAR Object Candidates Fusion (CLOCs) Deep Neural Network (DNN) architecture proposed by S. Pang et al. (27). This network combines LiDAR and camera data to locate and classify targets in 2D and 3D. Depending on the desired target detection, 2D or 3D, the system uses different perspectives obtained from the raw LiDAR data. As a result, this technique provides high accuracy location and classification of the targets in 2D and 3D. However, due to the complexity of its detectors, its memory consumption may not be suitable for the current edge devices.

[-15]Following the same research line for target detection using sensor fusion, J. Kim et al. (19) proposed a DNN to combine LiDAR depth maps and camera images. Its approach is based on extracting relevant features from both sensor data independently using the VGG-16 structure (28) before fusing the output feature maps at multiple levels. This approach is similar to our proposed network; however, we included the option of ignoring the result of the previous fusion layers to avoid including not highly relevant data in the final data fusion step. This leads to a more efficient approach where only relevant information is further studied. At the same time, this model presents the same problem regarding the network edge as the CLOCS Deep Neural Network. Due to the detector implemented as well as the structure of the layer used for the data fusion, the latency and memory consumption are larger than that obtained with our model.

LiDAR and camera are not the only sensors researched for target detection in the vehicle context. Other authors research techniques based on different devices such as camera, radar or ultrasonic sensors due to the high cost of the LiDAR sensors. Because of this, F. Nobis et al. (11) researched the fusion of radar and camera data for this task. In this pipeline, the radar data is preprocessed to generate 2D coordinates in the horizontal plane which could belong to possible targets. Their approach was based on DNN where the data is fused on multiple levels. During the training phase of the

model, the weight configuration of the DNN establishes at what level the fusion is more effective to obtain the desired output. The accuracy achieved with this pipeline in the NuScenes (3) dataset is 55.99 %, requiring 56 ms to study each frame.

Therefore, it can be observed how multiple approaches for the target detection are being researched and provide high accuracy results in some of the most popular datasets such as KITTI or NuScenes. In the case of single-sensor models, the problem of adversarial attacks or problems during the data acquisition are not solved, as (11) explains. This is one of the most relevant reasons to use a sensor fusion approach as other previously mentioned authors have done (11; 19; 27). However, none of these models take into account the constraints of the autonomous driving industry regarding latency and memory consumption. Because of this, our research faces the problem of the target detection from the Edge Computing perspective. The model integrates detectors that have been proven to provide state-of-the-art results at the network edge while also optimizing the model at layer and network level. Therefore, our proposed model considers the constraints of the autonomous vehicles paradigm while maintaining state-of-the-art accuracy results. A deeper comparison of the mentioned techniques, as our model, is shown in Section 7.4.

7.3. Proposed Multi-Level Sensor Fusion Network

The proposed Multi-Level Sensor Fusion (MLSF) Network aims to detect targets in the camera data by integrating LiDAR data as reinforcement data. Camera and LiDAR data are fused through this network at multiple levels, enabling the system to merge the features at the specific level or levels decided during the training phase of the model. The SSD (7) structure has been used as a reference for the proposed network due to its reduced memory consumption while maintaining high-performance accuracy results.

In order to optimize the process as well as the memory consumption, the LiDAR data preprocessing has been studied to reduce its dimensions while maintaining most of its relevant features for the object detection before it is fed into our proposed model.

7.3.1. LiDAR Depth Map Representation

A Laser Imaging Detection and Ranging (LiDAR) sensor has been used in this project to gather information regarding the environment. Differently from the camera, the LiDAR sensor transmits laser pulses and measures the time it takes until a reflection is received. Based on this, it calculates the distance of the target and its 3D coordinates since the angles used to send the laser pulse and the distance are known. Consequently, this sensor provides

information regarding surfaces rather than only images as the camera does.

The order of these 3D points, *(x, y, z)*, in datasets such as KITTI (2) and NuScenes (3) public datasets cannot be ensured since it depends on the sensor used to gather data as well as the environment. Therefore, it is usually assumed that the order of the points is unknown, unlike pixel arrays in images. In consequence, their integration in DNNs is not straightforward since the network must be invariant to these permutations of the input data in the feeding order.

To overcome this challenge, the voxel grid approach is applied by numerous authors by using 3D-CNN (29; 20). However, this technique increases the complexity of the DNN, leading to larger models, and uses layer structures that are not supported by some edge devices such as the 3D convolutional layer. Therefore, rather than this technique, depth maps from LiDAR data have been generated as shown in Figure 8.1. This technique ensures the low memory consumption and latency of the model in comparison with the voxel approach.

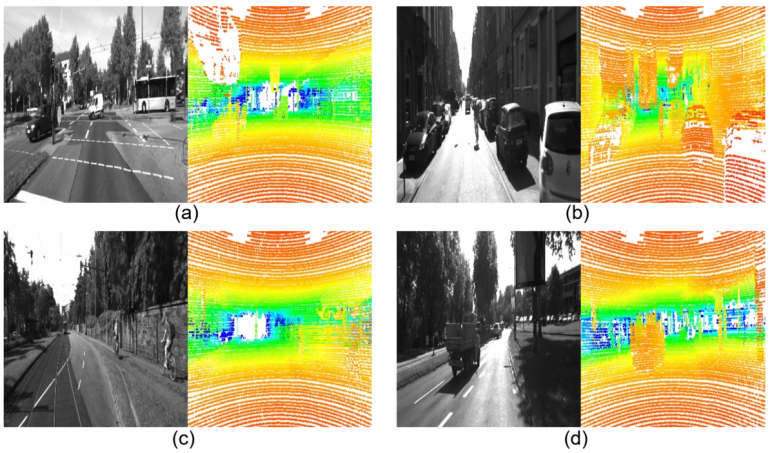

Figure 7.1: Camera images (left image in (**a**–**d**)) and LiDAR depth maps generated from LiDAR raw data (right image in (**a**–**d**)).

Depth maps representations reduce the dimensions of the LiDAR data to generate 2D images. The original 3D points are codified to generate these images following (1)–(4) procedures for each point. These equations transform the 3D points from Cartesian coordinates to the new representation system.

$$x_{DepthMap} = tan^{-1}(y_{3D}/x_{3D}) \tag{7.1}$$

$$d_{3D} = \sqrt{x_{3D}^2 + y_{3D}^2 + z_{3D}^2} \tag{7.2}$$

$$y_{DepthMap} = cos^{-1}(z_{3D}/d_{3D}) \tag{7.3}$$

$$color_{DepthMap} = d_{3D} \tag{7.4}$$

These new coordinates can be directly used for the representation of the depth maps by adding the color component. Since the new data type is a 2D image, the SSD structure can be implemented for the feature extraction from LiDAR depth maps. This preprocessing technique is applied to the raw LiDAR data before being fed into our proposed MLSF model.

This preprocessing of the LiDAR data enables a memory consumption reduction of 95.6 % to store the data when the depth maps are saved as 300 × 300 pixel images in comparison with the raw 3D point cloud.

7.3.2. Overall System Description

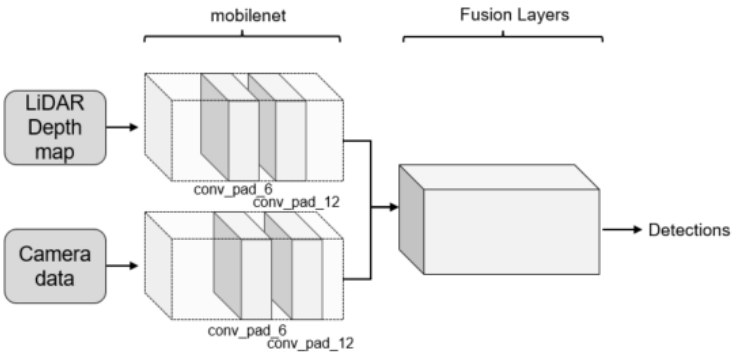

Figure 7.2: Proposed Multi-Level Sensor Fusion network structure for target detection.

The structure of the proposed Multi-Level Sensor Fusion network is shown in Figure 8.3. The camera images as well as the LiDAR depth maps are fed separately to our MLSF model. The two input data are studied using two separated CNNs following the SSD structure to generate the feature maps from each data type. These CNNs use the mobilenet network backbone. As a result of this, the initial number of layers is smaller than other structures such as VGGNet-16 used by other authors. The fusion of the feature maps is executed by our *fusion layers* shown in Figure 7.3.

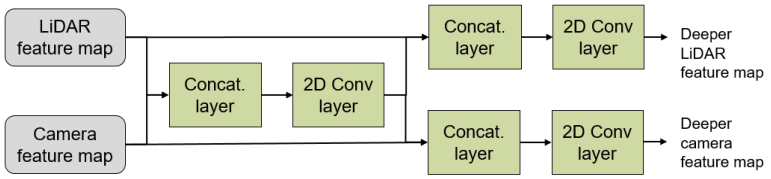

Figure 7.3: Proposed fusion layer.

The *fusion layers* combine the feature maps from both CNNs by concatenating them (in the channel axis) before applying a 2D convolutional filter (3×3) with ReLU activation function. After this, the new feature map is concatenated individually to each of the previous initial feature maps from the LiDAR and camera data to generate deeper feature maps. To maintain the shape of the initial feature map as well as to further process the data, another 2D convolutional filter (1×1) with ReLU activation function is applied to each of the new deeper feature maps. These feature maps can be used in a later step as input for another fusion layer, enabling the system to execute a multi-level fusion of the data. Because of this structure, the network learns during the training phase at what level it must execute the fusion of each feature extracted from the LiDAR and camera sensors.

In our network, four of these *fusion layers* have been implemented in order to extract the information of the localization of the targets and another four independent *fusion layers* for the classification. The initial feature maps for the classification are extracted from the *conv-pad-6* layer of the mobilenet networks that process the camera and LiDAR data. For the localization, the initial feature maps are extracted from the *conv-pad-12* layer in each network.

Since our approach is based on the SSD structure, the loss function implemented in the system is ruled by the SSD loss presented in (5)–(14).

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in (cx,cy,w,h)} x_{ij}^{k} \cdot S_{L1} \tag{7.5}$$

$$S_{L1} = smoothth_{L1}(l_i^m - \hat{g}_j^m) \tag{7.6}$$

$$smooth_{L1}(l_i^m - \hat{g}_j^m) \tag{7.7}$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \tag{7.8}$$

$$\hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \tag{7.9}$$

$$\hat{g}_j^w = log(\frac{g_j^w}{d_i^w}) \tag{7.10}$$

$$\hat{g}_j^h = log(\frac{g_j^h}{d_i^h}) \tag{7.11}$$

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p - \sum_{i \in Neg} log(\hat{c}_i^0) \tag{7.12}$$

$$\hat{c}_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)} \tag{7.13}$$

$$L(x, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{7.14}$$

where N indicates the number of matching default boxes, l represents the predicted boxes, g the ground truth boxes, x the coordinates of the bounding boxes and c the class confidences. These parameters also include the offset for the center points (cx, cy), the width of the box (w) and its height (h).

This DNN has later been optimized for the network edge by pruning the layers. This process removes the connections when the parameters of a layer-s/neurons do not highly modify the value of the input signal, consequently reducing the size of the model and the number of operations. At the same time, due to the constraints of the Google Coral TPU Edge Device where the model should be integrated, the model parameters have been quantized to 8-bit integer values. After the optimization process, the final model requires 56.4 MB in contrast to the initial model with a size of 185 MB.

Following these techniques, the full pipeline from the data acquisition to target detection is shown in Figure 8.2. In the first moment, after gathering the data from the LiDAR and camera sensors, the data need to be aligned to ensure that the coordinate origin is shared by both sensors. This is also used to filter out parts of the scene visualized only by the LiDAR that are not relevant for the target detection. After this step, the LiDAR raw data is used to generate the LiDAR depth maps previously explained. The input data also needs to be quantized to 8-bit integers before it is fed into the model due to the constraints imposed by the Google Coral TPU Development Board. Finally, our proposed MLSF model studies the input data to provide information about the targets in the current frame.

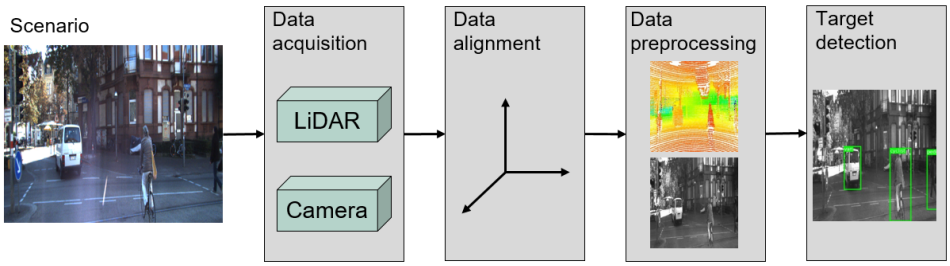

Figure 7.4: Full pipeline for target detection using our proposed model.

7.4. Experiment

In this section, our proposed MLSF model is evaluated using the KITTI dataset (2) to compare its accuracy results for the 2D target detection with state-of-the-art techniques. At the same time, latency and memory size are also included in the comparison since the goal is to design a target detection model for the network edge. Furthermore, the influence of the lighting conditions will also be discussed in this section.

Since targets could be located in both sensor data, the coordinatesórigin has been set to the ego vehicle for an easier final evaluation of the model results, as depicted in Figure 7.5.

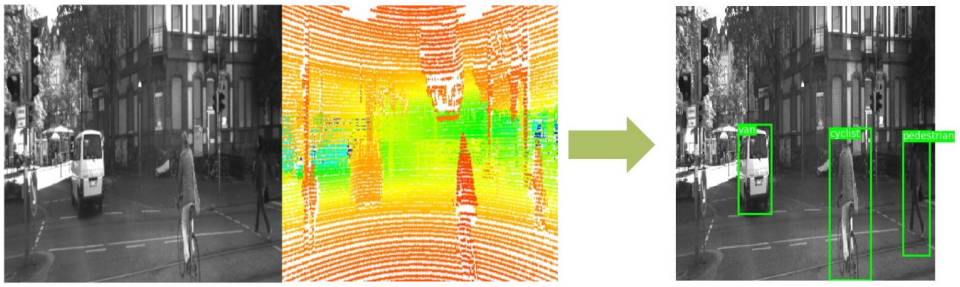

Figure 7.5: Input data on the left side of the figure and output on the right side.

7.4.1. Dataset

The dataset used for this experiment is the KITTI dataset (2) since it is one of the most popular databases when LiDAR and camera data is required. Due to this, numerous state-of-the-art results provide accuracy information of their algorithms with this dataset. The LiDAR sensor used in this dataset is the Velodyne HDL-64E (30) developed by Velodyne in San Jose, United States.

This dataset consists of 7481 training samples and 7518 testing samples. Both subsets include camera and LiDAR data. The labels in this dataset are provided using the ego vehicle coordinate system that can be used for all the sensors integrated in the KITTI dataset. Since our goal is the 2D target detection in the camera images, we have converted the LiDAR and camera data to this ego vehicle coordinate system in order to match the labels with the used data. The labeled targets in this dataset are: car, pedestrian, bicycle, tram, van, truck, misc and sitting down person. These targets'labels also include information about the difficulty to detect the target (easy, medium and hard). Therefore, we will also provide the accuracy result achieved on each of these subsets based on the difficulty of the targets.

Since labels are required to measure the final accuracy of the model, the training dataset has been split into training and evaluation data using 30 % of the dataset for the evaluation.

7.4.2. Hardware Architecture

As the goal of this paper is to present a sensor fusion pipeline for target detection at the network edge, the latency results were calculated in an edge device. The specific device is the Google Coral TPU Dev Board from Google, manufactured in China, that is shown in Figure 8.5. This platform is based on the integration of a TPU coprocessor to execute the tensor computations in a more efficient way than using traditional CPUs or GPUs. However, this device still integrates an Integrated GC7000 Lite Graphics GPU and an NXP i.MX 8M SoC CPU to execute nontensor operations. As a result of this, it is capable of performing 4 trillion operations per second (TOPS) while only consuming 0.5 W/TOPS.

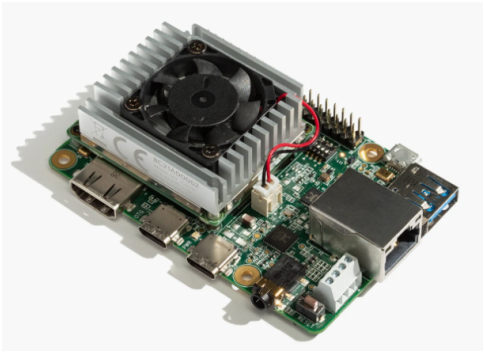

Figure 7.6: Google Coral TPU Development Board image from `https://coral.ai/` (accessed on 31-05-2021) (31).

Simultaneously, the software framework for the implementation of DNNs in the device enables the optimization of them. As a result, the latency, as well as the size of the model, can be reduced in most cases.

In order to provide an accurate comparison of the latency, the compared algorithms have been implemented in the same platform when possible. In other cases, the results of the algorithms have been extracted from the original papers and KITTI dataset results table (2).

7.4.3. Experimental Settings

The parameters used during the training of the model can be observed in Table 8.1. The loss function used for the training of the model is the same as that used in the SSD structure, which is explained in Section 8.2.2.

Parameter	Setting
Epochs	500
Batch size	6
Optimizer	SGD
Initial learning rate	$1{\cdot}e^{-3}$
Final learning rate	$1{\cdot}e^{-5}$
Momentum	0.9

Table 7.1: Parameters for the model training.

The number of labels for each of the classes of the KITTI dataset is not the same, having large variation between common classes such as car and uncommon classes such as tram. Therefore, the algorithm for the model training tries to use the same number of labels from each of the studied classes to overcome this class imbalance problem while still using all the training data.

The Intersection Over Union (IOU) algorithm has been implemented to measure the accuracy of the model. This technique can be applied to any system that predicts bounding boxes in scenarios where the ground truth is known. An IOU result above 0.5, as used in this experiment for Pedestrian and Cyclist classes, is normally considered as a good prediction. For cars, the IOU threshold selected is 0.7 as generally done when using the KITTI dataset. The algorithm itself is explained in (15), where A means the ground truth bounding box and B the predicted bounding box.

$$IOU = \frac{A \cap B}{A \cup B} \tag{7.15}$$

As well as the configuration of the experiment, it is also important to comment on the algorithms that have been selected for the comparison with our model. Due to the fact that the main goal of our research is the target detection at the network edge using sensor fusion, we have selected some of the most relevant target detection algorithms which have been tested using the KIITI dataset. Among those algorithms, we provide a comparison with the models based on LiDAR and camera sensor fusion. As a result, a comparison of models which share the dataset as well as the application goal is shown in the next section. These models are the Deep Gated Information Fusion Network (DGFN) (19), CLOCs (27), Multi-view 3d object detection network (MV3D) (32) and Multi-Scale Convolutional Neural Network (MS-CNN) (33) since they are the most relevant in this topic to the best of our knowledge.

At the same time, a comparison with single sensor models is also discussed in the next section. The goal of this comparison is the discussion of the advantages of the sensor fusion applied for the target detection rather

than using a single sensor. The models used for this comparison are the SSD (studied separately to detect targets using LiDAR and camera data) (7), the Deep Manta (23), the Structure Aware SSD (SA-SSD) (34) and the Mono-Pair (35) algorithms. These models have been chosen because SSD is one of the most efficient model structures for the network edge. The other models (based on camera or LiDAR) are some of the state-of-the-art target detection models based on a single sensor which achieves high accuracy results. Therefore, these models can provide further information regarding the state of our proposed model in comparison with the current state-of-the-art target detection models.

Finally, the influence of the lighting condition in the target detection task is discussed to prove the relevance of the sensor fusion. For this task, a reduced synthetic dataset has been created to generate night frames (500 frames) based on the original KITTI camera frames.

7.5. Results

In this section, a comparison of our proposed DNN and the state-of-the-art models for target detection will be presented for a deeper understanding of the advantages of our technique. The parameters that are compared in Table 8.3 are the size of the models in the second column, accuracy (for easy, medium and hard to detect targets according to the KITTI dataset) in the third column and latency in the fourth column. These parameters have been selected since they are the most relevant ones when executing target detection at the network edge taking into account the memory constraints of edge devices and latency requirements for autonomous vehicles.

Model	Size of the model	Accuracy (%) Easy - Medium - Hard	Latency (s)
MLSF (proposal)	56.4 MB	90.92 - 87.81 - 79.63	0.057
DGFN (19)	713 MB	98.69 - 90.31 - 82.16	0.73
CLOCs (27)	–	88.94 - 80.67 - 77.15	0.1
MV3D (32)	–	95.01 - 87.59 - 79.90	0.36
MS-CNN (33)	–	93.98 - 89.92 - 79.69	0.5

Table 7.2: Comparison of LiDAR-Camera fusion networks for target detection.

Table 8.3 shows how the achieved general accuracy results when using our proposed Deep Fusion Network model for easy, medium and hard to detect targets (shown in the third column of Table 8.3 respectively for all the classes in the studied dataset) are lower than the rest of the studied models. In this comparison, the DGFN algorithm achieves the highest accuracy results

for easy, medium and hard to detect targets. Nevertheless, the targets used during the training and testing of the DGFN and CLOCs algorithms were reduced subsets of the KITTI dataset. These subsets only include some of the classes instead of the whole list of targets. Consequently, this comparison must be understood as an estimation of the accuracy.

On the other hand, the model size of the proposed model is considerably smaller than the rest of the compared models. For example, it is 92.1 % smaller than the DGFN model. The rest of the compared models do not provide information about their memory size so a direct comparison is not possible. Therefore, the model size can only be compared with the DFGN model. Nevertheless, the MV3D model (32), which preprocesses 3 inputs individually, as well as the CLOCs (27), which executes simultaneously a 3D and 2D detection, and MS-CNN (33), which studies independently each subset in the LiDAR point cloud of each target, can be assumed to require a larger memory due to their high complexity in comparison to our proposed MLSF model.

The memory reduction has been achieved due to the optimization of the model for the network edge by applying multiple techniques such as quantization (36) and pruning (37). These techniques are further explained in Section 8.2.2. At the same time, the mobilenet detector integrated in the model, as well as the pruning of nonrelevant layers and parameters, leads to a faster execution as the previous table also shows.

As a conclusion from this table, it is possible to observe the tradeoff between the accuracy and memory/latency. Therefore, our algorithm has been designed following the network edge constraints in order to achieve a reduced latency and memory consumption. However, the accuracy results are 7.77 %, 2.50 % and 2.53 % (easy, medium and hard to detect targets, respectively) smaller than the DGFN model. This leads to the conclusion that this model should be implemented in a collaborative approach with other networks to ensure high reliability for applications such as autonomous driving.

From this point on, now our proposed model will be compared with other algorithms for target detection which are not based on sensor fusion. In this case, since the goal of the comparison is to discuss the improvement of the robustness as well as the general accuracy of the models for all the studied classes, the comparison will be based on the used input data, the latency and the accuracy achieved by the model, as shown in Table 7.3.

It is possible to observe in Table 7.3 how our proposed model outperforms the SSD structure using a single data input. This can result from the lack of information when using a single sensor as previously discussed, which leads to a lack of robustness during the target detection. Overcoming this problem is one of the main reasons to apply sensor fusion techniques, as explained in (19). In real scenarios such as the context of autonomous vehicles, the

Model	Data	Latency (s)	Accuracy (%) Easy - Medium - Hard
MLSF (proposal)	LiDAR-Camera	0.057	90.92 - 87.81 - 79.63
SSD (7)	Camera	0.003	87.14 - 84.37 - 75.74
SSD (7)	LiDAR	0.003	85.17 - 71.52 - 67.36
Deep Manta (23)	Camera	0.7	97.58 - 90.89 - 82.72
SA-SSD (34)	LiDAR	0.04	95.03 - 91.03 - 85.96
MonoPair (35)	Camera	0.06	96.61 - 93.55 - 83.55

Table 7.3: Comparison of of our model with no-sensor fusion algorithms.

weather conditions can affect the data acquisition of each individual sensor (i.e., water on the camera lens after the rain, LiDAR laser absorption in fog conditions, etc.). When using a sensor fusion approach, this problem can be mitigated due to the bidirectional reinforcement of the data.

On the other hand, the Deep Manta model (23) outperforms our model as well as the SA-SSD (34) and MonoPair (35) when it comes to target detection accuracy. Regarding the latency, except the SSD structure, the rest of the studied models have a similar execution time to our model even when they study the data from a single sensor. Therefore, even when they achieve similar results of latency and accuracy, the robustness of the single sensor models is reduced in comparison with our MLSF model, following the criteria of (19).

Finally, the suitability of these models for edge devices must also be considered, since most of these models integrate complex layers such as 3D Convolutions which are not supported in edge devices like the Google Coral TPU. On the other hand, our MLSF has been designed following the layer restrictions of this device to ensure its correct function.

Figure 7.7: Synthetic night frames (**top** row) and original images (**bottom** row).

The influence of the lighting conditions has been researched using some of

the frames from the KITTI dataset and reducing their luminosity to generate synthetic night frames as shown in Figure 7.7. Due to the darkness of these images, the accuracy of the target detection in these frames when using a SSD mobilenet trained with the original KITTI dataset dropped to 73.81 %–65.41 %–39.63 % (easy, medium and hard to detect targets) when using only the camera data. One possible reason is that in bad light conditions, the camera sensor may not be able to gather information about all the targets in the scenario. This problem can be observed in the top images of Figure 7.7 where it is hard to see the targets in comparison with the bottom images. However, when using our proposed sensor fusion algorithm, the model can detect targets that have not been located when using only the camera, as shown in Figure 7.8. LiDAR data is not affected by the poor light conditions due to the fact that it is based on measuring the time taken to receive the reflection of a transmitted laser pulse rather than measuring the external light reflected by the targets. The accuracy achieved with our model in these dark frames was 83.18 %–80.02 %–47.83 %. Consequently, it is possible to observe how the LiDAR data reinforce the camera data by generating a map of relevant areas of the image, leading to a higher accuracy.

Figure 7.8: Target detection in night frame with (**a**) sensor fusion algorithm and (**b**) only camera.

After these comparisons, it can be observed how our proposed model achieves high accuracy results for the researched task but it does not improve the state of the art. However, when taking into account the latency and memory constraints, our model achieves edge capabilities that are not present in the rest of the compared models while improving its robustness in comparison with single sensor models.

7.6. Conclusions

A Multi-Level Sensor Fusion deep neural network has been developed and tested in the Google Coral TPU Edge Device in this paper for target detection using camera and LiDAR sensors. The sensor fusion layers integrated in this model generate feature maps that are combined at multiple levels to produce a joint data representation that has been tested on the KITTI dataset.

The accuracy results do not surpass the state-of-the-art accuracy shown by other sensor fusion models such as DGFN (19). Nevertheless, our model still achieves a 90.92 %, 87.81 % and 79.63 % accuracy results for easy, medium and hard to detect targets in the challenging KITTI dataset while requiring a 92.1 % less memory than the DGFN model (19). As a result of the optimization applied to the proposed model, the latency of the model has also been reduced to 0.057 seconds, outperforming the rest of compared algorithms.

At the same time, the advantages of using a sensor fusion approach rather than a single sensor model have been studied by comparing our proposed algorithm with other single sensor target detection algorithms. This comparison is shown in Table 7.3 where it is possible to observe how some of the single sensor models achieve better accuracy than our proposed algorithm. However, we studied the effect of using only camera data for the target detection task and we showed how the accuracy drops to 73.81 %–65.41 %–39.63 % (easy, medium and hard to detect targets) in cases where the light is not good. Since the LiDAR data does not depend on the light of the scenario, using a sensor fusion approach helps to achieve an accuracy of 83.18 %–80.02 %–47.83 %. This proves that even if high accuracy can be achieved by a single sensor model, these algorithms are not robust when facing changes in the environment in comparison with a sensor fusion algorithm.

We can conclude that the proposed model has been designed in order to fit in edge devices as well as time constraints applications such as autonomous driving, taking into account a trade-off among the accuracy, latency and memory size.

References

[1] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Meireles Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, page 113816, 2020.

[2] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You

only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[6] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[7] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. *Lecture Notes in Computer Science*, pages 21–37, 2016.

[8] Nir Morgulis, Alexander Kreines, Shachar Mendelowitz, and Yuval Weisglass. Fooling a real car with adversarial traffic signs. *arXiv preprint arXiv:1907.00374*, 2019.

[9] Yujie Li, Xing Xu, Jinhui Xiao, Siyuan Li, and Heng Tao Shen. Adaptive square attack: Fooling autonomous cars with adversarial traffic signs. *IEEE Internet of Things Journal*, 2020.

[10] Chawin Sitawarin, Arjun Nitin Bhagoji, Arsalan Mosenia, Mung Chiang, and Prateek Mittal. Darts: Deceiving autonomous cars with toxic signs. *arXiv preprint arXiv:1802.06430*, 2018.

[11] F. Nobis, M. Geisslinger, M. Weber, J. Betz, and J. Lienkamp. A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection. *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF), Bonn, Germany*, pages 1–7, 2019.

[12] Andreas Pfeuffer and Klaus Dietmayer. Optimal sensor data fusion architecture for object detection in adverse weather conditions. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2018.

[13] Andreas Pfeuffer and Klaus Dietmayer. Robust semantic segmentation in adverse weather conditions by means of sensor data fusion. In *2019 22th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2019.

[14] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2267–2281, 2019.

[15] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z Morley Mao. Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 877–894, 2020.

[16] Teng Huang, Yongfeng Chen, Bingjian Yao, Bifen Yang, Xianmin Wang, and Ya Li. Adversarial attacks on deep-learning-based radar range profile target recognition. *Information Sciences*, 531:159–176, 2020.

[17] Jingtao Li, Adnan Siraj Rakin, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. Radar: Run-time adversarial weight attack detection and accuracy recovery. *arXiv preprint arXiv:2101.08254*, 2021.

[18] Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402, 2005.

[19] J. Kim, J. Choi, Y. Kim, J. Koh, C. C. Chung, and J. W. Choi. Robust camera lidar sensor fusion via deep gated information fusion network. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1620–1625, 2018.

[20] Zhizhong Kang, Juntao Yang, Ruofei Zhong, Yongxing Wu, Zhenwei Shi, and Roderik Lindenbergh. Voxel-based extraction and classification of 3-d pole-like objects from mobile lidar point cloud data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11:4287–4298, 11 2018.

[21] Dianlei Xu, Tong Li, Yong Li, Xiang Su, Sasu Tarkoma, and Pan Hui. A survey on edge intelligence. *arXiv preprint arXiv:2003.12172*, 2020.

[22] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

[23] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teuliere, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2040–2049, 2017.

[24] Martin Peter Christiansen, Morten Stigaard Laursen, Rasmus Nyholm Jørgensen, Søren Skovsen, and René Gislum. Designing and testing a uav mapping system for agricultural field surveying. *Sensors*, 17(12):2703, 2017.

[25] Ashok Kumar Patil, Young Ho Chai, et al. On-site 4-in-1 alignment: Visualization and interactive cad model retrofitting using uav, lidar's point cloud data, and video. *Sensors*, 19(18):3908, 2019.

[26] Jesus Zarzar, Silvio Giancola, and Bernard Ghanem. Pointrgcn: Graph convolution networks for 3d vehicles detection refinement. *arXiv preprint arXiv:1911.12236*, 2019.

[27] S. Pang, D. Morris, and H Radha. CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection. *arXiv preprint arXiv:2009.00784.*, pages 1–8, 2020.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] Zhongyang Zhao, Yinglei Cheng, Xiaosong Shi, and Xianxiang Qin. Classification method of lidar point cloud based on threedimensional convolutional neural network. *Journal of Physics: Conference Series*, 1168:062013, 02 2019.

[30] Velodyne. Velodyne hdl-64e sensor. `https://velodynelidar.com/products/hdl-64e/`. Accessed: 2021-05-31.

[31] Google. Google coral tpu dev board. `https://coral.ai/products/dev-board/`. Accessed: 2021-05-31.

[32] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[33] Xinxin Du, Marcelo H Ang, Sertac Karaman, and Daniela Rus. A general pipeline for 3d detection of vehicles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3194–3200. IEEE, 2018.

[34] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11873–11882, 2020.

[35] Yongjian Chen, Lei Tai, Kai Sun, and Mingyang Li. Monopair: Monocular 3d object detection using pairwise spatial relationships. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12093–12102, 2020.

[36] Taylor Simons and Dah-Jye Lee. A review of binarized neural networks. *Electronics*, 8(6):661, 2019.

[37] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Advances in neural information processing systems*, pages 2181–2191, 2017.

Chapter 8

Lidar-Radar Robust Multi-Level Sensor Fusion for Target Detection at the Network Edge

Javier Mendez[1,3], Manuel P. Cuellar[2], Diego P. Morales[3].

1. Infineon Technologies AG, Am Campeon 1-15, 85579 Neubiberg, Germany

2. Department of Electronic and Computer Technology, University of Granada, Avenida de Fuente Nueva s/n, Granada, 18071, Spain

3. Department of Computer science and AI, University of Granada, Avenida de Fuente Nueva s/n, Granada, 18071, Spain

Elsevier Measurement

- Received December 2021, Under review
- Impact factor: 3.927
- JCR Rank: 19/90 in category Engineering and Multidisciplinary (Q1)

Abstract: Perception subsystems of autonomous vehicles must be robust against weather hazards as well as sensor problems. Therefore, sensor fusion approaches are taking the lead in this context. Laser Imaging Detection and Ranging (LiDAR) and radar sensors are two of the most popular sensors for this task due to the complementary data that can be extracted from these devices. These features, in contrast with camera sensors, do not incur privacy concerns since only position and shape information is gathered. In this work, a multi-level sensor fusion approach based on LiDAR and radar sensors is proposed to locate and classify targets at the Network Edge, since resources in these vehicles are highly reduced in comparison to cloud servers. Consequently, the data representation and the algorithm have been optimized. As a result, an accurate, memory reduced and low latency algorithm has been developed. The proposed algorithm has been evaluated on a custom dataset as well as the challenging NuScenes dataset where the achieved accuracy is 88.3 % and 58.6 % respectively.

Sensor Fusion; Deep Learning; Edge Computing; LiDAR; Radar; Target Detection.

8.1. Introduction

Autonomous vehicles research has provided great advances in the last few years thanks to the recent advances in engineering fields such as robotics, communication protocols and computer vision techniques. As a result, numerous proposal are arising about target detection applied to this topic, aiming to improve the state of the art by integrating emerging sensors or computational techniques. The main focus in this manuscript are the computer vision techniques based on LiDAR and radar sensors to detect and classify relevant targets in the environment of the vehicle. This is a required to understand the environment before autonomous vehicles can decide the route to follow. Deep Learning (DL) techniques have been previously used for this task due to their robustness and trustworthiness in classification and detection tasks (1).

Some of previous approaches focus on single-sensor models based on high quality sensors, such as Laser Imaging Detection and Ranging (LiDAR), radar or camera sensors, that provide enough information to detect and classify targets. However, some of these sensors, such as camera sensors, may incur privacy problems when working inn public areas. Therefore, even when it may provide more information than LiDAR sensors, emerging techniques are based on LiDAR sensors rather than camera sensors (2; 3; 4).

An example of single LiDAR sensor target detection for autonomous vehicles is the approach proposed by Tianwei Y. et al. (2). This algorithm studies the 3D LiDAR point cloud from a top view to convert them into a

2D image that is used to detect targets from that perspective. This Deep Neural Network (DNN), following the CenterNet strategy (5), first locate the center of the targets and in a later step it provides information regarding features such as orientation and size of targets. This model was evaluated in the NuScenes dataset where it achieved a 67.4 % accuracy.

Following the same approach of using a single LiDAR sensor for target detection, Qian et al. (3) proposed a model, BANet, for 3D object detection from point clouds. This technique, rather than using a traditional 2 steps pipeline for the target detection, represents each possible target as a node for graph construction. As a result, local neighborhood graphs are generated for the target detection. This technique was evaluated in the KITTI dataset, where it achieved an accuracy of 95.61 %, 98.75 % and 90.64 % for easy, medium and hard to detect targets respectively in KITTI dataset.

Wu Z. et al. (4) also researched the target detection based on LiDAR data using a DNN following a Single Shot Detector (SSD) structure. They implement a teacher and a student network to used a distilled knowledge pipeline. At the same time, a new loss function was designed for this specific case to to supervise the student with constraints on the predicted box centers and orientations. This model was also evaluated in the KITTI dataset, where it achieved an accuracy of 95.60 %, 96.69 % and 90.53 % for easy, medium and hard to detect targets respectively in KITTI dataset.

Even when previous researchers achieved high performance results using a single sensor approach, this approach may incur problems in situations such as hazard weather conditions or because of data corruption. A technique to face this problem is the sensor fusion, which relies on multiple sensors data integration. As a result, this approach gathers more general and complete data that can complement each other or support in adverse situations while this approach can also increase the performance of the system in normal conditions. Sensor Fusion can be applied at different levels of data preprocessing, leading to different subcategories such as Early Fusion (fusion of low-level preprocessed data is used) and Late Fusion (high-level preprocessed data is used). Recently, a combination of these two subcategories, known as Intermediate Fusion, is gaining more relevance since it merges the data at an intermediate step or steps leading to a more flexible algorithm (6; 7; 8; 9).

An example of sensor fusion for target detection is Hojoon L. et al. (10). This author proposed a 2D LiDAR-radar based sensor fusion model for vehicle detection. This sensor fusion technique was used to extend the Field of View (FoV) of the LiDAR sensor before the detection, which is executed using a Kalman filter. However, since this technique is not based on DNN, it lacks of flexibility and robustness to changing scenarios/targets. It only detects targets without considering the target class or features.

Other authors, to ensure a full environment view, do not limit the number of sensors to 2 like previous author or the model proposed in this manus-

cript. Following this research line, Babak S.J. et al. (11) designed a sensor fusion framework based on LiDAR, camera and radar sensors among others for autonomous vehicles topics such as road segmentation, obstacle detection and target tracking. This framework uses a Fully Convolutional Neural Network (FCN) based on an encoder-decoder structure as well as a traditional Extended Kalman Filter (EKF) for the data fusion. Nevertheless, this framework, due to the large number of sensors, requires a large energy source to ensure the correct data gathering. At the same time, due to the numerous preprocessing steps to obtain the final detection and segmentation results, this model has not being optimized for the network edge in comparison with our proposed technique.

Therefore, it can be observed how multiple approaches for the target detection are being researched and provide high accuracy results in some of the most popular datasets such as KITTI or NuScenes (12; 13). In the case of single-sensor models, even when high accuracy results are achieved, they may incur in problems when facing hazard situations such as bad weather conditions or adversarial attacks, as (6) explains. This is one of the main reasons to follow a sensor fusion approach for target detection in autonomous vehicle scenarios. However, to the best of our knowledge, it has not been discussed yet the suitability of LiDAR and radar fusion for this purpose.

Our proposed LiDAR-Radar Multi-Level (LR-ML) algorithm, faces the problem of the target detection in autonomous vehicle scenarios using these sensors that can complement each other. LiDAR data provides reliable information of depth, shape and target presence while radar data is used as an attention mechanism for the LiDAR data. As a result, it is easier to locate relevant areas in LiDAR data in contrast to using only LiDAR where it may be difficult to differentiate targets near each other. At the same time, this combination provides a solution for scenarios where LiDAR data may be corrupted because of weather conditions. Our algorithm executes the data fusion at multiple levels using the Intermediate Fusion approach as a bidirectional attention to further improve both input data or feature maps, and it has been optimized to be executed in low-resource hardware such as Edge Intelligence devices.

In order to test this algorithm in a scenario close to a real autonomous vehicle scenario, it has been deployed in an Edge Device similar to the ones used in automotive industry, the Google Coral TPU (14). Because of this, the latency and memory consumption of the model have been used as constraints during the design of the model to ensure its suitability for the network edge.

The proposed model has been evaluated using a custom dataset where it achieved an accuracy of 88.3 % and the challenging NuScenes dataset (13).

The manuscript is organized as follows: Section 2 focuses on the proposed algorithm is described for a deep understanding of its structure as well as the required input data preprocessing. After that, Section 3 will focus on the

experiment where this algorithm has been applied, and finally Section 4 is the conclusion.

8.2. Proposed LiDAR-Radar Multi-Level Sensor Fusion Network

The proposed LiDAR-Radar Multi-Level Sensor Fusion (LRML) Network aims to detect targets based on the depth information extracted from LiDAR data. In this technique, radar data is used as an attention mechanism data since it provides information regarding areas where moving targets may be located but it does not reflect all target. The data from these sensors is fused at multiple levels rather than in a single step. As a consequence of this, the network can decide during training the optimal step or steps to fuse the information in order to get the desired results.

In order to optimize the process for the network edge, sensor data has been preprocessed to reduce its dimension while maintaining relevant features such as shapes in LiDAR data and target location in radar data. At the same time, during the preprocessing the memory size of the data has been reduced to fit the memory and latency limitations at the network edge.

8.2.1. Depth Map data representation

A Laser Imaging Detection and Ranging (LiDAR) sensor has been included in this project to gather information regarding shapes and positions of obstacles in the environment. However, the problem is the lack of order in the unsorted point cloud data resulting. Consequently, the order of these 3D points is usually assumed to be unknown, unlike pixel arrays in images. Because of this, the integration of LiDAR data in DNNs is not straightforward since the network must be invariant to these permutations of the input data in the feeding order.

To face this problem, there are multiple techniques to transform LiDAR 3D points into a structured format. The voxel approach is one of the most commonly used in the literature (15; 16). However, this technique still generates a complex data since it generates a 3D matrix to represent the studied space where each cell summarizes the information of that spatial area. This complexity leads to larger models as well as layer structures that are not supported usually by edge devices, such as 3D convolutional layers.

To solve this problem, a Depth Map approach has been followed. This technique converts 3D points into a 2D image which can be studied as a camera image. Apart from the simplicity of the resulting format, this technique also leads to a reduction of the memory size of the data, as discussed in (9). The process to convert 3D points to Depth Maps is ruled by equations (1), (2), (3) and (4) for each point. These equations transform the 3D points

from Cartesian coordinates to the new representation system.

$$x_{DepthMap} = arctan(y_{3D}/x_{3D}) \tag{8.1}$$

$$d_{3D} = \sqrt{x_{3D}^2 + y_{3D}^2 + z_{3D}^2} \tag{8.2}$$

$$y_{DepthMap} = arccos(z_{3D}/d_{3D}) \tag{8.3}$$

$$color_{DepthMap} = d_{3D} \tag{8.4}$$

These new coordinates and colors can be directly used to generate the new image data. Since the new data type is a 2D image, they can be studied using traditional approaches of computer vision using Convolutional Neural Networks.

When converting LiDAR data into Depth Maps, 3D points of the ground have been removed for an easier location of targets. This technique has also been applied to radar as previously studied by other authors (6).

In case of radar, the same Depth Map representation will be used in order to ease the attention in the Depth Map it provides to LiDAR data. Therefore, the first step when processing radar data is extracting the location of the relevant targets. To obtain this information, the radar raw data is first preprocessed to generate Range Doppler Images using a double Fourier transformation. This data can later be transformed into a Range Angle Map using the beamforming technique to generate an image with range and angle in the axis. Consequently, it is possible to locate the 2D position of clusters-possible targets in the horizontal plane. At this step, this information is similar to LiDAR data but without the height component. To overcome this problem, we have assumed all radar targets to have a height of 1.5 meters. This height has been selected since it is a representative height of pedestrians, bicycles and cars. Consequently, each radar detection has been understood as a continuous vertical detection of 1.5 meters (from the ground). Once the information from the X, Y and Z axis has been extracted from radar data, equations (1), (2), (3) and (4) can be used to generate radar Depth Maps as done with LiDAR data.

Once the data from LiDAR and radar sensors have been transformed to the same structure (Depth Maps) and coordinates, they can be fed into our DL model for target detection. A sample of the LiDAR and radar depth map is shown in Figure 8.1

8.2.2. Overall system description

The full system pipeline from data acquisition to target detection using the proposed model is shown in Figure 8.2. This figure shows how the first

Figure 8.1: From left to right: original environment camera image, LiDAR depth map and radar depth map.

step after the data gathering is the alignment of the data to ensure the use of the same coordinate origin. This process can be executed since the relative position of a sensor respect the other is known in advance. The following step is the data preprocessing to generate the previously explained Depth Maps, which contain the spatial information in an more optimized format than the raw data, and to fit the desired input format for the proposed algorithm. Finally, the proposed LR-ML model studies the input data to provide the classification and localization of the targets in the current frame.

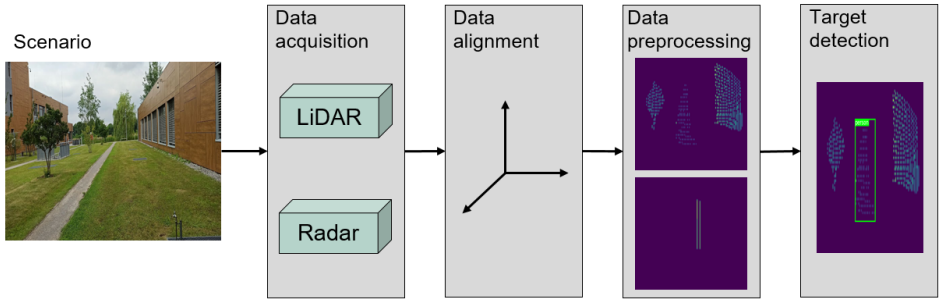

Figure 8.2: Full pipeline of the proposed target detection model.

As previously commented, the goal of the proposed LR-ML Sensor Fusion model is to extract relevant information from radar data that can be later used as reinforcement or attention pointer. In order to achieve this, data from LiDAR and radar are fed separately to a DNN following the structure presented in Figure 8.3. This network uses a SSD structure with mobilenet backbone (17) to preprocess the initial LiDAR depth maps to extract relevant feature maps. This structure has been selected due to its suitability for latency constrain scenarios as well as its reduced memory consumption while achieving high performance results (9).

A different branch of the network preprocesses the radar depth map using a set of 2D Convolutional layers to extract the attention feature maps.

Once both inputs have been preprocessed, they are merged at multiple levels. This fusion is executed using a Fusion Layer (9). This layer merges

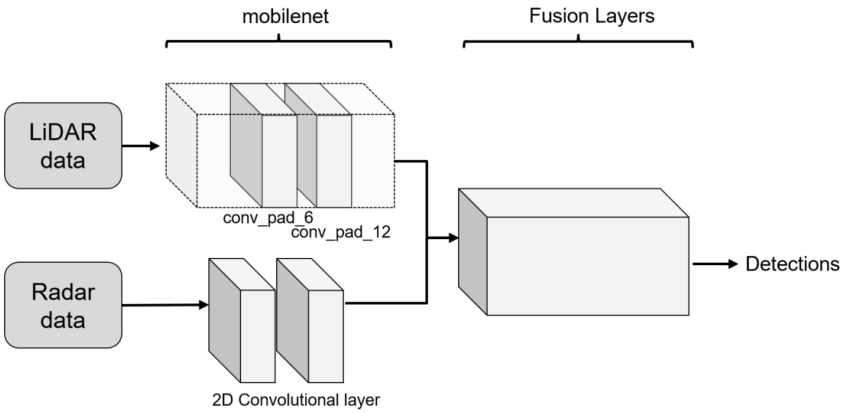

Figure 8.3: LiDAR-Radar Multi-Level Sensor Fusion network structure.

the information from the radar and LiDAR feature maps by concatenating the data in the last axis. Once the data has been concatenated, it is fed into a 2D Convolutional layer before concatenating its result with each of the initial feature maps. As a result, the information from both sensors is shared to improve the reinforcement feature maps of radar as well as using them to locate relevant areas in LiDAR feature maps.

Since the proposed model aims to classify and locate targets, it has two outputs. Consequently, 5 Fusion layers have been integrated into each of the output branches to ensure the fusion is executed for both outputs. This two branches approach for the fusion enables the model to use the same technique and fine tune its structure for the specific task of each branch. The initial feature maps for the classification are extracted from the *conv-pad-6* layer of the SSD mobilenet networks that preprocess the LiDAR data and the last 2D Convolutional layer that studies the radar data. For the localization, the initial feature maps are extracted from the *conv-pad-12* layer in the SSD mobilenet model and the last 2D Convolutional layer of the radar.

Since our approach is based on the SSD structure for target detection, the SSD loss function has also been implemented for the training of the proposed model. This loss function is ruled by equations (5)–(14).

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in (cx,cy,w,h)} x_{ij}^{k} \cdot S_{L1} \tag{8.5}$$

$$S_{L1} = smoothth_{L1}(l_i^m - \hat{g}_j^m) \tag{8.6}$$

$$smooth_{L1}(l_i^m - \hat{g}_j^m) \tag{8.7}$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \tag{8.8}$$

$$\hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \tag{8.9}$$

$$\hat{g}_j^{w} = log(\frac{g_j^{w}}{d_i^{w}}) \tag{8.10}$$

$$\hat{g}_j^{h} = log(\frac{g_j^{h}}{d_i^{h}}) \tag{8.11}$$

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p - \sum_{i \in Neg} log(\hat{c}_i^0) \tag{8.12}$$

$$\hat{c}_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)} \tag{8.13}$$

$$L(x,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)) \tag{8.14}$$

where N is used to represent the number of matching default boxes, l the predicted boxes, g the ground truth boxes, x the coordinates of the bounding boxes and c the class confidences. These parameters also include the offset for the center points (cx, cy), the width of the box (w) and its height (h).

Since the goal is to deploy this model in an Edge Device, it needs to be optimized to ensure its correct performance at the network edge. To do so, the model layers have been prunned (18) to remove connections (in the case of dense layers) or full layers (in the case of convolutional layers) when they do not highly modify the value of the input signal. As a result, operations that did not efficiently reduce or extract features are removed to not consume resources when it is not required. Consequently, the models needs to perform less operations and store less parameters. Apart from this, since the model will be deployed in the Google Coral TPU, the constrains of this Edge Device must also been taken into account. In this case, the constrain that must be followed is the quantization of the model to 8-bit integer values. This leads to a reduction in the accuracy in comparison with float point representation but it decreases the memory size to 101.5 MB in contrast to the initial size of 350.1 MB.

8.3. Experiments

In this section, our proposed LR-ML model is evaluated using a custom dataset to study the accuracy drop in hazard weather conditions. The NuScene dataset (13) has also been used during the evaluation of the proposed model to compare its accuracy results for the 2D target detection with state-of-the-art techniques based on LiDAR data as well as LiDAR sensor fusion

techniques. At the same time, latency and memory size are also included in the comparison since the goal is to design a target detection model for the network edge. In order to ensure the proposed approach is not limited to this dataset, it has been evaluated on a new dataset where data has been gathered using a different system from the proposed by NuScenes. More information regarding the specific sensors used for this task is provided in Subsection 8.3.2.

8.3.1. Datasets

A new dataset has been created to evaluate the proposed algorithm in a controlled environment. This dataset has been recorded using the Cube1 LiDAR sensor (19) and the radar sensor 60 GHz BGT60TR13C from Infineon Technologies (20). This dataset contains 1500 labeled frames (where 450 frames are used for test and 1050 for training). There are 5 possible classes in this dataset: pedestrian, bicycle, car, wall and tree. These classes have been selected since they are the most common targets in urban scenarios. In order to increase the number of frames for training, data augmentation techniques have been used to rotate, flip, translate and resize the initial images to generate a final dataset of 7500 frames.

A sample of the camera ground truth (used only during the data labeling), LiDAR depth map and radar depth map frames from this dataset is shown in Figure 8.4. The last sample remarks how targets are not always present in radar data since static objects are not not detected.

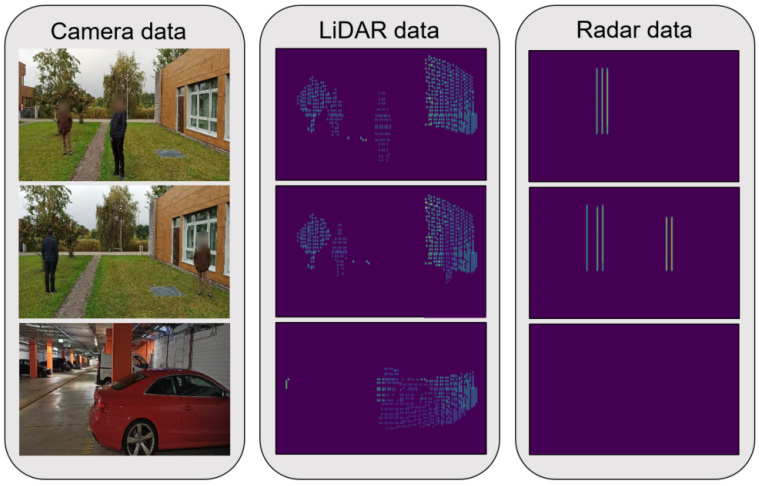

Figure 8.4: Camera ground truth, LiDAR depth map and radar depth map frame samples.

Apart from this dataset, the proposed algorithm has also been evaluated in the NuScenes dataset (13) to provide results in a generic dataset. This

dataset is one of the most popular databases when LiDAR and radar data is required due to the large quantity of data as well as their quality. The LiDAR sensor used in this dataset is the Velodyne HDL-64E (21). It also provides information of other sensors like cameras. As a result, numerous state-of-the-art LiDAR and radar DNN models have been evaluated with this dataset. This eases the comparison of new algorithms with the state of the art since it ensures models have been evaluated with the same input data.

This dataset contains information from 1000 scenes in Boston and Singapore. These scenes have been divided into training, validation and test data to ensure the fair comparison of models evaluated on this dataset. These subdatasets include 700 scenes (28,130 samples), 150 scenes (6019 samples) and 150 scenes (6008 samples) respectively. All subsets include LiDAR and radar data. Labeled targets include 10 different classes for target detection. Some of these classes are vehicles, pedestrians, mobility devices and other objects. The labels in this dataset are provided using the coordinates of the ego vehicle that can be later transformed to each specific sensor integrated in the NuScenes dataset.

8.3.2. Hardware architecture

The goal of this paper is to present a sensor fusion pipeline for target detection at the network edge. Consequently, the latency and accuracy results were calculated in an Edge Device to evaluate the performance of the model at the network edge.

The specific device used for the experiments is the Google Coral TPU Development Board, shown in Figure 8.5. This platform is based on the integration of a TPU co-processor to execute the tensor computations in collaboration with a CPU (NXP i.MX 8M SoC) and GPU (GC7000 Lite). Because of this structure, this device is able to also execute operations that are not based on tensors or that are not optimized for the TPU. As a result of this, it is capable of performing 4 trillion operations per second (TOPS) while only consuming 0.5 W/TOPS.

Simultaneously, the TensorFlow Lite software framework for the implementation of DNNs in the device enables their optimization. As a result, the latency, as well as the size of the model, can be reduced in most cases.

The results of the algorithms have been compared with the original papers and NuScenes dataset results table (13).

8.3.3. Experimental settings

The parameters used during the training of the model, for each dataset, can be observed in Table 8.1. The SSD loss function has been used for the training of the model since the general DNN structure follows the same

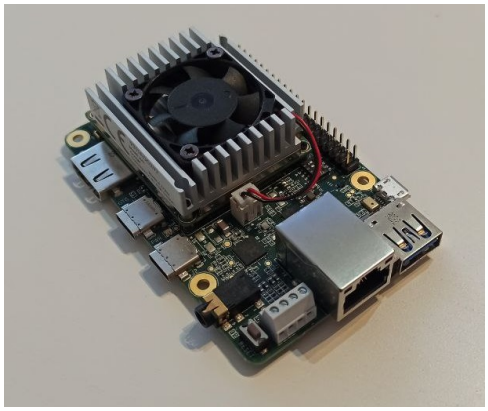

Figure 8.5: Google Coral TPU Development Board.

structure while only adding the Fusion layers to merge the feature maps from LiDAR and radar data.

Parameter	Setting (custom data)	Setting (NuScenes)
Epochs	600	400
Batch size	10	10
Optimizer	SGD	10
Initial learning rate	$1{\cdot}e^{-4}$	$1{\cdot}e^{-4}$
Final learning rate	$1{\cdot}e^{-5}$	$1{\cdot}e^{-5}$
Momentum	0.9	0.9

Table 8.1: Parameters for the model training in custom and NuScenes datasets.

The number of labels for each of the classes included in the NuScenes dataset is not the same. This is normal since some of the classes are highly common, such as pedestrian, while others are not so common in normal scenarios, such as bus. Therefore, the algorithm for the model training tries to use the same number of labels from each of the studied classes to overcome this class imbalance problem while still using all the training data. When studying our new dataset the same approach is followed in order to reduce the effect of the data class imbalance.

The Intersection Over Union (IOU) algorithm has been implemented to compare the predicted bounding box with the ground truth labels as an accuracy measurement. This technique can be applied to any system that predicts bounding boxes in scenarios where the ground truth is known. The algorithm is explained in (15), where A refers to the ground truth bounding box and B to the predicted bounding box. A threshold value for the IOU result must be set to decide if a predicted bounding box matches the initial

ground truth label. In this paper, the IOU threshold has been set to 0.5.

$$IOU = \frac{A \cap B}{A \cup B} \tag{8.15}$$

8.3.4. Results

In this subsection, the evaluation results of our proposed model in our custom and NuScenes datasets are provided and discussed for a deeper understanding of the advantages of our technique.

Our proposed model has first been evaluated in our custom dataset to study its performance in a controlled scenario where the exact configuration of sensors is known. The results in this dataset provided a final accuracy of 88.3 % while maintaining a low latency (38 milliseconds) and a reduced model size (101.5 MB).

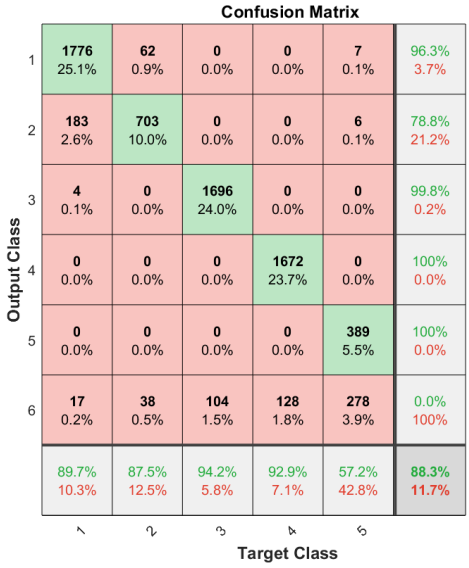

Figure 8.6: Confusion matrix result in custom dataset.

Figure 8.6 shows the exact results of the accuracy evaluation in the custom dataset. In this confusion matrix the different classes have been codified into: 1-pedestrian, 2-bicycle, 3-tree, 4-wall and 5-car. Columns represent ground truth labels and rows the detections of the model. It is possible to observe how there are 6 possible prediction classes (rows), where the last one represents false negative results. This confusion matrix shows how most of the wrong classifications occur between pedestrian and bicycle classes due to their similarities. In the rest of the cases, the accuracy drop is due to false negative detections, such as the case of the class car.

Apart from the previous performance results, as discussed in this work, the integration of the radar as an attention mechanism for LiDAR data improves the model performance in hazard weather conditions. To test this, the custom dataset has been used to generate another dataset where a 40 % of the LiDAR detections have been removed to simulate the effect of fog/rain. An example frame of this is shown in Figure 8.7

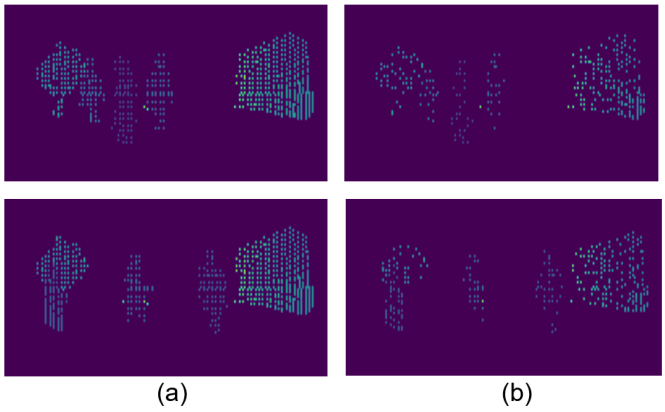

Figure 8.7: Sample of (a) original and (b) synthetic fog condition depth maps.

The difference in the accuracy results achieved by our proposed model, the CenterPoint v2 (2) and a single-sensor LiDAR SSD model when using the custom and synthetic fog datasets in shown in Table 8.2.

Model	Initial custom dataset	Synthetic fog dataset
Single LiDAR SSD	86.7 %	68.4 %
CenterPoint v2 (2)	90.3 %	79.1 %
LR-ML (proposal)	88.3 %	81.2 %

Table 8.2: Comparison of accuracy results with initial custom dataset and synthetic fog dataset.

It is possible to observe in Table 8.2 how the accuracy in the initial custom dataset only improves a 1.6 % when using our proposed model in comparison with a SSD structure. However, the CeneterPoint model based in LiDAR and camera sensors achieves a better accuracy result due to the integration of camera data that provides a high quantity of relevant features as well as the 3D convolutional study of the data.

When incurring in hazard situations such as the previously commented, this SSD model with a single sensor is highly affected since LiDAR data may not represent target as expected. The same effect is visible in the CenterPoint v2 network which also uses information from camera sensors, which would also be affected by hazard weather conditions such as rain or fog, to recover

missing information from the LiDAR data. Consequently, when studying hazard weather conditions these two approaches suffer an accuracy drop due to the missing data from the sensors. On the other hand, our sensor fusion approach uses radar data that is not affected by hazard weather conditions as an attention mechanism to select relevant areas of the LiDAR depth map data, what leads to a an accuracy drop of a 7.1 % in comparison to the case of the SSD where it dropped a 18.3 % and CenterPoint v2 a 11.2 %.

This result proves how the proposed approach, apart from improving the initial accuracy in comparison with a single-sensor approach, helps to maintain a high accuracy result in unexpected hazard situations in autonomous vehicle scenarios such as weather conditions or partial occlusion of sensors.

After discussing the performance results in our custom dataset, the proposed model has been evaluated in the NuScenes dataset to ensure its flexibility to achieve high performance results with different sensors and scenarios. The final accuracy achieved by our proposed algorithm in this dataset is 58.6 %. The individual accuracy for each of the classes as well as the error distribution in the NuScenes dataset is shown in Figure 8.8. In this confusion matrix, like in the previous one, columns represent target classes and rows the predictions. The different classes have been codified into: 1-car, 2-truck, 3-bus, 4-trailer, 5-construction, 6-pedestrian, 7-motorcycle, 8-bicycle, 9-traffic cone, 10-barrier and 11- no detected targets.

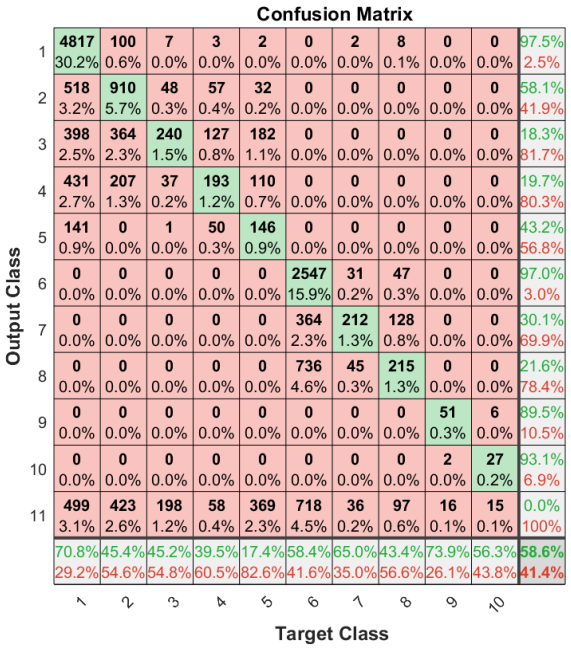

Figure 8.8: Confusion matrix result in NuScenes dataset.

It can be observed how the accuracy results highly differ among classes, such as the difference between the classes car and construction. This is the result of the high class imbalance in this dataset as well as the difficulty to detect smaller targets, such as traffic cones, with LiDAR sensors, which may not get enough points belonging to these targets if the are not close enough. However, the accuracy results for common targets such as car and pedestrian are not far from the state of the art results. At the same time, it is possible to observe how the error distribution follows the expected results where classes car, truck, bus and construction and often wrongly classify due to the similarities of these vehicles when studying only their shapes. The same problem is detected between classes pedestrian, motorcycle and bicycle since the main surface of these targets is the person.

After the study of the results on this dataset, the proposed algorithm and the state-of-the-art algorithms for target detection based on LiDAR data in the NuScenes dataset will be compared. The selected parameters to compare among all the discussed models in Table 8.3 are the size of the models, accuracy, latency and used sensors. These parameters have been selected since they are the most representative of the performance of a model at the network edge at the same time it is relevant to know what sensors have been used. This is relevant since the performance for a detection is highly affected by initial data, i.e. in the case of camera data, the state-of-the-art accuracy for detection is higher than LiDAR due to the larger number of features that can be extracted. However, camera sensors may incur into privacy problems as previously discussed. At the same time, camera sensors are sensitive to environmental conditions (weather, lighting, etc.) which can affect the performance in hazard situations.

Models tested on NuScenes dataset have been selected for the comparison in order to ensure the same data was used. At the same time, since the proposed LR-ML model requires LiDAR and radar data, it is the most representative dataset in contrast with other datasets, such as KITTI, which do not provide radar data. Among all models evaluated in the NuScenes dataset, 4 representative models have been selected due to their high positions in the benchmark NuScenes table as well as the information authors provided regarding these models and their trustworthiness. The chosen models for the comparison are: CenterPoint v2 (2), FusionPainting (22), MEGVII (23) and Shape Signature Networks (SSN) v2 (24). Nevertheless, since most of the authors do not focus on edge computing approaches, information regarding latency and memory size of the models is not public in some of the cases. Consequently, this comparison must be understood as a general view of the state of the art and not a direct comparison. Examples of LiDAR-radar fusion models have not been included in the comparison due to the lack of information regarding these models in the NuScenes benchmark table.

Table 8.3 shows how the accuracy results achieved by the proposed model

Model	Size of the model	Accuracy (%)	Latency (s)	Sensors
LR-ML (proposal)	101.5 MB	58.6	0.038	LiDAR-Radar
CenterPoint v2 (2)	106 MB	67.4	0.057	LiDAR
FusionPainting (22)	—	68.1	$\geq$ 0,124	LiDAR-Camera
MEGVII (23)	—	52.8	—	LiDAR
SSN v2 (24)	—	50.6	—	LiDAR

Table 8.3: Comparison of LiDAR-Camera fusion networks for target detection on NuScenes dataset.

does not surpass the state-of-the-art results such as the CenterPoint v2, which uses a voxel approachand 3D convolutional layers not supported by Edge Devices, (2) or the FusionPainting (22) models, which achieve a 8.8 % and 9.5 % higher accuracy respectively. On the other hand, the proposed model accuracy is better than the rest of the compared models.

Regarding latency, it is possible to observe how the proposed model is a 50 % faster than the CenterPoint v2. The FusionPainting latency is also higher than the achieved by the proposed model. This latency has been calculated based on the latency of each of the detectors and auxiliary modules required in the FusionPainting model, resulting in a minimum latency of 0.124 seconds. The rest of the models do not provide information regarding their latency in the NuScenes dataset so a direct comparison is not possible.

Similarly, the size of the models have been compared but not all the compared target detection models evaluated in the NuScenes dataset provide information about their memory size so a direct comparison is not possible. However, some of them do provide this information, such as the CenterPoint v2 model, which is 4.5 MB larger than the proposed model. The models that do not provide information regarding their memory size can be assumed larger than the proposed one due to their structure, such as SSN v2 (24) that prior the target detection reconstruct missing information of covered parts of the targets based on symmetry or the MEGVII (23) that requires the input data to be transformed into voxels to later study it using 3D convolutions. Similarly, the FusionPainting required to study the LiDAR 3D data at the same as the camera data prior their fusion to generate an attention map for the target detection. These preprocessing steps are based on semantic segmentation models, known by their high memory requirements. At the same time, none of these models have been optimized for the network edge as it is possible to observe by the layers, such as 3D Convolutional layers, that are not commonly supported by Edge Devices.

As a conclusion from this table, it is possible to observe how other models achieve better accuracy results due to larger and more complex structure that are not suitable for the network edge or the integration of sensors, i.e. camera

sensors, that may lead to privacy issues. Therefore, when taking into account the constrains at the network edge as well as the trend to avoid sensors that may record private data, our model proposed a different approach to face these problems in the topic of target detection.

8.4. Conclusions

A robust Multi-Level Sensor Fusion DNN based on LiDAR and radar sensors has been developed in this manuscript for target detection in autonomous vehicles scenarios. To better fit the constrains of these scenarios, network edge, the algorithm has been developed taking into account the memory usage and latency. To evaluate the proposed model, it has been implemented in a Google Coral TPU Edge Device where it achieved an accuracy of 58.6 % and a latency of 38 milliseconds in the challenging NuScenes dataset.

Even when the accuracy results do not surpass the state-of-the-art accuracy shown by other models, as a result of the optimization applied to the model, it achieved a reduced latency and memory requirements in contrast to the rest of compared models in this manuscript.

At the same time, the advantages of using a sensor fusion approach rather than a single sensor model have been studied by comparing our proposed algorithm and a single LiDAR sensor SSD model in the initial custom dataset and a synthetic dataset that emulates hazard weather conditions. In this test it was possible to observe the robustness of the proposed model to data corruption in comparison with single-sensor techniques.

We can conclude that the proposed model has been designed taking into consideration the constrains of Edge Devices as well as limitations of the final application such as autonomous driving. Consequently, the algorithm follows a trade-off among the accuracy, latency and memory size.

References

[1] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Meireles Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, page 113816, 2020.

[2] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021.

[3] Rui Qian, Xin Lai, and Xirong Li. Boundary-aware 3d object detection from point clouds. *ArXiv*, abs/2104.10330, 2021.

[4] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14494–14503, June 2021.

[5] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6569–6578, 2019.

[6] F. Nobis, M. Geisslinger, M. Weber, J. Betz, and J. Lienkamp. A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection. *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF), Bonn, Germany*, pages 1–7, 2019.

[7] J. Kim, J. Choi, Y. Kim, J. Koh, C. C. Chung, and J. W. Choi. Robust camera lidar sensor fusion via deep gated information fusion network. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1620–1625, 2018.

[8] Zhizhong Kang, Juntao Yang, Ruofei Zhong, Yongxing Wu, Zhenwei Shi, and Roderik Lindenbergh. Voxel-based extraction and classification of 3-d pole-like objects from mobile lidar point cloud data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11:4287–4298, 11 2018.

[9] Javier Mendez, Miguel Molina, Noel Rodriguez, Manuel P Cuellar, and Diego P Morales. Camera-lidar multi-level sensor fusion for target detection at the network edge. *Sensors*, 21(12):3992, 2021.

[10] Hojoon Lee, Heungseok Chae, and Kyongsu Yi. A geometric model based 2d lidar/radar sensor fusion for tracking surrounding vehicles. *IFAC-PapersOnLine*, 52(8):130–135, 2019. 10th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2019.

[11] Babak Shahian Jahromi, Theja Tulabandhula, and Sabri Cetin. Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles. *Sensors*, 19(20), 2019.

[12] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[13] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[14] Google. Google coral tpu dev board. `https://coral.ai/products/dev-board/`. Accessed: 2021-10-01.

[15] Hongwu Kuang, Bei Wang, Jianping An, Ming Zhang, and Zehan Zhang. Voxel-fpn: Multi-scale voxel feature aggregation for 3d object detection from lidar point clouds. *Sensors*, 20(3):704, 2020.

[16] Maxime Soma, Francois Pimont, and Jean-Luc Dupuy. Sensitivity of voxel-based estimations of leaf area density with terrestrial lidar to vegetation structure and sampling limitations: A simulation experiment. *Remote Sensing of Environment*, 257:112354, 2021.

[17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861, 2017.

[18] Russell Reed. Pruning algorithms-a survey. *IEEE transactions on Neural Networks*, 4(5):740–747, 1993.

[19] Blickfeld. Blickfeld cube 1 lidar sensor. `https://www.blickfeld.com/products/cube-1/`. Accessed: 2021-10-05.

[20] Infineon Technologies AG. BGT60TR13C evaluation board. `https://www.infineon.com/cms/de/applications/solutions/sensor-solutions/presence-detection/#!boards`. Accessed: 2021-10-05.

[21] Velodyne. Velodyne hdl-64e sensor. `https://velodynelidar.com/products/hdl-64e/`. Accessed: 2021-10-4.

[22] Shaoqing Xu, Dingfu Zhou, Jin Fang, Junbo Yin, Bin Zhou, and Liangjun Zhang. Fusionpainting: Multimodal fusion with adaptive attention for 3d object detection. *ArXiv*, abs/2106.12449, 2021.

[23] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019.

[24] Xinge Zhu, Yuexin Ma, Tai Wang, Yan Xu, Jianping Shi, and Dahua Lin. Ssn: Shape signature networks for multi-class object detection from point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 581–597. Springer, 2020.

References

[1] Hervé Abdi. Metric multidimensional scaling (mds): analyzing distance matrices. *Encyclopedia of measurement and statistics*, pages 1–13, 2007.

[2] Paolo Annibale, Jason Filos, Patrick A Naylor, and Rudolf Rabenstein. Tdoa-based speed of sound estimation for air temperature and room geometry inference. *IEEE transactions on audio, speech, and language processing*, 21(2):234–246, 2012.

[3] Sebastian Anzinger, Christian Bretthauer, Johannes Manz, Ulrich Krumbein, and Alfons Dehé. Broadband acoustical mems transceivers for simultaneous range finding and microphone applications. *2019 20th International Conference on Solid-State Sensors, Actuators and Microsystems & Eurosensors XXXIII (TRANSDUCERS & EUROSENSORS XXXIII)*, pages 865–868, 2019.

[4] Ben Beklisi Kwame Ayawli, Ryad Chellali, Albert Yaw Appiah, and Frimpong Kyeremeh. An overview of nature-inspired, conventional, and hybrid methods of autonomous vehicle path planning. *Journal of Advanced Transportation*, 2018, 2018.

[5] Igal Bilik, Oren Longman, Shahar Villeval, and Joseph Tabrikian. The rise of radar for autonomous vehicles: Signal processing solutions and future research directions. *IEEE signal processing Magazine*, 36(5): 20–31, 2019.

[6] Dennis A Bohn. Environmental effects on the speed of sound. *Journal of the audio engineering society, Audio Engineering Society Convention 83*, 1987.

[7] Mat Buckland. *Programming Game AI by Example.* Wordware Publishing, Inc., 2005.

[8] Aladin Carovac, Fahrudin Smajlovic, and Dzelaludin Junuzovic. Application of ultrasound in medicine. *Acta Informatica Medica*, 19(3): 168, 2011.

[9] Jian Chen, Fan Yu, Jianxin Yu, and Lin Lin. A three-dimensional pen-like ultrasonic positioning system based on quasi-spherical pvdf ultrasonic transmitter. *IEEE Sensors Journal*, 2020.

[10] Maximo Cobos, Fabio Antonacci, Anastasios Alexandridis, Athanasios Mouchtaris, and Bowon Lee. A survey of sound source localization methods in wireless acoustic sensor networks. *Wireless Communications and Mobile Computing*, 2017, 2017.

[11] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. *Version Control with Subversion.* O'Reilly, 2004. ISBN 0-596-00448-6.

[12] SAE On-Road Automated Vehicle Standards Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *SAE Standard J*, 3016:1–16, 2014.

[13] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. `https://github.com/open-mmlab/mmdetection3d`, 2020.

[14] David MJ Cowell and Steven Freear. Separation of overlapping linear frequency modulated (lfm) signals using the fractional fourier transform. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 57(10):2324–2333, 2010.

[15] Tobias Dahl, Joao L Ealo, Hans J Bang, Sverre Holm, and Pierre Khuri-Yakub. Applications of airborne ultrasound in human–computer interaction. *Ultrasonics*, 54(7):1912–1921, 2014.

[16] Amit Das, Ivan Tashev, and Shoaib Mohammed. Ultrasound based gesture recognition. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 406–410, 2017.

[17] Mathieu De Coster, Mieke Van Herreweghe, and Joni Dambre. Sign language recognition with transformer networks. In *12th International Conference on Language Resources and Evaluation*, pages 6018–6024. European Language Resources Association (ELRA), 2020.

[18] Jan De Leeuw and Patrick Mair. Multidimensional scaling using majorization: Smacof in r. 2011.

[19] Vin De Silva and Joshua B. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Technical report, Stanford University, 2004.

[20] Yao Deng, Xi Zheng, Tianyi Zhang, Chen Chen, Guannan Lou, and Miryung Kim. An analysis of adversarial attacks and defenses on autonomous driving models. In *2020 IEEE international conference on pervasive computing and communications (PerCom)*, pages 1–10. IEEE, 2020.

[21] Raúl Dominguez, Enrique Onieva, Javier Alonso, Jorge Villagra, and Carlos Gonzalez. Lidar based perception solution for autonomous vehicles. In *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 790–795. IEEE, 2011.

[22] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6569–6578, 2019.

[23] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6569–6578, 2019.

[24] W. Yu et al. A Survey on the Edge Computing for the Internet of Things. *IEEE Access*, 6:6900–6919, 2018.

[25] Jamil Fayyad, Mohammad A Jaradat, Dominique Gruyer, and Homayoun Najjaran. Deep learning sensor fusion for autonomous vehicle perception and localization: A review. *Sensors*, 20(15):4220, 2020.

[26] Martin Fowler. Continuous integration, 2006.

[27] Y. Gao, M. A. Maraci, and J. A. Noble. Describing ultrasound video content using deep convolutional neural networks. *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 787–790, 2016. doi: 10.1109/ISBI.2016.7493384.

[28] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[29] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[30] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[31] David G. Gobbi, Roch M. Comeau, and Terry M. Peters. Ultrasound probe tracking for real-time ultrasound/mri overlay and visualization of brain shift. pages 920–927, 1999.

[32] José Luis Guiñón, Emma Ortega, José García-Antón, and Valentín Pérez-Herranz. Moving average and savitzki-golay smoothing filters using mathcad. *Papers ICEE*, 2007, 2007.

[33] Sidhant Gupta, Daniel Morris, Shwetak Patel, and Desney Tan. Soundwave: using the doppler effect to sense gestures. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1911–1914, 2012.

[34] Chris Hand. A survey of 3d interaction techniques. 16(5):269–281, 1997.

[35] Simon Haykin, John Litva, Terence J Shepherd, et al. *Radar array processing*. Springer, 1993.

[36] G Hayward, F Devaud, and JJ Soraghan. P1g-3 evaluation of a bio-inspired range finding algorithm (bira). *2006 IEEE Ultrasonics Symposium*, pages 1381–1384, 2006.

[37] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11873–11882, 2020.

[38] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–1780, 1997.

[39] Ke-Nung Huang and Yu-Pei Huang. Multiple-frequency ultrasonic distance measurement using direct digital frequency synthesizers. *Sensors and Actuators A: Physical*, 149(1):42–50, 2009.

[40] Google I/O. Google tpu. URL `https://coral.ai/docs/edgetpu/faq/`. Accessed: 2022-02-07.

[41] J. C. Jackson, R. Summan, G. I. Dobie, S. M. Whiteley, S. G. Pierce, and G. Hayward. Time-of-flight measurement techniques for airborne ultrasonic ranging. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 60(2):343–355, 2013.

[42] Young Jeon, Taehong Kim, and Taejoon Kim. Fast and robust time synchronization with median kalman filtering for mobile ad-hoc networks. *Sensors*, 21(2):590, 2021.

[43] Moran Ju, Jiangning Luo, Panpan Zhang, Miao He, and Haibo Luo. A simple and efficient network for small target detection. *IEEE Access*, 7:85771–85781, 2019.

[44] Vijay Kakani, Van Huan Nguyen, Basivi Praveen Kumar, Hakil Kim, and Visweswara Rao Pasupuleti. A critical review on computer vision and artificial intelligence in food industry. *Journal of Agriculture and Food Research*, 2:100033, 2020.

[45] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219 – 235, 2019. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2019.02.050. URL `http://www.sciencedirect.com/science/article/pii/S0167739X18319903`.

[46] AKRMLPJ Khosravi, RNN Koury, L Machado, and JJG Pabon. Prediction of wind speed and wind direction using artificial neural network, support vector regression and adaptive neuro-fuzzy inference system. *Sustainable Energy Technologies and Assessments*, 25:146–160, 2018.

[47] J. Kim, J. Choi, Y. Kim, J. Koh, C. C. Chung, and J. W. Choi. Robust camera lidar sensor fusion via deep gated information fusion network. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1620–1625, 2018. doi: 10.1109/IVS.2018.8500711.

[48] Jelena Kocić, Nenad Jovičić, and Vujo Drndarević. Sensors and sensor fusion in autonomous vehicles. In *2018 26th Telecommunications Forum (TELFOR)*, pages 420–425. IEEE, 2018.

[49] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 22 (2):712–733, 2020.

[50] Taavi Laadung, Sander Ulp, Muhammad M. Alam, and Yannick Le Moullec. Active-passive two-way ranging using uwb. pages 1–5, 2020. doi: 10.1109/ICSPCS50536.2020.9309999.

[51] Patrick Lazik and Anthony Rowe. Indoor pseudo-ranging of mobile devices using ultrasonic chirps. pages 99–112, 2012.

[52] Hojoon Lee, Heungseok Chae, and Kyongsu Yi. A geometric model based 2d lidar/radar sensor fusion for tracking surrounding vehicles. *IFAC-PapersOnLine*, 52(8):130–135, 2019. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2019.08.060. 10th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2019.

[53] John J. Leonard and Hugh F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on robotics and Automation*, 7(3):376–382, 1991.

[54] Jacques Lewiner. Paul langevin and the birth of ultrasonics. *Japanese Journal of Applied Physics*, 30(S1):5, jan 1991. doi: 10.7567/jjaps.30s1.5. URL `https://doi.org/10.7567%2Fjjaps.30s1.5`.

[55] Wenbin Li and Matthieu Liewig. A survey of ai accelerators for edge environment. In *World Conference on Information Systems and Technologies*, pages 35–44. Springer, 2020.

[56] Xinrong Li. Collaborative localization with received-signal strength in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 56(6):3807–3817, 2007.

[57] Yehao Li, Ting Yao, Yingwei Pan, and Tao Mei. Contextual transformer networks for visual recognition. *arXiv preprint arXiv:2107.12292*, 2021.

[58] Ziyu Li, Yuncong Yao, Zhibin Quan, Wankou Yang, and Jin Xie. Sienet: spatial information enhancement network for 3d object detection from point cloud. *arXiv preprint arXiv:2103.15396*, 2021.

[59] Alejandro Lindo, Enrique Garcia, Jesus Urena, Maria del Carmen, and Alvaro Hernandez. Multiband waveform design for an ultrasonic indoor positioning system. *IEEE Sensors Journal*, 15(12):7190–7199, 2015.

[60] Kang Ling, Haipeng Dai, Yuntang Liu, and Alex X Liu. Ultragesture: Fine-grained gesture sensing and recognition. *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9, 2018.

[61] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[62] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

[63] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[64] Yunxiang Liu and Jinpeng Ren. Laser point cloud road 3d target detection based on deep learning. In *2021 2nd International Conference on Big Data and Informatization Education (ICBDIE)*, pages 78–81. IEEE, 2021.

[65] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3164–3173, 2021.

[66] J. M. Martın, A. R. Jiménez, F. Seco, L. Calderón, Jose L. Pons, and R. Ceres. Estimating the 3d-position from time delay data of us-waves: experimental analysis and a new processing algorithm. *Sensors and Actuators A: Physical*, 101(3):311–321, 2002.

[67] Robert Mecklenburg. *Managing Projects with GNU Make, 3rd edition.* O'Reilly Media, Inc, 2004. ISBN 0596006101.

[68] Carlos Medina, José Carlos Segura, and Angel De la Torre. Ultrasound indoor positioning system based on a low-power wireless sensor network providing sub-centimeter accuracy. *Sensors*, 13(3):3501–3526, 2013.

[69] Javier Mendez, Manuel Cuéllar, and Diego Morales. Lidar-radar robust multi-level sensor fusion for target detection at the network edge. *Elsevier Measurements - Under review*, 012 2021.

[70] Javier Mendez, Miguel Molina, Noel Rodriguez, Manuel P Cuellar, and Diego P Morales. Camera-lidar multi-level sensor fusion for target detection at the network edge. *Sensors*, 21(12):3992, 2021.

[71] Javier Mendez, Stephan Schoenfeldt, Xinyi Tang, Jakob Valtl, MP Cuellar, and Diego P Morales. Automatic label creation framework for fmcw radar images using camera data. *IEEE Access*, 2021.

[72] Javier Mendez, Kay Bierzynski, Manuel Cuéllar, and Diego Morales. Edge intelligence: Concepts, architectures, applications and future directions. *ACM Transactions on Embedded Computing Systems*, 01 2022. doi: 10.1145/3486674.

[73] Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory bfgs. *The Journal of Machine Learning Research*, 16 (1):3151–3181, 2015.

[74] José L. Morales. A numerical study of limited memory bfgs methods. *Applied Mathematics Letters*, 15(4):481–487, 2002.

[75] Felix Nobis, Maximilian Geisslinger, Markus Weber, Johannes Betz, and Markus Lienkamp. A deep learning-based radar and camera sensor fusion architecture for object detection. In *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–7. IEEE, 2019.

[76] Abdelmoumen Norrdine. An algebraic solution to the multilateration problem. 1315, 2012.

[77] Mahdi Panahi, Nitheshnirmal Sadhasivam, Hamid Reza Pourghasemi, Fatemeh Rezaie, and Saro Lee. Spatial prediction of groundwater potential mapping based on convolutional neural network (cnn) and support vector regression (svr). *Journal of Hydrology*, 588:125033, 2020.

[78] Dinesh Dash Partha Pratim Ray and Debashis. Edge computing for Internet of Things: A survey, e-healthcare case study and future direction. *Network and Computer Applications*, 140:1–22, 2019.

[79] Matti Pastell, Lilli Frondelius, Mikko Järvinen, and Juha Backman. Filtering methods to improve the accuracy of indoor positioning data for dairy cows. *Biosystems Engineering*, 169:22–31, 2018.

[80] Marco Patanè, Beatrice Rossi, Pasqualina Fragneto, and Andrea Fusiello. Wireless sensor networks localization with outliers and structured missing data. pages 1–7, 2017.

[81] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11, 2013.

[82] Marius-Constantin Popescu, Valentina E. Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. *Multilayer Perceptron: Architecture Optimization and Training with Mixed Activation Functions*. BDCA'17. Association for Computing Machinery, New York, NY, USA, 2017. ISBN 9781450348522. doi: 10.1145/3090354.3090427. URL `https://doi.org/10.1145/3090354.3090427`.

[83] Sorin C Popescu and Kaiguang Zhao. A voxel-based lidar method for estimating crown base height for deciduous and pine trees. *Remote sensing of environment*, 112(3):767–781, 2008.

[84] Antonin Povolny, Hiroshige Kikura, and Tomonori Ihara. Ultrasound pulse-echo coupled with a tracking technique for simultaneous measurement of multiple bubbles. *Sensors*, 18(5):1327, 2018.

[85] Veronika Putz, Julia Mayer, Harald Fenzl, Richard Schmidt, Markus Pichler-Scheder, and Christian Kastl. Cyber–Physical Mobile Arm Gesture Recognition using Ultrasound and Motion Data. 2020.

[86] J. Cao Q. Zhang Y. Li W. Shi and L. Xu. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[87] Jun Qi and Guo-Ping Liu. A robust high-accuracy ultrasound indoor positioning system based on a wireless sensor network. *Sensors*, 17 (11):2554, 2017.

[88] Rui Qian, Xin Lai, and Xirong Li. Boundary-aware 3d object detection from point clouds. *ArXiv*, abs/2104.10330, 2021.

[89] Yang Qifan, Tang Hao, Zhao Xuebing, Li Yin, and Zhang Sanfeng. Dolphin: Ultrasonic-based gesture recognition on smartphone platform. *2014 IEEE 17th International Conference on Computational Science and Engineering*, pages 1461–1468, 2014.

[90] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[91] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[92] Sara Casado-Vara Roberto Sittón-Candanedo Inés Alonso Ricardo Corchado Rodríguez, Juan Rodríguez. A Review of Edge Computing Reference Architectures and a new Global Edge Proposal. *Future Generation Computer Systems*, 2019.

[93] Wenjie Ruan, Quan Z Sheng, Lei Yang, Tao Gu, Peipei Xu, and Longfei Shangguan. Audiogest: enabling fine-grained hand gesture detection by decoding echo signal. *Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing*, pages 474–485, 2016.

[94] Daniel Ruiz, Jesús Ureña, Juan C. García, Carmen Pérez, José M. Villadangos, and Enrique García. Efficient trilateration algorithm using time differences of arrival. *Sensors and Actuators A: Physical*, 193: 220–232, 2013.

[95] T. A. Mohammed S. Albawi and S. Al-Zawi. Understanding of a convolutional neural network. *International Conference on Engineering and Technology (ICET), Antalya, 2017*, pages 1–6, 2017.

[96] Nasir Saeed, Haewoon Nam, Mian I. U.l Haq, and Dost B. Muhammad Saqib. A survey on multidimensional scaling. *ACM Computing Surveys (CSUR)*, 51(3):1–25, 2018.

[97] Borja Saez-Mingorance, Antonio Escobar-Molero, Javier Mendez-Gomez, Encarnacion Castillo-Morales, and Diego P Morales-Santos. Object positioning algorithm based on multidimensional scaling and optimization for synthetic gesture data generation. *Sensors*, 21(17): 5923, 2021.

[98] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674, 1991.

[99] Yu Sang, Laixi Shi, and Yimin Liu. Micro hand gesture recognition system using ultrasonic active sensing. *IEEE Access*, 6:49339–49347, 2018.

[100] M. Satyanarayanan. The emergence of edge computing. *Computer*, 50 (1):30–39, 2017.

[101] R. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34(3):276–280, 1986. doi: 10.1109/TAP.1986.1143830.

[102] Babak Shahian Jahromi, Theja Tulabandhula, and Sabri Cetin. Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles. *Sensors*, 19(20), 2019. ISSN 1424-8220. doi: 10.3390/s19204357. URL `https://www.mdpi.com/1424-8220/19/20/4357`.

[103] Maurice G Silk. *Ultrasonic transducers for nondestructive testing*. Adam Hilger Ltd., Accord, MA, 1984.

[104] Deepti Singhal and Rama M. Garimella. Simple median based information fusion in wireless sensor network. pages 1–7, 2012.

[105] Adam Smith, Hari Balakrishnan, Michel Goraczko, and Nissanka Priyantha. Tracking moving devices with the cricket location system. pages 190–202, 2004.

[106] Richard M. Stallman. *GNU Emacs Manual for Version 22, 16th Edition*. Free Software Foundation, 2007.

[107] Alan N Steinberg and Christopher L Bowman. Rethinking the jdl data fusion levels. *Nssdf Jhapl*, 38:39, 2004.

[108] Inc. ThoughtWorks. Cruise control, 2001.

[109] Michael W. Trosset and Carey E. Priebe. The out-of-sample problem for classical multidimensional scaling. *Computational statistics & data analysis*, 52(10):4635–4642, 2008.

[110] Jennifer Vesperman. *Essential CVS*. O'Reilly, 2003. ISBN 0-596-00459-1.

[111] General Vision. Neuroshield. (accessed: 20.04.2020). URL `https://www.general-vision.com/hardware/neuroshield/`.

[112] S. Elanayar V.T. and Y. C. Shin. Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems. *IEEE Transactions on Neural Networks*, 5(4):594–603, 1994.

[113] Miao Wang and Yi-Hsing Tseng. Incremental segmentation of lidar point clouds with an octree-structured voxel space. *The Photogrammetric Record*, 26(133):32–57, 2011.

[114] Shuxia Wang. Wireless network indoor positioning method using nonmetric multidimensional scaling and rssi in the internet of things environment. *Mathematical Problems in Engineering*, 2020, 2020.

[115] Tai Wang, ZHU Xinge, Jiangmiao Pang, and Dahua Lin. Probabilistic and geometric depth: Detecting objects in perspective. In *Conference on Robot Learning*, pages 1475–1485. PMLR, 2022.

[116] Wei Wang, Yujing Yang, Xin Wang, Weizheng Wang, and Ji Li. Development of convolutional neural network and its application in image classification: a survey. *Optical Engineering*, 58(4):040901, 2019.

[117] Zhangjing Wang, Yu Wu, and Qingqing Niu. Multi-sensor fusion in automated driving: A survey. *Ieee Access*, 8:2847–2868, 2019.

[118] Thomas M Ward, Pietro Mascagni, Yutong Ban, Guy Rosman, Nicolas Padoy, Ozanan Meireles, and Daniel A Hashimoto. Computer vision in surgery. *Surgery*, 169(5):1253–1256, 2021.

[119] Wikipedia. LaTeX.

[120] Christopher R. Wren, Ali Azarbayejani, Trevor Darrell, and Alex P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):780–785, 1997.

[121] Ling Xiao, Renfa Li, and Juan Luo. Sensor localization based on non-metric multidimensional scaling. *STRESS*, 2(1), 2006.

[122] Shaoqing Xu, Dingfu Zhou, Jin Fang, Junbo Yin, Bin Zhou, and Liangjun Zhang. Fusionpainting: Multimodal fusion with adaptive attention for 3d object detection. *ArXiv*, abs/2106.12449, 2021.

[123] Samir S Yadav and Shivajirao M Jadhav. Deep convolutional neural network based medical image classification for disease diagnosis. *Journal of Big Data*, 6(1):1–18, 2019.

[124] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1631–1640, 2020.

[125] Costas Yiallourides and Pablo P. Parada. Low power ultrasonic gesture recognition for mobile handsets. pages 2697–2701, 2019. doi: 10.1109/ICASSP.2019.8683781.

[126] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021.

[127] Qinglin Zeng, Zheng Kuang, Shuaibing Wu, and Jun Yang. A method of ultrasonic finger gesture recognition based on the micro-doppler effect. *Applied Sciences*, 9(11):2314, 2019.

[128] Zhuoran Zhao, Kamyar Mirzazad Barijough, and Andreas Gerstlauer. Deepthings: Distributed adaptive deep learning inference on resource-constrained iot edge clusters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2348–2359, 2018.

[129] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14494–14503, June 2021.

[130] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, Aug 2019. ISSN 1558-2256. doi: 10.1109/JPROC.2019.2918951.

[131] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019.

[132] Xinge Zhu, Yuexin Ma, Tai Wang, Yan Xu, Jianping Shi, and Dahua Lin. Ssn: Shape signature networks for multi-class object detection from point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 581–597. Springer, 2020.

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes