UNIVERSIDADE D
COIMBRA

Gledson Melotti

Reducing Overconfident Predictions in Multimodality Perception for Autonomous Driving

Doctoral thesis submited in partial fulfilment of the Doctoral Program in Electrical and Computer Engineering, area of expertise in Specialization in Automation and Robotics, supervised by Professor PhD Cristiano Premebida and Professor PhD Nuno Miguel Mendonça da Silva Gonçalves and presented to the Department of Electrical and Computer Engineering of the Faculty of Science and Technologic of the University of Coimbra.

September 2022

UNIVERSIDADE D
COIMBRA

Faculty of Science and Technology

Department of Electrical and Computer Engineering

Reducing Overconfident Predictions in Multimodality Perception for Autonomous Driving

Gledson Melotti

Thesis submitted to the Department of Electrical and Computer Engineering of the Faculty of Science and Technology of the University of Coimbra in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Supervisors: Professors Cristiano Premebida and Nuno Miguel Mendonça da Silva Gonçalves

Coimbra

September 2022

Acknowledgements

The presentation of this thesis was supported by my parents Edson Melotti and Lúcia Maria Gon Melotti, my sisters Monalisa Melotti and Margiana Melotti, who also encouraged me to develop myself personally, and professionally and I will be eternally grateful. I am also thankful to my friends Marcos Dias da Silva, Kelma Socorro Lopes de Matos, Vinicius Oliveira Casais, Luiza Vilas Boas Motta, Tales Argolo de Jesus, Cristina Sady Coelho da Rocha, Cleidson da Silva Oliveira, Patrícia Pereira Queiroz da Purificação, Erika Afonso Schmitz, Luciane Serrate Pacheco Bacheti, Nágila De Fátima Rabelo Moraes, Jardel Merlim Faria, Luís Carlos Artur da Silva Garrote, Alireza Asvadi, Eliete Rossmann Vorpagel, Jenniffer Miriã Vorpagel Parolli, Joana Girandelli, Érica Tatiane Soares Ciorici, Vasile Ciorici, Osvaldo Ferreira da Andrade Filho, Luciene de Souza Gomes, Luiz Felipe Magalhães Antunes de Almeida, Fabiana Bassani, Fernando Caixeta Lisboa, Daiane Godois Fritsch, Humberto Mendes Mazzini, Eduardo Mazoni Andrade Marçal Mendes, and all the people I made friends in Coimbra. In addition, I am grateful to my friends from Federal Institute of Espírito Santo-São Mateus-Brazil, and especially my friends from the Coordination of the Technical Course in Electrotechnical, for motivating me to pursue a doctorate degree.

I thank the supervisor Professor Ph.D. Cristiano Premebida for believing in me. He is able to share his knowledge, his constructive criticism, support me while solving issues, and especially encouraging me not to give upon my goals.

I also praise the supervisor Professor Ph.D. Nuno Miguel Mendonça da Silva Gonçalves, who gave me a vote of confidence and for being always willing to help.

I am grateful for my friends from the Institute of Systems and Robotics-Coimbra University who are always available to help me.

Thank you,
Gledson Melotti

Abstract

In the last recent years, machine learning techniques have occupied a great space in order to solve problems in the areas related to perception systems applied to autonomous driving and advanced driver-assistance systems, such as: road users detection, traffic signal recognition, road detection, multiple object tracking, lane detection, scene understanding. In this way, a large number of techniques have been developed to cope with problems belonging to sensory perception field. Currently, deep network is the state-of-the-art for object recognition, begin softmax and sigmoid functions as prediction layers. Such layers often produce overconfident predictions rather than proper probabilistic scores, which can thus harm the decision-making of "critical" perception systems applied in autonomous driving and robotics. Given this, we propose a probabilistic approach based on distributions calculated out of the logit layer scores of pre-trained networks which are then used to constitute new decision layers based on Maximum Likelihood (ML) and Maximum a-Posteriori (MAP) inference. We demonstrate that the hereafter called ML and MAP functions are more suitable for probabilistic interpretations than softmax and sigmoid-based predictions for object recognition, where our approach shows promising performance compared to the usual softmax and sigmoid functions, with the benefit of enabling interpretable probabilistic predictions. Another advantage of the approach introduced in this thesis is that the so-called ML and MAP functions can be implemented in existing trained networks, that is, the approach benefits from the output of the logit layer of pre-trained networks. Thus, there is no need to carry out a new training phase since the ML and MAP functions are used in the test/prediction phase. To validate our methodology, we explored distinct sensor modalities via RGB images and LiDARs ($3D$ point clouds, range-view and reflectance-view) data from the KITTI dataset. The range-view and reflectance-view modalities were obtained by projecting the range/reflectance data to the 2D image-plane and consequently upsampling the projected points. The results achieved by the proposed approach were presented considering the individual modalities and through the early and late fusion strategies.

Key words: Overconfident predictions; Deep neural network architectures; Perception systems; Autonomous driving systems; Object detection.

Resumo

Nos últimos anos, as técnicas de aprendizagem de máquina têm ocupado um grande espaço para solucionar problemas nas áreas relacionadas com sistemas de perceção aplicados à direção autónoma e sistemas avançados de assistência ao motorista, tais como: detecção de utilizadores de vias, reconhecimento de sinais de trânsito, deteção de vias, rastreamento de múltiplos objetos, deteção de pista, compreensão de cena. Desta forma, um grande número de técnicas tem sido desenvolvido para lidar com problemas pertencentes ao campo da perceção sensorial. Atualmente, deep network é o estado da arte para reconhecimento de objetos, iniciando funções softmax e sigmoid como camadas de previsão. Essas camadas geralmente produzem previsões de excesso de confiança em vez de pontuações probabilísticas adequadas, o que pode prejudicar a tomada de decisões de sistemas de perceção "críticos" aplicados em direção autónoma e robótica. Diante disso, propomos uma abordagem probabilística baseada em distribuições calculadas a partir dos scores (pontuações da saída) da camada logit de redes pré-treinadas que são então utilizadas para constituir novas camadas de decisão baseadas na inferência de Máxima Verossimilhança (ML) e Máxima a-Posteriori (MAP). Demonstramos que as funções ML e MAP daqui em diante são mais adequadas para interpretações probabilísticas do que previsões baseadas em softmax e sigmoid para reconhecimento de objetos, onde a nossa abordagem mostra um desempenho promissor em comparação com as funções usuais de softmax e sigmoid, com o benefício de permitir previsões probabilísticas interpretáveis. Outra vantagem da abordagem apresentada nesta tese é que as chamadas funções ML e MAP podem ser implementadas em redes já treinadas, ou seja, a abordagem beneficia da saída da camada logit de redes pré-treinadas. Assim, não há necessidade de realizar uma nova fase de treino, uma vez que as funções ML e MAP são utilizadas na fase de teste/previsão. Para validar a nossa metodologia, exploramos modalidades distintas de sensores por meio de imagens RGB e dados LiDARs ($3D$ point clouds, range-view e reflectance-view) do conjunto de dados KITTI. As modalidades range-view e reflectance-view foram obtidas projetando os dados de alcance/refletância para o plano de imagem $2D$ e, consequentemente,

upsampling dos pontos projetados. Os resultados alcançados pela abordagem proposta foram apresentados considerando as modalidades individuais e por meio das estratégias de fusão “early” e “late”.

Palavras-chave: Predições com excesso de confiança; Arquiteturas de redes neuronais profundas; Sistemas de perceção; Sistemas de condução autónoma; Reconhecimento de objetos.

Table of contents

List of Acronyms

ADAS	Advanced Driver Assistance System
ADS	Autonomous Driving Systems
AHS	Autonomous Highway System
AI	Artificial intelligence
AV/IV	Autonomous/Intelligent Vehicles
BF	Bilateral Filter
BNN	Bayesian Neural Network
CNN	Convolutional Neural Network
CS	Convoy Systems
DA	Driving Assistant
DM	Depth Map
ED	Eigenvalues Based Descriptors
FPFH	Fast Point Feature Histogram
GPS	Global Positioning System
GSH	Global Structure Histograms
HOG	Histogram of Oriented Gradients
HW	Haar Wavelets
ILSVRC	ImageNet Large Scale Visual Recognition Challenge

IMU Inertial Measurement Unit

KNN K-Nearest Neighbors

LBP Local Binary Patterns

LFSH Local Feature Statistics Histograms

LGP Local Gradient Patterns

LiDAR Light Detection and Ranging

LBP Local Binary Pattern

LRF Local Receptive Fields

MAP Maximum a-Posteriori

ML Maximum Likelihood

MLE Maximum Likelihood Estimation

MLear Machine Learning

MLP Multilayer Perceptron (MLP)

NN Neural Networks

PC Point Cloud

PCA Principal Component Analysis

PPF Point Pair Feature

RADAR Radio Detection and Ranging

RM Reflectance Map

SDG Sustainable Development Goals

SI Spin Image

SIFT Scale-Invariant Feature Transform

SIPF Scale Invariant Point Feature

SVM Support Vector Machine

SURF Speeded-up Robust Features

VFH Viewpoint Feature Histogram

List of figures

List of tables

Chapter 1

Introduction

Contents

1.1 Brief History of Autonomous Driving

In recent years, artificial intelligence and machine learning have achieved remarkable results in perception systems applied to intelligent vehicles (IV) and/or autonomous vehicles (AV) domains [49, 25, 177, 218], which allows practical applications for the automotive industry and beyond. Autonomous driving, as known today, dates back to the 80's, namely 1986, when the research groups of Ernst Dickmanns, at the Bundeswehr University Munich-Germany, and the Navlab at the Carnegie Mellon University-USA presented the first vehicles capable of autonomous driving. Besides being an actual vehicle, Navlab was also a 'living-lab' navigation system [98, 177, 218].

Some of the fundamental pillars of autonomous driving research, however, arose well before the 1980s [98, 134, 245, 145]. In 1925, the inventor Francis P. Houdina presented a prototype of driverless vehicle known as American Wonder[1] [98, 145]. Decades later, General Motors presented some prototypes in 1939 and 1956 in addition to the ones from Radio Company of America Labs in 1960, and so the Citroen DS 19 and Cabinentaxi of Demag/MBB in 1970 [98, 114]. Since then, research activities involving autonomous driving have reached remarkable achievements, for example: security drivers to monitor car operation, vehicle driving autonomously, and passenger transport without a driver [49, 217, 52, 45, 26, 58, 257, 80, 222], including the Drive.ai[2], and Waymo (that has already been deployed as a driverless taxi[3]).

Some examples of autonomous vehicles are shown in Fig. 1.1. Fig. 1.1a illustrates the Navlab's self-driving car [218] proposed in 1986. The Google's self-driving car (Toyota Priuses[4], in Fig. 1.1b, tested on highways around the years 2010 and 2011) is equipped with a Velodyne 64-beam laser (LiDAR-Light Detection and Ranging) as one of the main sensors in addition to the global positioning system (GPS), RADAR (Radio Detection and Ranging), and wheel encoder[5] [80]. Fig. 1.1c shows the Boss car[6] that won the 2007 DARPA Urban Challenge. This car contains several sensors: LiDARs, GPS, Inertial Measurement Units (IMU), RADARs and cameras. Lastly, the most recent Waymo car (tested on highways around the years 2020 and 2021), developed by Google, is shown in Fig. 1.1d.

[1] http://content.time.com/time/subscriber/article/0,33009,720720,00.html

[2] Drive.ai was acquired by Apple Inc in 2019.

[3] https://waymo.com/waymo-one, Accessed on 07/22/2021.

[4] https://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works, Accessed on 07/22/2021.

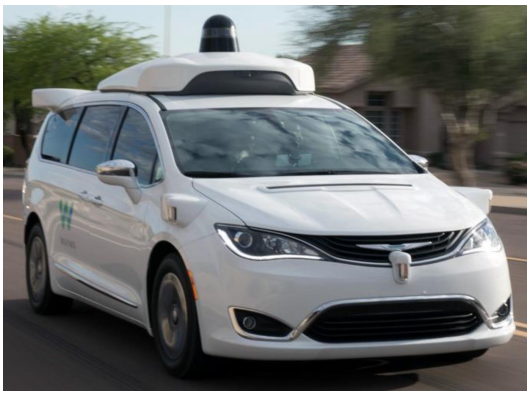
[5] The encoder measures the car movement in order to obtain a better estimation of its position.

[6] The name Boss is in honor of Charles Boss Kettering, the researcher who developed all-electric starters for cars, Freon refrigerant and the incubator for premature babies.

(a) The Navlab's self-driving car [218].

(b) The Google's self-driving car [80].

(c) The Boss's autonomous vehicle [222].



(d) The Waymo's self-driving car [232].

Fig. 1.1 Landmark examples of autonomous driving vehicles, from 1986 to 2021.

It is evident that the development of autonomous vehicles and the circulation of such vehicles on highways obey specific and existing regulations of each region[7], as the legislation proposed by the National Highway Traffic Safety Administration-USA[8] since 1996 [208], as well as the rules defined by Society of Automotive Engineers (SAE) in 2014, considering 6 levels of autonomy[9] [98]:

- Level 0 (no automation): the driver is responsible for all the driving, additionally, monitoring all the environment. For example: most of the cars produced in the previous decades.

[7] https://path.berkeley.edu/sites/default/files/ahs-milestone_2_report_task-c21.pdf. Accessed on 01/29/2021.

[8] https://www.federalregister.gov/agencies/national-highway-traffic-safety-administration. Accessed on 01/29/2021.

[9] www.sae.org/standards/content/j3016_201806. Accessed on 01/29/2021.

- Level 1 (driving assistant-DA): the vehicle system provides information to the driver in order to avoid accidents. For example: parking sensors and automatic emergency braking.

- Level 2 (partial automation): the driving system includes speed, direct-drive mechanism, and brake. However, the driver is responsible for monitoring the environment. For example: Tesla Autopilot, Audi Traffic Jam, and Volvo Pilot Assistant.

- Level 3 (conditional automation): the driving system may be able to participate in the driving tasks and also monitors the environment. Nevertheless, the driver must be attentive and assume when requested. Level 3 requires ideal roads and limited access. For instance: Audi AL8, Mercedez-Benz S-Class, and Honda Legend.

- Level 4 (high automation): the system allows the vehicle to perform in driving activities, such as accelerating, braking, monitoring the environment, operating actions when detecting certain on-road obstacles, as well as changing lanes. Nonetheless, such level of autonomy requires highways with ideal conditions. For example: Google's Waymo project in the U.S.

- Level 5 (full automation): this level presents, in driving system all the characteristics of level 4, as well as the ability to decision-making in dynamic environments (traffic jams) and the system does not actually require human intervention. For example: Audi's AI:CON and Google's Waymo projects under specific conditions (self-parking).

In addition to the definitions related to the autonomy levels, the IV/AV can be characterized according to the safety control functions (acceleration and braking) with regard to the driver and the driving system [51]:

- DA - Driving Assistant: it provides additional and complementary information to the driver to improving driving performance in terms of safety and efficiency.

- CS - Convoy Systems: the goal is the development of automated delivery convoys (vehicles "in line") in which the main vehicle is driven by a human operator, but the subsequent vehicles on the convoy are autonomously operated.

- ADS - Autonomous Driving System: the ultimate aim is replacing the human driver of the driving tasks, and then making the individual vehicle driverless or autonomous.
- AHS - Autonomous Highway System: the key principle that governs the AHS is by considering the entire highway as a unified-system which autonomously controls groups/fleet of vehicles.

On the other hand, a widely used concept in the studies of intelligent and/or automated vehicles includes, to some extent, the previous categories [98, 247, 43, 99, 41, 144]:

- ADAS - Advanced Driver Assistance System: a system that supports monitoring and alerting/warning, as well as breaking tasks, based on automated functionalities including advanced sensors and artificial intelligence/machine learning (AI/MLear) techniques.

The development of automated driving, whether or not equipped with one or more ADAS, DA, CS, ADS, AHS systems, typically comprises a combination of four key modules (or systems): **perception** (to capture data, including sensor-fusion, and interpret environment/surrounding conditions), **localization** (to define its own position and state), **planning** (to set the trajectory, plan the mission), and **control** (to ensure proper decisions/actions and safety). Among the key modules, this thesis deals with some of the machine learning components belonging to a perception system and concentrates on the classification scores (or top-label scores) of detected objects - which can be related to the confidence level - delivered by an object recognition module. Note that many studies of perception systems involving object recognition algorithms do not always provide the level of confidence about the object classification predictions *i.e.*, the score values do not reflect the certainty or uncertainty of classification, especially when the object is misclassified. Therefore, perception systems should not (ideally) make confusion regarding classification-score of a detected object with classification certainty-level because *in extremis* a classification error can lead to a fatality. Wherefore, object recognition inserted in the perception system module needs special attention.

1.2 Perception System

Autonomous driving encompasses several components and modules. It consists of connecting a “multitude” of hardware and software technologies. In terms of software,

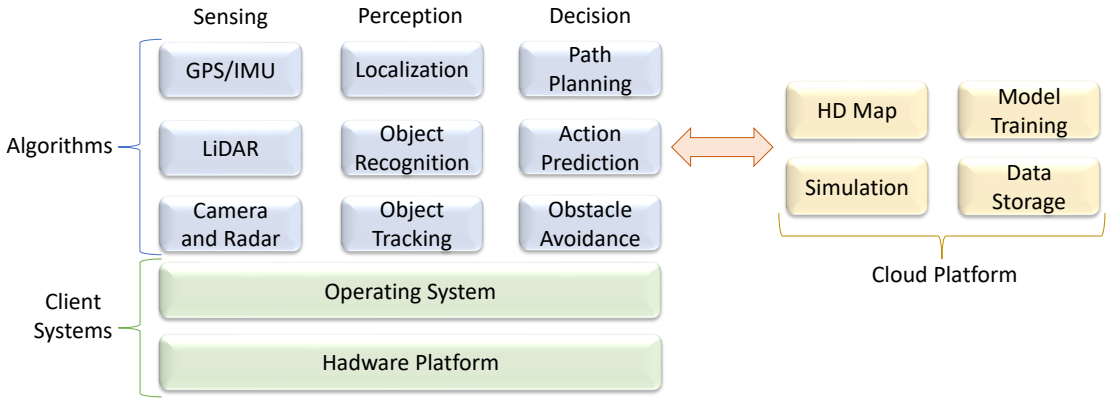


Fig. 1.2 Autonomous driving system architecture.

generally the architecture of an autonomous driving system can be represented by modules to process the data, the client system, and the cloud platform, according to Fig. 1.2 [134]. The software-modules decode information from the sensor data in order to comprehend the environment around the autonomous vehicle then, enabling decision-making about current and future events. The client system is responsible for ensuring that the information's processing-pipeline is shorter than the sensors' information captured in time. Meanwhile, the cloud platform enables offline computing of data storage, as well as training/updating software, updating data, such as high-definition maps (HD maps[10]), and finally improving decision models.

Sensor wise, autonomous vehicles typically comprise several sensors as follows: GPS, IMU, LiDAR, cameras and RADAR. Both GPS and IMU contribute to localize the autonomous vehicles themselves (inertial and global position estimation) and do not contribute directly on perception. Cameras and LiDARs, the standard perception sensors, can provide information about which objects are in the environment, as well as their position. Finally, RADAR provide information about object distances and speed as well. The data obtained from the sensors are processed by a perception system in order to understand the environment surrounding the vehicle. Simultaneously, it is able to decode information such as localization, detection, tracking objects, among others. As the information is gathered, the perception system makes a decision in "real time", taking into account path planning, action prediction, and obstacle avoidance [98, 134, 245, 145].

[10]Briefly, the pipeline involves processing raw data, depth information, $2D$ reflectance map, labels and finally the high-definition map (information about roads and structures, and it allows an accurate location of the vehicle).

All information obtained from the sensors should be ideally processed in real time, that is to say, the system of an autonomous vehicle should guarantee the processing in a shorter time than the update rates of the information provided by the sensors, and possibly guaranteeing the robustness during information processing (ability to recover from possible failures). The processing speed and the robustness of the system depend on the operating system and the components contained in the hardware platform (processors, GPUs, and power sources). In addition to the information from sensors and data processing, autonomous vehicles rely on cloud platforms keeping themselves updated, which include the use of new algorithms after simulations, production of high-definition maps and training models (deep learning). In fact, cloud platforms allow autonomous vehicles to process large amounts of sensor data and make real-time decisions [98, 134, 245, 145].

Regardless the number of sensors that contributes to the autonomous driving and decision-making, the development of a sensory perception system, including fusion strategies based on the information captured by the sensors seems to be essential to ensure the safety among road users and thus reduce the number of accidents *i.e.*, avoiding mistakes (fatalities and injuries) that might be caused by a lack of human perception while driving. Regarding safety, the modern deep learning based perception algorithms still tend to generalize in overconfident way on out-of-training-distribution test data. Besides, deep models are prone to providing overconfident results even for miss-classified objects consequently, which may have critical implications in safety [146, 81, 63].

1.3 Motivation and the Relevant Aspects of Perception Systems for IV/AV

According to WHO Global status report[11], 392.904 car occupants, 40.646 cyclists, 379.356 motorcycles, and 311.614 pedestrians died in 2020. Approximately more than 1,3 million people die every year on road traffic and millions live with several health problems as consequence of the accidents. Fig.1.3 illustrates the distribution of deaths among road users [234]. Note that the casualties by car occupants are higher in almost all regions. In fact, related deaths by occupants of motor vehicles (cars and motorcycles) are higher in all regions. The death rate among the most vulnerable road

[11] https://extranet.who.int/roadsafety/death-on-the-roads/#ticker/pedestrians. Accessed on 02/01/2021.

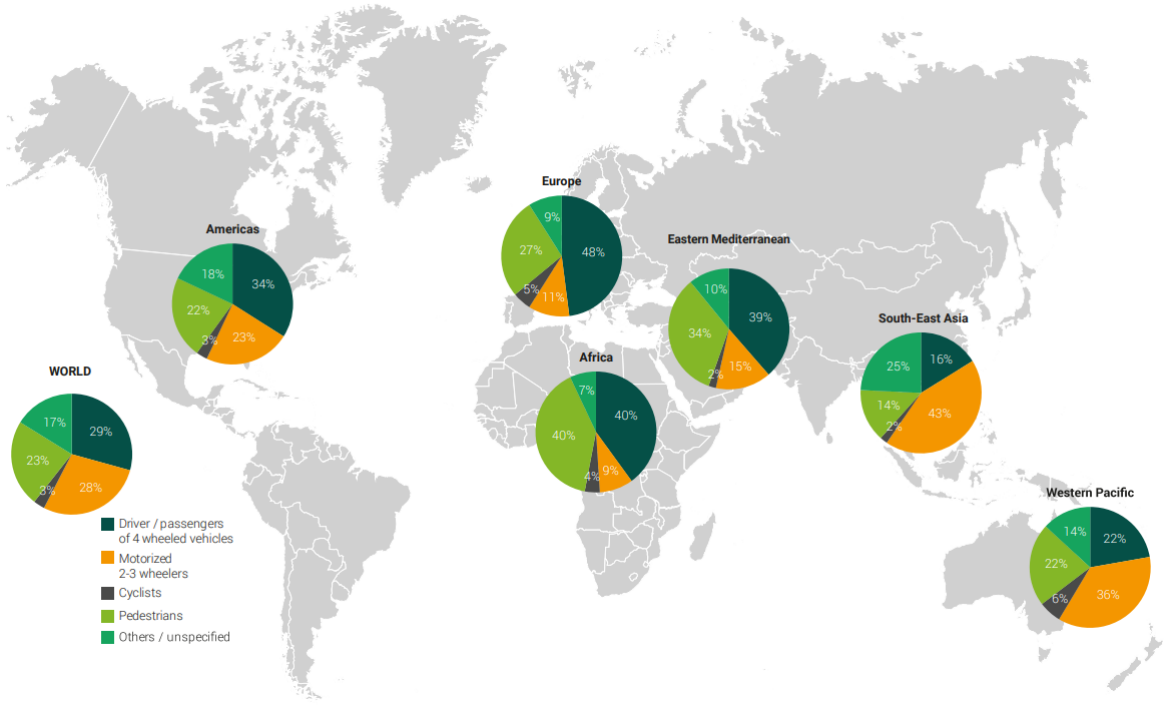


Fig. 1.3 Distribution of road traffic deaths by road user. Source [234].

users (pedestrian, cyclist and motorcyclists) is quite aggravating, exceeding 50% of road fatal losses.

In the last years, 123 countries have adopted regulations and strategies to reduce road accidents, such as encouraging the seatbelts use, limiting the concentration of alcohol per liter of blood, speed control rules, the use of helmets by motorcyclists and regulations on children safety in cars [234]. Nevertheless, although it reduces the damage caused by road accidents by applying laws or educational campaigns, such restrictions do not entirely solve the problem. Thus, perception systems and ADAS technologies with applications in AV/IV are expected to be an important step to avoid road accidents. Also, such technologies are useful to assist disabled people using these vehicles (by means of intelligent mobility technologies), to improve road traffics by reducing distances between vehicles and, lastly, to help drivers to do other activities during a certain route.

Although valuable and promising to undertake mitigation actions to increase safety, a sensory perception is not simple. They may suffer from lighting and occlusion, false positives and missing detection, poor probabilistic interpretation, also lack the detection of "non-rigid" entities *e.g.*, they may still suffer from articulation of the pedestrians' body [142, 4, 53, 160]. Therefore, several studies involving the areas of

Fig. 1.4 Example of an image containing objects (pedestrians and car categories/classes) captured from an RGB camera. The first row shows labels in $2D$ bounding boxes, while the second row presents labels in $3D$ bounding boxes. Image obtained from the Object Detection Evaluation of the KITTI dataset [69].

knowledge of electrical, mechatronics, mechanics and computing engineering, statistics, and MLear/AI techniques have contributed to vision and LiDAR-based perception continous achievements [98, 139, 19, 20, 60].

An example of sensory perception is illustrated in Fig. 1.4, that shows information of objects such as pedestrians and cars, obtained from the camera sensor. Another example of sensory perception is shown in the Fig. 1.5 from the LiDAR sensor, which provides $3D$ information (point cloud) containing detected cars. The two examples also illustrate the $2D$ and $3D$ bounding box of objects.

Image processing has achieved significant results through machine learning and deep learning algorithms. The algorithms of convolutional neural networks are currently, not only, responsible for most of the advances in pattern recognition to autonomous driving, but mainly for processing of $2D$ images [221, 241, 251, 253, 86, 203, 212, 249, 113, 123], as well as the $3D$ data processing [82, 122, 181, 184, 182, 110, 183].

On the one hand, RGB camera has the ability to capture information from objects (pedestrians, cars, cyclist, trucks, and vans), such as shape (contour patterns), color

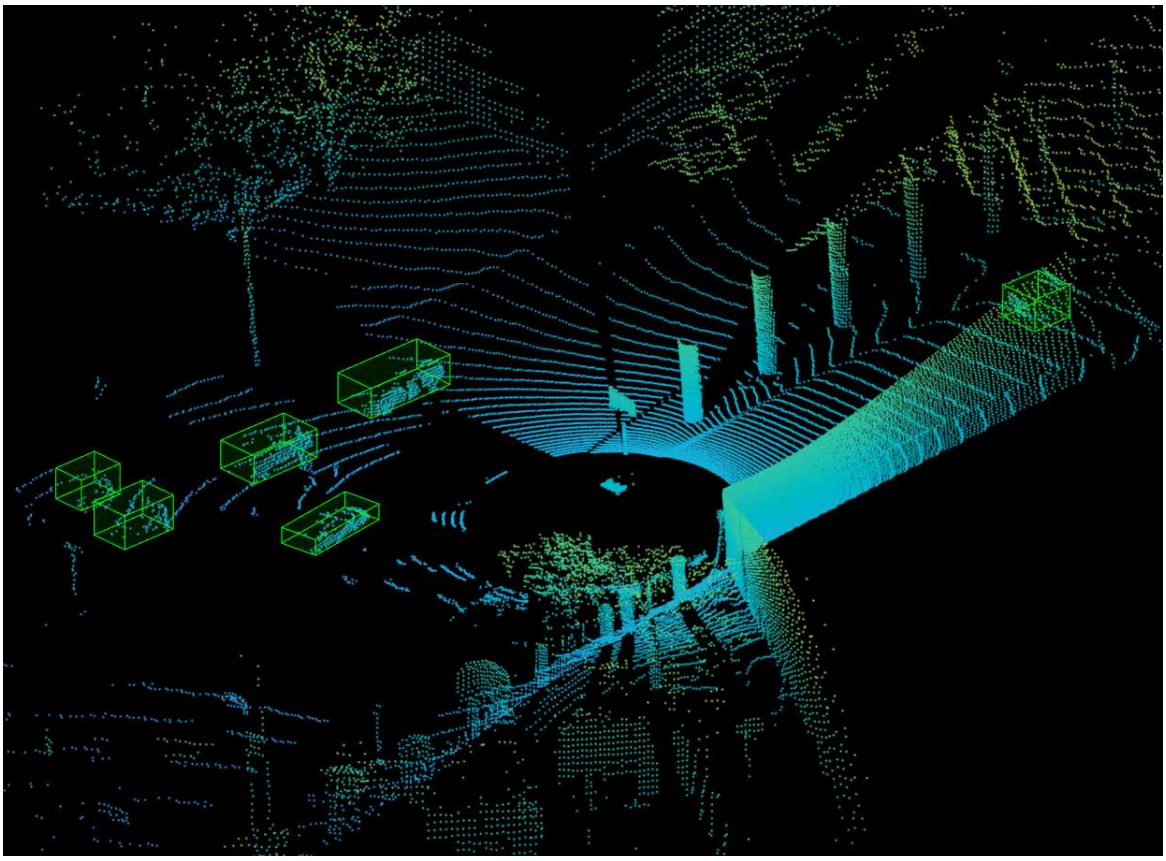


Fig. 1.5 Example of $3D$ point cloud captured by the Velodyne HDL-64E. $3D$ Point clouds were obtained from the Object Detection Evaluation of the KITTI dataset [69].

(pixel values) and texture (pixel arrangement structures). On the other hand, it does not have the ability to provide directly information regarding object size/height or depth (these measurements have to be estimated). Besides that, the cameras have sensitivity to light variation, and some cameras need extra light to capture images at night [8, 155].

In a different way, LiDAR sensors are capable of operating at night with no need of extra light-source. The advantage of such sensors is the ability to estimate $3D$ information of the environment with a 360 degree horizontal field view, including depth measurements and reflectance (intensity) of the detected objects, and such sensors can still capture millions of points per second. The disadvantages of LiDAR sensors are the limited range, reduced performance in snow, fog, rain and dust conditions [89, 59]. Additionally, the number of points per object can be different at the same time that such objects are captured, as well as the information can contain missing data (partially occluded objects), noise (outlier-random position in space), likewise they can

be corrupted by information from reflective and "transparent" objects [98, 18, 7, 8, 155, 187].

RADAR sensors (radio waves instead of light) are also effective for detecting object distances and speeds, in addition to the cameras and LiDARs thus, they can be complementary for detecting objects. RADARs cannot differentiate objects that are very close to each other but, some RADAR technologies have a wider range compared to LiDARs. However, RADARs are less accurate than LiDARs as they lose the target object reference in curves.

Still, many IV/AV systems contain ultrasonic sensors for short distance (parking assistant), infrared sensors (infrared spectrum) to identify objects in environments with low light conditions and odometry sensors indicating the distance traveled by the vehicle, taking wheels rotation into account. GPS, IMU, cameras, LiDARs, RADARs, infrared, odometry, ultrasonic sensors have advantages and disadvantages. The fusion of information from different sensors is important to ensure the correct detection of objects. The data fusion approaches depend not only on the quality of the sensor data but also on proper formulations of the problem and the implemented algorithms [98, 134].

Generally, pattern recognition algorithms, using machine learning or deep learning, provide a result (score) associated to the detection or classification of an entity (or object) without taking into account the model uncertainty during training and/or at the test phase *i.e.*, "how to guarantee a probabilistic interpretation and how reliable are these predictions?" is a pertinent question in this context. In plain words, this question can be answered by computing the uncertainties from the network weights or the predicted values. To obtain models uncertainties are important to increasing confidence in order to allow safer decision-making, especially considering that this decisions are managed by autonomous systems which may pose threat to people's lives. Thus, the fact is that adapting the model uncertainties in decision-making helps to mitigate unpredictable behavior.

The importance of obtaining uncertainty in the prediction might be better understood as follows: consider six networks trained to classify three classes: car, cyclist and pedestrian. Thus, the scores from the predictive values for each object are always referred to the training classes through a prediction layer, such as the softmax layer, according to Table 1.1, considering different types of neural network architectures[12]. The results of that table were quite satisfactory, since all objects of the test dataset

[12]We are not concerned with presenting the formalisms of network architectures, in addition to not presenting the advantages and disadvantages of each network.

Table 1.1 Classification results by different models.

Model	Car	Cyclist	Pedestrian	Average
LeNet [124]	99.17	89.08	93.79	94.02
AlexNet [113]	99.42	91.41	96.46	95.75
Inception V3 [213]	99.47	91.93	96.29	95.89
EfficientNetB1 [215]	99.84	97.43	98.74	98.67
ViT [47]	99.46	93.56	96.37	96.46
MLP Mixer [220]	98.98	87.47	92.42	92.96

belong to the classes used in the training. However, what happen when the network classifies an object belonging to a fourth class, similarly to the trees class? In other words, object of out-of-distribution test data (unseen/non-trained), it is to say that the fourth class was not used during training. The answer to the previous question can be obtained by analyzing Fig. 1.6. The trees class object was classified as belonging to one of the three classes used in the training and with a high score value. In this case, the results were greater than 90% (value very close to one) *i.e.*, result with extremist prediction (overconfident result). Others examples of overconfident are possible to be analyzed by the histograms, Fig. 1.7, which show results of predictions from an unseen class (person sitting, tree, pole) and separated by the training classes *i.e.*, the number of unseen objects that were classified as cars, cyclists and pedestrians (from left to right).

Overconfident results can be detrimental to draw a conclusion, particularly objects from out-of-distribution test data which appear in the perception system environment. It shows that the predicted value is not adequate to interpret the model's confidence.

A considerable complexity in perception systems is in defining the quantity or which sensors should be used, as well as the mathematical formulations and algorithms in order to convert the information captured into an information which might possibly be understood by the control systems and actuators of AV/IV. For this reason, the development of perception systems is such a quite challenging assignment. New algorithms or techniques should be developed to avoid possible errors while making predictions of pattern recognition systems. Therefore, taking into account the relevance of reducing road accidents and avoiding road user injuries and deaths, this thesis contributes to AV/IV advances of sensory perception systems, through studies that involve multi-sensor-modality datasets, focusing on reducing overconfident predictions for object recognition.

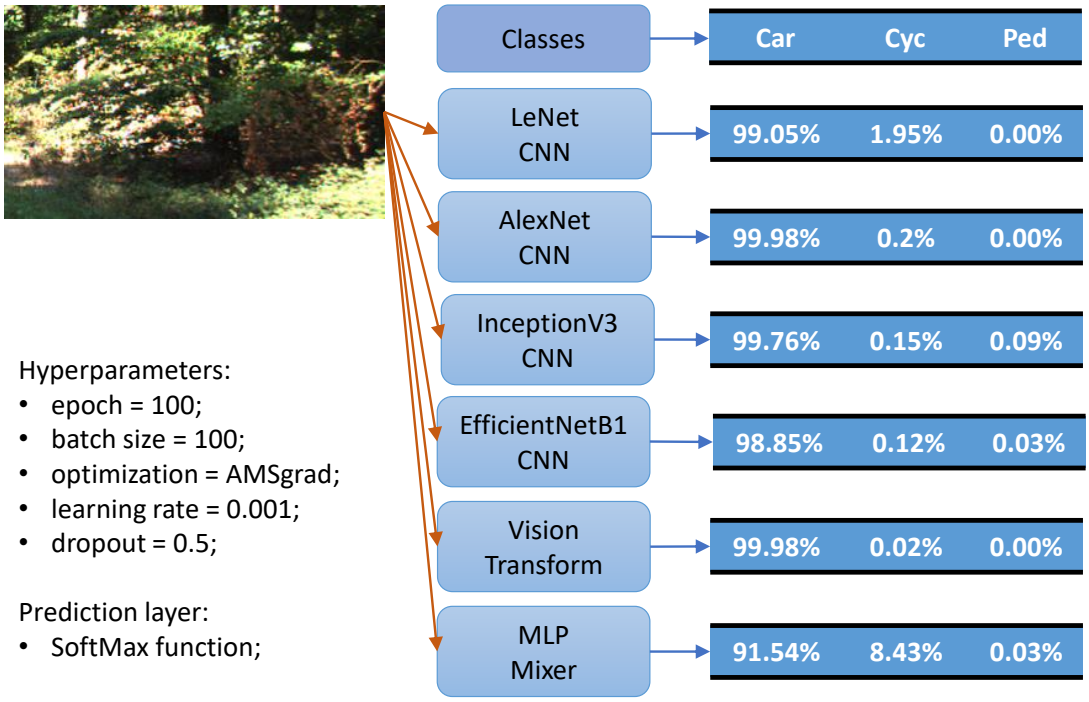
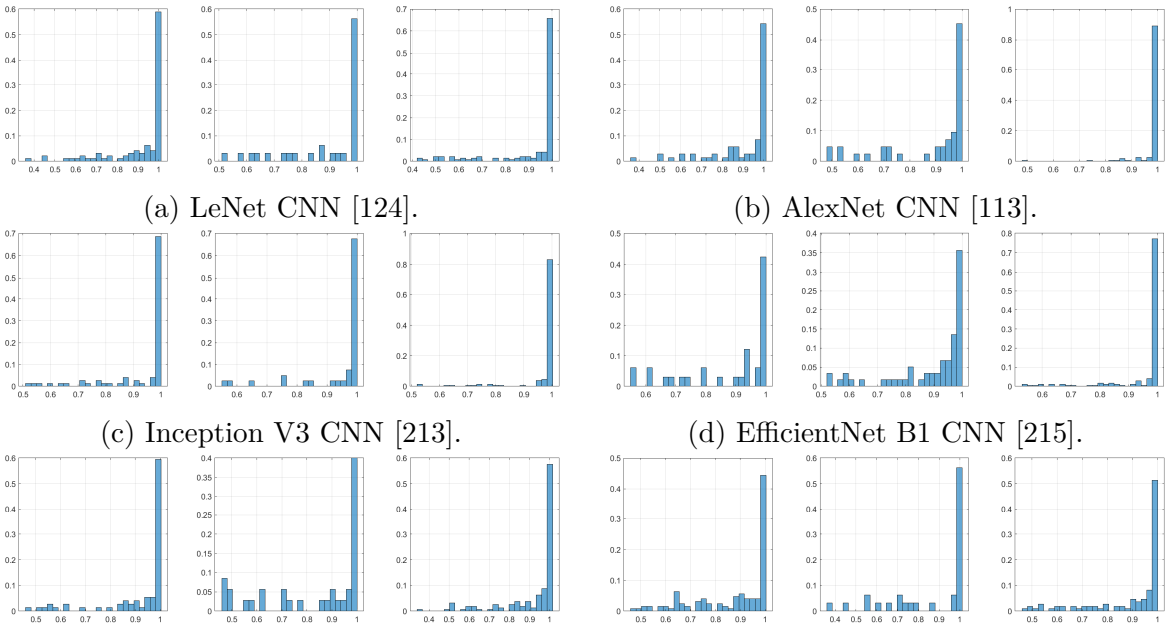


Fig. 1.6 Example of an object belonging to the trees class. Such class was not used during the training, where the prediction layer is the softmax function (SM). The object was classified by six different neural networks. All results were obtained with overconfident.

1.4 Summary of Contributions

The key contributions of this thesis are techniques to perform multimodality combination strategies (late and early fusion), and a novel approach to "smooth" overconfident results through Bayesian inference applied to the classification scores of the detected objects. To validate the proposed methodology, datasets from different modalities, such as RGB images, $3D$ point clouds, range-view and reflectance (intensity)-view maps have been considered. In summary, the contributions are the following:

- At a first moment, the thesis concentrates on the study of the influence of range-view (RaV) and reflectance/intensity-view (ReV) maps for objects classification. Such maps were obtained from the $3D$ point clouds projections (LiDAR) on the $2D$ image-plane, followed by an upsampling of the projected points. The upsampled map was obtained by using different sizes of masks (7×7, 9×9, 11×11, 13×13 and 15×15) and techniques, such as maximum, minimum, average and bilateral filter. Based on such map, a classification dataset was built to explore



(a) LeNet CNN [124]. (b) AlexNet CNN [113].

(c) Inception V3 CNN [213]. (d) EfficientNet B1 CNN [215].

(e) Vision Transform [47]. (f) MLP Mixer [220].

Fig. 1.7 Classification of objects of out-of-distribution test data (person sitting, tree, and pole classes) and separated according to training classes (car, cyclist and pedestrian- from left to right).

and study the multimodality approaches aiming to improve the object recognition capability of a perception system.

- Using the RGB, RaV and ReV modalities, it was possible to formulate early and late fusion strategies. For the early fusion strategy (training phase), the objects were concatenated into the entrance of the algorithm combining RGB, RaV and ReV maps. Regarding late fusion (prediction phase), the classification scores for each modality were combined using maximum, minimum, average, normalized-product, support vector machine (SVM), and genetic algorithm.

- By using $3D$ point clouds, RGB and RaV modalities, we formulated a multisensory-multimodality weighted-distance fusion strategy for object classification. The proposed strategy considers the relationship between the models classification performance and the distance of objects.

- Finally, and not least, it was considered an approach to reduce overconfident predictions through the concept of Bayesian inference. We model the likelihood function and the a-prior probability considering normalized histograms and

Gaussian functions of the training data, respectively. Such approach was applied both to object classification and detection datasets, presenting satisfactory results. These results are particularly significant when considering objects belonging to "unseen" samples which tend to be predicted with high score values by the deep learning models.

1.4.1 Publications

The results achieved during the doctoral studies have been reported in journals, conferences and workshop proceedings.

1.4.1.1 Journals: accepted and under-review

- Accepted: G. Melotti, C. Premebida, J. J. Bird, D. R. Faria and N. Gonçalves, "Reducing Overconfident Predictions in Autonomous Driving Perception", in *IEEE Access*, vol. 10, pp. 54805-54821, 2022, doi: 10.1109/ACCESS.2022.3175195.
- Under-review: G. Melotti, W. Lu, D. Zhao, A. Asvadi and N. Gonçalves, C. Premebida, "Probabilistic Approach for Road-Users Detection", in *IEEE Intelligent Transportation Systems Transactions*.

1.4.1.2 Conference Proceedings

- Melotti, G., Premebida, C., and Gonçalves, N. (2020). Multimodal Deep-Learning for Object Recognition Combining Camera and LiDAR Data. In *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Ponta Delgada, Portugal, pages 177 – 182, doi: 10.1109/ICARSC.49921.2020.9096138.
- Melotti, G., Premebida, C., Goncalves, N. M. M. d. S., Nunes, U. J. C., and Faria, D. R. (2018). Multimodal CNN Pedestrian Classification: A Study on Combining LiDAR and Camera Data. In *IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, USA, pages 3138 – 3143, doi: 10.1109/ITSC.2018.8569666.
- Melotti, G., Asvadi, A., Premebida, C. (2018). CNN-LiDAR pedestrian classification: Combining range and reflectance data. *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, Madrid, Spain, pages 1 – 6, doi: 10.1109/ICVES.2018.8519497.

1.4.1.3 Workshop paper

- Melotti, G., Premebida, C., Bird, J. J., Faria, D. R., and Gonçalves, N. (2020). Probabilistic Object Classification using CNN ML-MAP layers. In *ECCV Workshop on Perception for Autonomous Driving (PAD)*. Online.

1.4.1.4 Chapter in a Book

- Premebida C., Melotti G., and Asvadi A. (2019). RGB-D Object Classification for Autonomous Driving Perception. In: Rosin P., Lai YK., Shao L., Liu Y. (eds) RGB-D Image Analysis and Processing. Advances in Computer Vision and Pattern Recognition. Springer, Cham., pages $377-395$, doi.org/10.1007/978-3-030-28603-3_17.

1.5 Structure of the Thesis

The introduction of this thesis presented the basic concepts of autonomous/intelligent vehicles, as well as research motivations purposed. Chapter 2 presents an overview and discussion of the state-of-the-art on image and LiDAR classification and detection, fusion strategies, and finally probabilistic networks. Chapter 3 shows the mathematical formulations incorporated in the perception systems for object recognition, such as basic definitions of deep learning, activation functions, techniques to reduce overconfident predictions, as well as probabilistic formulations to obtain model uncertainty. In addition, Chapter 3 presents formulations about 3D point clouds and object detections. Chapter 4 explains the methodology adopted in the thesis *i.e.*, formulations using Bayesian inference to reduce overconfident predictions. The results of the research work on fusion strategies and reduction of overconfident predictions are shown in Chapter 5. The conclusion and future work are presented in the Chapter 6. Finally, the late fusion as Bayesian inference and Weighted Object Distance are explained in Appendix, as well as results from cumulative distribution.

Chapter 2

State of the Art

Contents

2.1 Object Recognition in Autonomous Driving

Perception systems for road users recognition is a key field of research and technology development in the automotive industry and in the academia, as it is the baseline of object detection and motion planning for ADS (Autonomous Driving Systems), ADAS (Advanced Driver Assistance System), and automotive protection systems. Such systems have considered different approaches and input data in order to obtain improve object detection performance, being cameras and LiDAR together with deep learning the most prevalent cases [37, 159, 77, 98, 225, 35, 2, 79].

Initially, the perception systems for autonomous driving had been dominated by the advancement of “traditional" machine learning algorithms and advanced feature-extractors, such as SVM and HOG-based descriptors [2, 160, 155]. The recent technological advances on deep networks (DNN), including convolutional neural networks (CNN), and vision transformers (ViT) [47, 215, 213, 86, 203, 212, 113, 249, 125], have made the DNNs the state of the art in pattern recognition tasks to process $2D$ and $3D$ data. These tasks consist of features extraction of the objects to be detected, selecting the most discriminating features, and constructing an end-to-end detector [55, 12]. In fact, many deep learning algorithms do not need extra algorithms of feature extraction (to identify a set of points of interest) or features descriptors (to describe a set of points) [11, 207].

2.1.1 Camera and LiDAR Modalities

Cameras that provide image from multiple domains (light spectra: visible and infrared) and multiple modalities (RGB, intensity/reflectance-view, depth/range-view, and motion) [15, 155] are one of the main sensors in autonomous driving applications, due to their ability to provide richer cues such as colors, shapes, and textures thus, forming patterns representations which, combined with machine learning techniques, contribute to the detection of objects by perception systems. However, recognizing patterns is not such an easy task, especially when there are objects with small dimensions or partially occluded objects. Therefore, choosing an algorithm to process images is a very crucial step, as object detection must be done in the shortest possible time and with the highest level of accuracy, especially for real-time applications, such as robotics and IV/AVs [55, 12, 98, 140]. Some of the main algorithms for objects detection tasks using vision/camera modality are:

- R-CNN [72]: it presented an approach that defined a selective search algorithm to generate 2000 regions within the image, as being proposed regions *i.e.*, candidate regions that can contain objects. Such regions are directed to a CNN that extracts features, and such features are inserted into a SVM, which determines if there are objects within the candidate regions.

- Fast R-CNN [71]: the approach is similar to R-CNN having the advantage of a faster object detection . The difference is that the input image goes straight to a CNN, which generates a convolutional feature map. From such maps, the proposed regions are obtained. Then the maps go through a pooling layer (region of interest) and are reshaped to a fixed length. Finally, they are processed through a completely connected layer (dense layer) and through other dense layers, the predicted classes and bounding boxes are obtained.

- Faster R-CNN [192]: the two detectors mentioned above use the selective search algorithm to define the proposed regions, which affects the training and testing phases. As an alternative, faster R-CNN replaces the search algorithm with a network that has the ability to learn the proposed regions *i.e.*, a separate network to predict the proposed regions. These regions are inserted into a pooling layer and the objects are detected, and finally the respective bounding-box values are estimated.

- RetinaNet [132]: it is a “simple" and efficient model for detecting small-scale objects through feature pyramid networks, as well as the introduction of Focal loss (a formulation that weights data from datasets with unbalanced classes).

- YOLOV4 [189–191, 23]: the ‘You Only Look One’ detector does not use a separate network to determine the proposed regions nor a search algorithm. YOLO takes advantage of convoluted networks to predict bounding boxes and detected object scores. Another difference from the detectors mentioned above is the input image divided into an $S \times S$ grid. Within each grid cell contains a number of defined bounding boxes (anchors). The network then generates bounding box values and detected object scores for each cell. Bounding boxes with scores greater than a threshold are selected to locate the object in the image.

In addition to the models mentioned above, we can cite the Mask R-CNN [84], R-FCN [42], SSD [137], as well as other more recent approaches defined in [128, 95, 252, 256, 101, 211, 30, 226, 239, 28, 186, 50, 126].

The approaches to detecting objects is not just restricted to $2D$ images. Several methodologies to detect and classify unstructured data like $3D$ point clouds have been presented in recent years and currently. The first approaches that achieved very significant results were through the volumetric and multi-view CNNs architectures, to classify $3D$ data obtained from CAD models and RGB-D sensors [183]. However, such $3D$ point clouds were modified before being processed *i.e.*, such data were transformed to regular $3D$ voxel grids or an image collection. This causes a disadvantage, since modifying the data can lead to loss of information.

To avoid any loss of information from the input $3D$ data, architectures of neural networks using $1D$ convolutions were proposed by [182, 184], having as main tasks the object classification, and semantic segmentation. Such networks were defined as PointNet, and PointNet++, and process each point independently of each other, and can learn features about $3D$ geometric data. In other words, PointNet, and PointNet++ capture the local structure of neighboring points and the interactions of combinations between the local structure. The main difference between the two networks is a cluster structure to capture more information about local features contained in PointNet++.

Currently, several papers presented similar architectures or ideas to $3D$ object recognitions from the PointNet and PointNet++, such as Generative PointNet [236] that proposes a generative model of unordered sets of points, Dynamic Graph CNN [228] that captures local geometric structure, describing the relationship between a point and its neighborhood through graphs, and PU-Net [246] that presents a technique of upsample of points, learning the characteristics in multi-levels per point.

Another model that showed satisfactory results in learning the $3D$ point clouds upsample was the PU-GCN [185], a model which considered an upsample learning structure through a graph convolutional networks (GCNs), which improves the capture of information on local points from neighboring points. In fact, GCNs show remarkable results for exploring the point interrelation [29, 238, 216, 200]. The ability to extract local information between neighboring points is not limited to graph networks, as the GeoCNN network [121] that extracts local geometric relations through the edge features extraction between the center and its neighboring points, performing a convolution operation for each point in its local neighborhood. In addition to the point cloud processing tasks we can cite PCNN and A-CNN [10, 111]. PCNN is a network that maps point cloud functions to volumetric functions using operators defined as extension and restriction, while the A-CNN copes with the sparsity and irregularities of the

geometric structure of point clouds by proposing a circular convolution network, which calculates convolution directly in $3D$ point clouds.

Development of architectures capable of processing $3D$ information is extremely important for autonomous driving applications, because many autonomous and intelligent vehicles use information obtained by LiDAR sensors [235, 6]. Among the various models for such applications, we can cite Ensemble Proposals [105], Frustum PointNets [181], BirdNet [17], MV3D [33], AVOD [115], PointRCNN [201], FastPointRCNN [34], Frustum ConvNet [229], 3DIoULoss [254], Point Pillars [122], TANet [138], HotSpotNet [31], Perspective Point Cloud (PPC) [29], and Recurrent PointPillars [147], BirdNet+[16], among others [255, 40, 82, 6].

2.1.2 Fusion Strategies to Combining Images and LiDAR Data

Fusion strategies are important components of perception systems, since it is possible to capture complementary and/or redundant information, both by different learning algorithms and by different data modalities [181, 168, 178, 3, 197]. We can address multimodality fusion by considering the possible combination of four components (layers): feature extraction, deformation manipulation models, occlusion manipulation models, and classifier. These components can then be combined through a deep model, in which the layer of deformation is incorporated, for example, into a CNN. Through the interaction between these independent components, the achieved results tend to show an improvement in the performance [168]. Considering, for instance, a fusion strategy that combines different machine learning algorithms, the research of [3] carried out pedestrian classification by combining the feature-space extracted by the AlexNet, VGG16 and Inception CNNs. Each extracted feature representation is then propagated to a hidden layer so that the rating output of each network has the same size. With the classification vectors of each network, a late fusion was applied through the product, sum, average, and max operations. The best result was achieved through the sum operator compared to the other operators and to the networks individually.

Regarding fusion strategies using different modalities, we can mention the works developed by [178] and [197], where the first performed pedestrian classification with three network architectures using the intensity, depth and flow modalities as input data. The first architecture presented three CNNs for each modality, and consequently a single vector was created with the outputs of each CNN, which was the input of a multilayer perceptron (MLP) that then provided the final classification. The second architecture considered one CNN having all modes in a single input vector, and consequently a

classification output was obtained. The latter architecture, carried out transfer learning by training a CNN with one modality and transferring the weights to a second CNN, which was trained with another modality, and likewise transferred the weights of the second trained model to a third CNN trained on the latest modality. The research showed that the last architecture, using weight transfer, outperformed the other two architectures. On the other hand, [197] performed pedestrian detection on two datasets: RGB and LiDAR data by upsampling the point clouds to obtain dense depth-maps by employing horizontal disparity (HD), height (H), and angle (A) techniques. Thus, the input data were the channels corresponding to the RGB images, HD, H, and the A maps. One fusion strategy involved the combination of the input-channels of such images and maps, and also an intermediate fusion across the CNN. The results showed that the fusion performed before the last fully connected layers (*i.e.*, intermediate fusion) presented better results than the combination of the channels in the network entrances (early fusion).

Fusion strategies can be further computed for $3D$ data representations, as described in [196], where the classification of $3D$ point clouds can be obtained by the Fisher encoding method (a measure of similarity between points, based on local shape statistical analysis) with spatial clusters presented by Gaussian Mixture Models[1]. Beyond the encoding method, techniques based on local shape descriptors (Fast Point Feature Histograms-FPFH, Spin Images-SI and Signatures of Histograms of Orientations-SHOT), and classification techniques (*e.g.*, kNN and SVM) were applied as well. The proposed method was then applied to pedestrian classification using artificial (Blender software) and real-world LiDAR (Stanford Track Collection) datasets involving partial occlusion, and non-standard shapes and poses. The classification also considered low-resolution data *e.g.*, a subset out of the set of points belonging to each object. The performance of the pedestrian classification approach was evaluated by means of the area under the curve (AUC) of the receiver operating characteristic (ROC) curve.

The combination of $2D$ and $3D$ sensor data tends to be crucial in some advanced perception systems, since images bring rich information related to the shape, color and texture of the objects, while $3D$ data (*e.g.*, provided by a LiDAR) brings depth information into the system, as explored and described in [102], who showed an alternative for classifying objects through the features obtained from point clouds and images. For point clouds, the features were geometric ones *e.g.*, height (mean

[1]They are probabilistic densities that represent normally distributed sub-populations within a whole population.

normalized height and height variance), plane (residual of plane fitting), eigenvalues, and others. After a segmentation stage, the image features were the average values of the R, G and B channels of all the image pixels, the brightness (calculated as the sum of the R, G and B average values divided by three), and then the R, G and B ratio (mean value divided by the sum of the three average values). Thus, the feature vector of each image was composed of height, plane, eigenvalue, mean values, brightness, and ratio. Such a vector was the input of a Bayesian Network that was structured by means of the concepts of mutual information *i.e.*, the entropy between the nodes. Finally, the classification decision was achieved by the computed posterior information *i.e.*, the posterior probability.

Alternatively, fusion strategies can be done using point clouds directly and images data, as presented by [181], who proposed a methodology that consisted of creating a space to extract a $3D$ bounding frustum of an object by extruding $2D$ bounding boxes from the images. Within each $3D$ space (frustum), it was performed $3D$ object instance segmentation and $3D$ bounding box regression using PointNet [182] and PointNet++ [184]. The point clouds passed through three stages before the object detection. The first step formed the frustum point cloud by projecting the camera matrix to a frustum, which defined a $3D$ search space for the object, where such points were collected. The next step performed the $3D$ instance segmentation of the point cloud using PointNet/PointNet++, which provided $3D$ mask coordinate. From the segmentation, it was possible to obtain a $3D$ location. The last step, $3D$ box estimation, estimated the coordinates of the object centre, then performed the detection. According to the authors, the results reached the state of the art when the method was evaluated on KITTI and SUN-RGB-D[2] benchmarks for $3D$ objects detection.

Another relevant research that combined $2D$ and $3D$ information was developed and described in [237], who showed satisfactory results for modeling $3D$ box estimation of cars, cyclists and pedestrians by combining RGB images and $3D$ point clouds applied in KITTI datasets (LiDAR and camera in driving scenes) and SUN-RGBD (indoor environments-RGB-D cameras). The inputs were processed by different networks. The point clouds were directly inserted into a variant of PointNet [182]. The RGB images were processed using ResNet pretrained on ImageNet. After the performed processing, the fusion was applied: point-wise and global features captured from PointNet, and images feature captured from ResNet-50. The fusion was the input of a MLP network that provided the $3D$ boxing and scores *e.g.*, the $3D$ bounding box for the object.

[2]SUN-Scene UNderstanding.

According to the authors the proposed architecture learned to combine image and depth sensor information, and the proposed methodology had no limitations for scene, object-specific, type and amount of depth sensors. Besides that, two functions of score losses were formulated: supervised and unsupervised scoring. The first trained the network to predict if a point was within the bounding box. The latter trained the network to choose the point that provided the optimal prediction. The authors showed that the unsupervised score function performed a bit better than the supervised score function.

Other early fusion strategies were proposed by [33] and [87]. [33] considered representations of point clouds bird's eye view (BEV), LiDAR front view, and camera front view. Such representations were processed by convolutional auto-encoder networks, and then were introduced into the pooling layers to obtain the regions of interest (RoI) of the detected objects. All these regions were concatenated and processed by a network of convolutions, which in turn generated the bounding boxes of the detected objects. [87] considered $3D$ point clouds and stereo RGB (two camera images of the same scene-right and left) to classify and detect 3D objects. Two convolution-deconvolution networks were applied to extract features from the images (one network for the images from the left camera and another network for the images from the right camera). At the same time, the corresponding pairs of images from each camera are concatenated and applied to another network, with the objective of generating $2D$ object proposals regions for the images from each camera. Both the outputs of the convolution-deconvolution networks and the proposed regions are inserted into the RoI pooling layers (one layer for images from the right camera and another for images from the left camera). RoI outputs are fused through the mean operation. The fused RoIs are fed into a new neural network architecture with the segmented $3D$ point clouds to generate the classifications and $3D$ bounding boxes.

In addition to the fusion of camera and LiDAR modalities, [48] showed a fusion strategy with information obtained from the RADAR sensor to detect $3D$ objects. The information from each sensor was inserted into a feature pyramid network to extract a space of individual characteristics of each modality. Such features are transformed into bird's-eye-view as a representation for an additive fusion operator, where convolution layers provide the classification and $3D$ bounding boxes.

2.2 Overconfident Predictions

Deep-learning based object detection, which is a key part of perception systems, tend to provide overconfident scores of the detected objects, regardless of whether the prediction is right or wrong. This is particularly common in non-calibrated or non-regularized networks, although even calibrated/regularised models are not exempt of overconfident behavior. Ideally, a well-calibrated or well-regularized model is expected to provide accurate predictions when they are right about object detection and, conversely, provide high uncertainty when they are inaccurate about a detection, as conceptualized in [242, 112, 44, 103, 219, 63]. Therefore, when a perception system is not able to provide an accurate determination of the detection of an object [164] a proper quantification of the uncertainty is desirable. Also, the idea of having a well-calibrated/regulated model is to guarantee that the resulting scores of the detected objects' classifications are actual probability estimates [167].

In general terms, formulations that act with predicted values after training the machine learning or deep learning algorithm has been defined as calibration techniques, while formulations that act during the training phase *e.g.*, by modifying the cost function (weight updates - to improve the generalization ability and, eliminate overconfidence) or even inserting perturbations into datasets during training, are defined as regularization techniques [154, 170, 179, 57, 258, 81, 174, 65, 107, 22, 106, 78]. Nevertheless, according to [68, 112] the name regularization can also be defined as calibration, as well as claim that such techniques to reduce or mitigate overconfidence still need to be improved. Also, in [112] introduced a kind of taxonomy based on three categories: (*i*) post-processing calibration, (*ii*) training the model with data augmentation, and (*iii*) probabilistic methods using Bayesian and non-Bayesian approaches.

2.2.1 Softmax and Sigmoid Prediction Layers

Softmax, a generalization of the sigmoid function for the multiclass case, is currently one of the most commonly employed functions to act as the prediction layer in deep learning. This is explained by the fact that softmax amplifies the weights (exponential term) of the correct class objects defined by the network, interfering in the updating of the weights, and thus may guarantees a better result in terms of classification performance. The price to be paid is that such weighting may contribute to overfitting, since the model becomes overconfidence on the training data [164, 148, 174]. Moreover, the softmax function does not provide any reliable confidence measurements for the

predicted values, and provides non-calibrated scores [38, 81, 90]. Also, it is possible to find in the literature works where the softmax's outputs are considered actual likelihood values [36, 133, 94, 227] (perhaps because they sum up to one) which tends to give an erroneous probabilistic interpretation about the results.

Softmax, as well as the logistic sigmoid function, are sensitive to adversarial attacks. In [75, 214], the authors have considered adversarial perturbations applied to the softmax and sigmoid prediction layers in deep-networks, and their findings have shown possible problems of underfitting on the weights.

In addition to the fact that softmax and sigmoid functions may providing poorly calibrated scores and being sensitive to adversarial attacks, such functions also seem inadequate to cope with detect objects belonging to out-of-distribution training data. This means that the model is trained on a training set and then evaluated on a set that does not contain the training classes, as demonstrated experimentally in [150, 44, 129, 90, 120, 63].

As alternatives to mitigate overconfidence, and reduce overfitting, the deep models can be learned using regulation or calibration technique [148, 174]. Both techniques have been subject of intensive studies in the last few years and continue to be a *hot* topic.

2.2.2 Regularization and Post-Processing Calibration Techniques

The improvement of the predicted values can be obtained by regularization techniques that avoid overfitting and contribute to reduce overconfident predictions [148, 164, 174], such as the transformation of network weights using $L1$ and $L2$ regularization [165], label and model regularization by a process of pseudo-label and self-training [258], label smoothing [141], knowledge distillation [91], and cost (loss) formulation that in the optimization process ensure low uncertainty for accurate predictions and high uncertainty for incorrect predictions [173, 36, 133, 112, 38, 44], as a cost function considering that each sample (object) can supervise all classes [36], as well as considering a margin on logit distances inserted in the cost function *i.e.*, the difference between the highest logit value in relation to the other logits of the same sample [133].

Other well-known regularization techniques are the batch normalization [96], stochastic regularization techniques such as dropout (randomly sets to zero the activation

function outputs of some neurons) [92], dropconect (randomly sets to zero some network weights) [224], multiplicative Gaussian noise [206].

Alternatively, highly confident predictions can often be mitigated by calibration techniques [250, 81] such as Isotonic Regression [248] which fits a piecewise constant non-decreasing function (piecewise-linear); Platt Scaling [176] which uses classifier predictions as features for a logistic regression model; Beta Calibration [117] which uses a parametric formulation that considers the Beta probability density function; Bayesian Binning performs Bayesian averaging to group multiple histogram binning calibration maps [162]; Temperature scaling that defines a parameter that rescales the logits values before applying the softmax function to compute the values of the scores of each class [194, 61, 81].

2.2.3 Predictive Uncertainty

Many deep learning methods used for perception systems (objects detection and recognition) do not capture the network uncertainties at training and test times. Probabilistic networks, such as the Bayesian Neural Network (BNN), can cope with uncertainties and can be carried out through distinct approaches. One way is to obtain the posterior distribution using variational inference after defining a prior distribution to the network weights [179, 202, 78]. Another method is the ensemble of multiple networks with the same architecture and different training sets for estimating predictive uncertainty [120].

Generally, uncertainties are defined as aleatory and epistemic uncertainties. Aleatory uncertainty is related to the inherent noises of observations (uncertainties arising from sensor inherent noise and associated with the distance of the object to be detected and the occlusion-model output), while the epistemic ones explain the uncertainties in the model parameters (uncertainties of the model associated with the detection accuracy thus, showing the limitations of the model). These uncertainties can be captured through Bayesian deep learning using probability distribution over the parameters of the model [103, 63, 65], Shannon Entropy (uncertainty in the prediction output) or by means of Mutual Information (confidence of the model) to measure the uncertainty of the classification scores [146, 57, 56].

The uncertainty of a prediction can also be achieved through Monte Carlo dropout strategy, using the dropout layers at test time *i.e.*, the predicted values depend on the randomly chosen connections between the neurons according to the dropout rate, that is, the same test example (an object) forwarded several times in the network can have

different predicted values (*i.e.*, the predicted values are not deterministic). In this way, it is possible to obtain the distribution, the average (final predicted value) and the variance (uncertainty) for each example [65, 64, 107].

2.3 Datasets

In any perception system study, one should consider the type of dataset to be used in order to decide whether technique is feasible or not, particularly in self-driving car studies [243]. Among the various datasets, we can cite KITTI-360 [130], ROAD [204], SensatUrban [93], Berkeley DeepDrive [244], Oxford RobotCar [142], Cityscapes [39], KITTI [69], nuScenes [27], A*3D [175], APOLLO [205], DrivingStereo [240], H3D [172], Lyft Level 5 [104], Waymo [232, 230, 231] and Mapillary [163]. Table 2.1 shows the main characteristics of the datasets mentioned above.

Among these, the KITTI dataset has earned a strong reputation in the AV/IV perception field, because the KITTI provides mono and stereo camera data, optical flow, visual odometry, $3D$ and $2D$ objects detection, $3D$ tracking, precise GPS data, high-resolution color cameras, LiDAR and the respective calibration parameters, providing raw data and ground-truth, as well as benchmarks for the tasks in rural areas and/or on highways.

2.4 State of the Art Summary

The tables 2.2, 2.5, 2.6, 2.7, 2.8, and 2.9 show a summary of the papers cited in State of the Art. In these tables, we can observe the types of data, the approach used in the papers and the datasets.

2.5 Discussion on the State of the Art

Despite the multitude and variety of sensors technologies used to build the many datasets mentioned before, we can cite as main sensors available on benchmarks and public datasets: monocular cameras, stereo, and LiDAR. Therefore, RADAR's technology is (presumably) an opportunity for research purposes. Such datasets are key ingredients to allow continuous research progress on perception systems for AV/IV applications and robotics. RGB images are crucial because they provide texture information (uniformity, luminosity, roughness and spatial distribution, etc.), thus

Table 2.1 Main features of some datasets for intelligent/autonomous vehicles.

Dataset	Multiple cities	Multiple weathers	Multiple time of day	Sensors
KITTI-360	No	No	No	Grayscale and color cameras, optics lenses, GPS/IMU, Velodyne 64 3D LiDAR
ROAD	Yes	Yes	Yes	monocular camera, GPS/INS
Berkeley DeepDrive	Yes	Yes	Yes	Video and GPS/IMU
Oxford RobotCar	Yes	Yes	Yes	Stereo and monocular camera, GPS/INS, SICK 2D LiDAR, SICK 3D LiDAR
Cityscapes	Yes	No	Yes	Stereo vision, stereo camera
KITTI	No	No	No	Grayscale and color cameras, optics lenses, GPS/IMU, Velodyne 64 3D LiDAR
nuScenes	Yes	Yes	Yes	Color cameras, LiDAR, RADARs, GPS/IMU
LyftLevel5	Yes	Yes	Yes	Color cameras, LiDAR, GPS/IMU
Apollo	Yes	−−	Yes	Color cameras, LiDAR, GPS/IMU
A*3D	No	Yes	Yes	Grayscale and color cameras, LiDAR
DrivingStereo	Yes	Yes	Yes	Color cameras, LiDAR, GPS/IMU
H3D	Yes	−−	Yes	Color cameras, LiDAR, GPS/IMU
Waymo	Yes	Yes	Yes	Color cameras, LiDAR, IMU
Mapillary	Yes	Yes	Yes	Color cameras

contribute to the identification of relevant patterns. On the other hand, point clouds provide the three-dimensional structure as well as depth and reflectance data.

It is clear that perception systems need formulations (representation) to interpret the sensory data. The representation depends on the ability of the formulation/s used to extracting information *i.e.*, feature extraction, from the images as well as from the point clouds, including the 'automatic' feature representation allowed in some deep learning frameworks (*e.g.*, CNNs) or the handcraft case (such as: LBP, LGP, and HOG).

Table 2.2 Summary of the papers cited in the state of the art regarding the camera and LiDAR modalities.

Reference	Title	Image	Point Cloud
Munder and Gavrila, [2009]	An experimental study on pedestrian classification	√	
Girshick et al., [2014]	Rich feature hierarchies for accurate object detection and semantic segmentation	√	
Miron et al., [2015]	An evaluation of the pedestrian classification in a multi-domain multi-modality setup	√	
Girshick, [2015]	Fast R-CNN	√	
Ren et al., [2015]	Faster R-CNN: Towards real-time object detection with region proposal networks	√	
Dai et al., [2016]	R-FCN: Object detection via region based fully convolutional networks	√	
Liu et al., [2016]	SSD: Single shot multibox detector	√	
Redmon et al., [2016]	You only look once: unified, real-time object detection	√	
He et al., [2017]	Mask R-CNN	√	
Redmon and Farhadi, [2017]	Yolo9000: Better, faster, stronger	√	
Qi et al., [2017a]	PointNet: Deep learning on point sets for $3D$ classification and segmentation		√
Qi et al., [2017b]	PointNet++: Deep hierarchical feature learning on point sets in a metric space		√
Qi et al., [2018]	Frustum PointNets for $3D$ Object Detection from RGB-D Data	√	√
Atzmon et al., [2018]	Point convolutional neural networks by extension operators		√
Redmon and Farhadi, [2018]	Yolov3: An incremental improvement	√	
Lu et al., [2019]	A review on object detection based on deep convolutional neural networks for autonomous driving	√	
Ahmed et al., [2019]	Pedestrian and cyclist detection and intent estimation for autonomous vehicles: A survey	√	
Komarichev et al., [2019]	A-CNN: Annularly convolutional neural networks on point clouds		√

Nowadays, recognizing patterns in RGB images is a well-consolidated task with the concepts of deep learning, particularly the CNN-based approaches. Several architectures were proposed for classification and detection of objects. However, there is no common consensus on what is the best CNN architecture. In fact, a good result can be achieved with a CNN for a dataset and an unsatisfactory result with another dataset, as illustrated in Section 2.1 through paper [253], which carried out a study with different CNNs and datasets. In addition, [253] has shown that there are several methodologies for extracting features and recognizing objects. The advantage of the CNNs is its ability to extract features such as edges and curves (convolutional layers) and classify the objects (fully connected layers) with no need to use formulations of handcrafted feature extractors.

Regarding the LiDAR data, some methodologies have been proposed for perception systems, achieving good results in objects recognition. The most notably formulations are the volumetric (voxels), the multi-view and the PointNet CNNs.

The volumetric CNNs subdivide the $3D$ space into a grid structure *i.e.*, discretizing the $3D$ space. We can explain the discretization as follows: if the voxel contains points, then its value is one; otherwise, its value is zero. The smaller voxel size, the more precise is its discretization, however, the computational cost is higher. The larger the voxel size, the less precise its discretization, as well as the loss of information. Thus, there is a tradeoff in the choice of voxel size and the data accuracy.

Multi-views use several $2D$ images from projections of the same $3D$ object in various views. Some information is lost because of the projections, but the various projections try to compensate these losses. However, with many $2D$ images of the same $3D$ object, the computational cost also turns out to be high.

Currently, some papers (Section 2.1.1 cf. paper [184] and Section 2.1.2 cf. papers [181] and [237]) embedded the PointNet network to extract and classify $3D$ data, because its architecture is a neural network that applies symmetric functions in several steps, and thus avoids the problem of ordering points (permutation).

Despite the advances of deep networks, the concepts of feature extraction using formulations, known as handcraft, which were shown in papers [102, 196] in Section 2.1.2 should not be ignored, since they present good results in the classification tasks.

Regardless of which concept we use to derive features from RGB images or point clouds, we can combine such concepts to improve the accuracy of classification, in other words, we can perform fusion with input data at the beginning of perception systems (early fusion), or with the data after the processing of each input in the perception

system (late fusion). These combinations are capable of improving the results of object recognition systems, as presented in the papers of the Section 2.1.2.

The processing of point clouds directly in the deep networks is still a challenge because the point clouds, which are obtained by the LiDAR sensor, are unstructured (no grid) and unordered data (a list of points with no order), in addition to the numbers of points not being fixed. This, we can ask the following question: what is the best methodology to be developed to insert the point clouds into a neural network, so that the order of points does not interfere with the results, as well as with the ability to define a grid and a fixed number of points?

We can state, taking into consideration the papers cited in the previous sections, that the deep learning models are the state of the art on perceptions systems. However, we should note that many models using deep learning do not provide uncertainty about predictions and about the model itself. These uncertainties are important for the tasks of pattern recognition applied in autonomous/intelligent vehicles, as presented in [56] and [103]. These papers, and others, have shown the importance of including Bayesian probability theory in deep learning models, which are known as Bayesian Deep Learning networks. We can define such networks as being models that are able to organize a given knowledge through a cause and effect relationship, and can also provide predictions or decisions even if they do not have complete information of a given situation; also, we can say they are graphs that relate probabilities between a group of variables. Thus, such networks provide the perception system output as well as the uncertainties contained in the model and output, but with relatively higher computational cost.

Deep models provide satisfactory results in determining classification scores. However, such results are presented with excessive confidence in the predictions, which can lead to an erroneous interpretation of the results, mainly when networks misclassify an object. To mitigate such errors, calibration and regularization techniques such as temperature scaling, isotonic regression, platt scaling, confidence penalty, and label smoothing aim to reduce the overconfidence problem when making predictions using relatively simple formulations. The disadvantage of these techniques is the inability to directly provide a measure of uncertainty regarding object classification scores.

Table 2.3 Continuation of Table 2.2.

Reference	Title	Image	Point Cloud
Lan et al., [2019]	Modeling local geometric structure of $3D$ point clouds using GEO-CNN	√	√
Wang et al., [2019b]	Dynamic graph CNN for learning on point clouds		√
Chen and Huang, [2019]	Pedestrian detection for autonomous vehicle using multi-spectral cameras	√	
Baker et al., [2020]	Local features and global shape information in object classification by deep convolutional neural networks	√	
Bochkovskiy et al., [2020]	Yolov4: Optimal speed and accuracy of object detection	√	
Carion et al., [2020]	End-to-end object detection with transformers	√	
Chen et al., [2020]	Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots		√
Janai et al., [2020]	Computer vision for autonomous vehicles: Problems, datasets and state of the art	√	√
Duan et al., [2020]	Corner proposal network for anchor-free, two-stage object detection	√	
Lin et al., [2020]	Focal loss for dense object detection	√	
Qiu et al., [2020]	Borderdet: Border feature for dense object detection	√	
Xu et al., [2020b]	Segment as points for efficient online multi-object tracking and segmentation	√	
Qian et al., [2021]	PU-GCN: Point cloud upsampling using graph convolutional networks		√
Xie et al., [2021]	Generative PointNet: Deep energy-based learning on unordered point sets for $3D$ generation, reconstruction and classification		√
Dosovitskiy et al., [2021]	An image is worth 16x16 words: Transformers for image recognition at scale	√	
Bansal et al., [2021]	$2D$ object recognition techniques: State-of-the-art work	√	
Chen et al., [2021]	MonoRUn: Monocular 3D object detection by reconstruction and uncertainty propagation	√	√

Table 2.4 Continuation of Table 2.2.

Reference	Title	Image	Point Cloud
Sun et al., [2021]	Sparse R-CNN: End-to-end object detection with learnable proposals	√	
Wang et al., [2021a]	Depth-conditioned dynamic message propagation for monocular 3D object detection	√	√
Feng et al., [2021]	Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges	√	
Zhang et al., [2021]	Objects are different: Flexible monocular 3D object detection	√	
Li and Zhao, [2021]	Monocular 3D detection with geometric constraint embedding and semi-supervised training	√	
Zhou et al., [2021]	Monocular 3D object detection: An extrinsic parameter free approach	√	
Joseph et al., [2021]	Towards open world object detection	√	
Zhou et al., [2022]	Leveraging Deep Convolutional Neural Networks Pre-Trained on Autonomous Driving Data for Vehicle Detection From Roadside LiDAR Data		√
Hussain et al., [2022]	Image Segmentation for Autonomous Driving Using U-Net Inception	√	
Li et al., [2023]	Contextual Transformer Networks for Visual Recognition	√	

Table 2.5 Summary of the papers cited in the state of the art regarding the fusion strategy.

Reference	Title	Image	Point Cloud
Schlosser et al., [2016]	Fusing LIDAR and images for pedestrian detection using convolutional neural networks	√	√
Akilan et al., [2017]	A late fusion approach for Harnessing multi-CNN model high-level features	√	
Chen et al., [2017]	Multi-view 3D object detection network for autonomous driving	√	√
Pop et al., [2017]	Incremental cross-modality deep learning for pedestrian recognition	√	
Ouyang et al., [2018]	Jointly learning deep features, deformable parts, occlusion and classification for pedestrian detection	√	
Qi et al., [2018]	Frustum PointNets for 3D object detection from RGB-D data	√	√
Savelonas et al., [2018]	Spatially sensitive statistical shape analysis for pedestrian recognition from lidar data		√
Xu et al., [2018]	PointFusion: Deep sensor fusion for 3D bounding box estimation	√	√
Arnold et al., [2019]	A Survey on 3D Object Detection Methods for Autonomous Driving Applications	√	√
Cui et al., [2021]	Deep Learning for Image and Point Cloud Fusion in Autonomous Driving: A Review	√	√
Wu et al., [2021]	Deep 3D Object Detection Networks Using LiDAR Data: A Review	√	√
Drews et al., [2022]	Deep 3D DeepFusion: A Robust and Modular 3D Object Detector for Lidars, Cameras and Radars	√	√
He et al., [2023]	Stereo RGB and Deeper LIDAR-Based Network for 3D Object Detection in Autonomous Driving	√	√

Table 2.6 Summary of the papers cited in the state of the art regarding the overconfident results.

Reference	Title	Image	Point Cloud
Platt, [1999]	Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods		
Zadrozny and Elkan, [2002]	Transforming classifier scores into accurate multiclass probability estimates		
Graves, [2011]	Practical variational inference for neural networks		
Hinton et al., [2012]	Improving neural networks by preventing co-adaptation of feature detectors	√	
Wan et al., [2013]	Regularization of neural networks using dropconnect	√	
Srivastava et al., [2014]	Dropout: A simple way to prevent neural networks from overfitting	√	
Kingma and Welling, [2014]	Auto-encoding variational Bayes	√	
Szegedy et al., [2014]	Intriguing properties of neural networks	√	
Ioffe and Szegedy, [2015]	Batch normalization: Accelerating deep network training by reducing internal covariate shift	√	
Hinton et al., [2015]	Distilling the knowledge in a neural network	√	
Goodfellow et al., [2015]	Explaining and harnessing adversarial examples	√	
Blundell et al., [2015]	Weight uncertainty in neural network	√	
Kingma et al., [2015]	Variational dropout and the local reparameterization trick	√	
Gal, [2016]	Uncertainty in Deep Learning	√	
Gal and Ghahramani, [2016a]	Bayesian convolutional neural networks with Bernoulli approximate variational inference	√	
Gal and Ghahramani, [2016b]	Dropout as a bayesian approximation: Representing model uncertainty in deep learning	√	
Kull et al., [2017]	Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers		

Table 2.7 Continuation of Table 2.6.

Reference	Title	Image	Point Cloud
Hendrycks and Gimpel, [2017]	A baseline for detecting misclassified and out-of-distribution examples in neural networks	√	
Pereyra et al., [2017]	Regularizing neural networks by penalizing confident output distributions	√	
Guo et al., [2017]	On calibration of modern neural networks	√	
Kendall and Gal, [2017]	What uncertainties do we need in bayesian deep learning for computer vision?	√	
McAllister et al., [2017]	Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning	√	
Lakshminarayanan et al., [2017]	Simple and scalable predictive uncertainty estimation using deep ensembles	√	
Liang et al., [2018]	Enhancing the reliability of out-of-distribution image detection in neural networks	√	
DeVries and Taylor, [2018]	Learning confidence for out-of-distribution detection in neural networks	√	
Neumann et al., [2018]	Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection	√	
Feng et al., [2018]	Towards safe autonomous driving: Capture uncertainty in the deep neural network for LiDAR $3D$ vehicle detection		√
Corbière et al., [2019]	Addressing failure prediction by learning model confidence	√	
Zou et al., [2019]	Confidence regularized self-training	√	
Feng et al., [2019]	Leveraging heteroscedastic aleatoric uncertainties for robust real-time lidar $3D$ object detection		√

Table 2.8 Continuation of Table 2.6.

Reference	Title	Image	Point Cloud
Mesquita et al., [2019]	LS-SVR as a bayesian RBF network		
Nixon et al., [2019]	Measuring calibration in deep learning	√	
Thulasidasan et al., [2019]	On mixup training: Improved calibration and predictive uncertainty for deep neural networks	√	
Shridhar et al., [2019]	A comprehensive guide to Bayesian convolutional neural network with variational inference	√	
Zhang et al., [2020]	Mix-n-Match: Ensemble and compositional methods for uncertainty calibration in deep learning	√	
Lukasik et al., [2020]	Does label smoothing mitigate label noise?	√	
Melotti et al., [2020a]	Probabilistic object classification using CNN ML-MAP layers	√	
Passalis et al., [2020]	Probabilistic knowledge transfer for lightweight deep representation learning	√	
Krishnan and Tickoo, [2020]	Improving model calibration with accuracy versus uncertainty optimization	√	
Meister et al., [2020]	Generalized Entropy Regularization or: There's Nothing Special about Label Smoothing		
Posch and Pilz, [2021]	Correlated parameters to accurately measure uncertainty in deep neural networks	√	
Wang et al., [2021b]	Energy-Based Open-World Uncertainty Modeling for Confidence Calibration	√	
Cheng and Vasconcelos, [2022]	Calibrating Deep Neural Networks by Pairwise Constraints	√	
Liu et al., [2022a]	The Devil is in the Margin: Margin-based Label Smoothing for Network Calibration	√	
Frenkel and Goldberger, [2022]	Network calibration by temperature scaling based on the predicted confidence	√	

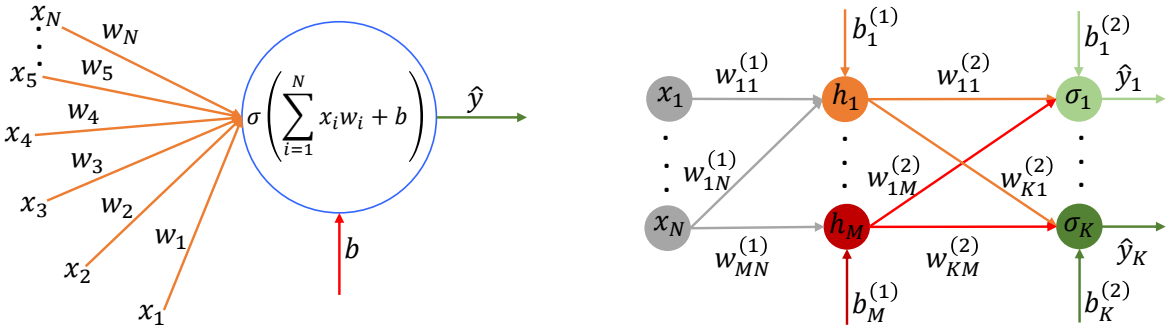
Table 2.9 Continuation of Table 2.6.

Reference	Title	Image	Point Cloud
Yeung et al., [2022]	Calibrating the Dice Loss to Handle Neural Network Overconfidence for Biomedical Image Segmentation	√	
Roitberg et al., [2022]	Is My Driver Observation Model Overconfident? Input-Guided Calibration Networks for Reliable and Interpretable Confidence Estimates	√	
Patra et al., [2023]	Calibrating Deep Neural Networks using Explicit Regularisation and Dynamic Data Pruning	√	

Chapter 3

Background

Contents



(a) Example of a simple neuron with N inputs and one output.

(b) Example of a neural network with N inputs and K outputs.

Fig. 3.1 Diagrams of two neural networks.

3.1 Artificial Neural Networks

Artificial neural networks (ANNs) were originally inspired on concepts of animals biological neurons. The formulation of a neural network was then defined by means of synaptic connections which bind input information to neurons. Mathematically, such connections are formulated as synaptic weights and measure the information on the neurons, as illustrated in Fig. 3.1a and Fig. 3.1b. Such weights are computed during an optimization process that minimizes the error between predicted and expected values. Typically, an ANN is composed of two stages: feedforward (which processes the input data and provides a prediction - from input to output), and backpropagation (feedback process to update the referred weights through the error - from output to input) [74].

3.1.1 Feedforward and Backpropagation in Neural Networks

A simple neural network model (NN), as the Fig. 3.1a and expressed by (3.1), is composed of a layer for data processing (perceptron) [195], an observation matrix with the input data of the network, and an output vector denoted by

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{X}\mathbf{w} + b\right) \tag{3.1}$$

where $\mathbf{X}$ represents the observation matrix, each observation contains N inputs for the network, b is the bias value, $\sigma(\cdot)$ is an element-wise non-linearity function, $\hat{\mathbf{y}}$ is the model output, and $\mathbf{w} = \{w_1, \cdots, w_N\}$ represents the model weights. Equation 3.1 can be better understood considering only one observation $\mathbf{x}$, let us say $\mathbf{x} = \{x_1, \cdots, x_N\}$,

according to (3.2)

$$\hat{y} = \sigma\left(\sum_{i=1}^{N} x_i w_i + b\right), \tag{3.2}$$

where the perceptron receives $x_1, \cdots, x_N$, and it provides the weights for the inputs, sum the inputs with weights, and generates the output after the non-linearity function.

The network weights can change their values according to a cost function. Generally for classification, the simplest cost function is determined by (3.3), while for regression is established by (3.4)

$$E_{cl}^{\mathbf{w},\mathbf{b}} = -\frac{1}{N_o}\sum_{i=1}^{N_o} y_i log(\hat{y}_i) \tag{3.3}$$

$$E_{rs}^{\mathbf{w},\mathbf{b}} = \frac{1}{2N_o}\sum_{i=1}^{N_o} ||y_i - \hat{y}_i||^2, \tag{3.4}$$

where N_o represents the outputs for the observed inputs, y_i is the label (target), and $\hat{y}_i$ is the predicted value by the neural network [63].

The best result of a prediction is the smallest difference (error) between the output value of the neural network (predicted value) and the real value. Therefore, the result takes into account the minimization of the errors in virtue of a relation including the predicted value in each parameter of the network. Such relationship is formulated by means of derivatives using the chain rule, for the reason that neural networks work with sequences of inputs and outputs through intermediate layers (or not) until reaching the final output. Thus, after determining the error and the gradient value, the network weights are updated according to (3.5)

$$\mathbf{w} = \mathbf{w} - \eta\frac{\partial E}{\partial \mathbf{w}}. \tag{3.5}$$

The training phase of a NN may suffer of overfitting, which impairs a good performance of the final prediction. An alternative to mitigate the overfitting is through regularization techniques, as L_2 regularization, by weighing the NN's parameters by a decay rate λ_i, resulting in an optimization process to minimize the cost function given by (3.6)

$$\mathcal{L}(\mathbf{w},\mathbf{b}) = E^{\mathbf{w},\mathbf{b}} + \lambda_1||\mathbf{w}||^2 + \lambda_2||\mathbf{b}||^2. \tag{3.6}$$

Briefly, the mechanism for calculating the predicted value from input to output is designated as the feedforward process, while the backpropagation is a method of calculating partial derivatives from the cost function with respect to all parameters of the model, while the process of minimizing the cost function is performed by an optimization method.

The model structure expressed by (3.2) can be extended to models with more neurons and layers, as shown in Fig. 3.1b and expressed by (3.7), with an intermediate layer (hidden layer) and K outputs

$$\hat{\mathbf{y}}_K(\mathbf{x}, \mathbf{W}) = \sigma \left[\sum_{j=1}^{M} w_{Kj}^{(2)} h \left(\sum_{i=1}^{N} w_{ji}^{(1)} + b_i^{(1)} \right) + b_j^{(2)} \right], \tag{3.7}$$

where $\hat{\mathbf{y}}_{1\ldots K}$ are the predicted values, $\mathbf{x}$ is vector of a given observation with N inputs, $\mathbf{W} = \{\mathbf{w^{(1)}}, \mathbf{w^{(2)}}\}$ is weights matrix, begin $\mathbf{w^{(1)}} = \{w_{M1}, w_{M2}, \ldots, w_{MN}\}$ and $\mathbf{w^{(2)}} = \{w_{K1}, w_{K2}, \ldots, w_{KM}\}$ vectors, M is the amount of neurons in the hidden layer, $h(\cdot)$ is the activation function, and $\sigma(\cdot)$ is the prediction function.

Considering Fig. 3.1b, with only two inputs and two neurons in the intermediate layer, the relationships of parameters $w_{11}^{(1)}$ and $w_{11}^{(2)}$ with the predicted values $\hat{\mathbf{y}}_1$ and $\hat{\mathbf{y}}_2$ are given by

$$\frac{\partial \mathcal{L}(W)}{\partial w_{11}^{(1)}} = \frac{\partial \mathcal{L}(W)}{\partial \sigma_1} \frac{\partial \sigma_1}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial h_1} \frac{\partial h_1}{\partial z_1^{(1)}} \frac{\partial z_1^{(1)}}{\partial w_{11}^{(1)}} + \frac{\partial \mathcal{L}(W)}{\partial \sigma_2} \frac{\partial \sigma_2}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial h_1} \frac{\partial h_1}{\partial z_1^{(1)}} \frac{\partial z_1^{(1)}}{\partial w_{11}^{(1)}} \tag{3.8}$$

$$\frac{\partial \mathcal{L}(W)}{\partial w_{11}^{(2)}} = \frac{\partial \mathcal{L}(W)}{\partial \sigma_1} \frac{\partial \sigma_1}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial w_{11}^{(2)}}, \tag{3.9}$$

where σ_1 and σ_2 are the prediction functions for outputs $\hat{\mathbf{y}}_1$ and $\hat{\mathbf{y}}_2$ respectively, index $^{(1)}$ refers to the hidden layer and index $^{(2)}$ refers to the output layer, $z_1^{(1)}$, $z_1^{(2)}$, $z_2^{(2)}$ are the linear combinations between the inputs and the weights of the network, as $z_1^{(1)} = w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2$. To update all the weights in the example above with 8 parameters ($w_{11}^{(1)}$, $w_{11}^{(2)}$, $w_{21}^{(1)}$, $w_{12}^{(1)}$, $w_{21}^{(2)}$, $w_{12}^{(2)}$, $w_{22}^{(1)}$, $w_{22}^{(2)}$), 52 partial derivative terms are required.

3.1.2 Convolutional Neural Networks

From the work presented in [113], the convolutional neural networks (CNNs) became the state of the art in image recognition [74, 253] which, among other scientific fields,

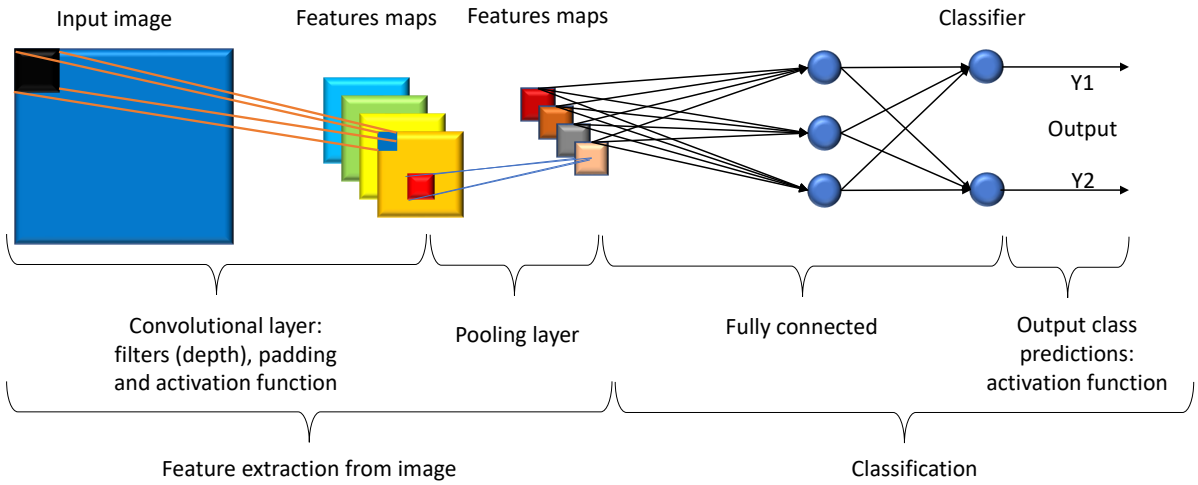


Fig. 3.2 Example of a basic CNN structure showing feature extraction and the classification, as well as the respective layers. The quantity of feature maps indicates the number of filters applied to the image during the convolution process.

have significantly contributed to the advances in perception systems for autonomous vehicles [9] and robotics. An example of a basic CNN structure, that contains the feature extraction and classification layers, is showed in Fig. 3.2. Within the feature extraction part, there are layers corresponding to the image input, the convolution layer (depth filters, padding and activation function), and the pooling layer, while the classification part corresponds to fully connected layers and output (final result-scores). In a "high-level" perspective, the convolutional part, including feature extraction, seeks to learn a local representation of the input data, while the fully connected part seeks to learn and make a decision with "global" characteristics.

Concisely, the key concepts and characteristics of a CNN structure can be explained as follows:

- Deep: represents the number of hidden layers.
- Depth: the number of feature maps also representing the number of filters (kernel).
- Stride (S): is the number of elements (cell of the image-row/column) that slides over the image.
- Padding (P): occasionally, we need to fill the matrix with zeros at the edges in order to perform the filter. It enables to prevent the filter from passing through a region out of the image.

- Convolution: refers to the idea of the mathematical concept of convolution, e.g., an operation between two functions to generate a third function, in other words, convolution is the integral/sum (continuous/discrete) of the point-to-point multiplication of the two functions, where one function is translated in time. Posterior to the convolutions, the feature maps are obtained.
- Activation function: this allows the network to learn more characteristics, in addition to the linear relationships of dependent and independent variables, according to Subsection 3.2.
- Pooling: reduces the convolution scaling while maintaining relevant image's information. Some functions of pooling are: maximum, minimum, average, sum, among others.
- Fully connected layer: all neurons connected to all neurons *i.e.*, each neuron in the previous layer is connected to each neuron in the next layer.
- Output layer: provides the classification scores by means of the activation function.

3.2 Activation Functions

When building a neural network, it is essential to define which activation function should be used, both in the hidden layers and in the prediction layer. The activation function has the ability to control how the neurons will be activated.

In fact, a neural network without an activation function is equivalent to a linear regression model, and therefore it loses the ability to learn features of more complex tasks. Thus, the activation function aims to introduce a non-linearity in the neuron output, and this contributes to decrease the error between the true value (label) and the predicted value during training, as well as ensuring a more accurate value of prediction with data not used during training.

Currently, several activation functions have been proposed [5], such as Tangent-Hyperbolic ($TanH$), Exponential Linear Unit (ELU), Gaussian Error Linear Unit ($GELU$), Rectified Linear Unit ($ReLU$), Leaky Rectified Linear Unit ($LReLU$), Scaled Exponential Linear Unit ($SELU$), Swish (Sh), Hard-Swish (H_{Sh}), Mish (Mh), Sigmoid (SgM), Softmax (SM), and Softplus (SP). Such functions are defined according to the expressions below,

- Tangent-Hyperbolic

$$TanH = \frac{e^z - e^{-z}}{e^z + e^{-z}}; \tag{3.10}$$

- Exponential Linear Unit:

$$ELU = \begin{cases} z, & \text{if} \quad z > 0 \\ \alpha(e^z - 1), & \text{if} \quad z \leq 0; \end{cases} \tag{3.11}$$

- Gaussian Error Linear Unit:

$$GELU = 0.5z\left(1 + tanh\left(\sqrt{\frac{2}{\pi}}(z + 0.044715z^3)\right)\right); \tag{3.12}$$

- Rectified Linear Unit:

$$ReLU = \begin{cases} 0, & \text{if} \quad z \leq 0 \\ z, & \text{if} \quad z > 0; \end{cases} \tag{3.13}$$

- Leaky Rectified Linear Unit:

$$LReLU = \begin{cases} 0.01z, & \text{if} \quad z < 0 \\ z, & \text{if} \quad z \geq 0; \end{cases} \tag{3.14}$$

- Scaled Exponential Linear Unit:

$$SELU = \lambda \begin{cases} \alpha(e^z - 1), & \text{if} \quad z < 0 \\ z, & \text{if} \quad z \geq 0; \end{cases} \tag{3.15}$$

- Swish:

$$Sh = z\frac{1}{1 + e^{-\beta z}}; \tag{3.16}$$

- Hard-Swish:

$$H_{Sh} = z\frac{min(max(0, z+3), 6)}{6}; \tag{3.17}$$

- Mish:

$$Mh = ztanh(ln(1+e^z)); \tag{3.18}$$

- Sigmoid:

$$SgM = \frac{1}{1+e^{-z}}; \tag{3.19}$$

- Softmax:

$$SM(\hat{z_j}) = \frac{e^{(\hat{z_j})}}{\sum_{k=1}^{K} e^{(\hat{z_k})}}, \tag{3.20}$$

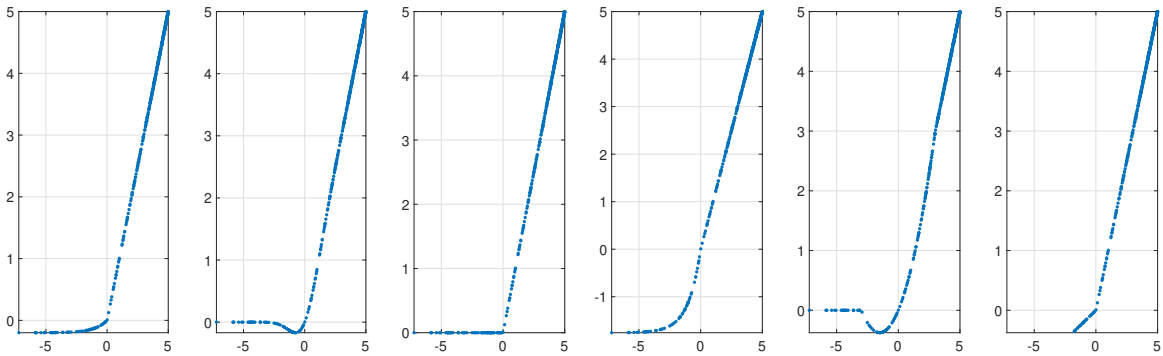
  where $j = 1, \ldots, K$;

- SoftPlus:

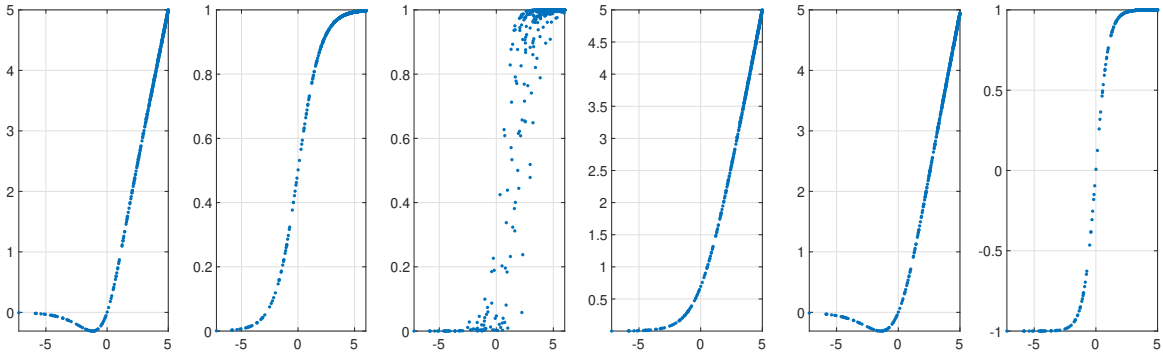
$$SP = ln(1+e^z). \tag{3.21}$$

Figures 3.3a and 3.3b illustrate the behavior of the outputs of the activation functions mentioned above. Note that for input values greater than zero, most of the activation functions behave similarly, with the exception of the Sigmoid, Softmax, and TanH functions. So far, it has not been possible to find a formulation or a method that define which activation function should be used in a CNN to obtain the best performance. However, softmax and sigmoid functions are already well accepted by the machine learning community as activation functions on the prediction layers, due to the fact that they tend to present better prediction results than the other activation functions.

From the normalized histograms, shown in Fig. 3.4, it is possible to see the behavior of these activation functions when they are employed as prediction functions. Note that the second, third, and last graphs on the second row of the Fig. 3.4b represent an overconfident prediction, which is good when the predicted objects belong to the same classes as the trained objects *i.e.*, when all predictions are correct. However, it can be problematic when an object is considered to be out-of-distribution test data [94] (class objects that were not considered during the training phase) or when objects are misclassified.



(a) The plots, from left to right, correspond to ELU, $GELU$, $ReLU$, $SELU$, H_{Sh}, $LReLU$ activation functions.

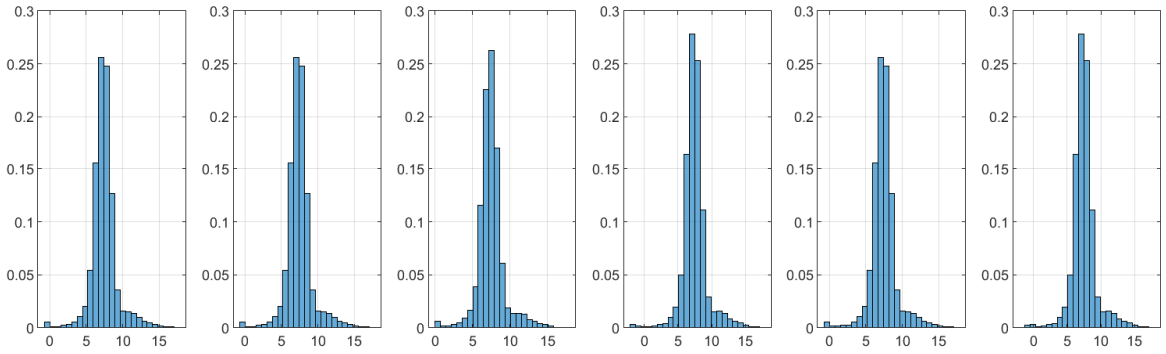


(b) The graphs, from left to right, correspond to Mh, SgM, SM, SP, Sh, $TanH$ activation functions.

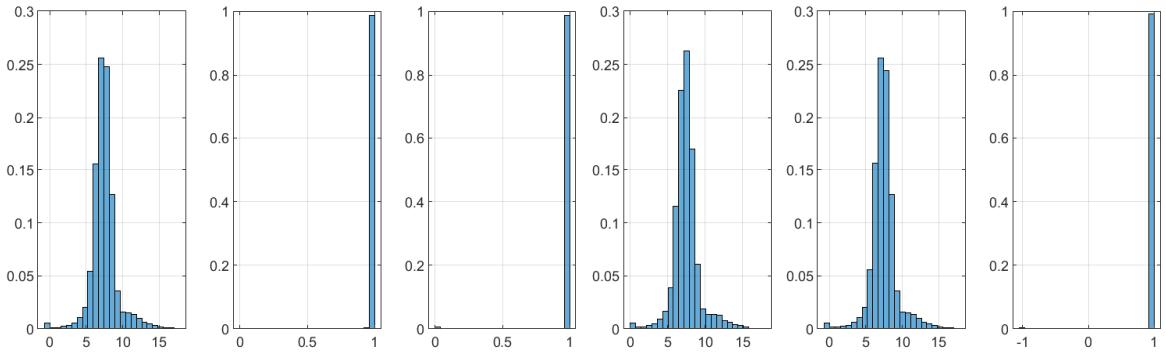
Fig. 3.3 Behavior of activation functions.

3.3 Overconfident Predictions

Oftentimes, regardless of the network architecture or input modalities, most object recognition techniques from deep learning models provide normalized prediction scores (the outputs) via a softmax layer [209] *i.e.*, the prediction values are in the interval of $[0, 1]$. Furthermore, such models are often implemented through deterministic neural networks, and the prediction itself does not consider uncertainty for the predict class of an object during the decision-making [199]. In fact, in most cases, the decision-making takes into account only the prediction value provided directly by a deep learning algorithm *i.e.*, disregarding a proper level of uncertainty of the prediction (unavailable for most networks). Therefore, evaluating the prediction confidence or uncertainty is crucial in decision-making whereas an erroneous decision may lead to a fatality, especially considering the autonomous driving, where the safety of human lives depends on the automation algorithms.



(a) The histograms from left to right correspond to ELU, $GELU$, $ReLU$, $SELU$, H_{Sh}, $LReLU$ activation functions.



(b) The histograms from left to right correspond to Mh, SgM, SM, SP, Sh, $TanH$ activation functions.

Fig. 3.4 Histograms corresponding to the activation functions on a test sample.

In this regard, the main techniques to mitigate the overconfident results in deep networks are calibration [81, 118, 1, 153] and regularization [179, 258, 1, 153]. Calibration acts directly in the network output result, while regularization aims at penalizing network weights through a variety of methods, adding parameters or terms directly to the network cost/loss function [179, 258, 174]. Consequently, the latter demands that the network should be retrained. However, the papers proposed by [133, 36, 68, 112] define regularization techniques as a type of calibration.

3.3.1 Out-of-Distribution Test Data

Generally, neural networks are trained considering that test data are similar to training data. However, in real situations the data can vary greatly when compared to the training data, for example objects that do not belong to the training classes, which are defined as out-of-distribution data or unseen data [94, 63], and sometimes defined as data that are far from the training data [81, 63, 21]. Such variation can

Table 3.1 KITTI dataset: objects from the out-of-distribution test data.

	Unseen Objects		
	Tram/Truck/Van	**Tree/lamppost**	**Person-sitting**
Training	-	-	-
Validation	-	-	-
Testing	511/1094/2914	45	222

significantly compromise the results provided by the perception systems, as illustrated in Fig. 1.7 in Subsection 1.3. Thus, training systems must be able to generalize to out-of-distribution test data.

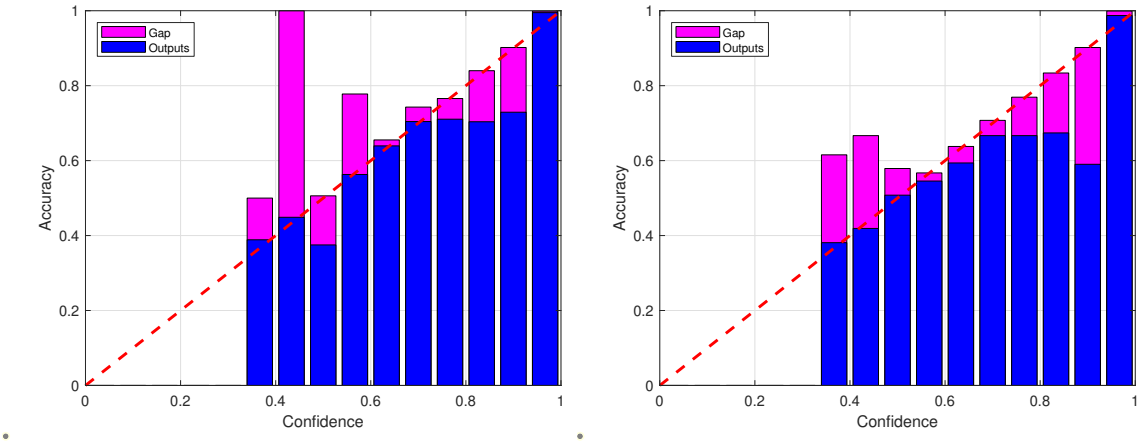
From the out-of-distribution test data as an alternative to validate the proposed methodology of reducing overconfident predictions, we considered object from the tram, truck, van, tree (stem), lamppost and person-sitting classes *i.e.*, object classes not used during the training. The objective, in this case, is to avoid that erroneously classified objects have a prediction with high score value. The number of objects that make up the unseen dataset is shown in the Table 3.1.

3.3.2 Model Calibration

In [81] states that "confidence calibration is the problem of predicting probability estimates representative of the true correctness likelihood". Mathematically, the idea of calibration can be defined as follows: let h to be a model of machine learning, where $h(X) = (\widehat{Y}, \widehat{P})$, considering a distribution generated over the K possible classes of the model for a given input X, begin $\widehat{Y}$ the predicted class with a associated predicted confidence defined as $\widehat{P}$. The *perfect calibration*is given by:

$$\mathbb{P}(\widehat{Y} = Y | \widehat{P} = p) = p, \quad \forall\, p \in [0,1], \tag{3.22}$$

whereas the probability is over a joint distribution. The formulation (3.22) can be better understood by the example given by [81]: "given 100 predictions, each with confidence of 0.8, we expect that 80 should be correctly classified." Thus, for every subset of predicted samples of a given class with score values equal to S, the proportion of samples that are actually of that class is S too. Nonetheless, the calibration formulation is an approximation process that depends on a calibration



(a) Example using RGB modality. (b) Example using range view modality.

Fig. 3.5 Example of reliability diagrams on the testing set from the KITTI dataset, considering Inception V3 CNN using the softmax layer (SM) as the prediction layer.

measure, which can be obtained by separating the predictions into multiple bins, as reliability diagram [116, 81, 166, 61, 136, 73, 88].

3.3.2.1 Reliability Diagram

Typically, predictions are analyzed in the form of reliability diagram representation, which illustrate the relationship of the model's prediction scores in regard to the true correctness likelihoods [116, 81, 166, 61], as shown in Fig. 3.5. Reliability diagrams show the expected accuracy of the samples as a function of confidence.

The scores (predicted values) are grouped into M bins (histogram) in the reliability diagrams [81, 73, 88]. Each sample (classification score of an object) is allocated within a bin, according to the maximum prediction value (prediction confidence). Each bin has a range $I_m = \left(\frac{(m-1)}{M}, \frac{m}{M}\right]$, where $m = 1, .., M$ *i.e.*, range I_m set to the score values in the range $(0, 1]$. The accuracy[1] $acc(BM) = \frac{1}{BM}\sum_i 1(\hat{y}_i = y_i)$ is calculated in each range I_m, where $\hat{y}_i$ and y_i are the predicted value and true class label for sample i, as well as the average confidence $conf(BM) = \frac{1}{BM}\sum_i \hat{p}_i$, where $\hat{p}_i$ is the confidence for sample i and BM is the amount of objects in each I_m. Thus, a perfect calibration will have $acc(BM) = conf(BM)$ for all $m \in 1, ..., M$. In addition, a gap can be obtained *i.e.*, the difference between accuracy and average confidence in each range (I_m). Thus, the greater the gap, the worse the calibration result in the respective bin. Furthermore, through reliability diagrams, it is possible to obtain calibration errors, such as the

[1] True fraction of positive cases.

Expected Calibration Error[2] (ECE) and the Maximum Calibration Error[3] (MCE):

$$ECE = \sum_{m=1}^{M} \frac{BM}{n} |acc(BM) - conf(BM)|, \tag{3.23}$$

$$MCE = \max_{m \in \{1,...,M\}} |acc(BM) - conf(BM)|, \tag{3.24}$$

where n is the number of samples, and $|acc(BM) - conf(BM)|$ represents the difference between the predict confidence and model accuracy.

Moreover, the reliability diagrams illustrate the identity function (diagonal-dashed line) that represents a perfectly calibrated output, while any deviation from the diagonal represents a calibration error. In other words, a perfectly calibrated model has all points on the identity function *i.e.*, for each bin the average predicted score is equal to the fraction of true positive samples [116, 81, 166, 61, 136, 73, 88].

3.3.3 Post-Processing Calibration Techniques

An efficient way to mitigate poorly calibrated models is through predictions after training. Among the methods capable of such calibration, we can mention Platt scaling, isotonic regression and tempaerature scaling [176, 248, 81]. Post-processing calibration techniques present the advantage of being easily applied to pre-trained models.

3.3.3.1 Platt Scaling

Platt Scaling [176] uses classifier predictions as features for a logistic regression model (sigmoid function). In fact, the idea is to transform the outputs of a classifier into posterior probabilities by passing them through the sigmoid function:

$$P(y = 1|f(x)) = \frac{1}{1 + exp(Af(x) + B)}, \tag{3.25}$$

where $f(x)$ is the output of a given model, A and B are parameters to be fitted with the training set $(f(x), y)$ by minimizing the negative log likelihood.

[2]Weighted average of the distance from the model calibration curve to the identity function (diagonal).

[3]Maximum distance between the model calibration curve and the identity function.

3.3.3.2 Isotonic Regression

This method determines that the function is isotonic (increasing monotonically). Given the model outputs (predicted values) and the respective labels, the isotonic regression [248] is defined as:

$$y_i = m(f_i) + \epsilon_i, \tag{3.26}$$

where m is the isotonic function determined by minimizing $\sum_{i=1}^{n}(f_i(x_i) - y_i)^2$, where f_i is the prediction value from a function to be learned for each training x_i, and y_i is the label.

3.3.3.3 Temperature Scaling

An alternative to reduce overconfident predictions was proposed by [81] and defined as temperature scaling (TS). The value of TS is obtained by minimizing the negative log likelihood (NLL) on the validation set. All the values of the logit vector (before the prediction layer) are multiplied by scalar parameter $\frac{1}{TS}$ with $TS > 0$ during the prediction at test time. Simply, the temperature scaling parameter can be included in the softmax function (SM) according to (3.27)

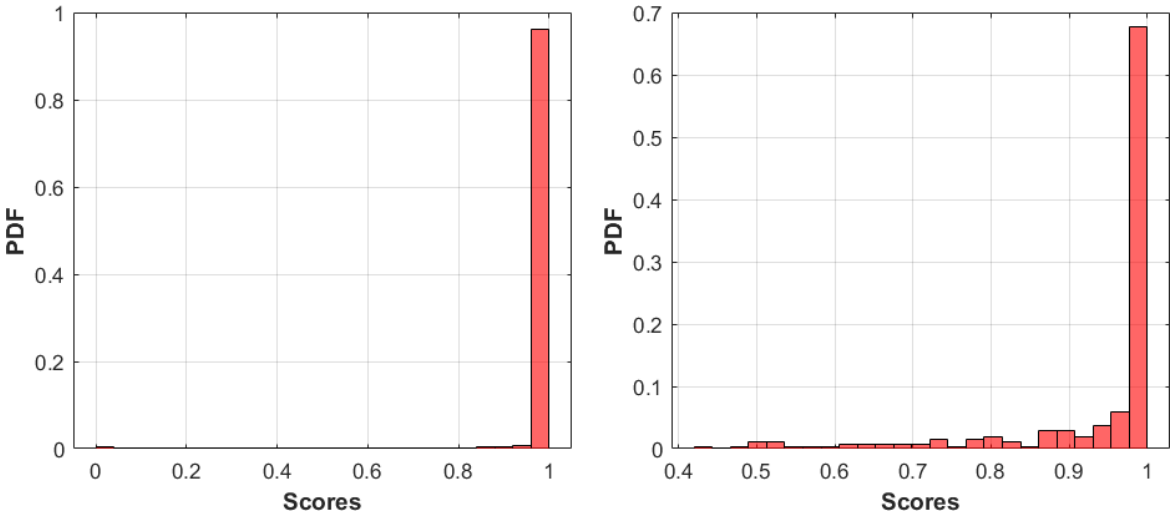
$$Softmax(\hat{z}_j) = \frac{e^{(\hat{z}_j/TS)}}{\sum_{k=1}^{K} e^{(\hat{z}_k/TS)}}, \tag{3.27}$$

where $k \in \{1, \ldots, j, \ldots, K\}$, K is the number of classes, $\hat{z}_j$ is the output of the predicted logit layer.

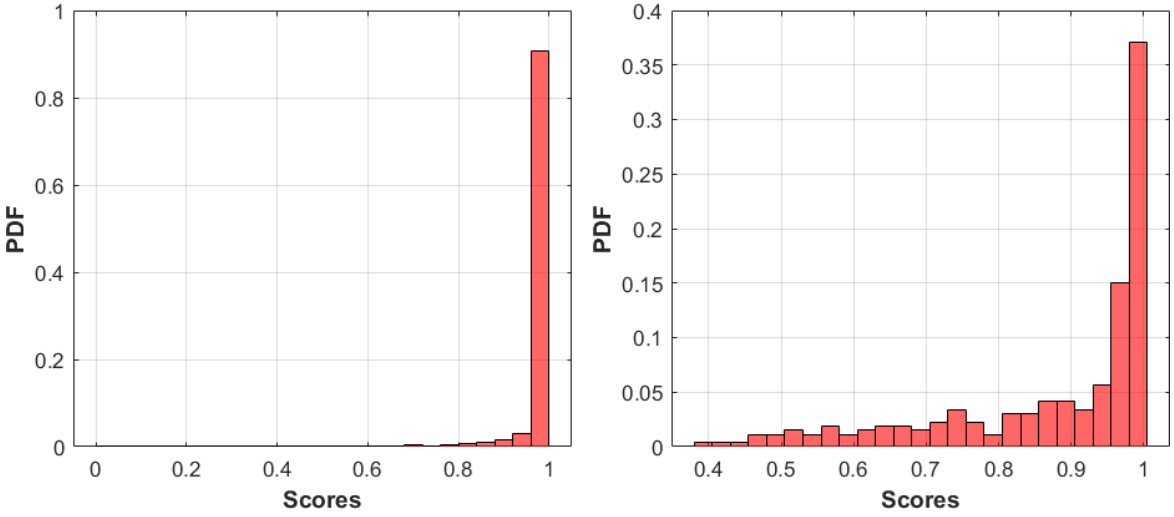
The score results after the classification using temperature scaling is showed in the Fig. 3.6b, where the first column represents scores of classified objects such as car, cyclist and pedestrian, while the second column represents scores for unseen (out-of the training distribution) datasets. Note that distributions from SM, Fig. 3.6a, are more extreme than distributions from temperature scale with SM.

3.3.4 Regularization Techniques

Different from the post-processing techniques, several regularization techniques, which act on the updates of the neural network weights, have been proposed to avoid overfitting, including L1/L2 regularization, dropout, early stopping and batch



(a) Classification result without calibration and regularization techniques.



(b) Result after applying the temperature scaling calibration with $TS = 1.8174$.

Fig. 3.6 Distribution of scores obtained from Inception V3 CNN training, considering the scores of objects belonging to the classes used during training (comprising the car, cyclist and pedestrian classes together) in the first column, while in the second column are the unseen dataset scores.

normalization. However, these techniques are not enough to get smoother output distributions. To mitigate overconfident outputs, mathematical formulations such as confidence penalty [148, 174], and label smoothing [127, 161, 174] have been applied to neural networks in order to acquire a smoother prediction output.

3.3.4.1 Loss Function

For classification problems, the label is defined as $\mathbf{Y} = \{0, 1, \ldots, K\}$ for object i from the training dataset with the input variable $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$ *i.e.*, (x_i, y_i) for object i, with the label defined as one-hot encoding vector, and the loss function, $\mathcal{L}$, for N objects with K classes is given by

$$\mathcal{L} = -\sum_{i}^{N} [p(y_i|x_i) log(p(\hat{y}_i|x_i)] , \tag{3.28}$$

where $p(y_i|x_i)$ is the distribution of the label (ground-truth), and $p(\hat{y}_i|x_i)$ is the predicted distribution. Equation (3.28) was defined for multiclass classification (more than 2 classes). When the number of classes is 2, the loss function is definde by

$$\mathcal{L} = -\sum_{i}^{N} [p(y_i|x_i) log(p(\hat{y}_i|x_i)) + (1 - p(y_i|x_i)) log(1 - p(\hat{y}_i|x_i))] . \tag{3.29}$$

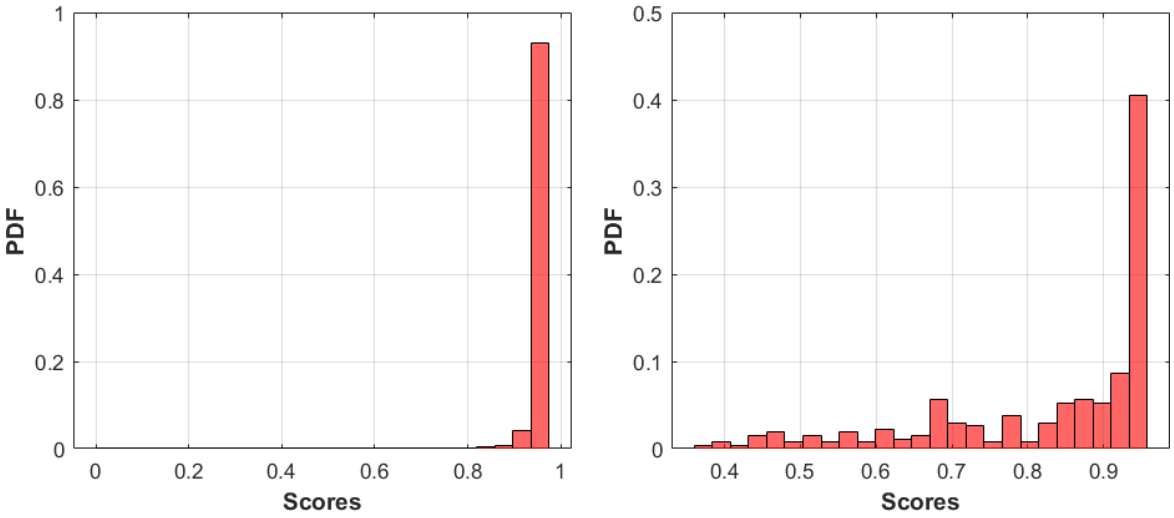
The cost function $\mathcal{L}$ can also be represented by the average, which is multiplied by the factor $\frac{1}{N}$.

3.3.4.2 Confidence Penalty

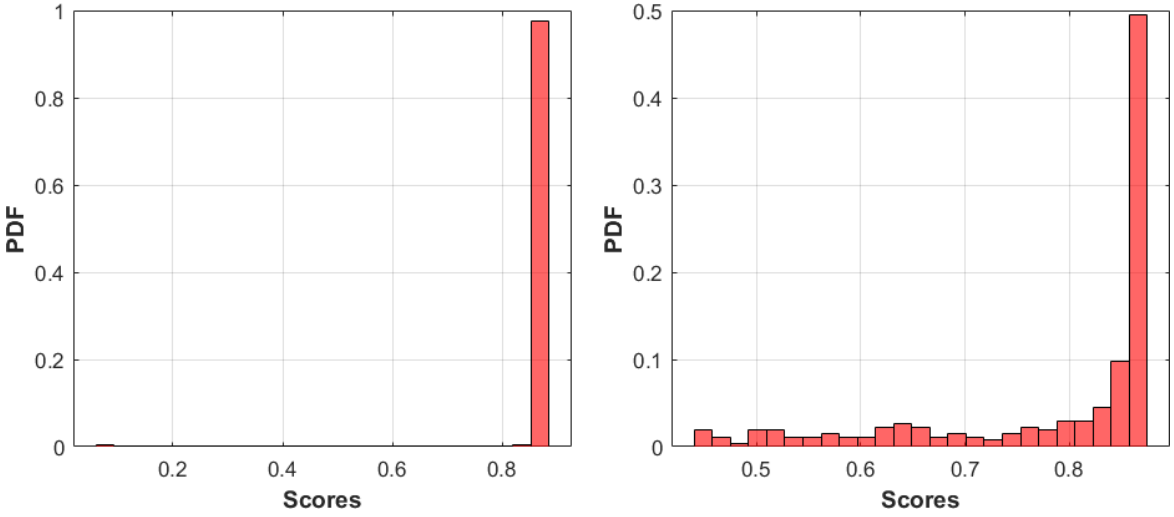
Models that provide overconfident distributions of outputs have such distributions with low entropy [133, 148]. Thus, the formulation of confidence penalty, expressed in (3.30), includes a weighting term in the cost function given in (3.28). Mathematically, the additional term is the entropy of the predicted values, and β is the parameter that controls the confidence penalty [174, 148],

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} [p(y_i|x_i) log(p(\hat{y}_i|x_i)) - \beta p(\hat{y}_i|x_i) log(p(\hat{y}_i|x_i))] . \tag{3.30}$$

The confidence penalty term can contribute to the model output distribution being more entropic [133, 148]. An example of the result considering confidence penalty on object classification is shown in Fig. 3.7a, that illustrates an overconfident result.



(a) Inception V3 trained with confidence penalty, considering $\beta = 0.3$.



(b) Inception V3 trained with label smoothing, considering $\epsilon = 0.2$.

Fig. 3.7 Distribution of scores obtained from Inception V3 CNN training, considering the pedestrian, car and cyclist classes in the first column, and unseen dataset in the second column.

3.3.4.3 Label Smoothing

Unlike confidence penalty, the label smoothing technique does not directly interfere with the mathematical formulation of the cost function (entropy), making the model less certain about the provided predictions. In fact, label smoothing modifies the values of the one-hot encoding vector, as defined in (3.31) [213, 174],

$$y_{new_{i,k}} = (1 - \epsilon) y_{i,k} + \frac{\epsilon}{K} \tag{3.31}$$

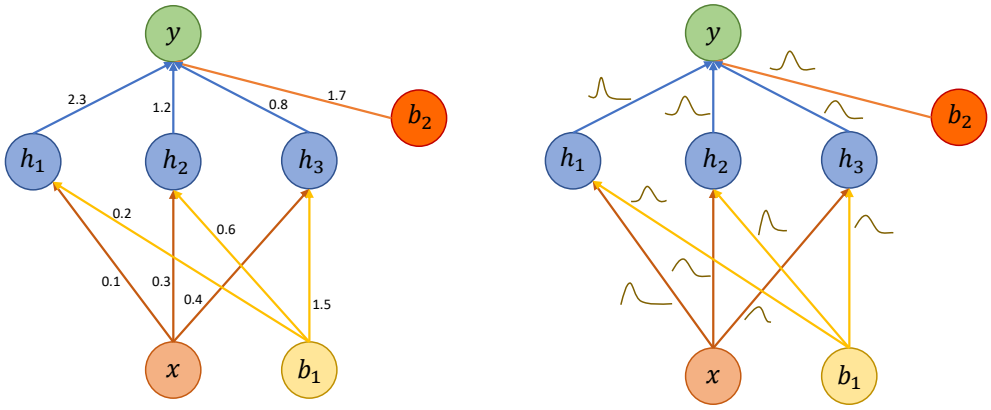


Fig. 3.8 The figure on the left represents the structure of a neural network where the weights are point-estimates, while the one on the right illustrates the weights with probability distributions.

where $y_{i,k}$ is the object i in the class k, $y_{new_{i,k}}$ is the new label value, ϵ is the smoothing parameter arbitrarily defined, and K is the number of classes. Label smoothing reduces the difference between the values of the labels of the correct class against the values of the other classes, interfering in the updating of the weights of the network. In other words, label smoothing prevents the network from assigning the total score value to a single class, maintaining a reasonable distance between the values of the ground-truth class scores and the other classes, as well as contributing to the network provide a better calibrated result [36, 148, 133, 174]. According to [213], not using the label smoothing technique may result in overfitting, and thus reduces the ability of the model to adapt. This happens because the model becomes too confident about its predictions.

Considering the vector $Y = \{[0,0,1],[0,1,0],[1,0,0]\}$ as the classification label (one-hot encoding), and the smoothing parameter as 0.2, the new label is given by the vector $Y_{new} = \{[0.066,0.066,0.868],[0.066,0.868,0.066],[0.868,0.066,0.066]\}$ and $\sum_{k}^{3} y_{new_{i,k}} = 1$. The result using label smoothing is shown in Fig. 3.7b, which still shows overconfident behaviour.

3.4 Probabilistic Model and Confidence

A probabilistic model has the ability of informing how certain/uncertain is the prediction value, as well as the model confidence..

The model confidence can be captured by mathematical formulations that consider the networks weights as distributions instead of single values [74, 22, 21], such as the probabilistic network shown in Fig. 3.8 on the right, considering Bayesian neural network. Other formulations consider the results from deterministic networks in order to calculate the model's uncertainty, such as Monte Carlo Dropout [65, 103] and Ensemble Modeling [120].

3.4.1 Probabilistic Modeling

The relationship between input data $\mathbf{X}$ and output data $\mathbf{Y}$, defined by $Data = (\mathbf{X}, \mathbf{Y})$, with learning parameters $\mathbf{W}$ from a supervised classification system, can be formulated according to a random experiment by considering a sample space $\mathbf{S}$. The numerical outcome obtained from each element of $\mathbf{S}$ is related to a real number, defined by the random variable (*rv*) $\mathbf{D}$ which relates the input and output *i.e.*, the input and output data are conditioned to the *rv* $\mathbf{W}$. Formally, the *rv* is a function that maps each element of the sample space with a real number of the set $\mathbb{R}$, which can be simply expressed as $\mathbf{D} : \mathbf{S} \rightarrow \mathbb{R}$. In other words, a *rv* is a function $\mathbf{D}$ that outputs a real number $\mathbf{D}(\zeta)$ for each element $\zeta \in \mathbf{S}$ of a random experiment. From the sample space, an event (subset of $\mathbf{S}$) can be defined and associated with a probability $\mathbf{P}$ between the interval $[\xi,\, \xi + \Delta\xi]$. Such probability is a distribution function and its derivative is the probability density function (PDF) $f_D(D = \xi|\mathbf{W})$ *i.e.*, the density function is conditional to $\mathbf{W}$, as in (3.32) [169],

$$f_D(D = \xi|\mathbf{W}) = \lim_{\Delta\xi \rightarrow 0} \frac{\mathbf{P}\{\xi \leq \mathbf{D} \leq \xi + \Delta\xi|\mathbf{W}\}}{\Delta\xi} \tag{3.32}$$

where $f_D(D = \xi|\mathbf{W}) \geq 0 \; \forall \; \xi$, considering ξ continuous. The integral of (3.32) represents the probability $\mathbf{P}$ with the random variable $\mathbf{D}$ contained in the interval. Consequently, if the interval $[\xi,\, \xi + \Delta\xi]$ is sufficiently small of specific length $\Delta\xi$, the probability will be $\mathbf{P}\{\xi \leq \mathbf{D} \leq \xi + \Delta\xi|\mathbf{W}\} \simeq f_D(D = \xi|\mathbf{W})\Delta\xi$ *i.e.*, the probability of the random variable $\mathbf{D}$ is proportional to $f_D(D = \xi|\mathbf{W})$. Thus, the probability will be maximum if the interval $[\xi,\, \xi + \Delta\xi]$ contains the most likely value of $\mathbf{D}$, where $f_D(D = \xi|\mathbf{W})$ will be maximum. If the random variable is discrete, a probability mass function (PMF) is used instead of a probability density function (PDF) [169].

3.4.1.1 Probabilistic Inference

Assuming that the class-conditional probability $P(\mathbf{D}|\mathbf{W})$ (likelihood) is known from (3.32), the posterior probability $P(\mathbf{W}|\mathbf{D})$ is obtained through Bayes' rule (also known as Bayes' theorem). Actually, Bayesian approach depends on a prior distribution, $P(\mathbf{W})$, which expresses the uncertainty about the value $\mathbf{W}$ before any observed data [74, 21, 169], considering $\mathbf{W}$ to be the neural network weights [74, 21].

The aforesaid Bayes' rule determines the relationship between distributions, in which the posterior distribution is obtained according to (3.33)

$$P(\mathbf{W}|\mathbf{D}) = \frac{P(\mathbf{D}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{D})}, \tag{3.33}$$

where $P(\mathbf{D}) \neq 0$ is the model evidence. Thus, the posterior distribution captures the most likely parameters given the observed data [83].

3.4.1.2 Bayesian Neural Networks

Bayesian Neural Networks are modelled using Bayesian inference (3.34) to assign probabilities to events, and thus capturing uncertainties in model predictions [103, 65, 120, 21], by considering the network weights as a probability distribution parameter instead of a deterministic value (like in traditional deep neural networks). The posterior probability of the weights given the input ($\mathbf{X}$) and the output ($\mathbf{Y}$-target/class) data can be formulated by Bayes' rule, which determines the relationship between the prior and the conditional probabilities defined by

$$P(\mathbf{W}|\mathbf{X},\mathbf{Y}) = \frac{P(\mathbf{Y}|\mathbf{X},\mathbf{W})P(\mathbf{W})}{P(\mathbf{Y}|\mathbf{X})}, \tag{3.34}$$

where $p(\mathbf{Y}|\mathbf{X},\mathbf{W})$ is the class conditional density (likelihood function), $P(\mathbf{Y}|\mathbf{X}) \neq 0$ acts as a normalizing constant for $P(\mathbf{W}|\mathbf{X},\mathbf{Y})$.

The calculation of the posterior $P(\mathbf{W}|\mathbf{X},\mathbf{Y})$ is not trivial, due to the fact that the density function $P(\mathbf{Y}|\mathbf{X})$ may not have a known analytical form, whereas the prior $P(\mathbf{W})$ can be specified from some previous knowledge and the likelihood conceivably obtained from the data. For this reason, in complex models - like deep neural networks - the posterior becomes intractable. Thus, a possible solution is to perform an approximation by means of variational inference [106, 233, 66, 157, 74, 22, 107, 21]. Nonetheless, variational inference still presents some challenges in terms of computa-

tional complexity, specially when dealing with large models and large quantities of data.

3.4.1.3 Variational Inference

As stated in Subsection 3.4.1.2, the posterior distribution may not have an analytical solution due the model evidence [21] is not known. Thus, a possible solution is to perform an approximation by means of a variational inference [106, 233, 157, 74, 22, 107, 21] that defines a new distribution $q_\theta(\mathbf{W})$, over the weights $\mathbf{W}$ and parameterized by θ *i.e.*, an approximation for the true posterior distribution $q_\theta(\mathbf{W}) \approx P(\mathbf{W}|\mathbf{X},\mathbf{Y})$, where $q_\theta(\mathbf{W})$ can be assumed to be a Gaussian and θ includes mean and variance for each weight (weights are independent) [22, 78].

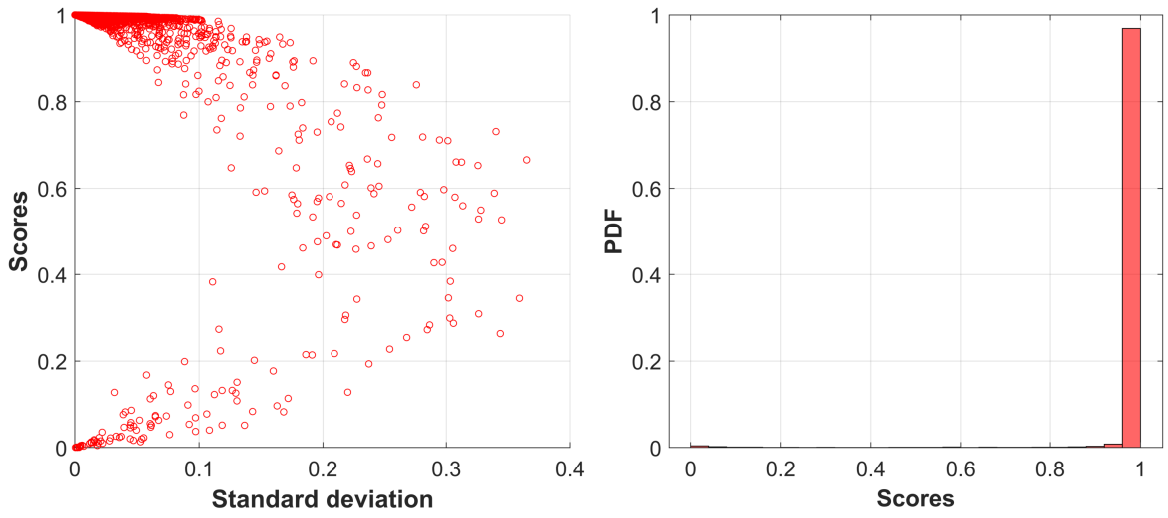
The difference between the approximate distribution and the true posterior distribution is measured by Kullback-Leibler (KL) divergence ($P(\mathbf{W}|\mathbf{X},\mathbf{Y})$ from $q_\theta(\mathbf{W})$), considering the parameter θ to be optimized *i.e.*, the mean and variance parameters are estimated by minimizing the KL [74]. Thus, the smallest divergence between $q_\theta(\mathbf{W})$ and $P(\mathbf{W}|\mathbf{X},\mathbf{Y})$ is obtained as a result of an optimal θ defined as $\theta^{opt} = arg\min_\theta KL[q_\theta(\mathbf{W})||P(\mathbf{W}|\mathbf{X},\mathbf{Y})$.

An example of the Bayesian Convolutional Neural Networks (BCNN) to classify cars, cyclists and pedestrians, using Inception V3 CNN with Bayesian layers from TensorFlow[4], is shown in Fig. 3.9. The results were analyzed through standard deviation of prediction scores from the car, cyclist, pedestrian, and unseen classes.

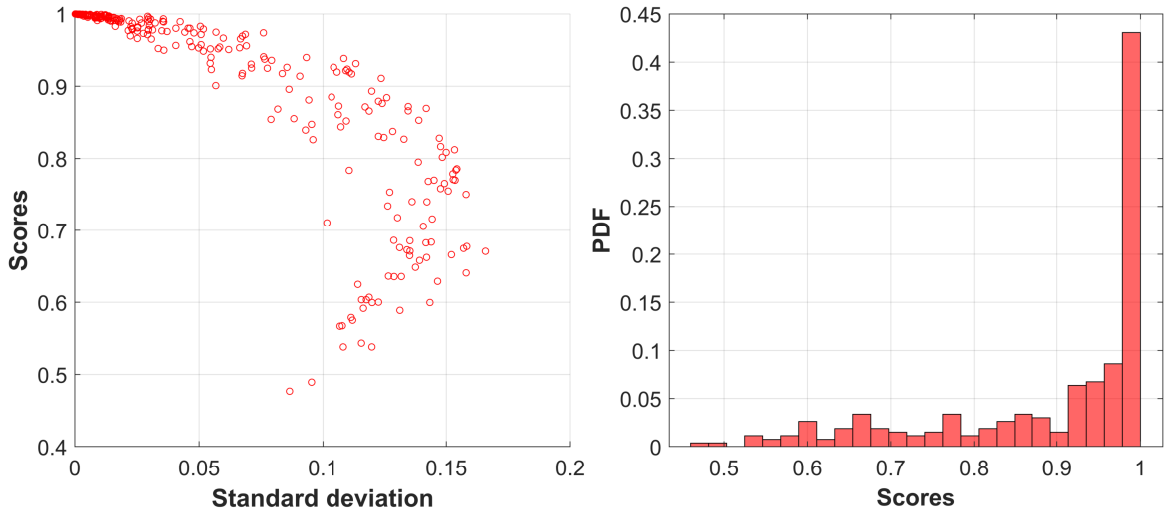
3.4.1.4 Point Estimation

From probabilities over a categorical distribution, $P(\mathbf{Y}|\mathbf{X},\mathbf{W})$, to an extent that the weights are the result of a learning process in order to explain the data, being $\mathbf{X} = \{x_1,\ldots,x_N\}$ the model input, and $\mathbf{Y} = \{y_1,\ldots,y_N\}$ the model output, where y_i is output corresponding to the input x_i, the result can be a point estimate obtained through the Maximum Likelihood and Maximum a-Posteriori.

[4]The platform/library to train Inception V3 BCNN was TensorFlow 2 and TensorFlow Probability, using the concept of flipout [233], where the formulation of the Bayesian neural network creates perturbations in the weights, and the expected negative log likelihood is computed by sampling the weights by means of Monte Carlo (approximation of the distribution integrating over the weights and bias), while KL divergence is approximated by the regularization term [106, 157, 22, 107]. For the sake of computational cost, the Inception V3 BCNN training has considered only dense layers (fully connected) as Bayesian layers.



(a) Classification scores for car, cyclist and pedestrian classes.



(b) Classification scores for unseen class.

Fig. 3.9 Classification using Inception V3 BCNN. For each classified object, 300 prediction samples were performed. The final result is these average of the predictions for each object. In the first line, the graphs represent the uncertainty in the predictions for the classes used in the training (car, cyclist and pedestrian), while in the second line represents the uncertainty and the overconfident prediction for the unseen class (person sitting, tree and stem) through the histogram.

The weights that maximize $P(\mathbf{Y}|\mathbf{X},\mathbf{W})$ through the Maximum Likelihood Estimation (MLE) is given by $w_{MLE} = arg\max_{w} P(\mathbf{Y}|\mathbf{X},\mathbf{W})$. Thus, the formulation of the

w_{MLE} can be defined as [74]

$$w_{MLE} = arg\max_{w} \prod_{i=1}^{N} P(y_i|x_i, \mathbf{W}). \tag{3.35}$$

Training to maximize the posterior $P(\mathbf{W}|\mathbf{X}, \mathbf{Y})$ is also possible. In this way, the model learns Maximum a-Posteriori (MAP), $w_{MAP} = arg\max_{w} P(\mathbf{W}|\mathbf{X}, \mathbf{Y})$, which includes the prior distribution over the weights. Thus, $w_{MAP} = arg\max_{w} P(\mathbf{Y}|\mathbf{X}, \mathbf{W})P(\mathbf{W})$ is given by

$$w_{MAP} = arg\max_{w} \prod_{i=1}^{N} P(y_i|x_i, \mathbf{W})P(\mathbf{W}). \tag{3.36}$$

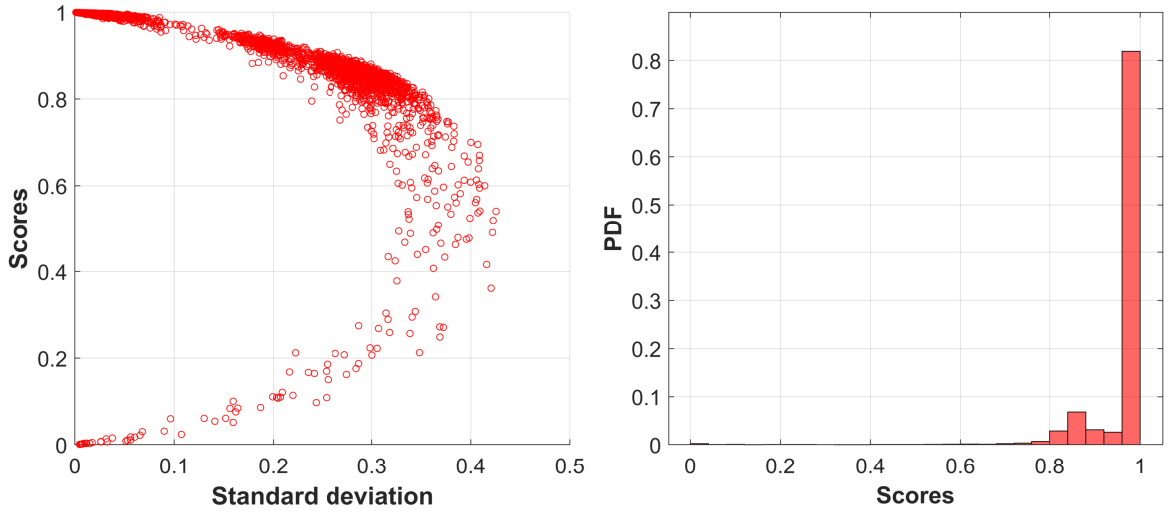
From the obtained weights by MLE and MAP functions, the predictions might be calculated by the model using the weights learned as $P(\hat{y}|\hat{x}) = P(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w_{MLE}})$ or $P(\hat{y}|\hat{x}) = P(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w_{MAP}})$. Notice that, although the Bayesian formulation takes distributions into account, MLE and MAP compute a single estimate rather than a distribution.

3.4.2 Monte Carlo Dropout

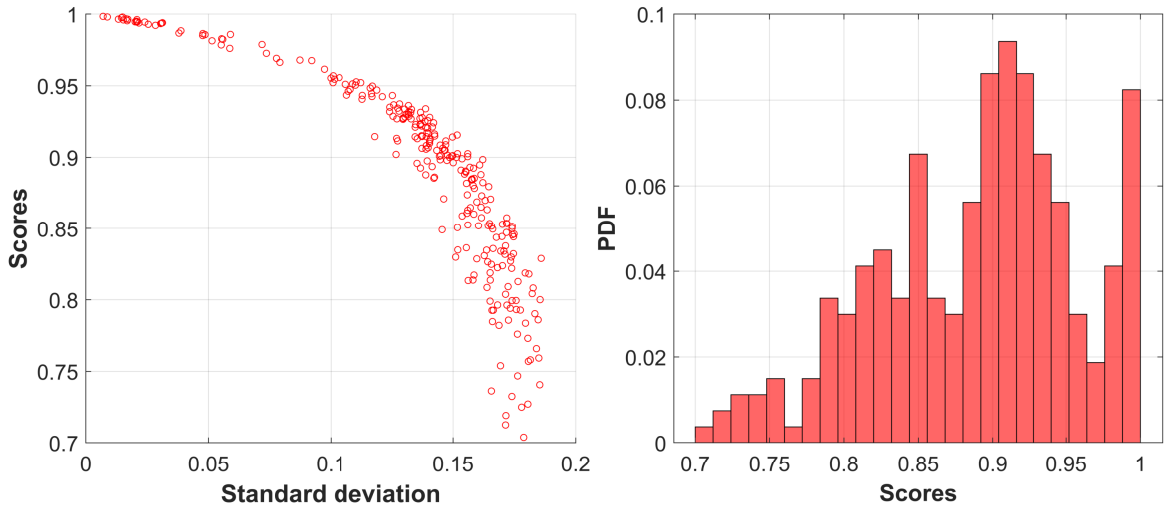
Dropout is a regularization technique [74, 206, 63], which might potentially be included in a neural network, contributing to avoid overfitting. It is usually used during training, but therefore emerges the question about what occurs if the dropout shall be used during test time? The answer is simple. The predicted values will not be deterministic *i.e.*, such values depend on which connections between the neurons will be chosen at random in procedure to perform the prediction. In fact, the same test sample forwarded several times in the network can have different predicted values.

The idea of applying dropout at test time is to obtain a probability distribution for each sample. According to the authors in [103, 66, 65], the multiple prediction values from the same sample can be analyzed as the model confidence *i.e.*, the model confidence is given by the predicted values variance and average, as illustrated in Fig. 3.10a, where 300 predictions were made for each object at test time using Inception V3 CNN.

Notably, Fig. 3.10a shows classes with scores around 85% and prediction uncertainties about 25% *i.e.*, predictions with high scores and low certainty in the result



(a) Classification scores for car, cyclist and pedestrian classes.



(b) Classification scores for unseen class.

Fig. 3.10 Monte Carlo Dropout to calculate the uncertainties in the predictions by means of standard deviation. For each classified object, 300 prediction samples were performed. The final result is these average of the predictions for each object. In the first line, the graphs represent the uncertainty in the predictions of objects classified as car, cyclist and pedestrian, while in the second line represents the uncertainty and the overconfident prediction for the unseen class (person sitting, tree and stem) through the histogram.

presented by the network. In addition, Fig. 3.10b highlights the scores from the unseen dataset. Such scores are values with excessive confidence and low uncertainties (scores around 100%). The ideal model should provide low score values and low uncertainty for objects in the unseen class, as well as not recognizing such objects. Interestingly,

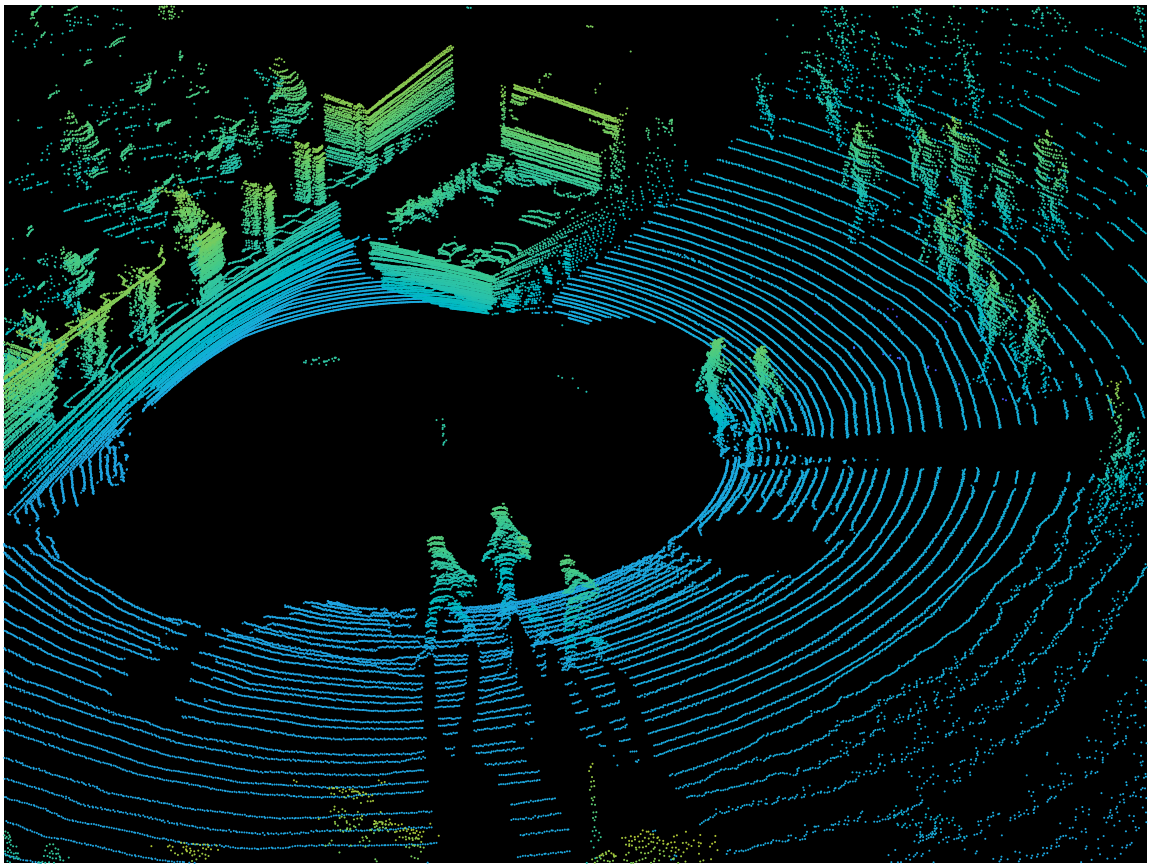


Fig. 3.11 Example of an image generated from the HDL-64E Velodyne 3D LiDAR. Image obtained from the Object Detection Evaluation of the KITTI dataset [69, 70].

the dropout approach at test time presents the robustness of the trained model, in other words, how reliable the model is. The disadvantage is the execution time because many forward passes have to be executed for each sample.

3.5 3D Point Cloud

The LiDAR is a sensing mechanism composed mainly of a laser and a scanner (scanning system), which outputs a set of measurements of objects in the surrounding environment. Such mechanism rotates 360° [69, 70, 98], allowing full-covered horizontal field of view. The lasers are vertically stacked creating multiple beams, consequently allowing the observation of the scenery at different heights and providing a set of 3D point cloud in 360°, as shown in Fig. 3.11.

The operating principle of LiDAR is based on the emission of laser pulses and their reflection after illuminating an object. The differences between the emissions and returns of pulses, both related to times and wavelengths, can be used to obtain a $3D$ representation of the objects that received and reflected the laser pulses.

The LiDAR sensor output (or scan) is a set of $3D$ points clouds (PCs) defined in a coordinate system as $\mathbf{P}^{3\mathbb{R}} = \{pc_1, \ldots, pc_n\}$, where each element $pc_{i=1,\ldots,n} = (x_L, y_L, z_L)_i$ is a LiDAR measured point that can be represented in Cartesian coordinates. In addition to the coordinate points (x_L, y_L, z_L), the LiDAR sensor can provide the reflectance value (intensity) [69, 70, 98].

3.5.1 Projecting 3D Point Cloud on the 2D Image-Plane

The set $\mathbf{P}$ is projected in the $2D$ image-plane reference system (RGB camera) *i.e.*, pc_i can be transformed into image pixel coordinates $(u, v)_i$, according to (3.37) and (3.38), considering that the calibration matrices between LiDAR and RGB camera are known [69, 70, 98].

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_i = P_{rect}^{(i)} R_{rect}^{(i)} T_{Lidar}^{cam} pc_i \tag{3.37}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_i = \underbrace{\begin{bmatrix} f_u^{(i)} & 0 & c_u^{(i)} & -f_u^{(i)} b_x^{(i)} \\ 0 & f_v^{(i)} & c_v^{(i)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & R_{12} & R_{13} & 0 \\ R_{21} & 1 & R_{23} & 0 \\ R_{31} & R_{32} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{Projection} \\ \text{matrix}}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{LiDAR to} \\ \text{camera}}} \underbrace{\begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix}}_{\substack{\text{LiDAR} \\ \text{coordinates}}} \tag{3.38}$$

where u and v represent the position of the pixel coordinates, P_{rect}^{i} is projection matrix, $R_{rect}^{(i)}$ is rotation matrix from camera reference, T_{Lidar}^{cam} is the matrix containing the rotation and translation matrices (LiDAR to camera), f_u and f_v are focal lengths, c_u

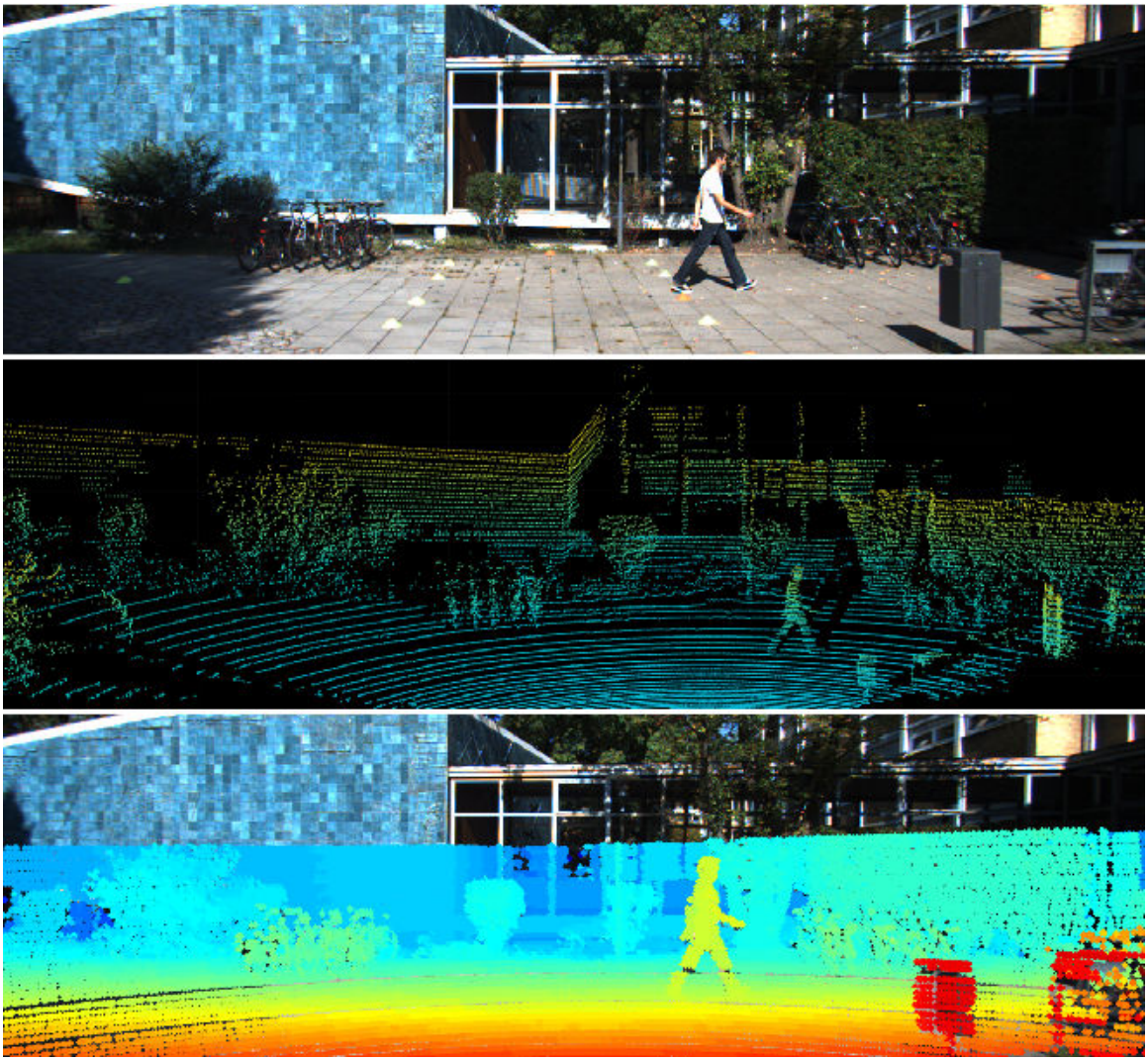


Fig. 3.12 Example of an image obtained through a passive sensor (RGB camera) in the first row, while the second row shows the same image in $3D$ point clouds, obtained by an active sensor (HDL-64E Velodyne $3D$ LiDAR). The last row shows the projection of the $3D$ point clouds in the $2D$ image-plane. Image obtained from the Object Detection Evaluation of the KITTI dataset [69].

and c_v are principal point coordinates, and $b_x^{(i)}$ denotes a baseline (in meters) with respect to camera zero reference. An example of projection of the $3D$ point clouds in the $2D$ image-plane is illustrated in Fig. 3.12.

3.5.2 Range-view and Reflectance-view Maps

The Range-view (RaV) and Reflectance/Intensity-view (ReV) maps - which are forms of representation - are constructed from (i) the depth and reflectance data of the $3D$ point clouds and (ii) the projected points on the $2D$ image-plane *i.e.*, the desired Rav/ReV maps have to approximate the image resolution (in pixels). However, the number of positions on the image-plane without projected points is high due to the data sparsity characteristic of the LiDAR. Therefore, in order to obtain a high-resolution map, it is necessary to estimate the values of RaV and ReV in unsampled positions of the maps. Such estimates can be performed by considering a mask C_{mask} of size $c \times c$ pixels, and by using the sliding window principle. The sampled point, center of C_{mask}, is weighted by the number of neighboring points defined by the mask size *i.e.*, the formulation combines the intensity and distance values of a pixels group which are inside the mask C_{mask}, being $c_0 = (c_h, c_v)$ the M center, which is the localization of interest, and $\hat{r_0}$ the value to be estimated at c_0 from the r_i (RaV or ReV), where c_h and c_v are the positions in the horizontal and vertical directions respectively [180].

Unsampled points estimate were performed using the Average (Ave), Minimum (Min), Maximum (Max), Inverse Distance Weighting (IDW), and Bilateral Filter (BF) formulations as described in [180]. The formulations for BF and IDW are given in (3.39) and (3.42) respectively, as follows

- BF:

$$\hat{r_0} = \frac{1}{W_{BF}} \sum_{i=1}^{n} G_{\sigma_s}(||c_0 - c_i||) G_{\sigma_r}(|r_0 - r_i|) r_i, \tag{3.39}$$

  where c_i is the sample located in the local region of interest, $W = \sum_{i=1}^{n} G_{\sigma_s}(||c_0 - c_i||) \times G_{\sigma_r}(r_0 - r_i)$ is a normalization factor that ensures the weighting terms sum to one, G_{σ_s} weights the point c_i inversely proportional to a distance (we used the Euclidean distance) to the position of interest c_0, and G_{σ_r} weights the sampled points from their values r_i. G_{σ_s} and G_{σ_r} were considered as

$$G_{\sigma_s} = \frac{1}{1 + (||c_0 - c_i||)} \tag{3.40}$$

$$G_{\sigma_r} = \frac{1}{1 + (|r_0 - r_i|)}. \tag{3.41}$$

- IDW:

$$\hat{r_0} = \frac{1}{W_{IDW}} \sum_{i=1}^{n} W_i(\mathbf{x}) r_i, \tag{3.42}$$

  where $W_{IDW} = \sum_{i=1}^{n} W_i(\mathbf{x})$, c_i is the sample located in the local region of interest, $W_i(\mathbf{c}) = d_i^{-p}$, $d = ||c_0 - c_i||$ is the Euclidean distance and p is a power parameter (positive real number).

For both IDW and BF we use the notation $|\cdot|$ for the absolute value and $\|\cdot\|$ for the Euclidean distance between pixel locations.

Figure 3.13 shows the RaV and ReV maps from the Ave, Min, Max, IDW, and BF formulations with respect to Fig. 3.12. In this case, the mask size M is 13×13.

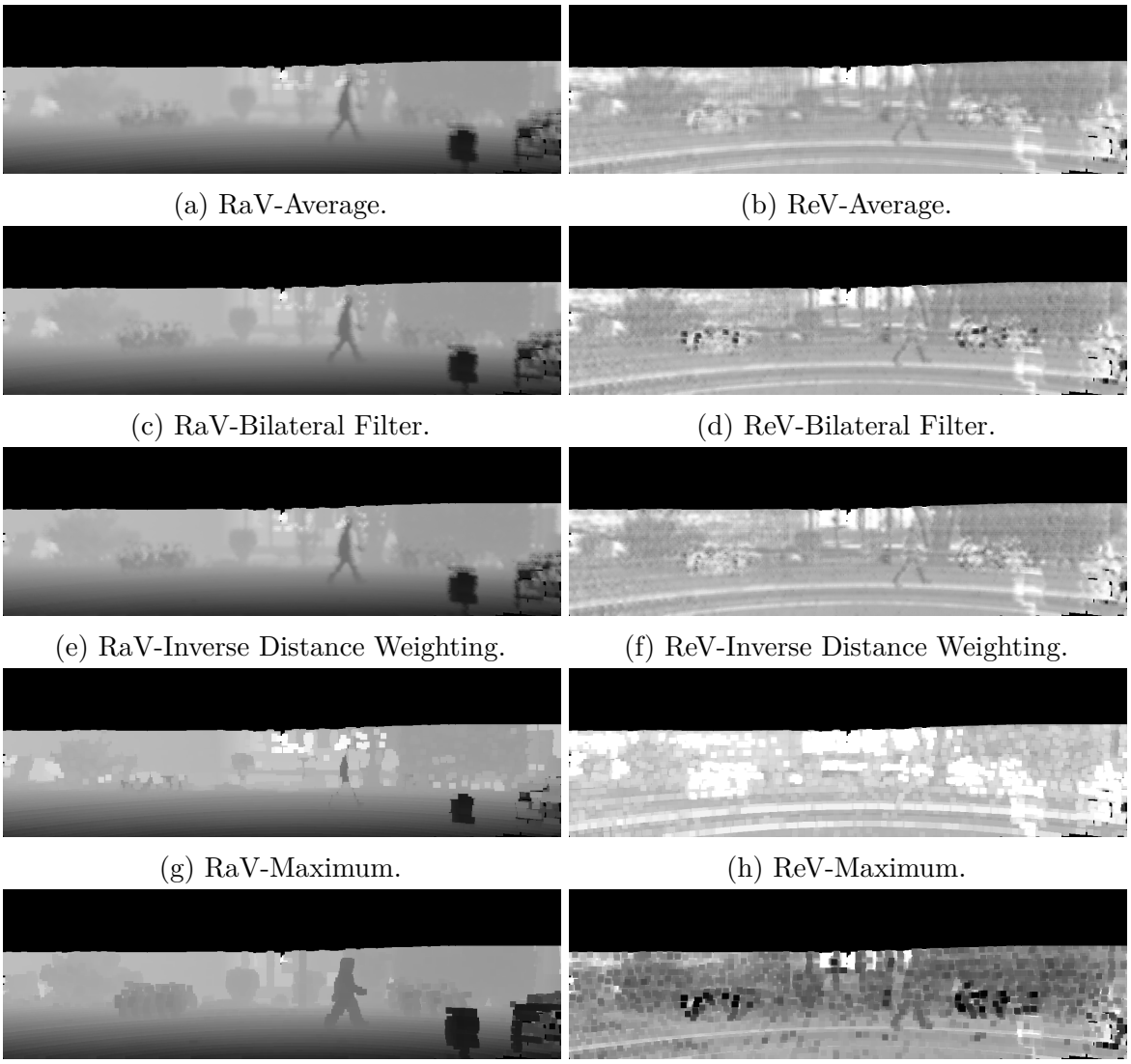
3.6 Object detection

Currently, the state-of-the-art in pattern recognition are defined by object detection models, which has become one of the most important areas of computer vision. Such models have as main objective to estimate the objects bounding boxes and the associated class/category scores, as illustrated in Fig. 3.14. Generally, such models estimate the $2D$ bounding boxes considering the coordinates of the center (C_W,C_H), width (W) and height (H) of the boxes. To be more precise, detection models also estimate the classification score, the predicted class, and (in some approaches) an objectness score (confidence threshold). The formulations to define the bounding boxes and the classification of objects depend on comparisons across thresholds between predicted and ground-truth bounding boxes. In summary, the formulations to determine the positions of the bounding boxes, classes, and confidence levels of the classes are direct or indirectly built upon the following concepts [189–191, 23],

- Anchor boxes: a bounding boxes set with predefined values. Such anchors may be defined with many 'fixed' sizes, in order to capture objects having different sizes, according to the central image of Fig. 3.15.

- Classification score: a score value for each class, or category, of the detected object.

- Objectness score (OS): parameter which defines whether a region in the image contains an object or not (*c.f.*, grid of the image on the left in Fig. 3.15). For



(a) RaV-Average. (b) ReV-Average.

(c) RaV-Bilateral Filter. (d) ReV-Bilateral Filter.

(e) RaV-Inverse Distance Weighting. (f) ReV-Inverse Distance Weighting.

(g) RaV-Maximum. (h) ReV-Maximum.

(i) RaV-Minimum. (j) ReV-Minimum.

Fig. 3.13 Range-view and reflectance-view maps using a mask size $M = 13 \times 13$.

each grid in the image, the network provides a set of bounding box with a given class but, such network will have to compute which is the best bounding box that represents a given object through a objectness threshold (τ_{Obj}). In other words, OS is used to evaluate which bounding box centered on a grid best represents the detected object. Depending on the τ_{Obj} value, some detected objects will still have more than one bounding box.

- Intersection over union (IOU): relationship between the area of intersection over the area of union between two boxes (overlap measure), for example, measure

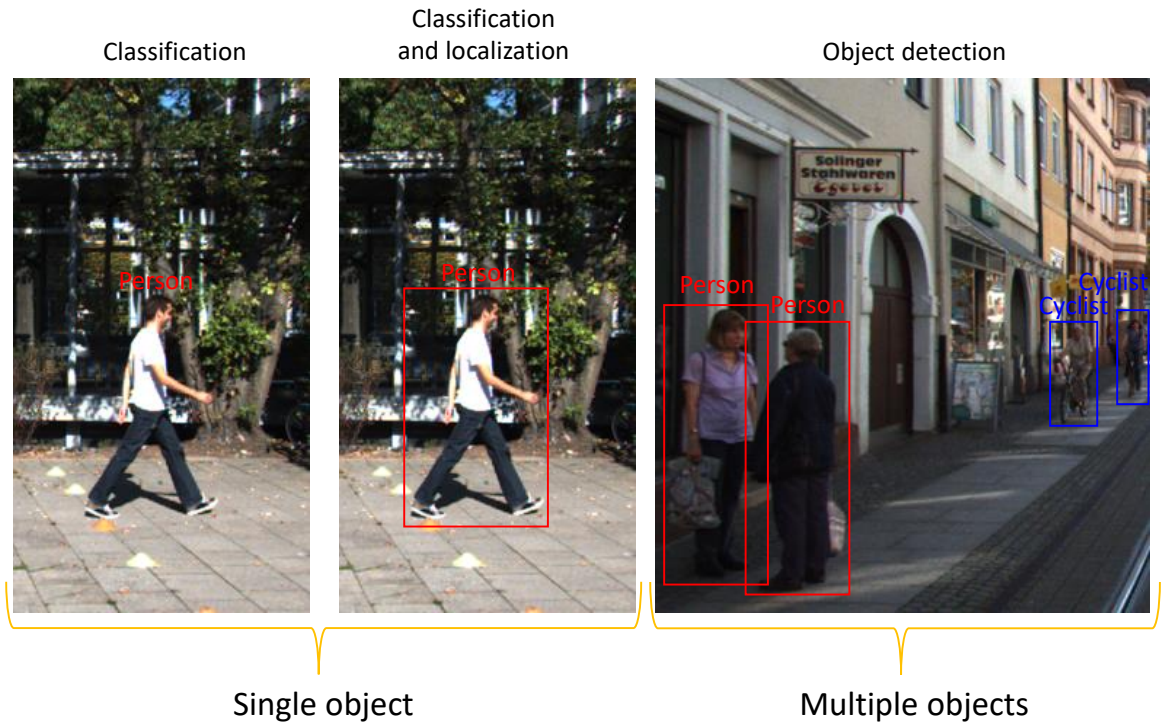


Fig. 3.14 Example of classified and detected objects with their respective $2D$ bounding boxes.

the predicted bounding box (B_p) with the ground-truth bounding box (B_{gt}), as expressed by

$$IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}, \tag{3.43}$$

where IOU measures the similarity between two sets and must be a value between 0 and 1. The closer IOU to 1, the better the detection result. However, a detected bounding box is said to be correctly if the value of the IOU is greater than a given threshold. Otherwise, the prediction is incorrect (*i.e.*, when the IOU is less than the threshold). Since the detectors also classify the detected object, then the IOU is calculated only for the detected bounding boxes that correspond to the same ground-truth class.

- Non-maximum suppression: the model can predict some candidate bounding boxes for the same object (according to the τ_{Obj}), so the model needs to filter the candidates through the criterion, as non maximum suppression (NMS), which suppresses the candidates least likely using the IOU metric, and considering

Fig. 3.15 Images with anchor boxes, and bounding boxes after NMS threshold. Source: https://pjreddie.com/darknet/yolov1/

a threshold (τ_{nms}). In fact, NMS loops through all classes, and for each class it checks for overlap (IOU) between all bounding boxes. After the NMS, the numbers of bounding boxes are reduced to obtain the best bounding box for each detected object, as the image on the right in Fig. 3.15.

After the NMS step, the detection of each object is defined as being positive or negative, and true or false, according to the following definitions:

- true positive (TP): a correct detection *i.e.*, ground-truth bounding box detected correctly.
- false positive (FP): wrong detection of an existing object or an wrong detection of a non-existent object.
- false negative (FN): a ground-truth not detected *i.e.*, the model failed to detect an object considered as ground-truth.
- true negative (TN): would be all bounding boxes that were not correctly detected, that is because this concept does not apply to object detection.

In short, such concepts are defined from the correct or incorrect detections (non-existing object or the erroneous detection of an existing object) of bounding boxes, as well as unidentified ground-truths.

The concepts of TP, FP and FN enable define metrics such as:

- Precision (P_r): relationship between true positives (TP) with the sum of detected positives ($TP + FP$).
- Recall (R_c): relationship between true positive (TP) with the total ground-truth positives ($TP + FN$).

- Precision-Recall Curve: curve which computes the precision-recall pairs for different thresholds, calculating the values of the accumulated TP and FP detections.

Among the various detection models, we chose to use YoloV4, which has become the state of the art in the COCO dataset [23].

YoloV4 "backbone" (feature extraction) considers cross-stage-partial-conections (CSP) *i.e.*, feature maps are divided into two parts, one part goes straight through the dense block, and the second part concatenates with the final result of the dense block, having as main network the Darknet53 [188, 189]. Even at the beginning of the network, just after generating the first feature maps, the "backbone" contains a layer designated as a spatial attention module (SAM), that is used to refine such maps.

The output of the last dense block is directed to the spatial pyramid pooling layers (SPP) [85] *i.e.*, a maximum pooling is applied with different filter sizes (sliding kernel), and generated feature maps are concatenated. Then such maps undergo convolution layers that perform upsample (top-down stream) and downsamples (bottom-up stream) in terms of maps sizes *i.e.*, the network contains feature map information coming from bottom-up and top-down streams, and this step is known as path aggregation network (PAN) [135], and such maps are concatenates from the different layers and goes through a fusion layer using max operation. The entire structure involving SSP and PAN is known as "neck".

The result of the fusion layer, aforementioned, is directed to the prediction layer, defined in the same way as YoloV3 [191] *i.e.*, the predictions consider the anchor boxes. This last step in the YoloV4 architecture is named "head", and the prediction output (sigmoid function to get the classification score values, and objectness scores) are three matrices with different sizes ($Y_m \times Y_m \times 3$ with m being an integer), as the first part in Fig. 3.16.

In fact, the final detection needs to encode the information from the matrices output through the concepts of intersection over union, NMS, and class threshold. Each position of the three matrices bring information of the bounding boxes (center, height and width measurements, OS, and classification score), according to the second part in Fig. 3.16.

The differences between YoloV4 and previous versions are the formulations such as dropout in the convolution layers (DropBlock regularization), and the types of connections between the different layers, such as cross-spatial-partial connections, pyramid pooling layers, path aggregation network.

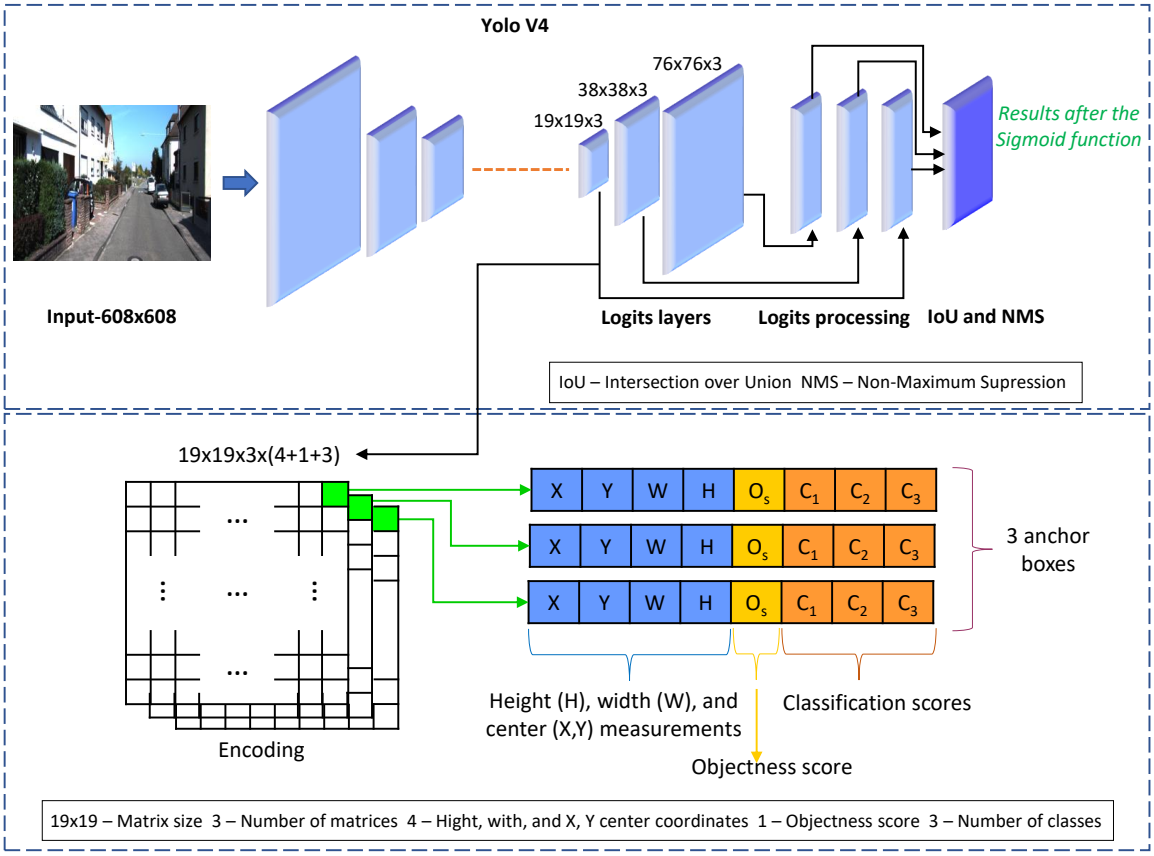


Fig. 3.16 Summary structure of YoloV4 considering the prediction layers, as well as the step of encoding to get bounding boxes, objectness scores, and classification scores.

Chapter 4

Inference and Fusion Strategies

Contents

Table 4.1 KITTI dataset for classification: number of objects per class and subsets.

KITTI dataset - 7481 Frames			
	Car	**Cyclist**	**Pedestrian**
Training	18103	1025	2827
Validation	2010	114	314
Testing	8620	488	1346

4.1 Datasets from Maps and 3D Point Clouds

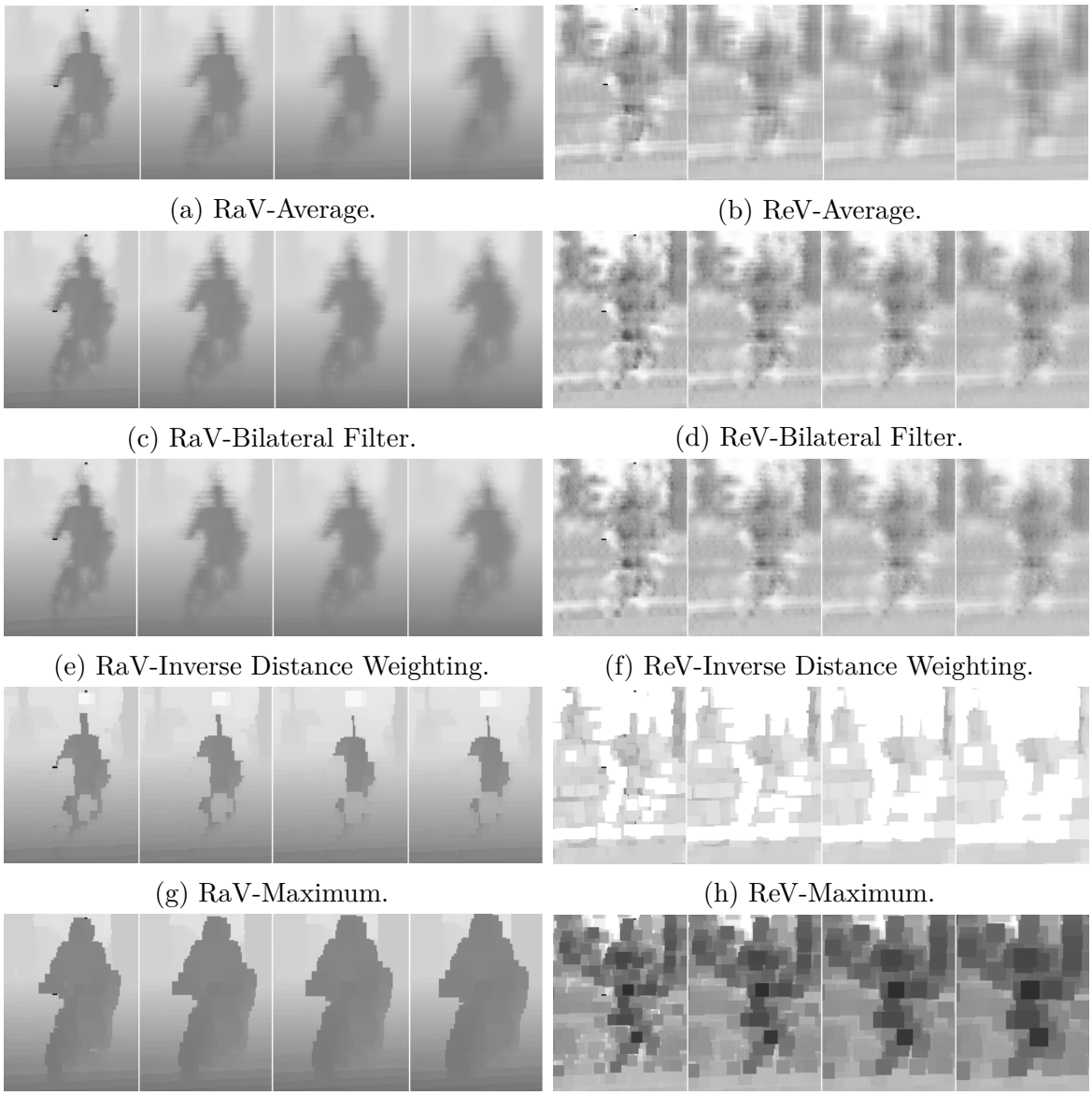
A key contribution to the growing improvement of perception systems for autonomous driving is the availability of representative datasets, considering different modalities such as RGB, LiDAR, and RADAR [27, 175, 205, 240, 172, 163] with several objects from different classes. Particularly, the classes of interest in this research are pedestrians, cars, and cyclists. In addition, some extra objects belonging to unseen/non-trained classes (object classes not used during training), for instance, "a person sitting", "tram", "truck", "van", "tree", "lamppost", "signpost", "bus", and "motorcycle" will be considered in the test/prediction phase, to verify the erroneous overconfidence from the prediction layers of the trained networks. The unseen classes can be understood as "adversarial" cases. Note, however, that this research does not target, in particular, adversarial network architectures.

4.1.1 Cropped Range-View and Reflectance-View Maps

From the range-views and reflectance-views maps [149, 151], new datasets were created using different mask size C_{mask}, as shown in Fig. 4.1. The maps only used the $3D$ point clouds data *i.e.*, the camera calibration data were solely used to visualize the projected $3D$ points on the $2D$ image-plane.

Each line in Fig. 4.1 represents a configuration to perform the upsample of the projected $3D$ point clouds on the $2D$ image-plane, as described in Subsection 3.5.1. From left to right, the masks size were $C_{mask} = 9 \times 9$, $C_{mask} = 11 \times 11$, $C_{mask} = 13 \times 13$ and $C_{mask} = 15 \times 15$, respectively.

From the RGB images, RaV, and ReV, a total of 41 new sub-datasets of objects were generated (five formulations for upsample and four mask sizes, and RGB images). The number of objects per class in each dataset is shown in Table 4.1, considering KITTI dataset as baseline.



(a) RaV-Average. (b) ReV-Average.

(c) RaV-Bilateral Filter. (d) ReV-Bilateral Filter.

(e) RaV-Inverse Distance Weighting. (f) ReV-Inverse Distance Weighting.

(g) RaV-Maximum. (h) ReV-Maximum.

(i) RaV-Minimum. (j) ReV-Minimum.

Fig. 4.1 Range-view and reflectance-view maps obtained from the BF, IDW, AVE, MAX and MIN formulations using different mask sizes: $C_{mask} = 9 \times 9$, $C_{mask} = 11 \times 11$, $C_{mask} = 13 \times 13$ and $C_{mask} = 15 \times 15$, from the left to right, respectively.

4.1.2 Cropped and Upsampled 3D Point Clouds

The crop of the $3D$ objects (point clouds) is obtained through the $2D$ bounding boxes, after the projections of the point clouds into pixel-coordinates frame [152]. The LiDAR points, and so the respective bounding boxes, that have been lying outside the camera's field of view were eliminated from the image-plane as shown in Fig 4.2. Thus,

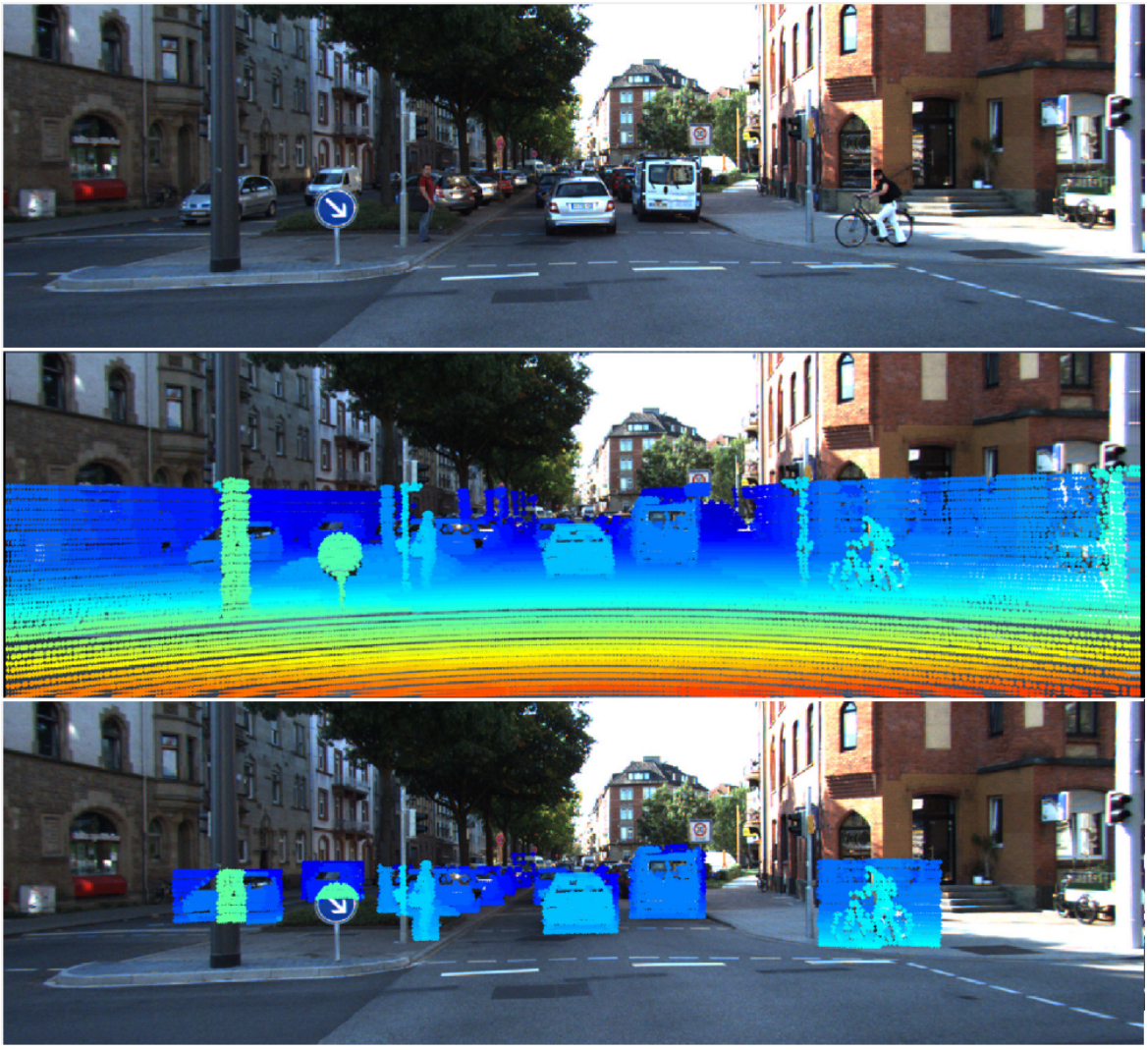


Fig. 4.2 Example obtained from KITTI 2D Object Detection Dataset showing the environment as "observed" by AV/IV sensors. The 3D point clouds are coloured proportionally to the measured range. In the last row, we can see the projected point sets in the neighbor region of pedestrians, vehicles, and a cyclist.

using Algorithm (1), the 3D points that have generated the projected points *i.e.*, inside the bounding boxes, are then considered for the upsampling phase. Nonetheless, there are 3D points that do not belong to the cropped 3D objects therefore, such points are defined as backgrounds or foregrounds points, as illustrated in Fig. 4.3, and can be removed through a clustering technique based on the distance between points in order to define which points belong to the respective object, as indicated in Algorithm (2).

Algorithm 1: Cropped $3D$ Point Cloud.

Input: Data from the LiDAR sensor and $2D$ bounding boxes.
Output: Cropped $3D$ point clouds without cluster.
Getting the 3D point clouds
$pc \leftarrow OpenLiDAR(data)$;
$indices \leftarrow pc(:,1) < 5$; /* Points that do not belong to the $2D$ image-plan are removed (the value is an approximation) */
$pc(indices,:) \leftarrow [\ ]$;
Project PC for image-plane
$pc_{proj} \leftarrow P_{rect} R_{rect} T_{LiDAR}^{Cam} PC$; /* Equation 3.38 */
$pc_{proj}(:,1) \leftarrow pc_{proj}(:,1)/pc_{proj}(:,3)$;
$pc_{proj}(:,2) \leftarrow pc_{proj}(:,2)/pc_{proj}(:,3)$;
Defining the points inside the bounding box
$Boxes = [x_{min} \quad y_{min} \quad x_{max} \quad y_{max}]$;
$indices \leftarrow [\ ]$;
for $i \leftarrow 1 : Size(pc_{proj})$ **do**
  if $(pc_{proj}(i,1) >= Boxes(1)$ **and** $pc_{proj}(i,1) <= Boxes(3) + 1)$ **and** $(pc_{proj}(i,2) >= Boxes(2)$ **and** $pc_{proj}(i,2) <= Boxes(4) + 1)$ **then**
    $indices \leftarrow [indices;\ i]$
  end
end
$PC_{WithoutCluster} = pc(indices,:)$;

An elemental disadvantage of LiDAR mapping is the inability to provide the same number of points for the detected objects, because the distance to the objects varies. So, to standardize the number of range points that characterizes the objects, six upsampling strategies have been carried out (after the clustering step), with different number of points: 64, 128, 256, 512, 1024 and, 2048. Consequently, six additional point cloud datasets were created and then used for training separately. Figure 4.4 shows an example of the upsampled output obtained for an object (a pedestrian).

The LiDAR-points upsampling approach was performed using the k-nearest neighbors, initially considering k equal to three and the Euclidean distance of the point-coordinates (x, y, z), (x, y), (x, z) and (y, z) *i.e.*, the upsample was performed after calculating four different values of distances. The sampled point is obtained from the average of the coordinates. The upsampling implementation controls the value of k; therefore, k increases gradually if the object's upsample does not reach the maximum quantity of points determined. On the other hand, if an object has the number of points higher than the predetermined, then a downsample is randomly applied to the cluster of points.

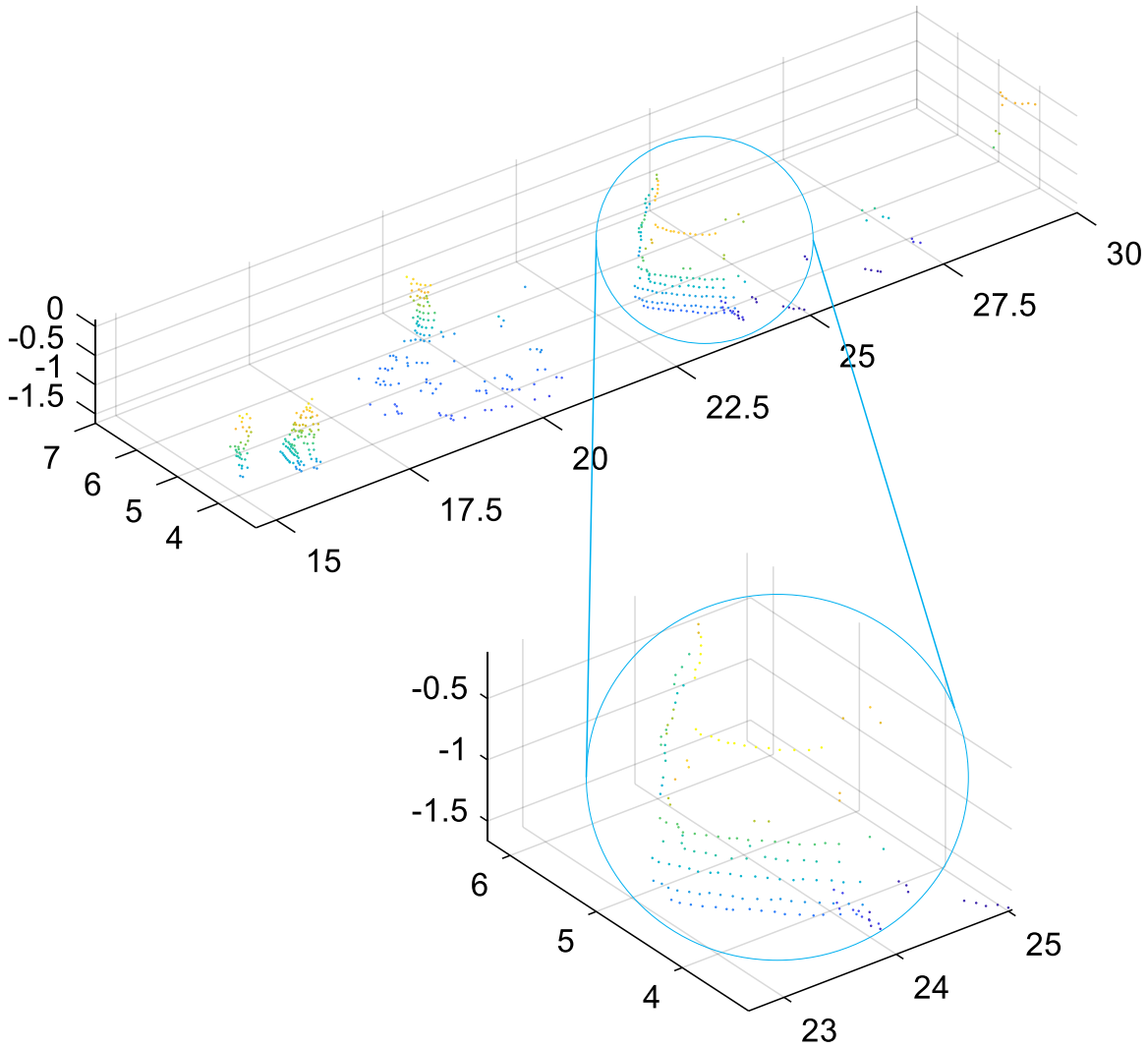


Fig. 4.3 Example of cropped $3D$ object without cluster at the top of the figure, presenting background and foreground points. At the bottom of the figure, there is the object after the clustering operation.

Algorithm 2: Cluster.

Input: Cropped $3D$ point cloud and distance between points.
Output: $3D$ point clouds with a cluster.
Compute the distance
$reference \leftarrow 0$;
$Dist_{pc} \leftarrow EuclideanDistance(PC_{WithoutCluster}, reference)$;
$indice \leftarrow [1:1:Size(PC_{WithoutCluster})]$;
$Dist \leftarrow [Dist_{pc} \quad indice]$;
$Dist \leftarrow SortRows(Dist, 1)$; /* Shortest distance to longest distance */
Compute the cluster
$distance \leftarrow 0.25$;
$id_{cluster} \leftarrow Zeros([Size(PC_{WithoutCluster}),\ 1])$
$id_{master}(1) \leftarrow 1$;
for $i \leftarrow 2 : Size(PC_{WithoutCluster})$ **do**
  if $Dist(i,1) - Dist(i-1,1) <= distance$ **then**
    $id_{cluster} \leftarrow id_{master}$;
  else
    $id_{master} \leftarrow id_{master} + 1$;
    $id_{cluster} \leftarrow id_{master}$;
  end
end
Check the cluster and compute the histogram count
$Cluster \leftarrow Unique(id_{cluster})$;
$HC \leftarrow HistogramCount(id_{cluster}, Cluster)$;
$confidence \leftarrow 1$ /* Confidence level */
$cl \leftarrow Size(HC)$ /* Number of clusters in the sample */
for $i \leftarrow 1 : Size(cl)$ **do**
  $HC(i) \leftarrow HC(i) * \left(confidence - \frac{(i-1)}{cl}\right)$
end
$[ClusterCount \quad Position] \leftarrow Max(HC)$;
$ct \leftarrow 1$;
for $i \leftarrow 1 : Size(id_{cluster})$ **do**
  if $id_{cluster}(i) == Cluster(Position)$ **then**
    $PC_{Cluster}(ct,:) \leftarrow PC_{WithoutCluster}(Dist(i,2),:)$;
    $ct \leftarrow ct + 1$;
  end
end

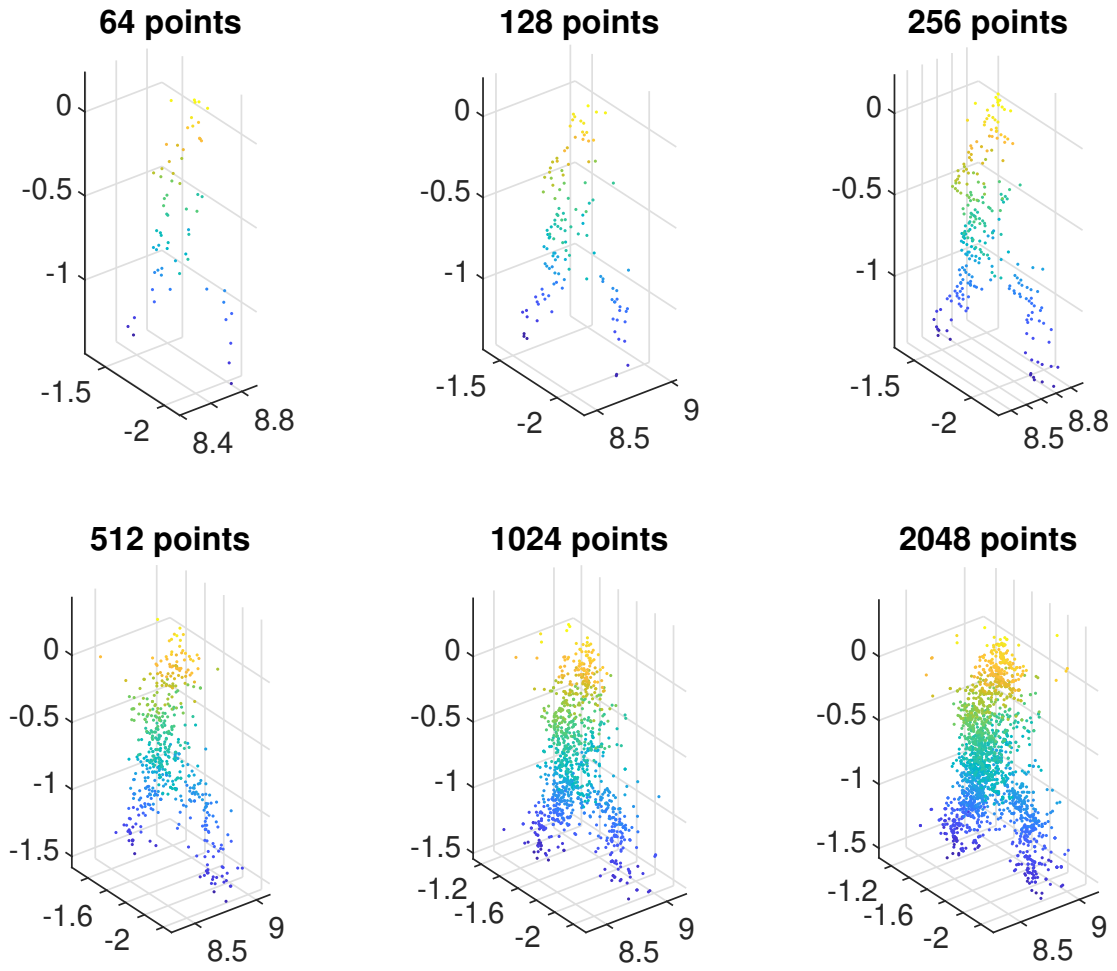


Fig. 4.4 Point cloud after the upsample. The same object (pedestrian) with 64, 128, 256, 512, 1024 and, 2048 points.

4.2 Probabilistic Inference

This section presents the formulations to reduce overconfident predictions, through Maximum Likelihood (*ML*) and Maximum a-Posteriori (*MAP*) functions, based on the Bayes' rule (3.33), and using expressions (3.35) and (3.36) presented in Section 3.4.1.4, including non-parametric and parametric modeling to define the posterior probability, likelihood function, and prior probability as well.

Similarly to (3.32), the output scores of a classification system, denoted $\mathbf{Sc} = \{sc_1, \ldots, sc_{nc}\}$, with nc classes, can be defined as a *rv*, considering $\mathbf{C} = \{c_1, \ldots, c_{nc}\}$ as the set of classes *i.e.*, $\mathbf{Sc}$ is dependent on the variable $\mathbf{C}$ for the formulation of *ML* and *MAP*. Therefore, the density function is conditional to $\mathbf{C}$ [169] and, from the (3.32) yields

$$f_{Sc}(Sc = \xi|\mathbf{C}) = \lim_{\Delta\xi \to 0} \frac{\mathbf{P}\{\xi \leq \mathbf{Sc} \leq \xi + \Delta\xi|\mathbf{C}\}}{\Delta\xi}. \tag{4.1}$$

Through the Bayes' rule, the posterior probability $P(\mathbf{C}|\mathbf{Sc})$ is given by

$$P(\mathbf{C}|\mathbf{Sc}) = \frac{P(\mathbf{Sc}|\mathbf{C})P(\mathbf{C})}{P(\mathbf{Sc})}, \tag{4.2}$$

where $P(\mathbf{Sc}|\mathbf{C})$ is the likelihood, $P(\mathbf{C})$ is the prior probability and $P(\mathbf{Sc}) \neq 0$ is the model evidence, considering the pior and likelihood known. From the law of the total probability [21], (4.2) can be rewritten using the *per-class* expression,

$$P(c_i|\mathbf{Sc}) = \frac{P(\mathbf{Sc}|c_i)P(c_i)}{\sum_{i=1}^{nc} P(\mathbf{Sc}|c_i)P(c_i)}. \tag{4.3}$$

Given (4.3), an inference can be made on the test set about the "unknown" *rv* $\mathbf{C}$ from the dependence with $\mathbf{Sc}$ *i.e.*, the value of the posterior distribution of $\mathbf{C}$ is determined after observing the value of $\mathbf{Sc}$.

4.2.1 Softmax and Sigmoid as Posterior Probability

Softmax and Sigmoid functions are generally used as prediction functions to classify a given input *i.e.*, they act as decision-making. Such non-linear functions can be taken

in the probabilistic context [21], considering the concept of Bayes' theorem, according to (4.3), and the decision problem for problems of two or more classes [21].

Defining a two-class problem, the posterior probability can be defined for each class as follows,

$$P(c_1|\mathbf{Sc}) = \frac{P(\mathbf{Sc}|c_1)P(c_1)}{P(\mathbf{Sc}|c_1)P(c_1)+P(\mathbf{Sc}|c_2)P(c_2)}, \tag{4.4}$$

$$P(c_2|\mathbf{Sc}) = \frac{P(\mathbf{Sc}|c_2)P(c_2)}{P(\mathbf{Sc}|c_1)P(c_1)+P(\mathbf{Sc}|c_2)P(c_2)}. \tag{4.5}$$

Dividing the numerator and denominator of (4.4) by $P(\mathbf{Sc}|c_1)P(c_1)$, the posterior for class 1 is given by:

$$P(c_1|\mathbf{Sc}) = \frac{\frac{P(\mathbf{Sc}|c_1)P(c_1)}{P(\mathbf{Sc}|c_1)P(c_1)}}{\frac{P(\mathbf{Sc}|c_1)P(c_1)}{P(\mathbf{Sc}|c_1)P(c_1)}+\frac{P(\mathbf{Sc}|c_2)P(c_2)}{P(\mathbf{Sc}|c_1)P(c_1)}} = \frac{1}{1+\frac{P(\mathbf{Sc}|c_2)P(c_2)}{P(\mathbf{Sc}|c_1)P(c_1)}}. \tag{4.6}$$

Considering the exponential function and natural logarithm, the expression (4.6) is given by (4.7),

$$P(c_1|\mathbf{Sc}) = \frac{1}{1+e^{-\xi}} = \sigma(\xi), \tag{4.7}$$

where $\xi = ln\left(\frac{P(\mathbf{Sc}|c_1)P(c_1)}{P(\mathbf{Sc}|c_2)P(c_2)}\right)$, and $\sigma(\xi)$ is defined as the logistic sigmoid function, while the inverse of such a function is given by $\xi = ln\left(\frac{\sigma}{1-\sigma}\right)$ and is defined as logit Function [21]. Note that (4.7) is the posterior probability rewritten in an alternative form, and can also be deduced in the same way for $P(c_2|\mathbf{Sc})$.

For a classifier with more than two classes, the posterior probability in (4.3) can be rewritten based on a softmax function, re-applying the exponential and the natural logarithm [21]. Thus, the posterior probability can be expressed as,

$$P(c_i|\mathbf{Sc}) = \frac{e^{\xi_i}}{\sum_{i=1}^{nc} e^{\xi_i}} = \sigma(\xi_i), \tag{4.8}$$

where $\xi_i = ln(P(\mathbf{Sc}|c_i)P(c_i))$ and $\sigma(\xi_i)$ is defined as softmax function (normalized exponential) *i.e.*, $P(c_i|\mathbf{Sc})$ was rewritten as being an exponential. Furthermore, (4.8) is considered a multiclass generalization of the logistic sigmoid function [21].

Expressions (4.7) and (4.8) were obtained considering probabilistic generative models in classifications[1] *i.e.*, the class-conditional densities and the prior are modeled, and then the posterior probabilities are computed through the Bayes' theorem [21]. In other words, first we have to determine the class-conditional density $P(\mathbf{Sc}|c_i)$ for each class c_i individually and then the class prior probability $P(c_i)$. Equivalently, the joint distribution $P(\mathbf{Sc}|c_i)$ can be directly modeled and later normalized to obtain the posterior probability. In fact, the classification result is given through two stages, the first being inference[2] and the second being decision-making[3].

Alternatively, many traditional approaches to classification problems are of the type called discriminative models[4] or discriminant functions[5] [21]. The first tries to model a posterior $P(c_i|\mathbf{Sc})$ directly using a parametric model in the inference stage, and consequently optimizing such parameters through the training set. Given the posterior model, for each new entry, it assigns a top-class label. The second case *i.e.*, discriminant function, the approach defines a function which uses the training data to map each entry directly to a certain class (input-output mapping), and according to Bishop [2006] the "probabilities play no role" *i.e.*, it is not possible to access posterior probabilities $P(c_i|\mathbf{Sc})$. In this case the inference and decision stages are into a single learning algorithm.

The result achieved by a machine learning algorithm, such as a classifier considering the prediction as the logistic function or softmax function, should be carefully analyzed, so that the prediction result is not considered as a probabilistic value. To obtain probabilistic results, the structure of the learning algorithms must encompasses probabilistic formulations. Through Bayesian inference it is possible to define the predicted values as being probabilistic, using the training data to model the class-conditional density (likelihood function) and a prior probability. An example of this was proposed in this thesis considering neural networks trained with the softmax function. However, the modeling of likelihood functions and prior probabilities of each class were obtained with

[1] Naive Bayes, Bayesian networks and Hidden Markov Models.
[2] Distribution modeling.
[3] Classification.
[4] Logistic regression and support vector machine.
[5] Traditional neural networks and k-nearest neighbors.

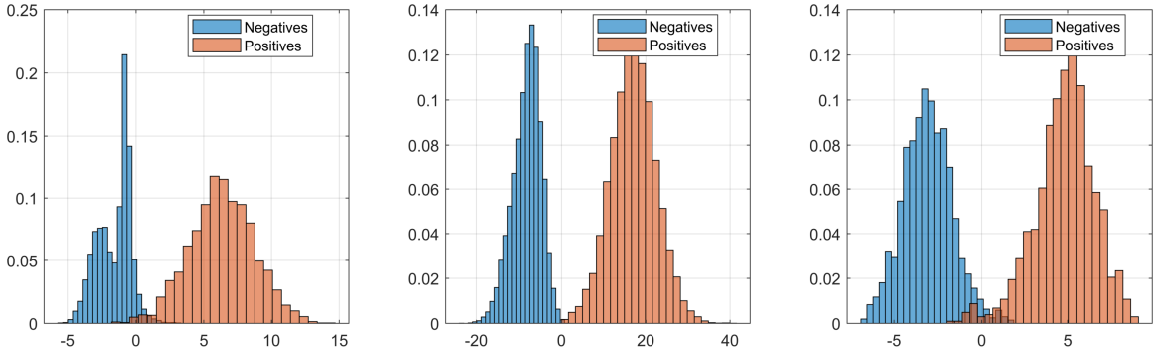


Fig. 4.5 Probability density functions (PDFs), using normalized histograms, for the logit layers data on the training sets of the KITTI datasets. The graphs are organized from left-right by classes (pedestrian, car and cyclist, where the positives are in orange) using the RGB modality.

the values before the prediction layers and after the trained *i.e.*, before the softmax function (SM) or sigmoid function (SgM).

4.2.2 ML and MAP Functions

Generally, the distribution over the values at the logit layer are far more appropriate to represent a PDF, as shown in Fig. 4.5, rather than the score values out of the softmax/sigmoid. Therefore, the basis argumentation is that ML and MAP functions are more adequate to perform probabilistic inference in regard to permitting decision-making under uncertainty, which is particularly relevant in autonomous driving and robotic perception systems. Thus, the ML and MAP functions make inference based on PDFs obtained from the logit layer prediction scores, where the densities are extracted by using the training set. This is illustrated in Fig. 4.6, where the horizontal axes represent the random variable **Sc** (from the logit layer) and the vertical axes are proportional to the class-conditional probability.

As expressed in (4.3), the posterior probability depends on the class-conditional probability (likelihood function) and on the prior probability *i.e.*, the MAP estimation is dependent on a distribution of both densities, while ML only depends on $P(\mathbf{Sc}|\mathbf{C})$ because, in the ML case, $P(\mathbf{C})$ is usually assumed to be uniform and identically distributed.

The probabilities $P(\mathbf{Sc}|\mathbf{C})$ are modeled by means of non-parametric estimates over the predicted scores of the logit layer for each class on the training set, as showed in the first column of Fig. 4.6. Each predicted value in the test set from the logit

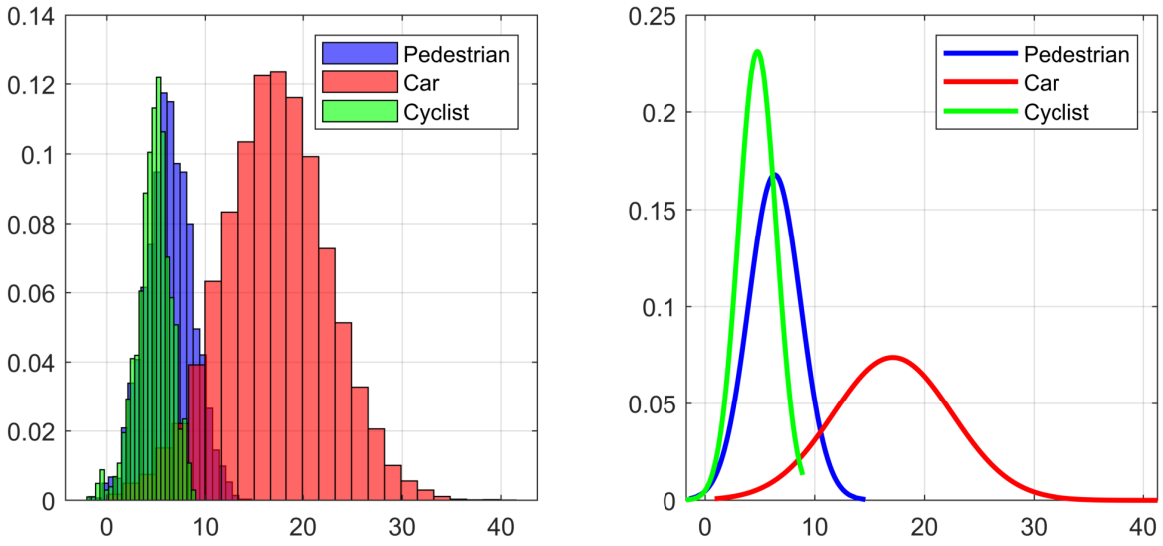


Fig. 4.6 From left-right respectively, normalized histograms and Gaussian distributions were calculated with the logit layer values for each class on the training set (RGB modalities from KITTI Dataset).

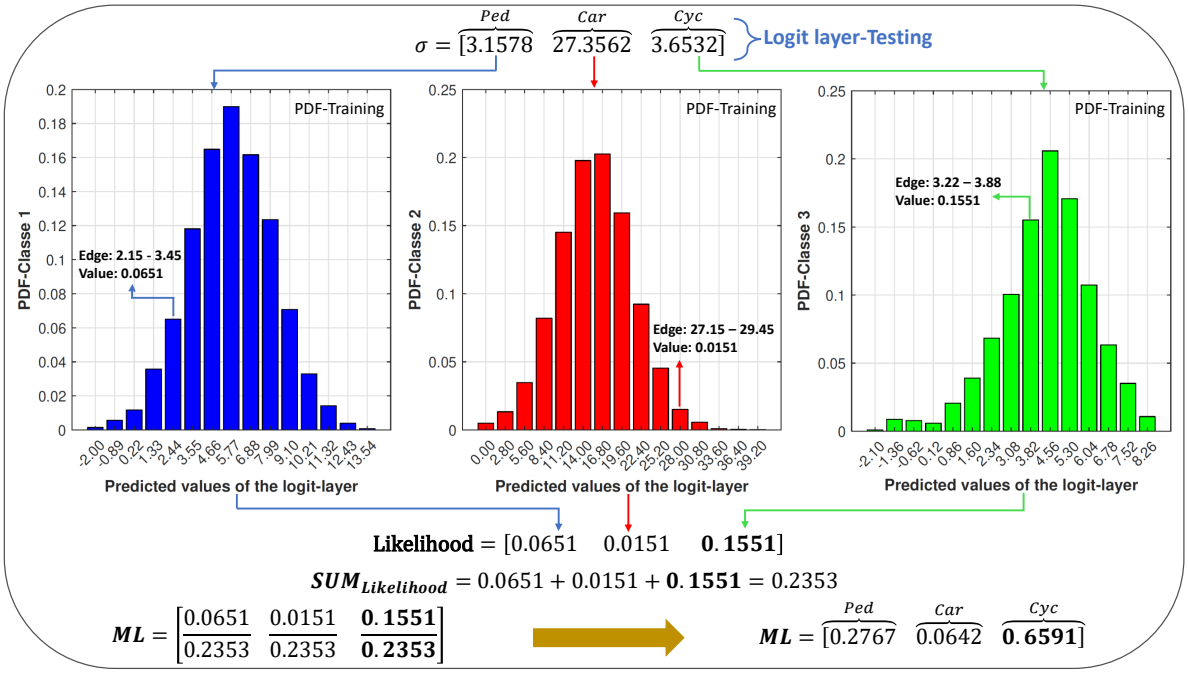
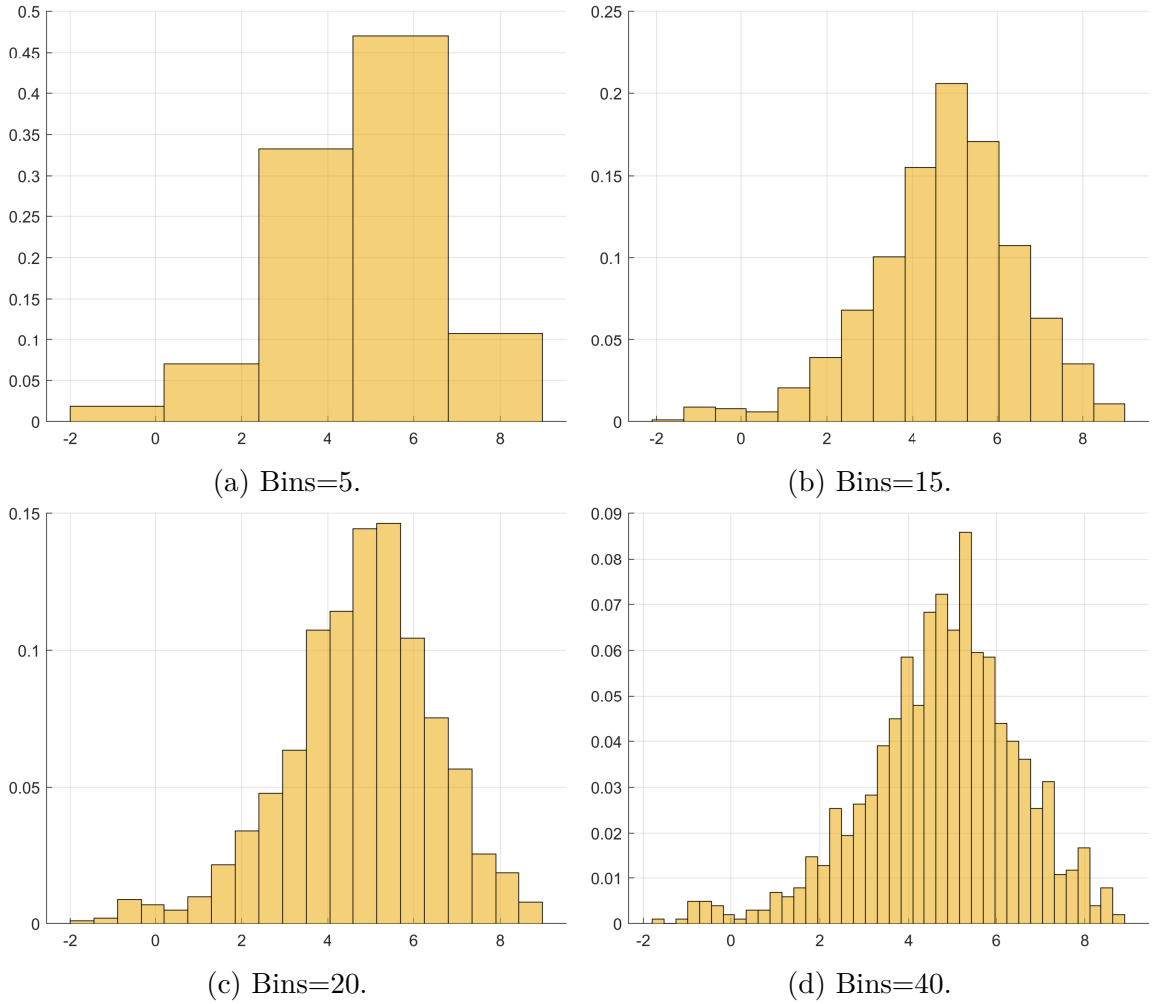


Fig. 4.7 Example of probability values of a normalized histogram generated with the training data of the logit layer.

layer has a score value corresponding to its bin range in the respective class histogram (normalized histogram from the training set), which is illustrated in Fig. 4.7.



(a) Bins=5.

(b) Bins=15.

(c) Bins=20.

(d) Bins=40.

Fig. 4.8 The greater the number of bins, the greater the number of variations *i.e.*, the greater the number of bumps.

Histograms are graphical ways of summarizing or describing a variable in a simple way, in other words, histograms show how variables (in this case, the network's logits) are distributed, also histograms provide information about the frequencies of observations. The number of bins $(nbins)$[6] determines the smoothness of the histogram [198], as seen in Fig. 4.8. As in C. Bishop [21], "we can view the histogram as a simple way to model a probability distribution given only a finite number of points drawn from that distribution".

For our methodology, we have chosen *nbins* empirically to guarantee a result very close to or better than the results provided by the *SM* and *SgM* layers, in order to maintain a smooth histogram.

[6]Often, the number of bins of a histogram are chosen to have the same width.

Regarding the *MAP* function, the prior is modeled by a Gaussian distribution that guarantees prediction values to be smoother as well, as observed within the second column of Fig. 4.6. Thus, $P(\mathbf{c}) \sim \mathcal{N}(\mathbf{Sc}|\mu, \sigma^2)$ with mean μ and variance σ^2 is calculated per class, using the training set comprising the variable *Logits*.

The normal distribution is feasible for modeling an unknown distribution, by reason of its maximum entropy[7]. Hence, the greater entropy might guarantee a more informative distribution and, simultaneously, less confident information around the mean, that is, it contributes to the reduction of the overconfident inferences. Therefore, a Gaussian distribution was defined for prior $P(c_i)$ to express a high degree of uncertainty in the value of variable $\mathbf{C}$ before observing the data. Furthermore, a prior distribution with high entropy is said to be a prior distribution with high variance [21].

The use of different densities to represent the distributions, Gaussians and normalized histogram in this case, aims at capturing potentially complementary information from the training data as shown in Fig. 4.6.

Additionally, to avoid the "zero" probability problem, as well as to incorporate some controlled uncertainty levels in the final prediction, the Additive Smoothing method [223, 32, 131] is implemented during the *ML* and *MAP* predictions. The value assigned for the Additive Smoothing, λ (selected as a value not too great or small), does not depend on specific information from the training dataset. This value was determined empirically *i.e.*, by observing which value would preserve approximately the "original" distribution without compromising the final result. The estimated probabilities with the Additive Smoothing are shown in (4.9) and (4.10) *i.e.*, a small correction is incorporated into the *ML* and *MAP* estimate. Consequently, no prediction will have a "zero" probability, despite being unlikely.

ML function is straightforwardly calculated by normalizing $P(\mathbf{Sc}|\mathbf{C})$ by the $P(\mathbf{Sc})$ during the prediction phase, as in (4.9), since the priors $P(\mathbf{C})$ are set uniformly and identically distributed for the set of classes $\mathbf{C}$,

$$ML = arg\max_{i} \frac{(P(\mathbf{Sc}|c_i) + \lambda)}{\sum_{i=1}^{nc} (P(\mathbf{Sc}|c_i) + \lambda)}. \tag{4.9}$$

[7]In the case of continuous distribution, the distribution that maximizes the entropy is a Gaussian distribution. Thus, entropy increases as the variance increases [21].

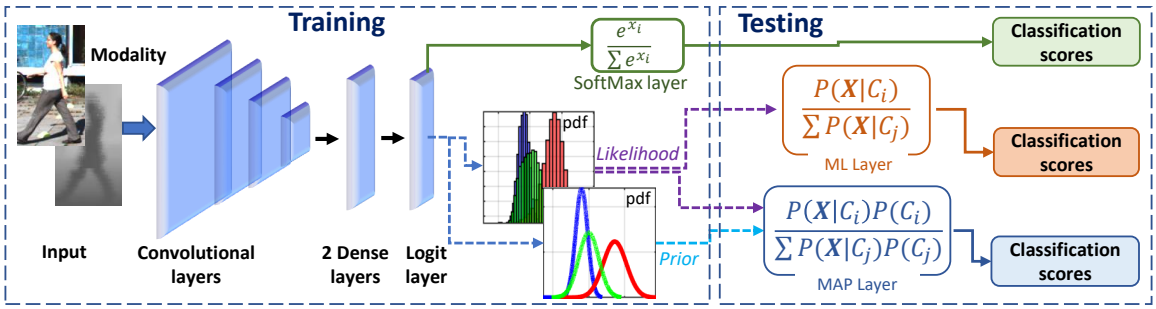


Fig. 4.9 Inception V3 CNN representation with logit and softmax layers, Maximum Likelihood (ML) and Maximum a-Posteriori (MAP) functions. CNN's training was done with the softmax layer. After training, the softmax layer was replaced by the ML and MAP functions *i.e.*, the CNN was not trained with the ML and MAP functions.

Alternatively, the inference using MAP function is given in (4.10) as follows,

$$MAP = arg\max_{i} \frac{(P(\mathbf{Sc}|c_i)P(c_i) + \lambda)}{\sum_{i=1}^{nc}(P(\mathbf{Sc}|c_i)P(c_i) + \lambda)}. \tag{4.10}$$

The sequential steps for calculating the ML and MAP are summarized in the Algorithm 3, where class-conditional $P(\mathbf{Sc}|\mathbf{C})$ is modelled by a normalized histogram. On the other hand, to get the maximum posterior probabilities (MAP) the priors are modelled by normal $\mathcal{N}(test_{Lg}|\mu_{train}, \sigma^2_{train})$. The subscript Lg in the Algorithm 3 indicates that the data is obtained from the logit layer (*i.e.*, the layer before the network prediction layer) *i.e.*, both likelihood and prior are extracted from the logit layer using the training data[8]. In addition to the results considering probability density functions, we consider cumulative distribution functions, as can be seen in the Appendix A.1 Appendix A.1.

The experiments reported throughout the remainder of this work are based on the premise that, after training the network, the proposed ML and MAP functions then replace the softmax and sigmoid prediction layers on the test set only, according to Fig. 4.9.

Another alternative to model the prior probability is through the Kernel Density Estimation (KDE), given in (4.11), which computes the average of several probability

[8]The code for training the network, obtaining the logit layers, and computing the ML and MAP function is available at github.com/gledsonmelotti/ML-MAP-Layers-for-Probabilistic.

Algorithm 3: compute *ML* and *MAP*.

Input

- Number of classes used in training (nc);
- Number of histogram bins (nbins);
- Values of the logit layer on the training set ($train_{Lg}$);
- PDF's parameters (normalized histogram and normal on the training set, see Fig. 4.6);
- Values of the logit layer on the testing set ($test_{Lg}$).
- Additive smoothing (λ).

Output

- Maximum Likelihood (ML) and Maximum a-Posteriori (MAP).

Getting the normalized frequency histograms:
$hc \leftarrow histogram(ScoresLogitsTrain(classes))$**;**
Getting the edge values of each bin of each histogram:
$BinLow \leftarrow BinEdgesLow(hc)$**;**
$BinHigh \leftarrow BinEdgesHigh(hc)$**;**
Getting the normalized frequency values of each bin of each histogram:
$Values \leftarrow Values(hc)$;
Getting the likelihood:
$P(\mathbf{Sc}|\mathbf{C}) \leftarrow zeros(size(test_{Lg}), nc)$;
$Y \leftarrow ScoresLogitsTest$;
for $k \leftarrow 1 : size(test_{Lg})$ **do**
  for $cla \leftarrow 1 : nc$ **do**
    for $i \leftarrow 1 : size(Values)$ **do**
      if $(BinLow(cla,i) \leqslant Y(k,cla)) \& (Y(k,cla) < BinHigh(cla,i))$ **then**
        $P(\mathbf{Sc}|c)(k,cla) \leftarrow Values(cla,i)$;
      end
    end
  end
end
Getting the Prior:
$P(\mathbf{C}) \leftarrow \mathcal{N}(test_{Lg}|[\mu_{train}, \sigma^2_{train}])$;
Calculating the *ML* and *MAP*:
$ML \leftarrow P(\mathbf{Sc}|\mathbf{C}) + \lambda$;
$ML \leftarrow (ML/\text{sum}(ML))$;
$MAP \leftarrow P(\mathbf{Sc}|\mathbf{C})P(\mathbf{C}) + \lambda$;
$MAP \leftarrow (MAP/\text{sum}(MAP))$;

density function to obtain a kernel estimate of the probability density function [198].

$$\widehat{f}_{ker}(d) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{d - Sc_i}{h}\right), \tag{4.11}$$

where h is a smoothing parameter[9] called window width or bandwidth (bw)[10], n is the number of observations (number of samples - scores), d is a value set (domain) that evaluates the function $\widehat{f}_{ker}(d)$, Sc_i are the predicted values (scores) of each object, $K\left(\frac{d-Sc_i}{h}\right) = K(t)$ is the kernel, having the condition that $\int K(t)dt = 1$ to ensure that the estimate in (4.11) is a proper probability density function. Such function is computed at each data point and then taking the average of them. Often the kernel is set to a standard Normal density.

Considering a univariate kernel estimator, the procedure to obtain the estimation is given by the following steps:

- define a kernel and the window width h;
- defines a set of d values to evaluate the $\widehat{f}_{ker}$ function;
- obtain the predicted values of each object (scores), Sc_i, which will generate the estimated probability density functions;
- for each Sc_i, evaluate the kernel for all values in domain d:

$$K_i = K\left(\frac{d-Sc_i}{h}\right), \quad i = 1,2,\ldots,n \tag{4.12}$$

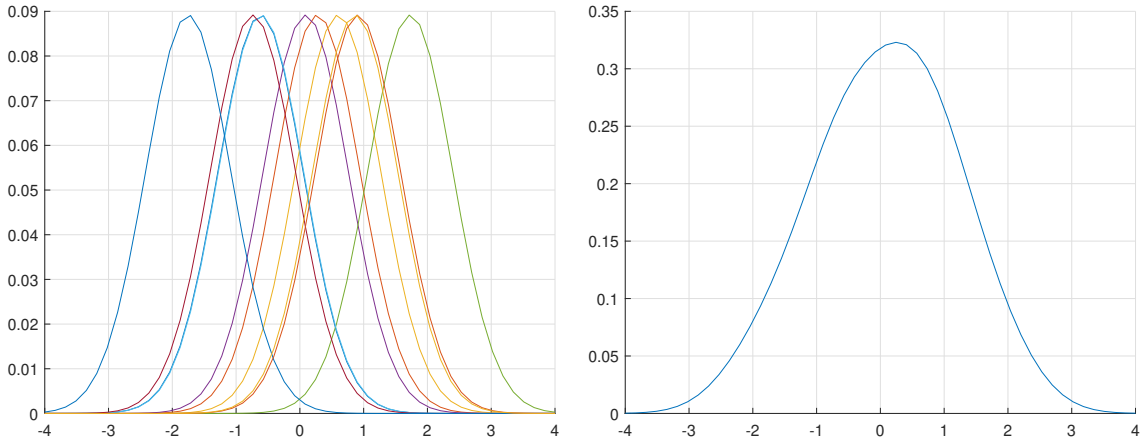
  where the result is a set of n curves, one for each point Sc_i;

- weight each curve by the factor $\frac{1}{h}$;
- for each value of d, compute the average of the weighted curves.
- the average of the weighted curves is the prior probability, $P(\mathbf{C})$.

Figure 4.10 illustrates a KDE example, considering $n = 10$ random variables, generating 10 individual curves that are averaged together to obtain an estimate of the probability density function, according to Algorithm (4). The kernel and estimated function are defined according to the (4.11), (4.13) and (4.15).

[9]This smoothing parameter is not related to the smoothing parameter of Bayesian inference functions (*ML* and *MAP*).

[10]Small values generate rough curves, while larger values generate smoother curves.



(a) Individual estimates of the probability density function.

(b) The average of the weighted curves to obtain the estimate.

Fig. 4.10 Kernel density estimation considering $n = 10$ random variables and kernel as the probability density function.

$$K(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}, \tag{4.13}$$

$$K(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\left(\frac{d-Sc_i}{h}\right)^2}{2}} = \frac{1}{\sqrt{2\pi}} e^{-\left(\frac{1}{2h^2}\right)(d-SC_i)^2}, \tag{4.14}$$

$$\widehat{f}_{ker}(d) = \frac{1}{nh} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\left(\frac{1}{2h^2}\right)(d-SC_i)^2}, \tag{4.15}$$

where $t = \frac{d-Sc_i}{h}$, $h = 1.06n^{\left(\frac{-1}{5}\right)}$ as proposed by [198], and each kernel function is evaluated at d centered at data Sc_i.

4.3 Fusion Strategies

The choice of a sensor depends on the design constraints, accuracy, range, calibration, hysteresis, linearity, dimension, weight, cost, energy consumption, among others. The restrictions presented by one type of sensor may not be demonstrated by another, and vice-versa. It means that, suing a single sensor is very difficult or even impossible to

Algorithm 4: compute the estimated probability density function.

Input

- Data Sc_i normally distributed with length $n = 10$;
- d is the set of values to evaluate $\widehat{f_{ker}}$ with 50 values linearly spaced;
- $h = 1.06 * n^{\frac{-1}{5}}$;
- kernel function defined as probability density function.

Output

- The average of the individually estimated probability density functions.

$f_{hat} \leftarrow zeros(size(d))$; /* take the average of weighted curves */
for $k \leftarrow 1 : n$ **do**
    $f \leftarrow \frac{1}{n*h} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\left(\frac{1}{2h^2}\right)*(d-SC_i)^2}$
    $f \leftarrow f/h$ /* weight each curve by $1/h$ */
    $f_{hat} \leftarrow f_{hat} + f/n$
    $plot(d, f/(n*h))$
end
$plot(d, f_{hat})$

obtain complete, consistent or accurate data [158, 54]. Therefore, combining information from different sensors is crucial to improve robustness and reliability of decision-making in applications such as perception systems for autonomous driving.

In fact, perception systems that benefit from multisensory data fusion may reduce the uncertainties of information captured from different sensors [67, 143, 109, 76], caused by the limited resolution and measurement of noise. In addition, the fusion of information from different sensors may reduce uncertainties of a dynamic scenario (past information is no longer present in the environment), because each sensor may provide data from different regions of the environment, leading to greater confidence and reducing ambiguity in decision-making, meaning that it also contribute to such systems being less prone to perturbations.

Currently, the fusion strategies widely used in deep learning are the early and late fusion applied in different fields of science [143, 171, 109, 210, 156]. Such strategies are illustrated by Fig. 4.11, where early fusion combines information from different modalities (RGB camera and $3D$ point cloud) after processing the features extraction

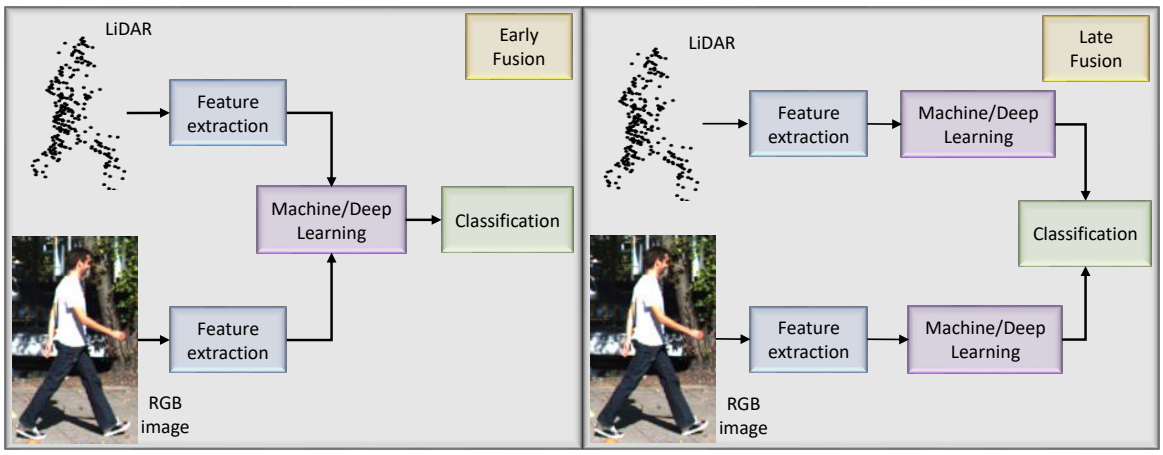


Fig. 4.11 Representation of two types of sensor fusion. The one on the left represents the early fusion strategy, while the other on the right represents the late fusion strategy.

(feature descriptors), while late fusion combines information/data of the top-class scores out of the output layer. In other words, the early fusion strategy combines the information at the input of learning algorithms, while the late fusion combines the information in the prediction layers or at the test stage *i.e.*, combining the predicted values on the test phase.

Before applying a fusion strategy, we have performed the classification of objects using a series of sub-sets separately, as described in Section 4.1.1. The datasets that generated the best range-view and reflectance-view results, considering the F-score metric[11], were considered as baseline datasets for the study of fusion strategies. Therefore, the final number of datasets has been reduced to three depending on the sensory modalities: RGB images, LiDAR-based range-view (RaV - or front-view) and reflectance-view (ReV) obtained through Bilateral Filter using a mask size $C_{mask} = 13 \times 13$.

4.3.1 Early Fusion

The early fusion strategy considered concatenating the images/maps at the beginning of the deep network. Thus, four classification results were obtained through the concatenation of the RGB images, RaV, and ReV maps channels *i.e.*, images with two, four and five channels, as illustrated in Fig. 4.12.

[11] F-score= $2(P.R)/(P+R)$, where P is defined as precision and R is defined as recall, $P = TP/(TP+FP)$ and $R = TP/(TP+FN)$.

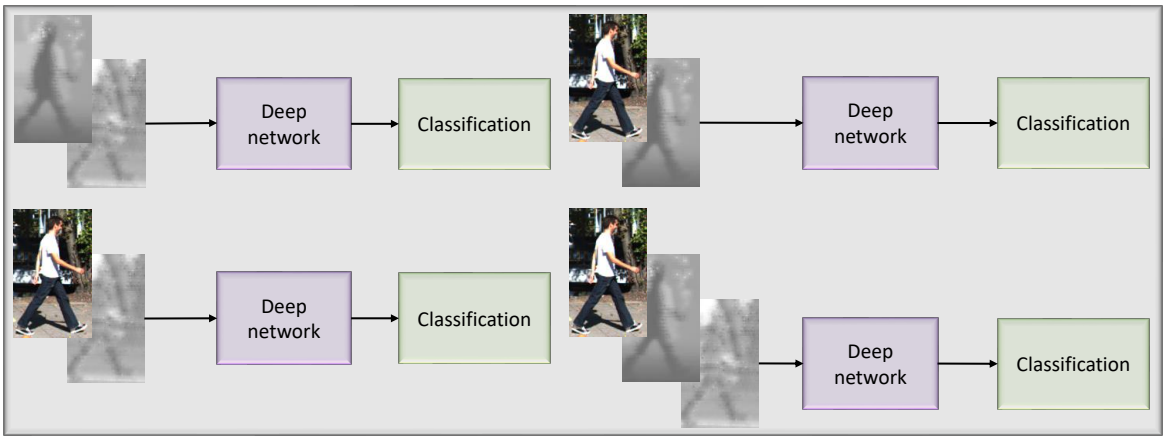


Fig. 4.12 Representation of early fusion strategies. The images were concatenated at the beginning of the deep network.

4.3.2 Late Fusion

For this strategy, we have implemented non-learning formulations, considering traditional formulations as maximum, minimum, average, normalized product [108, 119] as in (4.16)-(4.19), assuming the learning models are independent.

$$\mathcal{S}_{max} = max_m(\mathcal{S}_m) \tag{4.16}$$

$$\mathcal{S}_{min} = min_m(\mathcal{S}_m) \tag{4.17}$$

$$\mathcal{S}_{aver} = \frac{1}{M}\sum_{m=1}^{M}\mathcal{S}_m \tag{4.18}$$

$$\mathcal{S}_{prod} = \frac{\prod_{m=1}^{M}\mathcal{S}_m}{\prod_{m=1}^{M}\mathcal{S}_m + \prod_{m=1}^{M}(1-\mathcal{S}_m)}, \tag{4.19}$$

where M is the number of models and S_m is the top-class score of a given model *i.e.*, the score is the actual output value predicted by the network.

Chapter 5

Evaluation and Results

Contents

5.1 Bayesian Inference: ML and MAP

The CNN output scores indicate a degree of certainty of the given prediction. The level of certainty can be defined as the confidence of the model in a given prediction, and in an object recognition problem, represents the maximum value within the prediction layers. However, the output scores may not always represent a reliable indication of certainty with regard to a given class, especially when unseen (non-trained *i.e.*, out-of-training distribution) objects occur in the prediction (test) stage; this is particularly relevant for real world application domains involving autonomous robots and vehicles, since unpredictable objects are likely to be encountered which may be misclassified by prediction layers; even worst when such objects are misrecognized with a high degree of certainty. With this in mind, in addition to the trained classes (pedestrian, car, and cyclist), a set of unseen objects were introduced on the classification dataset, according to Subsection 3.3.1. Regarding the objects detection, the unseen classes are already present in the test dataset's own frames. The object detection results are presented by means of precision-recall curves considering the easy, moderate and hard cases (based on object size, occlusion state, and truncation level), according to the devkit-tool provided by the KITTI benchmark.

5.1.1 Classification

All classes, on the training dataset, were extracted directly from the KITTI Vision Benchmark Suite-2D object, with the exception of the "tree", "lamppost" and "signpost" classes which were manually extracted from the data for this study. The rationale behind this is to evaluate the prediction confidence of the networks on objects that do not belong to any of the trained classes, and as such the consistency of the models can be assessed. Ideally, if the classifiers are perfectly consistent in terms of probability interpretation, the prediction scores would be identical (equal to 1/3) for each class in each sample on the unseen dataset. Results on the testing set are shown in Table 5.1 in terms of F-score, False Positive Rate (FPR), the average ($Ave.Scores_{FP}$) and variance ($Var.Scores_{FP}$) of the False Positives (FP). The average ($Ave.Scores_{unseen}$) and the variance ($Var.Scores_{unseen}$) of the predicted scores are also shown for the unseen testing set.

In reference to Table 5.1, where the results are reported based on the KITTI-classification test set, it can be observed that the FPR, $Ave.Scores_{FP}$ and $Var.Scores_{FP}$ values of the ML and MAP are considerably lower than the results presented by the

Table 5.1 Comparison between the classifications obtained by the *SM* layer, *ML* and *MAP* layers in terms of average F-score and FPR (%). The performance measures on the unseen dataset are the average and the variance of the prediction scores.

			KITTI Dataset			
Modalities	**F-score**	**FPR**	**Average Score$_{FP}$**	**Variance Scores$_{FP}$**	**Average Score$_{unseen}$**	**Variance Scores$_{unseen}$**
SM$_{RGB}$	95.89	1.60	0.853	0.021	0.982	0.005
ML$_{RGB}$	94.84	1.19	0.487	0.018	0.708	0.025
MAP$_{RGB}$	95.22	1.15	0.355	0.002	0.388	0.003
SM$_{RaV}$	92.40	2.12	0.878	0.029	0.961	0.014
ML$_{RaV}$	92.07	1.73	0.495	0.024	0.721	0.049
MAP$_{RaV}$	92.17	2.12	0.371	0.007	0.672	0.077
SM$_{ReV}$	92.82	1.73	0.824	0.031	0.967	0.009
ML$_{ReV}$	92.52	1.61	0.692	0.039	0.889	0.029
MAP$_{ReV}$	92.10	1.66	0.442	0.023	0.515	0.016

Table 5.2 Number of bins for *ML* and *MAP* functions.

Modality	RGB			RaV			ReV		
Bins per class	Ped	Car	Cyc	Ped	Car	Cyc	Ped	Car	Cyc
ML layer	25	25	25	5	10	7	9	4	26
MAP layer	30	30	30	5	3	8	6	8	35

softmax layer (*SM*) for both of the sensor modalities. Regarding the F-scores of the proposed approach (*ML* and *MAP*) compared to the *SM*, the methodology resulted in a reduction of $1,10\%$ (*ML*) and 0.70% (*MAP*) (percentage point) for the RGB modality, 0.36% (*ML*) and 0.25% (*MAP*) for RaV modality, and for the ReV modality a reduction of 0.32% (*ML*) and 0.78% (*MAP*). Such reductions of the F-scores are relatively small and thus did not compromise the classification ability. The parameters assigned for the number of bins and additive smoothing are shown in Table 5.2 and Table 5.3. Such parameters do not depend on previous information of the training set. These parameters were determined empirically *i.e.*, by observing which value would preserve approximately the "original" distribution without compromising the final result.

Further experiments have been carried out as a complementary analysis concerning the network's overconfident behaviour, on a so-called unseen test set, by means of the network's average scores defined as $Ave.Scores_{unseen}$. Note that for *ML* and *MAP* functions, the results are smaller than the *SM* layer as can be seen in Table 5.1.

Table 5.3 Smoothing parameter (λ) for *ML* and *MAP* functions.

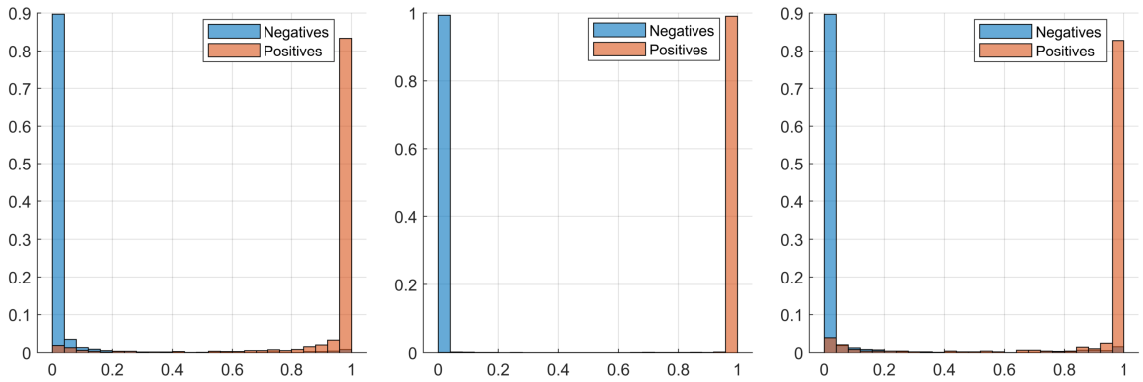
Modality	RGB	RaV	ReV
Layer	Additive Smoothing	Additive Smoothing	Additive Smoothing
ML	1×10^{-2}	1×10^{-6}	1×10^{-4}
MAP	1×10^{-2}	1×10^{-6}	1×10^{-4}

This indicates that the probabilistic inference are significantly better balanced *i.e.*, enabling more reliable decision-making, when "new" objects of "non-trained" classes are presented to the CNNs.

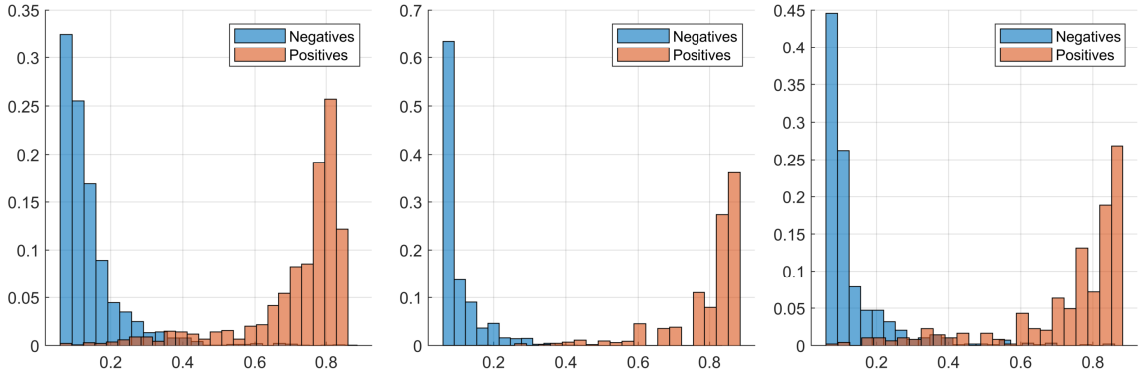
The distributions of the score for the pedestrian, car, and cyclist classes (columns from left to right) obtained through the *ML* and *MAP* functions are illustrated by the histograms in Fig. 5.1, Fig. 5.2, Fig. 5.3, while Fig. 5.4 represents the distribution for the unseen dataset. We can see that the aforementioned graphs show less "extreme" results than those provided by the softmax layer.

Another way of analyzing the results is through reliability diagrams, as shown in the Fig. 5.5, considering uncalibrated, *ML* and *MAP* data. Furthermore, as a way of validating our methodology, we compared our results achieved with the temperature scaling calibration technique. Note that the results presented in the reliability diagrams are shown through the MCE and ECE metrics. From these metrics we cannot say which is the best calibration technique, because for a given technique the lowest value for the MCE was obtained, while for another technique the lowest value for the ECE was obtained. However, we show that the proposed approach contributed to reduce the calibration errors *i.e.*, to reduce the values of the MCE and ECE metrics when compared to the uncalibrated data, and consequently we provide a more reliable result, as well as the contribution to reduce the overconfident predictions.

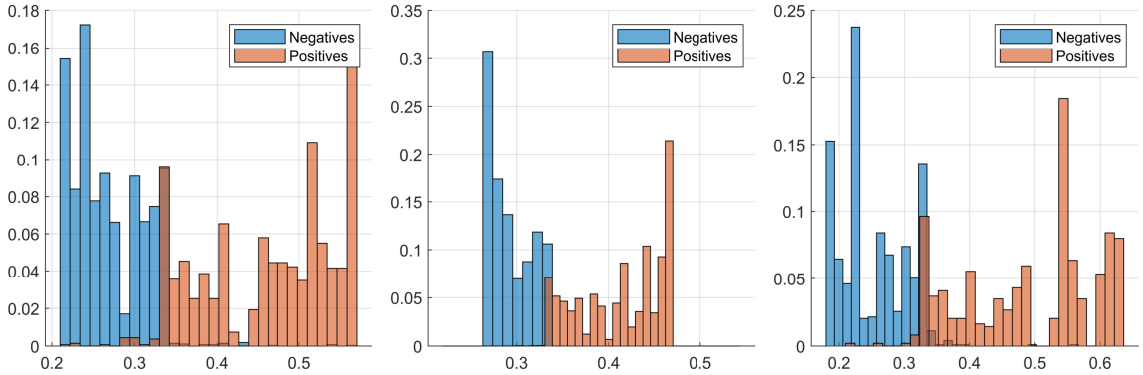
The results from Kernel Density Estimation (KDE) are shown in Table 5.4. Note that the reduction of F-scores does not compromise the results already achieved by the traditional *SM* layer. In the case of RGB modality, the reduction was 0.49% for *MAP*. For the RaV dataset the reduction was 0.04% for *MAP*, while for the ReV dataset the *MAP* had a reduction of 1.11%. Values assigned for bandwidth from KDE are shown in Table 5.5, as well as the additive smoothing for *MAP*. The score distributions from the proposed methodology (*MAP*) using KDE are illustrated in Fig. 5.6 - Fig. 5.8. Regarding the calibration metrics, the results were not significant. In fact, the *MAP*



(a) Softmax function scores.



(b) *ML* function scores.

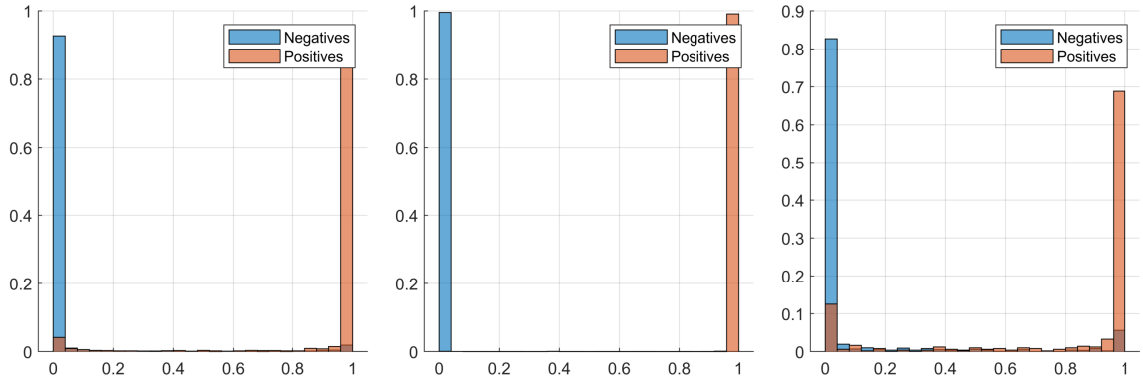


(c) *MAP* function scores.

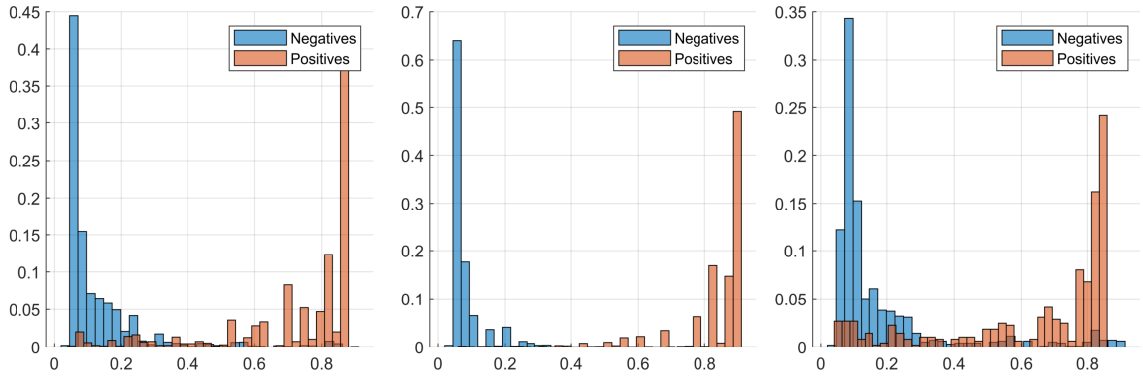
Fig. 5.1 From the RGB modality, the prediction scores were calculated using the softmax function, *ML* and *MAP* functions on the KITTI dataset test set.

achieved a better result than the uncalibrated scores when analyzed with the MCE metric. The results of such a calibration are illustrated in Fig. 5.9.

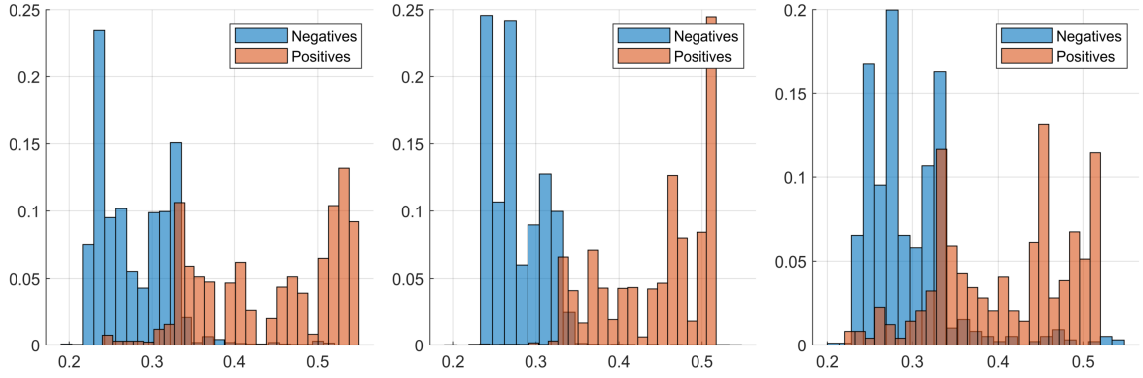
In addition to the results presented in the Table 5.4, through the F-scores, the Fig. 5.10 presents the distributions of the scores through the histograms of the unseens datasets, being possible to observe the reduction of the values of the scores.



(a) Softmax function scores.

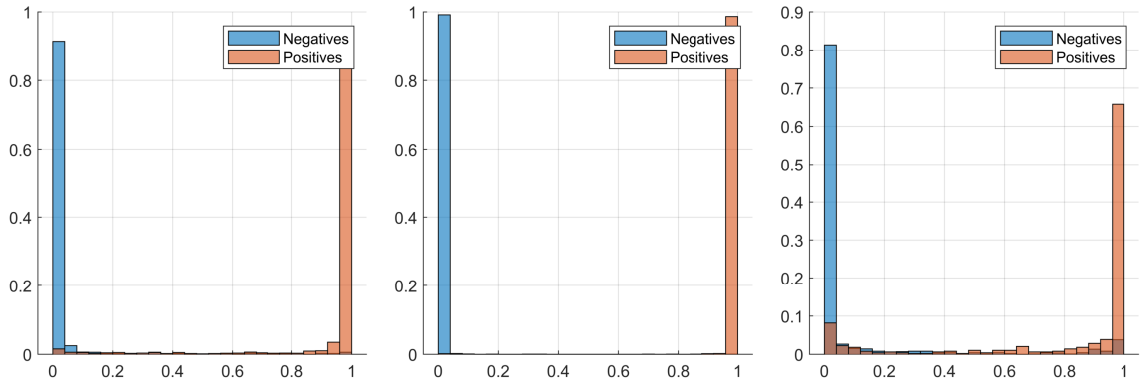


(b) *ML* function scores.

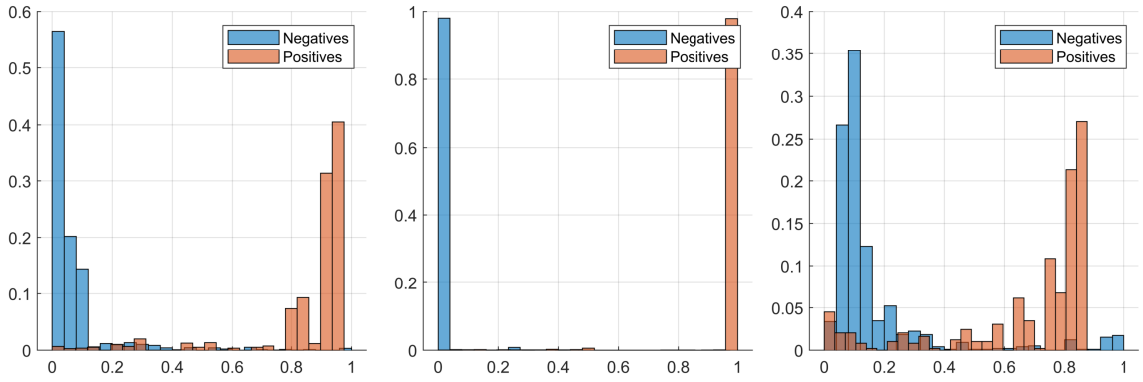


(c) *MAP* function score.

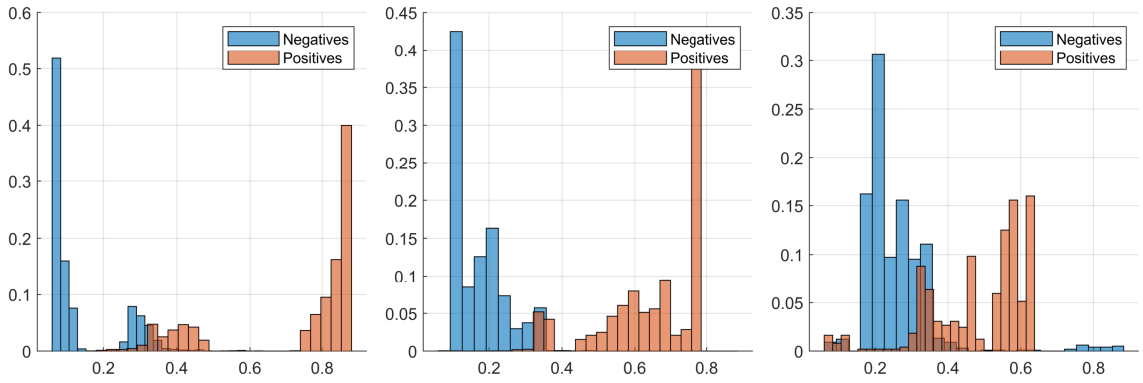
Fig. 5.2 From the LiDAR (RaV) modality, the prediction scores were calculated using the softmax function, *ML* and *MAP* functions on the KITTI dataset test set.



(a) Softmax function scores.

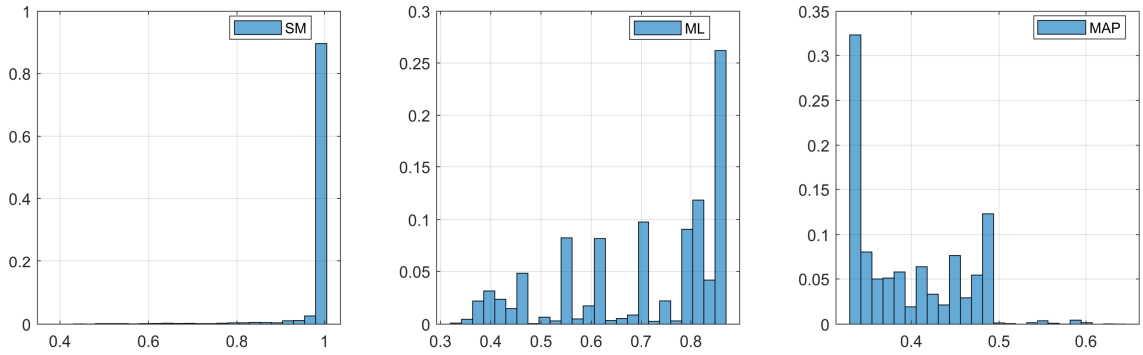


(b) *ML* function scores.

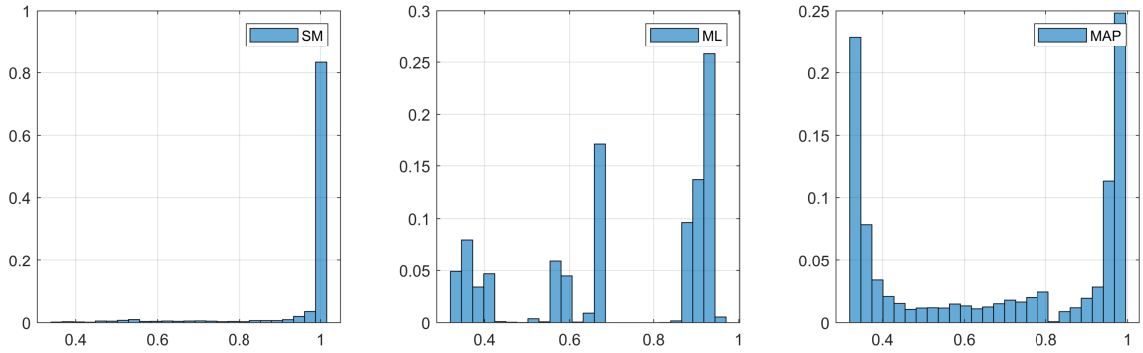


(c) *MAP* function score.

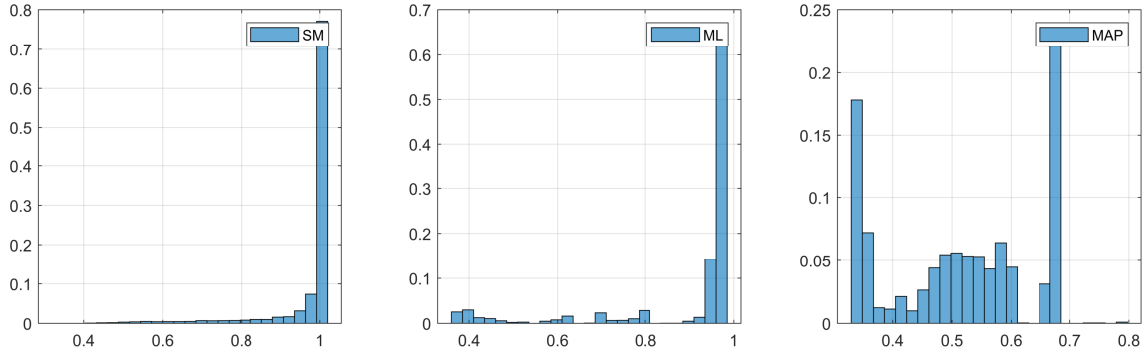
Fig. 5.3 ReV modality from the LiDAR sensor, the prediction scores were calculated using the softmax function, *ML* and *MAP* functions on the KITTI dataset test set.



(a) *SM*, *ML*, *MAP* scores on the RGB unseen KITTI-set.

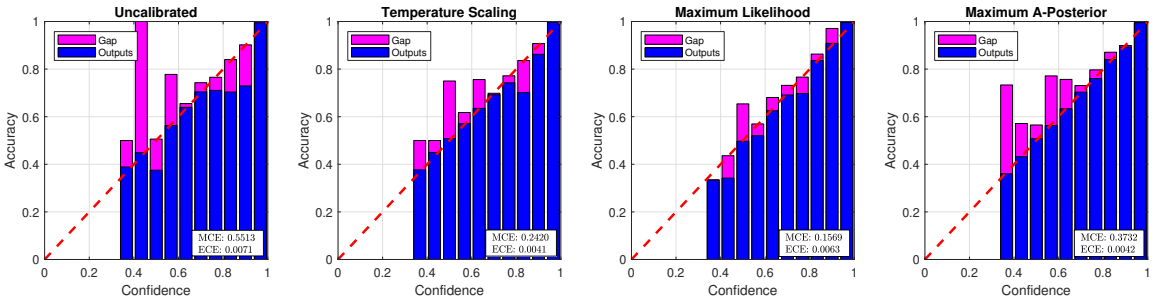


(b) *SM*, *ML*, *MAP* scores on the RaV unseen KITTI-set.

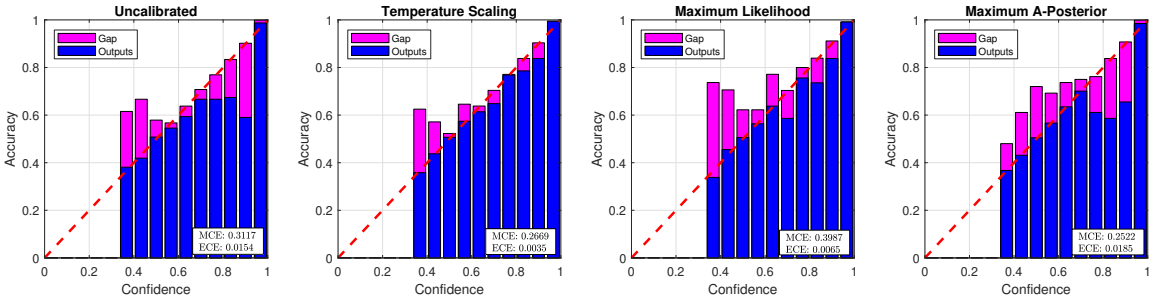


(c) *SM*, *ML*, *MAP* scores on the ReV unseen KITTI-set.

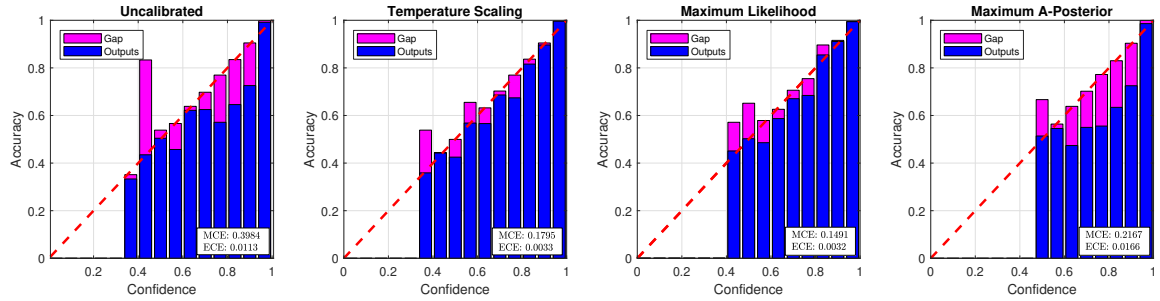
Fig. 5.4 Prediction scores on the unseen/non-trained data (comprising the classes: person sitting, tram, tree/lamppost/signpost, truck, van), using *SM* layer (left side), and the proposed *ML* (center) and *MAP* (right side) functions, where the likelihood functions were defined as normalized histograms, and the priors were modelled by Gaussian distributions.



(a) Reliability diagrams for RGB images from KITTI dataset, considering the number of bins $= 15$ and $TS = 1.31$.



(b) Reliability diagrams for RaV images from KITTI dataset, considering the number of bins $= 15$ and $TS = 2.26$.



(c) Reliability diagrams for ReV images from KITTI dataset, considering the number of bins $= 15$ and $TS = 1.61$.

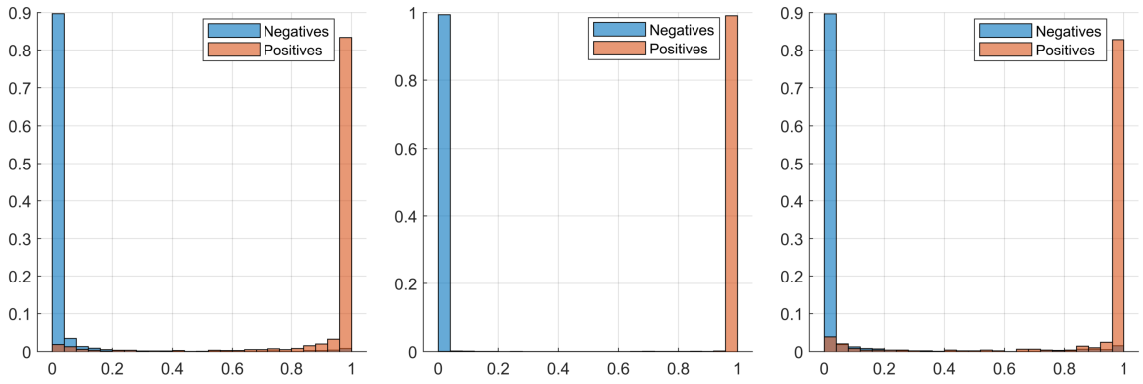
Fig. 5.5 The graphs, from left to right, represent uncalibrated score values, followed by score values calibrated through Temperature Scaling, then scores obtained by the *ML* and *MAP* layers respectively.

Table 5.4 Comparison between the classifications obtained by the *SM* layer, *ML* and *MAP* functions in terms of average F-score and FPR (%), considering the prior probability estimated from KDE. The performance measures on the unseen dataset are the average and the variance of the prediction scores.

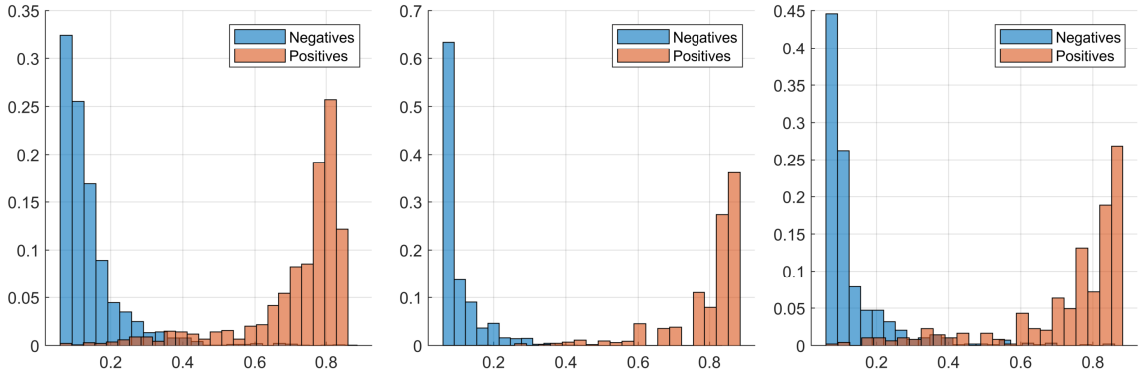
	KITTI Dataset					
Modalities	**F-score**	**FPR**	**Average Score$_{FP}$**	**Variance Scores$_{FP}$**	**Average Score$_{unseen}$**	**Variance Scores$_{unseen}$**
SM$_{RGB}$	95.89	1.60	0.853	0.021	0.982	0.005
ML$_{RGB}$	95.09	0.97	0.526	0.024	0.650	0.023
MAP$_{RGB}$	95.42	1.41	0.809	0.037	0.740	0.055
SM$_{RaV}$	92.40	2.12	0.878	0.028	0.961	0.013
ML$_{RaV}$	92.07	1.73	0.495	0.024	0.721	0.049
MAP$_{RaV}$	92.36	1.89	0.376	0.012	0.508	0.048
SM$_{ReV}$	92.82	1.73	0.824	0.031	0.967	0.009
ML$_{ReV}$	92.52	1.61	0.692	0.039	0.889	0.029
MAP$_{ReV}$	91.79	1.46	0.397	0.021	0.393	0.010

Table 5.5 Smoothing parameter (λ) for *MAP* function, and bandwidth for KDE.

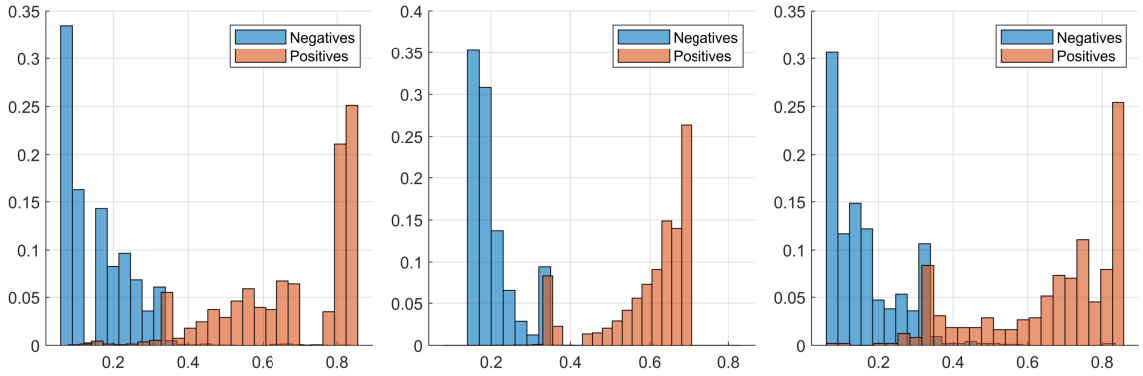
Modality	RGB		RaV		ReV	
Layer	Bandwidth	Additive Smoothing	Bandwidth	Additive Smoothing	Bandwidth	Additive Smoothing
MAP	0.500	1×10^{-2}	0.105	1×10^{-2}	0.104	1×10^{-3}



(a) Softmax function scores.

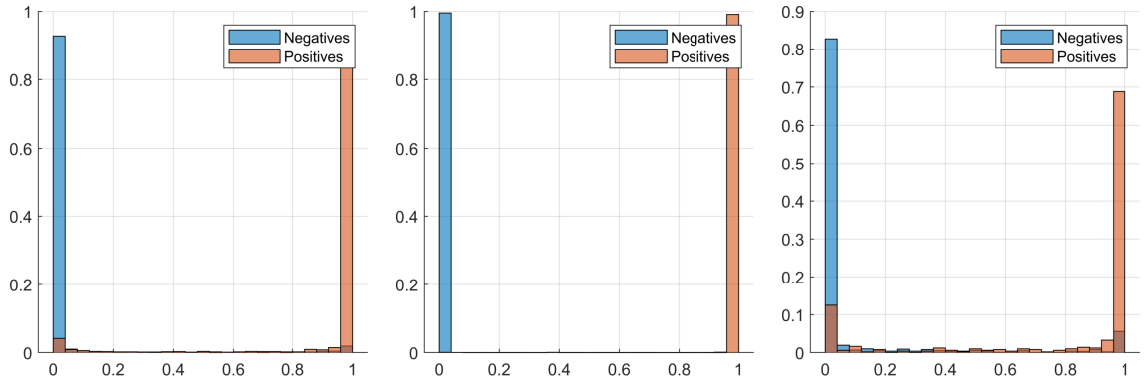


(b) *ML* function scores.

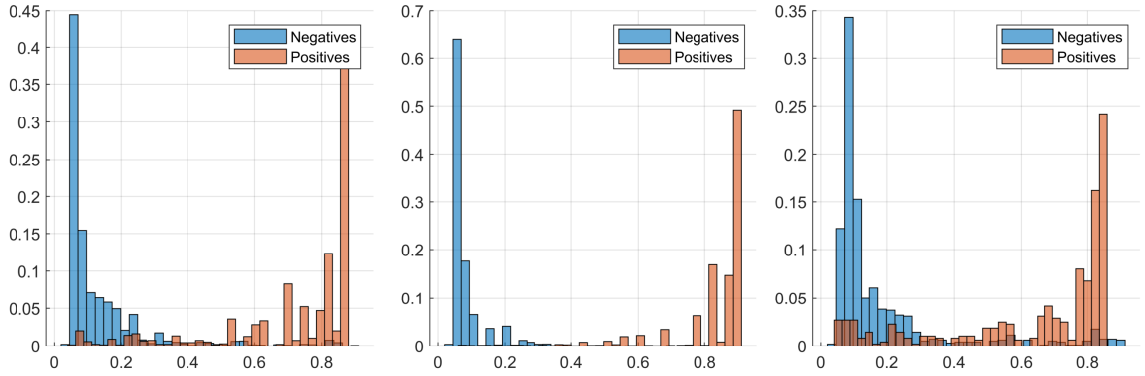


(c) *MAP* function scores.

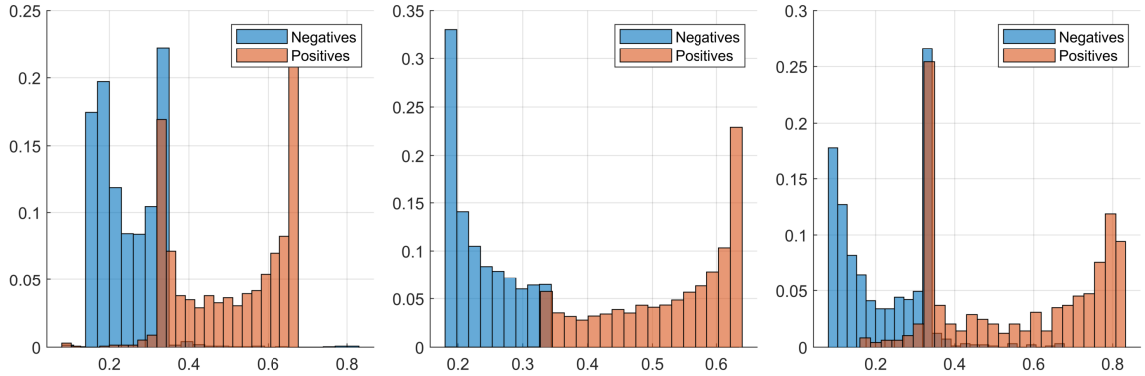
Fig. 5.6 From the RGB modality, the prediction scores were calculated using the softmax function, *ML* and *MAP* functions, considering KDE on the KITTI dataset test set.



(a) Softmax function scores.

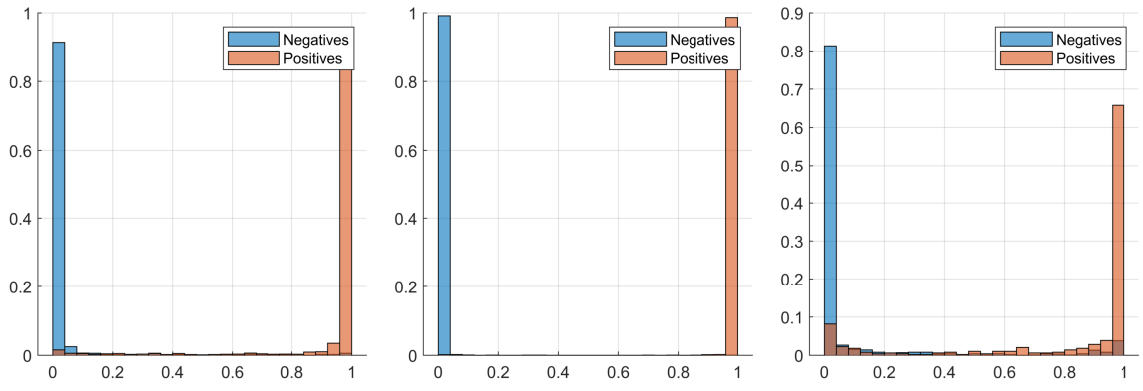


(b) *ML* function scores.

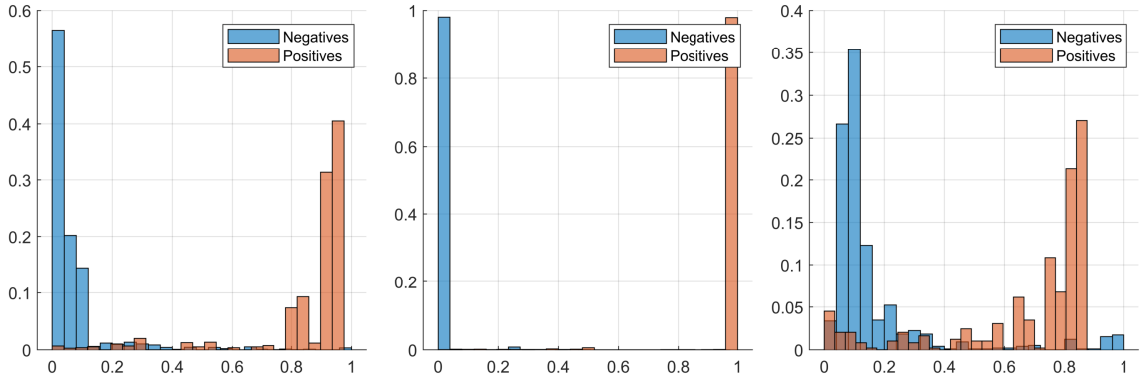


(c) *MAP* function score.

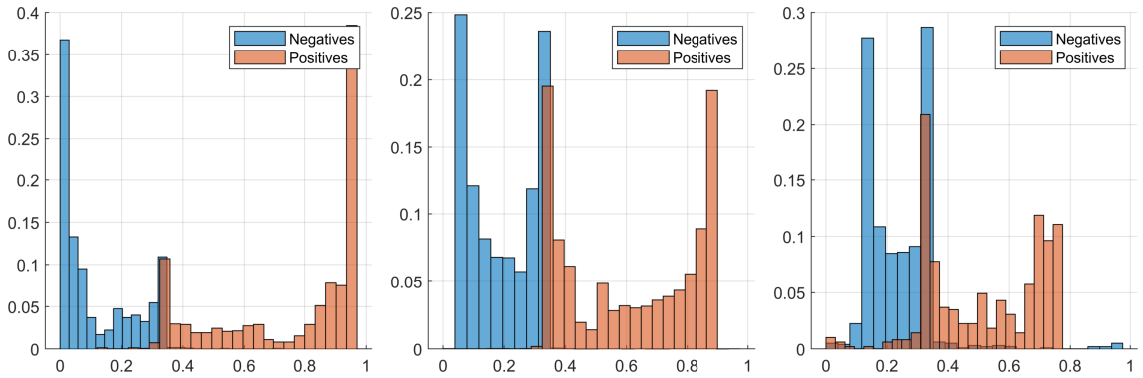
Fig. 5.7 From the LiDAR (RaV) modality considering KDE, the prediction scores were calculated using the softmax function, *ML* and *MAP* functions on the KITTI dataset test set.



(a) Softmax function scores.

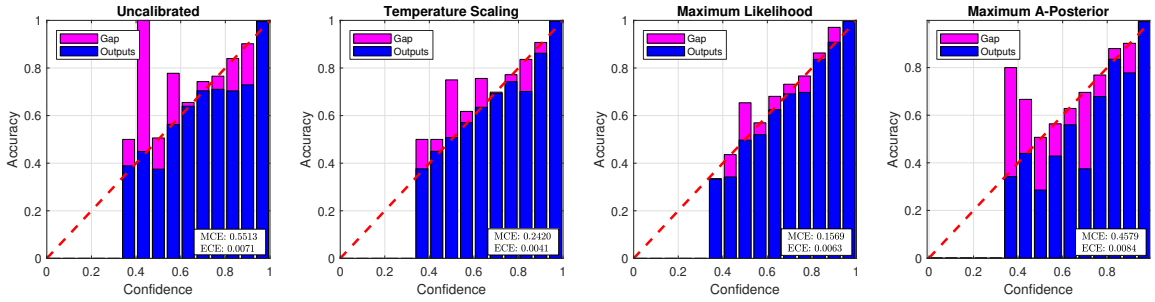


(b) *ML* function scores.

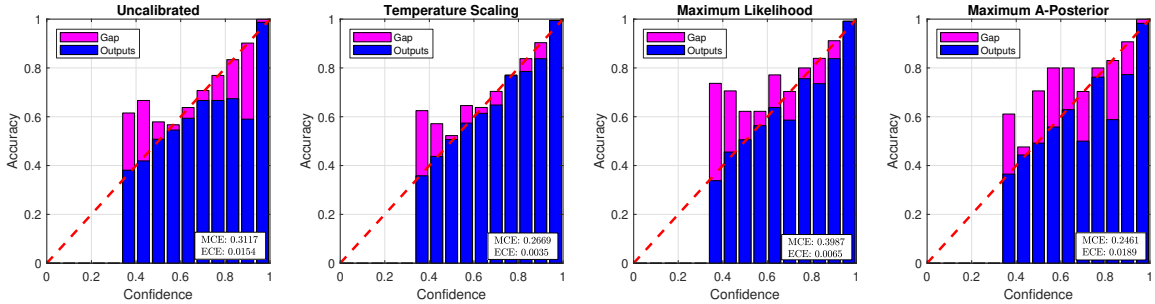


(c) *MAP* function score.

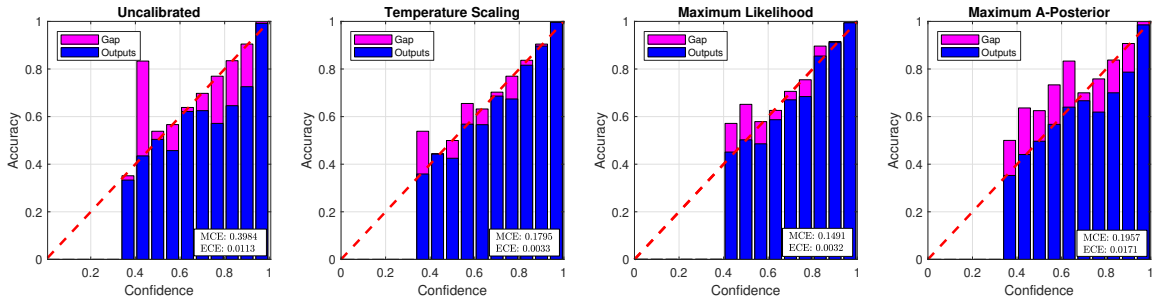
Fig. 5.8 ReV modality from the LiDAR sensor, the prediction scores considering KDE were calculated using the softmax function, *ML* and *MAP* functions on the KITTI dataset test set.



(a) Reliability diagrams for RGB images from KITTI dataset, considering the number of bins $= 15$ and $TS = 1.31$.

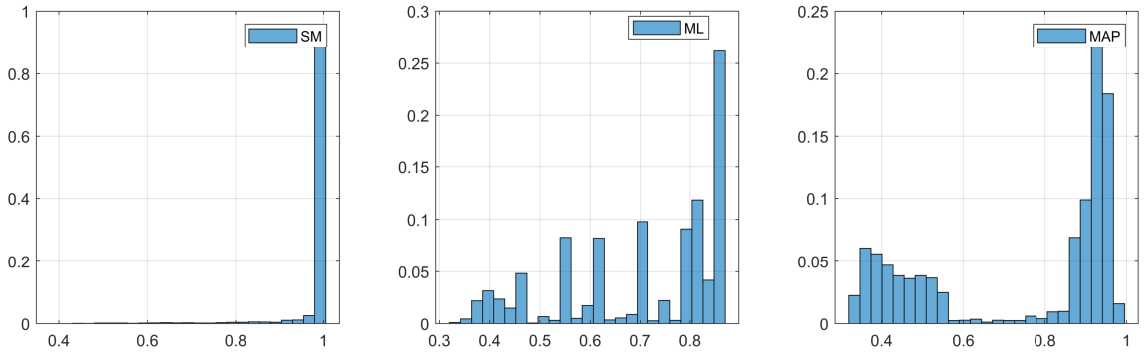


(b) Reliability diagrams for RaV images from KITTI dataset, considering the number of bins $= 15$ and $TS = 2.26$.

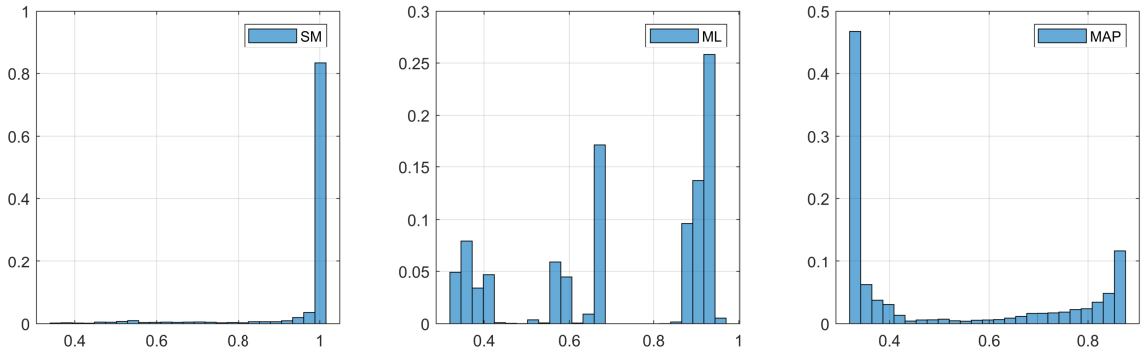


(c) Reliability diagrams for ReV images from KITTI dataset, considering the number of bins $= 15$ and $TS = 1.61$.

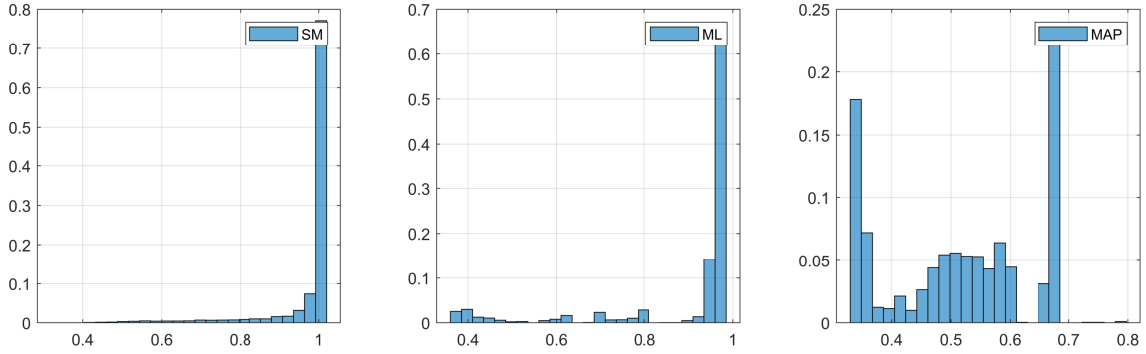
Fig. 5.9 The reliability diagrams considering KDE, from left to right, represent uncalibrated score values, followed by score values calibrated through Temperature Scaling, then scores obtained by the *ML* and *MAP* layers respectively.



(a) *SM*, *ML*, *MAP* scores on the RGB unseen KITTI-set.



(b) *SM*, *ML*, *MAP* scores on the RaV unseen KITTI-set.



(c) *SM*, *ML*, *MAP* scores on the ReV unseen KITTI-set.

Fig. 5.10 Prediction scores on the unseen/non-trained data (comprising the classes: person sitting, tram, tree/lamppost/signpost, truck, van), using *SM* layer (left side), and the proposed *ML* (center) and *MAP* (right side) functions, considering the KDE as estimate for the prior probability.

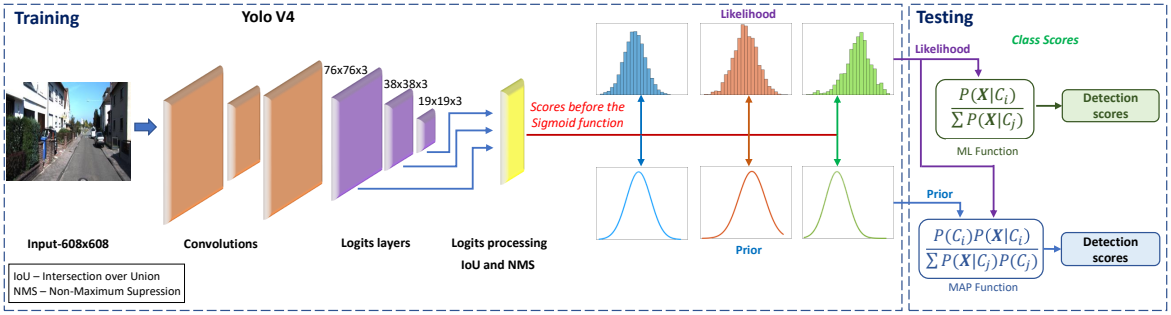


Fig. 5.11 Yolo V4 representation with logits and sigmoid (*SgM*) layers, Maximum Likelihood (*ML*) and Maximum a-Posterior (*MAP*) functions. After training, the predicted values from the sigmoid layer were replaced by the scores from *ML* and *MAP* functions. Notice that the Yolo V4 was not trained or re-trained with the *ML/MAP* functions.

5.1.2 Detection

The prediction layer in the detection algorithm was the sigmoid (SgM) function, instead of the softmax (SM) function. The procedure for reducing the values of the prediction scores with overconfidence (false positives) is the same as for object classification, as shown in Fig. 5.11.

The results on the *per-modalities* test sets are shown in Figures 5.12, 5.13, and 5.14 through precision-recall curves (Pr-Rc) for YOLOV4. Note that the curves are presented to the three different difficulty levels (easy, moderate and hard), according to the KITTI dataset methodology for object detection.

To facilitate the comparison analysis from the results given by the Pr-Rc curves, we further present a quantitative comparison, between the baseline (designated by sigmoid, or simply *SgM*) and the proposed *ML*, and *MAP* functions, using the areas under the curve (AUC), as shown in Table 5.6. Note that the most expressive result was achieved for the cyclists class (Cyc), both for the RGB, RaV and ReV modalities in the three difficulty levels. With respect to the car (Car) and pedestrians (Ped), the proposed approach also showed some improvement.

Additionally, the graphs in figures 5.15 and 5.16 show, when using the YOLOV4 detector, the distribution of the output-scores for the proposed approach and the baseline (*i.e.*, using sigmoid). We can see that the baseline results achieved by YOLOV4 (shown in the first row) present many false positives (FP) with overconfident scores, while the *ML* and *MAP* layers have reduced the overconfidence on the FPs, whereas the performance on the true positives (TP) is relatively unaffected, according

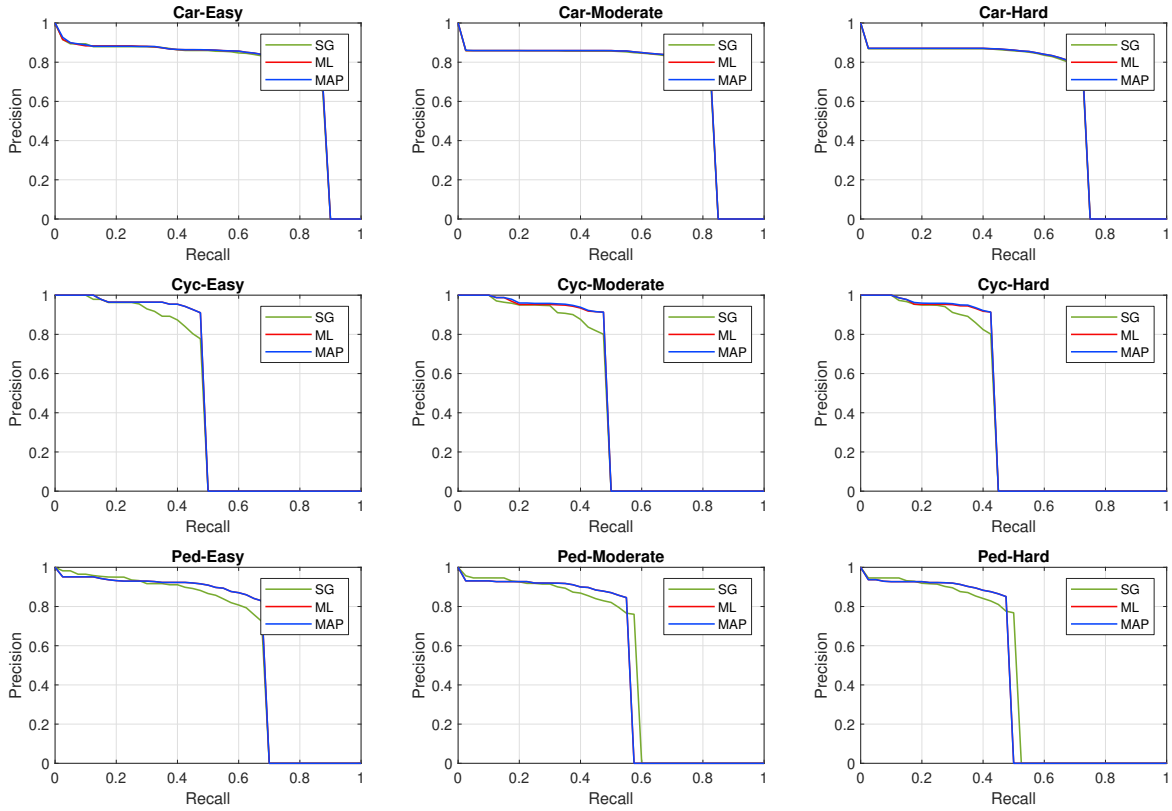


Fig. 5.12 Precision-recall curves for car, cyc. and ped. classes using the RGB modality, with $\lambda_{ML} = 1.6 \times 10^{-6}$, $Bins_{ML} = 22$, $\lambda_{MAP} = 1.0 \times 10^{-8}$, and $Bins_{MAP} = 24$.

to Table 5.7. Furthermore, the proposed approach can be compared quantitatively with the baseline through the ECE metric, as shown in Table 5.8 (RGB, RaV and ReV modalities). Based on such results, we can see that the ECE was reduced for the proposed methodology.

Note that the proposed methodology is dependent on the number of bins (nbins) and the parameter λ. Thus, the values of the scores may vary according to the values of these parameters. For the particular case of the cyclist class, the proposed methodology achieved better performance compared to the baseline (results in Table 5.6). In this search, we have chosen to use a single set of parameters for all the three cases (*i.e.*, the same values of λ and nbins for each class). Given the proposed approach, we note that a set of tailored parameters for each class can be used instead, as the distributions (PDF's) are carried out individually.

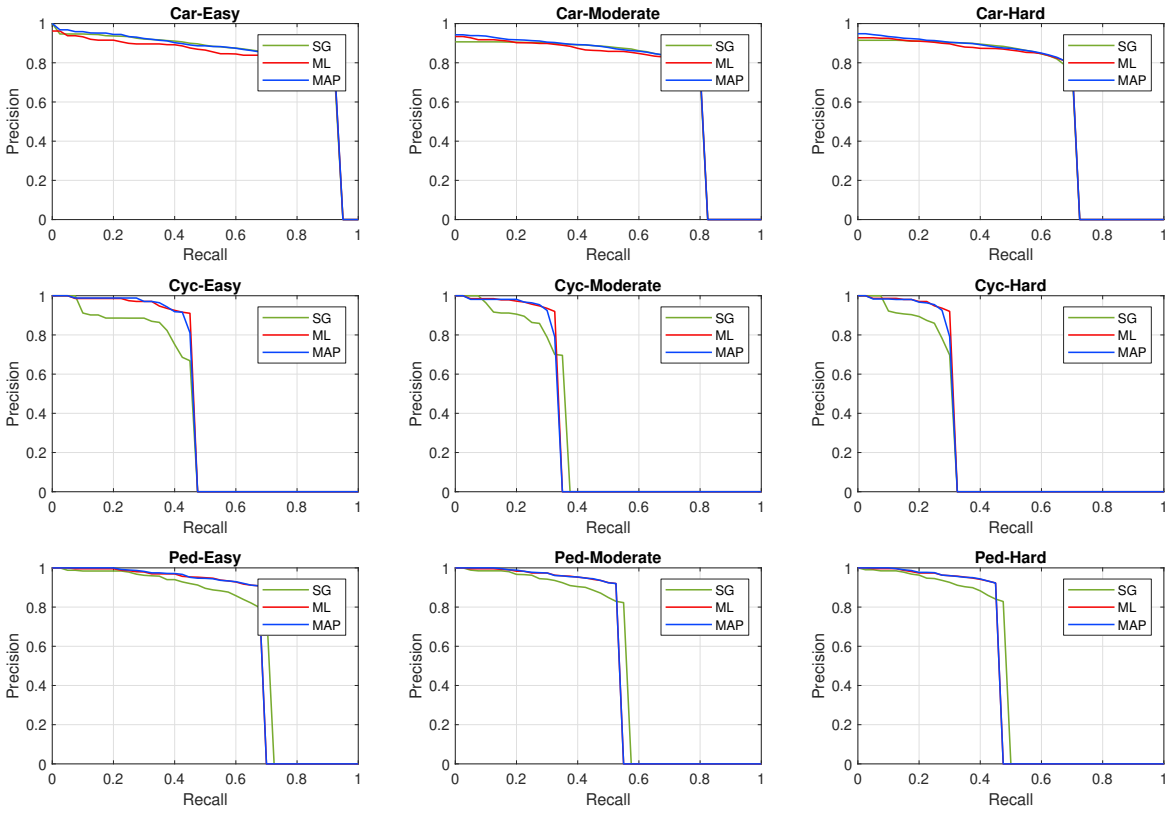


Fig. 5.13 Precision-recall curves for RaV modality, with $\lambda_{ML} = 1.3 \times 10^{-3}$, $Bins_{ML} = 20$, $\lambda_{MAP} = 1.7 \times 10^{-5}$, and $Bins_{MAP} = 24$.

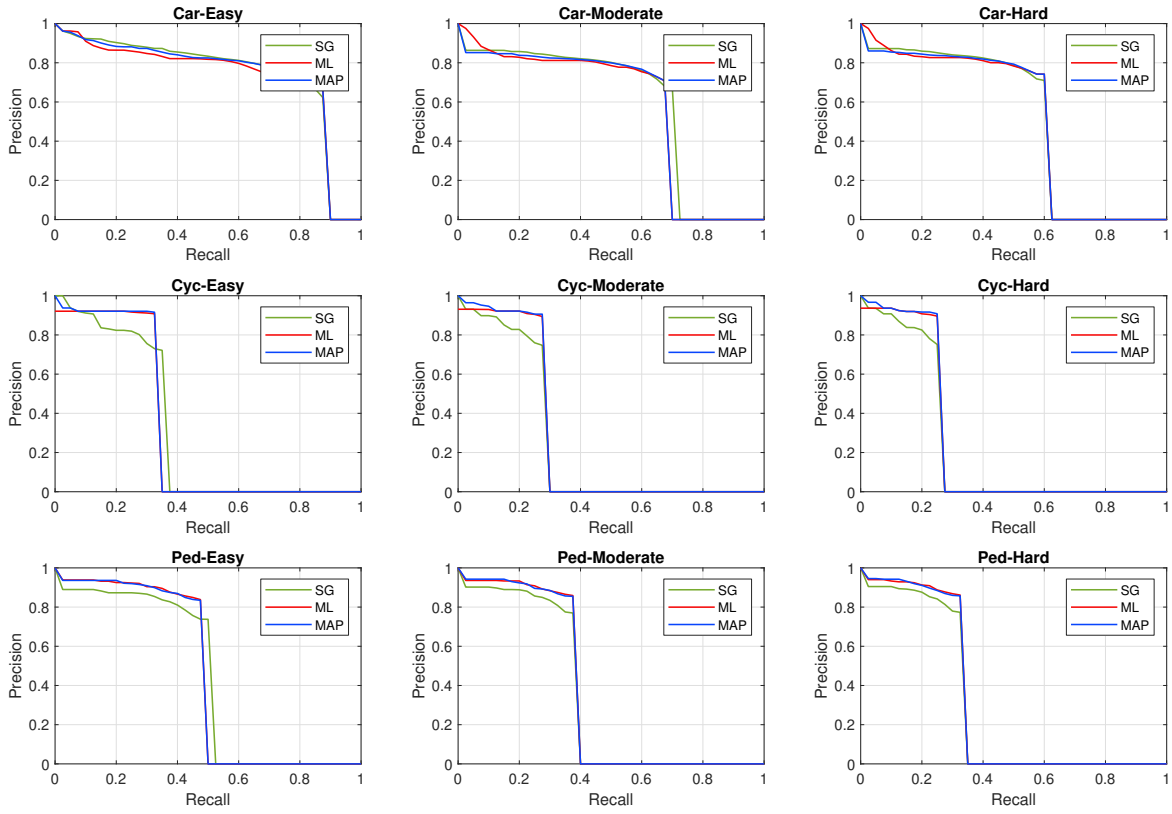


Fig. 5.14 Precision-recall curves for ReV modality, with $\lambda_{ML} = 1.3 \times 10^{-3}$, $Bins_{ML} = 23$, $\lambda_{MAP} = 8.0 \times 10^{-5}$, and $Bins_{MAP} = 5$.

Table 5.6 Comparison of the areas under the curves (%) between the sigmoid layer (*SgM*), *ML* and *MAP* functions from the precision-recall curves.

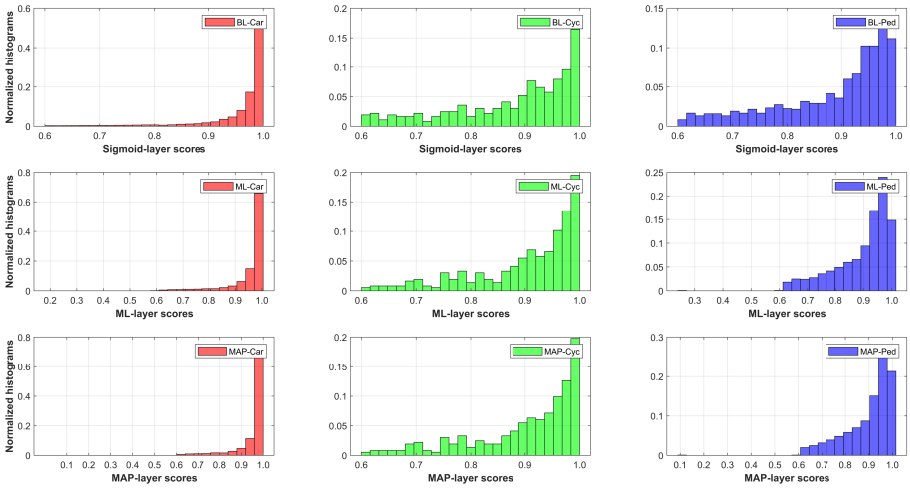
RGB Modality											
Easy				**Moderate**				**Hard**			
Case	***SG***	***ML***	***MAP***	**Case**	***SG***	***ML***	***MAP***	**Case**	***SG***	***ML***	***MAP***
Car	75.48	75.93	**75.95**	**Car**	70.67	70.90	**71.00**	**Car**	63.04	**63.36**	**63.36**
Cyc	45.47	**47.20**	**47.20**	**Cyc**	45.47	46.83	**46.99**	**Cyc**	40.94	42.09	**42.22**
Ped	61.84	**63.05**	**63.05**	**Ped**	**52.27**	51.25	51.24	**Ped**	**45.65**	44.52	44.52
RaV Modality											
Easy				**Moderate**				**Hard**			
Case	***SG***	***ML***	***MAP***	**Case**	***SG***	***ML***	***MAP***	**Case**	***SG***	***ML***	***MAP***
Car	82.99	81.13	**83.21**	**Car**	71.07	72.16	**71.78**	**Car**	62.97	62.80	**63.53**
Cyc	40.48	**44.80**	44.73	**Cyc**	32.28	**32.74**	32.43	**Cyc**	28.13	**30.39**	29.99
Ped	66.27	66.45	**66.60**	**Ped**	**52.56**	52.22	52.22	**Ped**	**45.57**	44.93	44.96
ReV Modality											
Easy				**Moderate**				**Hard**			
Case	***SG***	***ML***	***MAP***	**Case**	***SG***	***ML***	***MAP***	**Case**	***SG***	***ML***	***MAP***
Car	**74.42**	72.68	73.92	**Car**	**58.13**	56.14	56.35	**Car**	50.83	50.69	50.52
Cyc	30.80	31.00	**31.25**	**Cyc**	24.65	26.46	**26.86**	**Cyc**	22.73	24.21	**24.53**
Ped	43.51	**44.35**	44.26	**Ped**	33.62	35.44	**35.45**	**Ped**	29.32	**30.88**	30.87

Table 5.7 The average of the scores after the proposed approach, considering the results from the YOLOV4.

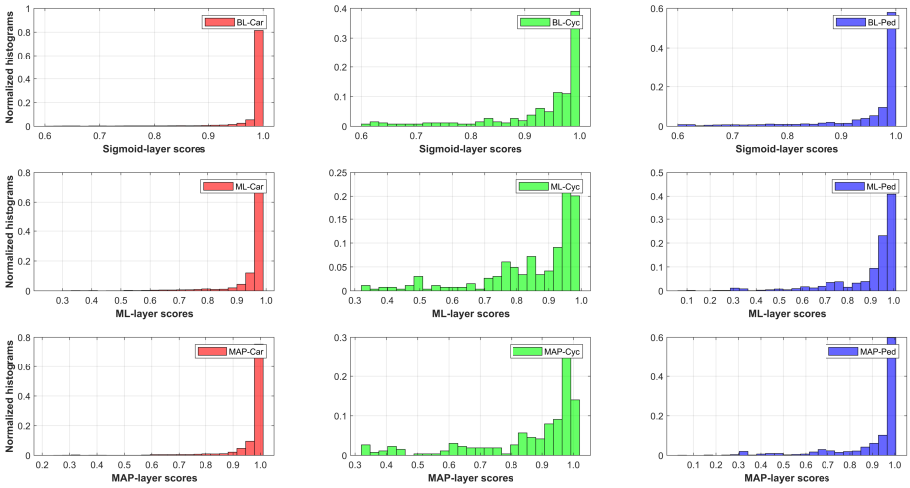
	Modality	**RGB**			**RaV**			**ReV**		
True Positives	Approach	*SG*	*ML*	*MAP*	*SG*	*ML*	*MAP*	*SG*	*ML*	*MAP*
	Average	0.947	0.950	0.950	0.974	0.940	0.955	0.970	0.934	0.951
	Variance	0.007	0.006	0.006	0.004	0.010	0.011	0.005	0.011	0.012
	Modality	**RGB**			**RaV**			**ReV**		
False Positives	Approach	*SG*	*ML*	*MAP*	*SG*	*ML*	*MAP*	*SG*	*ML*	*MAP*
	Average	0.788	0.806	0.806	0.867	0.780	0.786	0.872	0.795	0.817
	Variance	0.013	0.013	0.013	0.015	0.037	0.044	0.014	0.034	0.030

Table 5.8 ECE on the different modalities, when using YOLOV4 as detector.

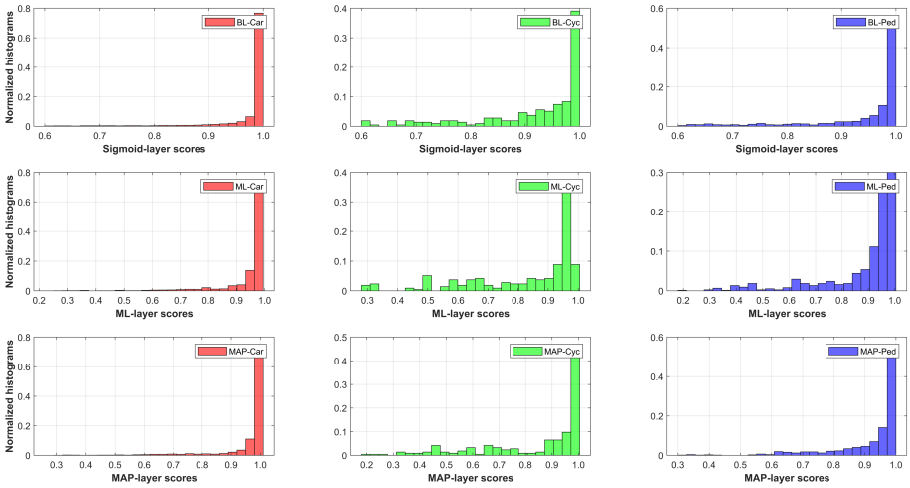
Modality	RGB			RaV			ReV		
Approach	*SgM*	*ML*	*MAP*	*SgM*	*ML*	*MAP*	*SgM*	*ML*	*MAP*
ECE	0.007	0.005	0.005	0.0367	0.013	0.027	0.031	0.013	0.031



(a) RGB modality.

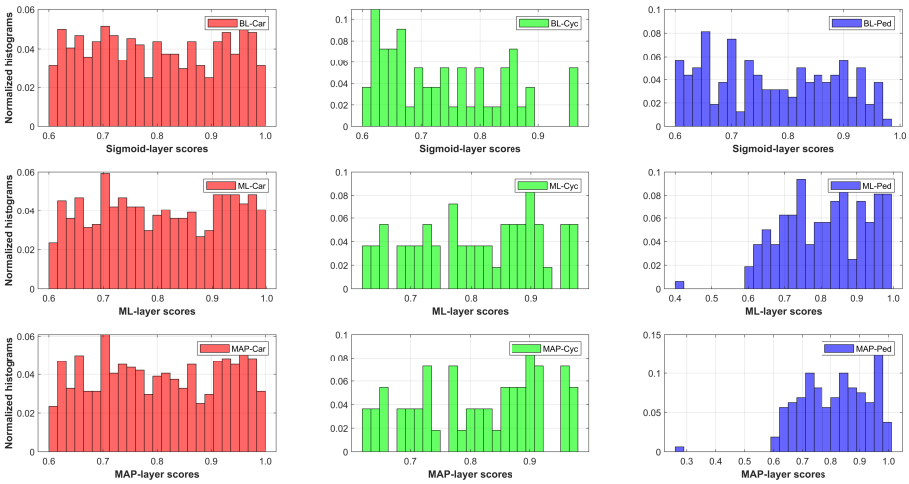


(b) RaV modality.

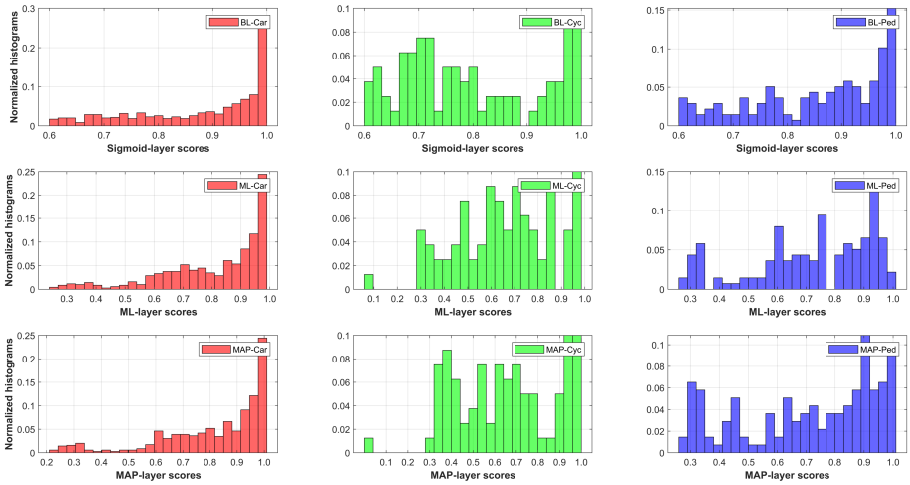


(c) ReV modality.

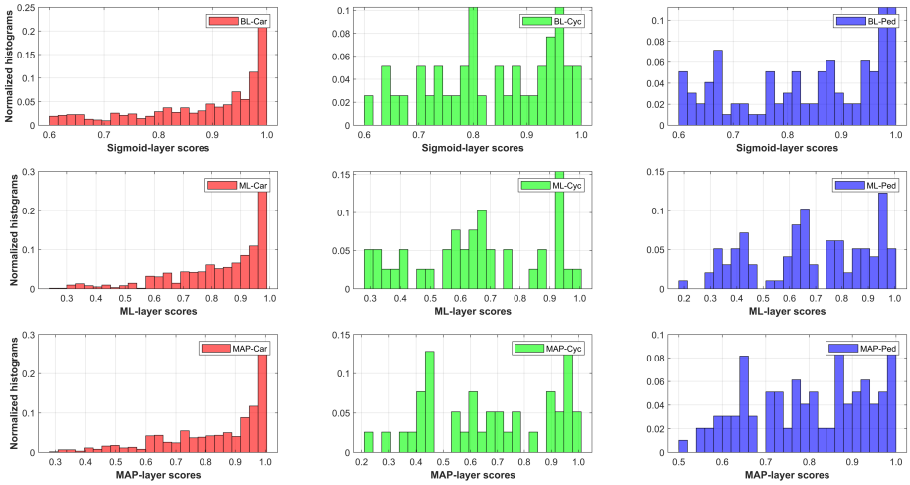
Fig. 5.15 Score distributions considering TP objects from YOLOV4 detector.



(a) RGB modality.



(b) RaV modality.



(c) ReV modality.

Fig. 5.16 Score distributions considering FP objects from YOLOV4 detector.

Table 5.9 Classification of objects without fusion strategies (Ped-Pedestrian, Car-Car, Cyc-Cyclist).

F-score									
Modalities	Image			3D Point Cloud					
Classes	RGB	RaV	ReV	64	128	256	512	1024	2048
Ped	96.29	93.33	94.46	84.67	85.21	84.36	84.14	80.30	84.70
Car	99.47	99.30	99.31	97.84	98.03	98.00	97.97	97.85	97.86
Cyc	91.93	84.56	84.70	64.97	67.07	64.29	66.06	52.23	62.51
Ave	**95.89**	92.40	92.82	82.43	**83.44**	82.22	82.72	76.79	81.69

5.2 Sensor Fusion Strategies

One of the best ways to increase confidence in decision-making is to combine data from different sensors (fusion strategies - early and late multimodal sensor fusion) *i.e.*, data from different sensors contains different information, such as cameras and LiDARs. In other words, the fusion of different modalities generally tend to improve the performance of object recognition systems. It is evident that the improvement of the result does not depend only on the data, but on the formulations of the algorithms, both non-learning and learning methods. This means that fusion strategies will not always provide better results than individual sensor information [62, 24, 14, 3, 100, 46].

Before presenting the results on fusion strategies, we need to emphasize that we work with the RGB, RaV, ReV, and $3D$ point clouds modalities. The results of the classifications using single modalities are shown in the Table 5.9.

5.2.1 Early Fusion

Early fusion fuses the data at the beginning of the machine learning/deep learning algorithm. Some data can be fused directly without any pre-processing, while other data need feature extractions before starting the training of a given algorithm [62, 24, 100, 46].

From the definition of early fusion presented in the previous paragraph, considering the combination of data without any pre-processing, we define the combinations between RGB images, RaV and ReV modalities, according to Fig. 4.12 (Subsection 4.3.1). The best result was achieved considering the RGB_RaV fusion strategy, according to the results presented in the Table 5.10.

Table 5.10 Classification of objects with early fusion strategies (Ped-Pedestrian, Car-Car, Cyc-Cyclist).

	F-score			
Classes	RaV_ReV	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	94.91	97.40	95.19	94.31
Car	99.49	99.66	99.50	99.36
Cyc	86.02	94.34	87.96	87.25
Ave	93.48	**97.14**	94.22	93.64

5.2.2 Late Fusion

Generally, late fusion strategies are simple, combining the final results after processing the training algorithms, such as the strategies presented by the equations (4.16), (4.17), (4.18), (4.19) *i.e.*, maximum, minimum, average, and product[1] respectively [108, 119, 62, 24, 14, 3, 100].

Taking into account the fusion strategies by the equations presented in the previous paragraph, we show the late fusion results in the Table 5.11, Table 5.12 and Table 5.13 where the best result was achieved through the product late fusion strategy, using the RGB, RaV, ReV and PC modalities.

5.2.2.1 ML and MAP

Taking advantage of the idea of late fusion, the results achieved by *ML* and *MAP* approach were fused according to the equations (4.16), (4.17), (4.18), (4.19). The best result achieved was through Average/Product Late Fusion, as shown in Table 5.14 and Table 5.15, considering the fusion with the RGB and RaV modalities. Furthermore, the results considering KDE as estimate for the prior probability are presented in the Table 5.16, being the Minimum strategy as the best result.

5.3 Discussion

Within the experiments performed in this thesis, a probabilistic approach for CNNs was addressed as distributions in the logit layer to best represent the outputs classification. The results reported here in the experiments are very promising, given

[1] In Appendix A.3 other late fusion results using Bayesian inference are presented.

Table 5.11 Classification of objects with late fusion strategies (Ped-Pedestrian, Car-Car, Cyc-Cyclist).

F-score				
Average Late Fusion				
Classes	RaV_ReV	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	95.60	96.89	97.50	97.62
Car	97.21	99.65	99.61	99.67
Cyc	87.74	92.68	93.76	93.08
Ave	94.43	96.41	96.85	96.79
Maximum Late Fusion				
Classes	RaV_ReV	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	95.48	96.67	97.42	97.09
Car	99.48	99.61	99.58	99.61
Cyc	88.03	92.34	93.63	92.72
Ave	94.33	96.21	96.88	96.47
Minimum Late Fusion				
Classes	RaV_ReV	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	95.76	97.16	97.54	97.35
Car	99.57	99.69	99.63	99.68
Cyc	88.56	93.02	93.28	92.91
Ave	**94.63**	92.62	96.81	96.65
Product Late Fusion				
Classes	RaV_ReV	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	95.57	96.97	97.68	97.58
Car	99.52	99.66	99.61	99.65
Cyc	88.41	93.00	93.97	94.02
Ave	94.50	**96.55**	**97.09**	**97.08**

that *ML* and *MAP* noticeably reduced the classifier overconfidence, providing a more significant distribution in terms of probabilistic interpretation.

The improvement is not as significant when analyzing objects defined as true positives. However, our concern was to develop a methodology that might reduce the values of false positives (mainly objects of the unseen class: which may be critical in robotics and autonomous driving applications) without degrading the results achieved by true positives. Note that, we have included two metrics in Table 5.1 and Table 5.4, in order to show the reduction of score values for the unseen class (in particular) and

Table 5.12 Classification of objects with late fusion strategies considering the point cloud modality (Ped-Pedestrian, Car-Car, Cyc-Cyclist).

F-score				
Average Late Fusion				
Classes	RGB_PC	RaV_PC	ReV_PC	RGB_RaV_PC
Ped	96.83	94.46	95.58	97.22
Car	99.47	99.25	99.41	99.54
Cyc	91.72	86.00	88.48	93.60
Ave	96.01	93.24	94.82	96.79
Maximum Late Fusion				
Classes	RGB_PC	RaV_PC	ReV_PC	RGB_RaV_PC
Ped	96.82	94.31	96.27	96.73
Car	99.46	99.25	99.38	99.52
Cyc	91.45	86.00	88.29	91.96
Ave	95.91	93.19	94.64	96.07
Minimum Late Fusion				
Classes	RGB_PC	RaV_PC	ReV_PC	RGB_RaV_PC
Ped	96.91	94.72	96.48	97.04
Car	99.52	99.35	99.46	99.58
Cyc	91.98	86.62	88.75	92.19
Ave	96.13	**93.56**	94.89	96.27
Product Late Fusion				
Classes	RGB_PC	RaV_PC	ReV_PC	RGB_RaV_PC
Ped	96.90	94.67	96.54	97.46
Car	99.50	99.31	99.43	99.58
Cyc	92.29	86.53	888.84	93.66
Ave	**96.23**	93.51	**94.94**	**96.90**

also demonstrating that the overconfident behavior has been mitigated for TPs and FPs.

To assess the classifier's robustness or the model uncertainty when predicting objects, we consider the ones obtained from the unseen dataset. Overall, the results are promising, since the prediction distributions were not extremities related to the results from the *SM* layer. In other words, the average scores using *ML* and *MAP* layers were significantly lower than the softmax prediction layer (the baseline), and thus, the CNNs are less prone to overconfident.

Table 5.13 Continuation of Table 5.12 (Ped-Pedestrian, Car-Car, Cyc-Cyclist).

F-score		
Average Late Fusion		
Classes	RGB_ReV_PC	RGB_RaV_ReV_PC
Ped	97.46	97.80
Car	99.57	99.60
Cyc	93.18	93.10
Ave	96.74	96.84
Maximum Late Fusion		
Classes	RGB_ReV_PC	RGB_RaV_ReV_PC
Ped	97.80	97.31
Car	99.51	99.54
Cyc	93.12	92.23
Ave	96.81	96.36
Minimum Late Fusion		
Classes	RGB_ReV_PC	RGB_RaV_ReV_PC
Ped	97.77	97.35
Car	99.58	99.60
Cyc	93.56	92.59
Ave	96.97	96.51
Product Late Fusion		
Classes	RGB_ReV_PC	RGB_RaV_ReV_PC
Ped	97.83	98.25
Car	99.56	99.66
Cyc	93.89	94.42
Ave	**97.09**	**97.44**

One potential way to improve the F-scores achieved by the *ML* and *MAP* layers would be to obtain a "perfect" match between the smoothing parameter (λ) and the number of bins in the histograms. As a consequence of the Additive Smoothing (λ), the score values equal to 0.0 and 1.0 are excluded from the prediction values. The influence of the λ parameter on the data distribution can be seen from the figures in Appendix A.2, particularly with respect to objects of the unseen class.

The results for calibration were presented through reliability diagrams, taking into account the MCE and ECE metrics. In fact, such metrics indicate the amount of predicted score values calibrated, that is, the best calibration has to present the lowest value for the MCE and ECE. However, we observed that depending on the dataset

Table 5.14 Classification of objects with late fusion strategies considering *ML*.

F-score			
Average Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	97.07	95.79	96.77
Car	99.65	99.47	99.62
Cyc	92.88	88.30	91.20
Ave	**96.53**	94.52	**95.86**
Maximum Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	96.99	95.71	95.94
Car	99.68	99.44	99.51
Cyc	92.67	87.08	88.02
Ave	96.45	94.08	94.49
Minimum Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	97.00	95.65	95.76
Car	99.68	99.45	99.47
Cyc	93.06	88.69	87.23
Ave	96.58	**94.60**	94.78
Product Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	97.07	95.60	96.20
Car	99.65	99.43	99.51
Cyc	92.88	88.13	89.92
Ave	**96.53**	94.39	95.21

and sensor modality, our approach obtained the best result in only one of the metrics *i.e.*, either the lowest value for the MCE metric or the lowest value for the ECE metric. Such fact can also be compared against the temperature scaling calibration - served as baseline and still one of the most used state-of-the-art calibration technique.

Additionally, another key factor, which contributes to validate the proposed approach, is the use of different datasets in terms of both RGB, Range-view, and Reflectance-view ($3D$ point clouds-LiDARs) modalities, considering that the sensors of the datasets have different resolutions, mainly the LiDAR sensor.

The proposed methodology also obtained good results for object detection, without degrading the results when compared to the *SgM* prediction layer, presenting better

Table 5.15 Classification of objects with late fusion strategies considering *MAP*.

F-score			
Average Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	97.07	94.09	94.27
Car	99.66	99.21	99.22
Cyc	92.67	80.22	80.78
Ave	**96.47**	91.17	91.42
Maximum Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	97.03	94.05	94.05
Car	99.66	99.21	99.21
Cyc	92.58	80.15	80.15
Ave	96.42	91.13	91.13
Minimum Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	96.96	94.35	94.43
Car	99.66	99.20	99.23
Cyc	92.47	81.18	81.52
Ave	96.36	**91.58**	**91.72**
Product Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	97.07	94.12	94.27
Car	99.66	99.21	99.22
Cyc	92.67	80.33	80.78
Ave	**96.47**	91.22	91.41

results in all cases. The improvement is more evident for the cyclist class, which contains the least number of examples. This is an interesting result that could be further investigated in future work.

Regarding the formulations of probabilistic distributions, the prior modeling by a Gaussian distribution was shown to guarantee a smoother distribution for the prediction values. Unlike the prior, the likelihood function was modeled by means of a normalized histogram *i.e.*, by a non-parametric formulation showing the probability distributions. If both the prior and the likelihood function were modeled by a uniform distribution, the final result would be similar to those achieved by the *SM* and *SgM* layers, giving that it would not offer any smoothing for the prediction values. In fact, a uniform

Table 5.16 Classification of objects with late fusion strategies considering KDE as estimate for the prior probability (*MAP*).

F-score			
Average Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	97.33	95.36	95.70
Car	99.64	99.33	99.50
Cyc	93.56	83.98	86.95
Ave	96.84	**92.89**	**94.05**
Maximum Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	97.29	95.32	95.47
Car	99.64	99.33	99.35
Cyc	93.56	83.90	84.23
Ave	96.83	92.85	93.02
Minimum Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	95.89	93.38	94.57
Car	99.40	99.28	99.17
Cyc	90.98	84.42	81.63
Ave	**96.95**	92.87	93.04
Product Late Fusion			
Classes	RGB_RaV	RGB_ReV	RGB_RaV_ReV
Ped	97.33	95.36	95.48
Car	99.64	99.33	99.38
Cyc	93.56	83.98	84.73
Ave	96.84	**92.89**	93.20

prior or likelihood would add constancy to the training data modeling, which would have little effect on the prediction values obtained by the *ML* and *MAP*. In addition to the results considering probability density functions (normalized histograms and Gaussian distributions), we present classification results considering the cumulative distribution function, as can be seen in the Appendix A.1.

The prior probability was also modeled using the KDE methodology. The results through KDE were very significant (F-score and FPR), both for RGB, RaV, and ReV modalities, as well as to the results achieved with the unseen (out-of-training) dataset.

In order to improve the classification results provided by neural networks, we present early and late fusion strategies. In fact, we verified how much the fusion of information from different modalities (sensors) contributes to the improvement of individual results provided by neural networks. The best result of the early fusion strategy was the combination of RGB and RaV modalities with an F-score of 97.14% *i.e.*, a gain of 1.3% when compared to the individual result for the RGB modality, according to the Table 5.9 and Table 5.10. Regarding the late fusion strategy, the best result was with an F-score of 97.44%, with a gain of 1.61% when compared to the RGB modality. This result was obtained through the product late fusion strategy considering the RGB, RaV, ReV, and point cloud modalities, as shown in Table 5.13.

It is interesting to note that the late fusion strategy presented a significant result when combined to the individual results provided by the proposed methodology in this thesis *i.e.*, the results obtained through the *ML* and *MAP* layers with an average gain of 0.64% of F-score, when compared to the RGB modality individually, according to tables 5.9, 5.14, and 5.15.

In conclusion, our results present an alternative to reduce overconfident predictions, providing a smoother prediction, especially with respect to the misclassified objects. In addition, fusion strategies also present alternatives to improve the results already achieved by neural networks.

Chapter 6

Conclusion and Future Work

Contents

6.1 Conclusion

In this chapter we discuss the main conclusions drawn from the thesis, including the experiments and developments on multimodality datasets, fusion strategies, as well as strategies to reduce the score values of overconfident predictions, focusing on perception systems for autonomous/intelligent vehicles. Modern perception systems have achieved satisfactory results with recent advances in deep learning and sensory technology, through data such as RGB cameras, LiDAR, RADAR, stereo, and RGB-D. The work carried out in this thesis mainly deals with object recognition and detection using deep network learned on dataset comprising RGB and LiDAR modalities.

The thesis reflects the research work and studies conducted using camera and LiDAR technologies to classify cars, cyclists, and pedestrians based on deep networks. The modalities, and their respective representations, are RGB images, range-view (RaV) and reflectance-view (ReV) maps, where range and reflectance representations have been obtained from the 3D point clouds projections (LiDAR sensor). For the last two modalities, different techniques have been employed (maximum, minimum, average, inverse distance weighting and bilateral filtering) to upsample the projected $3D$ points cloud on the $2D$ mapping coordinates, resorting by varying mask sizes (7×7, 9×9, 11×11, 13×13, and 15×15) - as detailed in the Subsection 4.1.1. The objects captured by the LiDAR sensor contain sparse points and different amounts of points (because of the distance to the objects and their dimension), making it difficult to upsample both the $3D$ points and the projected $3D$ points on the $2D$ image-plane. It cannot be overlooked that a perfect technique to sample the projected $3D$ points on the $2D$ image-plane is still an open challenge. In other words, there is no explicit closed-form technique, as far as the author knows, that upsamples the projected $3D$ points on the $2D$ image-plane.

From the different modalities we run the convolutional networks AlexNet, Inception V3[1] and PointNet to classify cars, cyclists, and pedestrian. We observed that the best result achieved was attained by Inception V3 with RGB images. However, the object classification using the range-view and reflectance-view modalities showed very satisfactory results. Unlike AlexNet and Inception V3 networks, PointNet directly runs the $3D$ point clouds. Thus, we have studied datasets containing objects having 64, 128, 256, 512, and 1024 points *i.e.*, five different datasets of $3D$ point clouds representing the

[1]In this thesis we only present the results achieved by Inception V3 CNN, because such results were better than the results achieved by AlexNet CNN. However, results with AlexNet CNN have been presented at conferences [149, 151].

objects of interest. Such datasets were generated according to the approach described in Subsection 4.1.2: *i*) objects are cropped from each frame of the $3D$ point clouds; *ii*) for each object, a clustering technique is performed to eliminate the points that do not belong to the object *i.e.*, background returns; *iii*) through upsample or downsampling step, the desired amount of points for each object is then interpolated.

In the second part of this study, fusion strategies have been addressed as an alternative to combine features of different modalities and potentially improve the performance of the models. Such strategies are known as early fusion and late fusion. In the case of early fusion we concatenate RGB with RaV images channels, RGB with ReV images channels, RGB with RaV and ReV images channels, and RaV with ReV images channels, before inserting them into the convolutional neural network *i.e.*, we obtain images with 4 (RGB and RaV), 4 (RGB and ReV), 5 (RGB, RaV, and ReV) and 2 (RaV and ReV) channels, being the combination of RGB with RaV image channels the best result using early fusion, according to the results reported in Subsection 5.2.1. With respect to late fusion strategies, we use Average, Maximum, Minimum and Normalized Product with the scores obtained from the individual classifications. The best result was achieved considering the scores of the classifications of RGB images, RaV images, ReV images, and point clouds, using the product formulation, as shown in Subsection 5.2.2. Both combining strategies achieved better results than individual classifications using single modalities.

After the classification studies, the last part of this thesis, object detection have been explored through deterministic deep neural networks, often, we observed that the top-class classification scores were overconfidence, regardless of the type of network architecture or input modalities.

In particular, the research focused on deep networks where the output represents normalized prediction scores through the softmax and sigmoid layers *i.e.*, the prediction values are in a range of $[0, 1]$. In other words, normalized prediction values by softmax and sigmoid layers generate values that are very close to zero or one ('extreme' results), which are very satisfactory results indeed but, as long as the predictions are correct. The problem with getting 'extreme' results - overconfident behaviour - is particularly critical concerning the mispredictions *i.e.*, false positives with high score values. Furthermore, most of the traditional deep networks do not consider the model's actual confidence for the predicted class in the decision-making phase. In fact, in most cases, the decision-making takes into account only the prediction value provided directly by a deep learning algorithm disregarding a proper level of confidence of the prediction (which

is unavailable for most networks) *i.e.*, quantification of uncertainty on the prediction is often overlooked. Thus, after a careful study of overconfident predictions, this thesis proposed a methodology based on Bayesian inference, considering the Maximum Likelihood (ML) and Maximum a-Posteriori (MAP) formulations, as described in Chapter 4. Such methodology provides a alternative for the scientific community, as a technique that smoothes the values of the scores provided by the prediction layers (softmax and sigmoid functions), mainly for objects detected as false positives. Unlike many calibration techniques that reduce the values of overconfident predictions, our methodology allows for a more adequate probabilistic interpretation of the predicted scores.

Additionally, we observed that the problem related to overconfidence is not related to the fact that deterministic networks do not provide an estimate of uncertainty, but because the networks are poorly calibrated. Several researches have been proposed to overcome problems related to overconfidence, as well as to identify sources of uncertainty (model uncertainty and data uncertainty) and quantify such uncertainties (Bayesian neural networks, ensemble of neural networks, and Monte Carlo Dropout). Generally, despite some overlap and controversy, such recent field can be categorized as belonging to calibration or regularization techniques. Calibration techniques act directly with the predicted values *i.e.*, after training. Differently, the regularization techniques act on the cost/loss function of the network during the training stage *i.e.*, regularization interferes with the updating of the network weights and consequently it is more complicated to perform on complex and deeper models. However, some researches emphasize that regularization techniques are also defined as calibration techniques.

An important fact regarding regularization techniques is the need for a cost/loss function, which does not follow a fixed formulation *i.e.*, the function can vary according to the type of network/model and its implementation. Thus, the cost function applied during the training of the networkshould be well suited to not compromise the uncertainty estimates and/or in the reduction of overconfident predictions. In the same way, calibration techniques must guarantee a reduction of overconfident predictions without compromising classification results.

The estimation of the prediction uncertainty of a deep neural network model can take into account the aleatoric uncertainty *i.e.*, uncertainties in the data (data acquisition process), and epistemic uncertainty *i.e.*, the network structure itself (number of layers, number of neurons, and activation functions, optimization algorithm, regularization, batch size, learning rate, number of training epochs, and so on). Evaluating the

prediction confidence or uncertainty is crucial in decision-making because an erroneous decision can lead to disaster, especially in autonomous driving where the safety of human lives are dependent on the automation algorithms.

6.2 Future Work

During the PhD study, it has been verified that the output of the softmax and sigmoid functions do not provide any measure of uncertainty of the predictions, as explained in the Subsection 3.3, where we presented different formulations to reduce overconfident predictions, as well as different formulations to determine uncertainties in deep neural networks. Basically, both the softmax and sigmoid functions provide a direct measure of classification through the maximum class score.

The prediction functions mentioned above also do not provide any information regarding the certainty that the model itself has about the predictions. Therefore, some networks consider a posterior distribution on the parameters of the learned model, and thus obtain an estimate of the uncertainty of the model with the prediction *i.e.*, considering a posterior distribution, the outputs of the previously referred layers become a random variable, and so one can have a measure of uncertainty (evaluate the output variation). In this case, the most common measures are Mutual Information, Kullback-Leibler Divergence, and the predictive variance. However, evaluating the quality of uncertainty estimates is still a challenge for the following reasons:

- uncertainty estimates depend on sophisticated methods that tend to become intractable for deep models containing millions of parameters thus, many approaches are performed by means of approximations;
- uncertainty estimates depend on the sample size as well *i.e.*, the sample size can provide a certain degree of confidence that such a sample is representative but, representativeness of a dataset seems to be an open problem in many domains - including perception systems;
- it is not easy to obtain a ground truth about uncertainty estimates of the datasets. In fact, during the studies that conducted this thesis a satisfactory/reliable ground truth about uncertainty estimates were not found;

Although the uncertainty measures, such as entropy, mutual information, Kullback-Leibler Divergence, and predictive variance, can be captured using Bayesian neural

networks, ensemble methods, or test-time data augmentation methods. However, the uncertainty obtained from out-of-distribution data (unseen data) is an open challenge. Thus, research on measurements/quantification of uncertainties in such out-of-distribution data have not been backed yet by solid formulations, and therefore further studies concerning this research direction deserve to be pursued by the community.

Appendix A

Contents

Table A.1 Comparison between the classifications obtained by the *SM* layer, *ML* and *MAP* layers in terms of average F-score and FPR (%). The performance measures on the unseen dataset are the average and the variance of the prediction scores.

			KITTI Dataset			
Modalities	**F-score**	**FPR**	**Average Score$_{FP}$**	**Variance Scores$_{FP}$**	**Average Score$_{unseen}$**	**Variance Scores$_{unseen}$**
SM$_{RGB}$	95.89	1.60	0.853	0.021	0.982	0.005
ML$_{RGB}$	95.84	1.49	0.646	0.019	0.868	0.022
MAP$_{RGB}$	95.92	1.50	0.426	0.022	0.681	0.068
SM$_{RaV}$	92.40	2.12	0.878	0.029	0.961	0.014
ML$_{RaV}$	92.18	1.81	0.634	0.032	0.804	0.055
MAP$_{RaV}$	92.27	2.07	0.436	0.029	0.709	0.083
SM$_{ReV}$	92.82	1.73	0.824	0.031	0.967	0.009
ML$_{ReV}$	92.56	1.64	0.724	0.019	0.871	0.018
MAP$_{ReV}$	92.02	1.58	0.495	0.029	0.722	0.046

Table A.2 Number of bins for *ML* and *MAP* functions.

Modality	RGB			RaV			ReV		
Bins per class	Ped	Car	Cyc	Ped	Car	Cyc	Ped	Car	Cyc
ML Layer	7	6	26	14	7	35	6	8	17
MAP Layer	9	4	10	4	3	30	5	5	20

A.1 Cumulative Distribution Function

As an alternative to the probability density functions presented in Subsection 4.2.2, cumulative distribution functions (CDF) were defined for the calculations of the *ML* and *MAP* functions. The CDF of a real valued random variable **Sc**is defined the probability that **Sc** will take a value less than or equal to **Sc**.

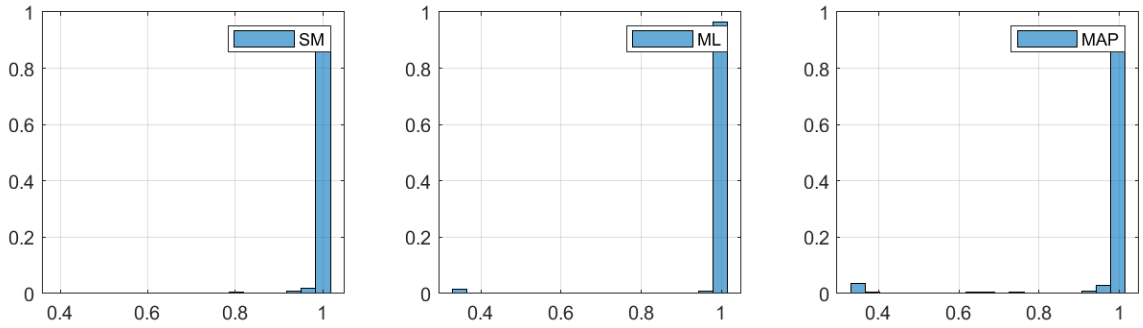
For the proposed approach, the likelihood functions were considered to be the cumulative values from the normalized histograms. However, for the prior function, we determine the cumulative distribution functions from Gaussian distributions. The results are shown in the Table A.1, while the bin numbers and the smoothing parameter are shown in Table A.2 and Table A.3, respectively.

Table A.3 Smoothing parameter (λ) for *ML* and *MAP* functions.

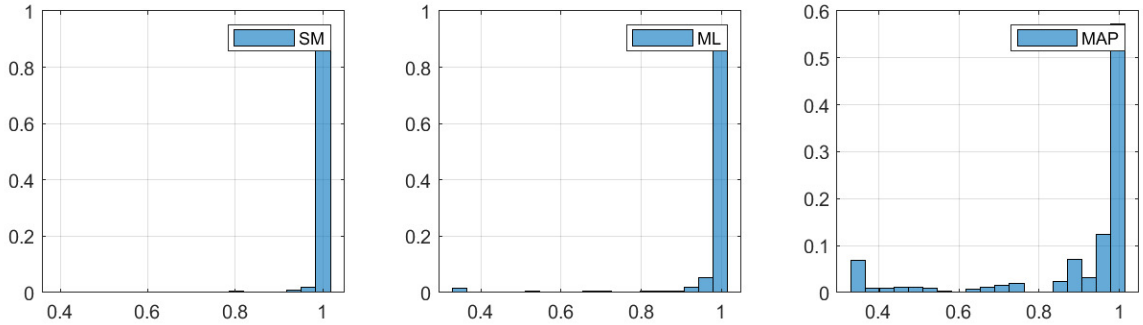
Modality	RGB	RaV	ReV
Layer	Additive Smoothing	Additive Smoothing	Additive Smoothing
ML	1×10^{-2}	1×10^{-2}	1×10^{-2}
MAP	1×10^{-2}	1×10^{-2}	1×10^{-2}

A.2 Smoothing Parameter Influence

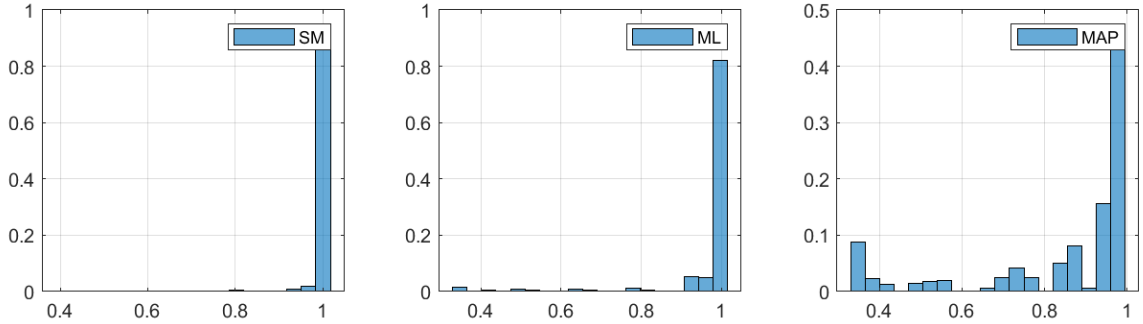
Additionally to the results presented in Subsection 5.1.1, we have implemented the proposed methodology on another state-of-the-art network, the EfficientNetB1. The performance achieved by the EfficientNetB1 to classify RGB images was a F-score of 98.67% using the softmax layer (as baseline). The result achieved through the *ML* layer is equivalent to the baseline *i.e.*, F-score = 98.67%, while using the *MAP* layer the network achieved 98.66% (almost the same). By keeping $nbins = 19$ for both cases, we have performed several runs by changing the values of λ, and the resulting F-score stabilized around 98.66% *i.e.*, very close to the F-score provided by the softmax layer (baseline). A way to choose the best values for nbins and λ could be, for instance, by reducing the values of the scores of the objects classified as false positives without degrading the results of the true positives, as illustrated by figures A.1, A.2, and A.3, where the distributions in each row were obtained through a given value for the λ parameter, considering classifications from the unseen dataset. Note that as the value of λ increases, the distributions tend to move away from the extreme values (0.0 and 1.0).



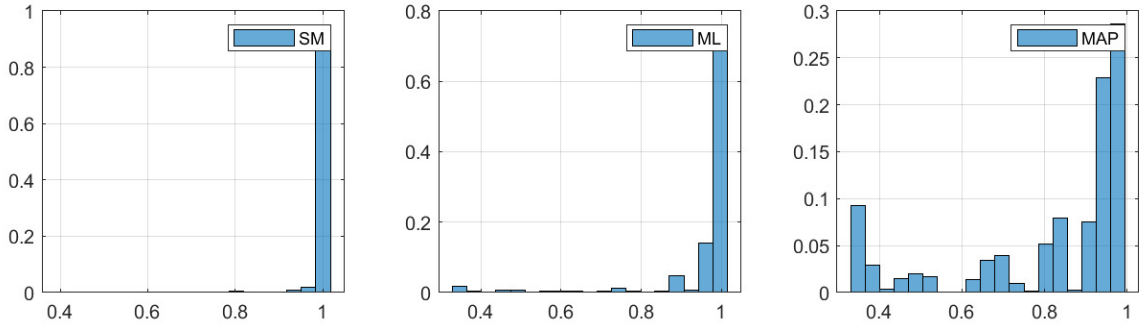
(a) $\lambda_{ML} = 1.0 \times 10^{-6}$ and $\lambda_{MAP} = 1.0 \times 10^{-6}$.



(b) $\lambda_{ML} = 9.1 \times 10^{-5}$ and $\lambda_{MAP} = 9.1 \times 10^{-5}$.

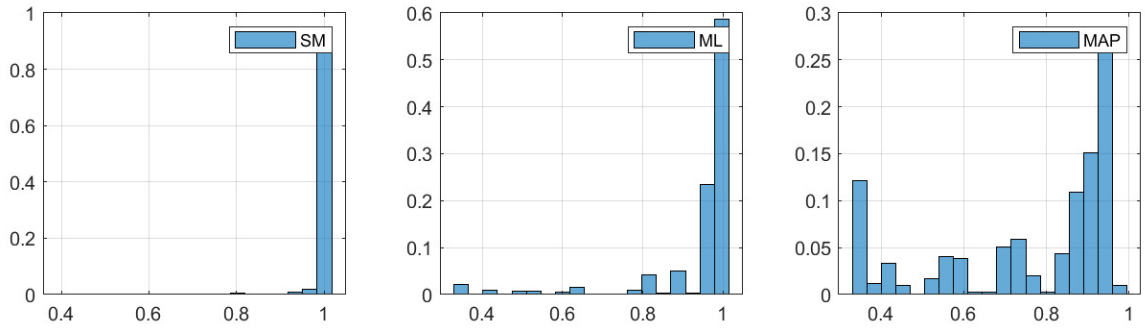


(c) $\lambda_{ML} = 2.91 \times 10^{-4}$ and $\lambda_{MAP} = 2.91 \times 10^{-4}$.

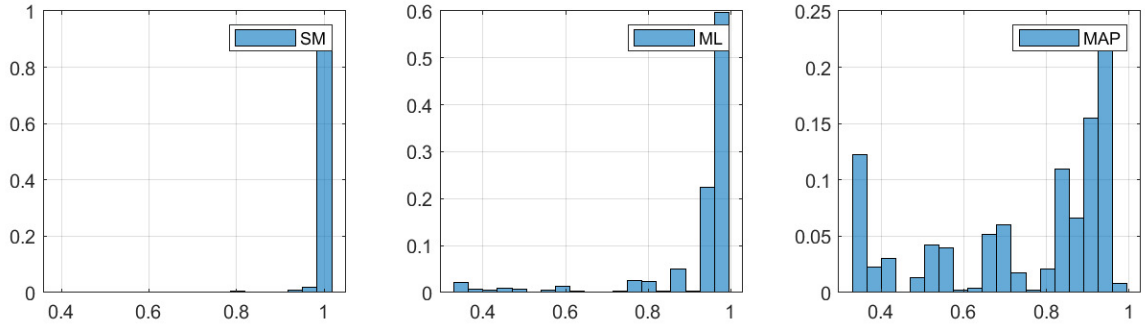


(d) $\lambda_{ML} = 3.91 \times 10^{-4}$ and $\lambda_{MAP} = 3.91 \times 10^{-4}$.

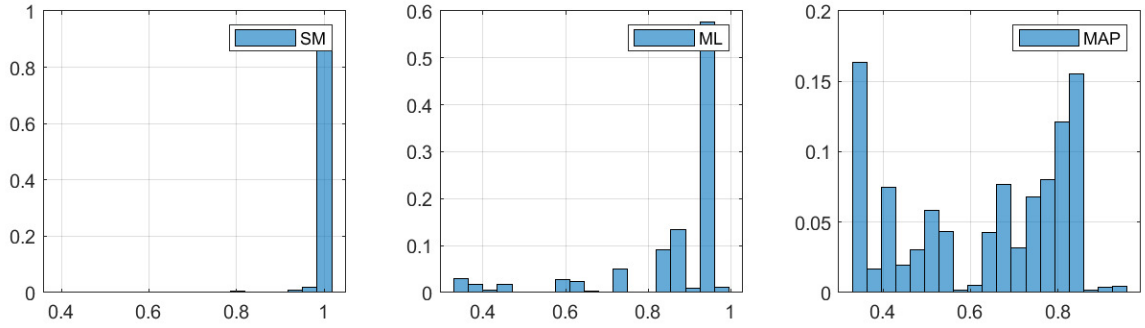
Fig. A.1 Prediction scores on the RGB unseen/non-trained data (untrained data), using *SM* layer (left side), and the proposed *ML* (center) and *MAP* (right side). The *SM* case, that does not depend on λ, serves as baseline for comparison.



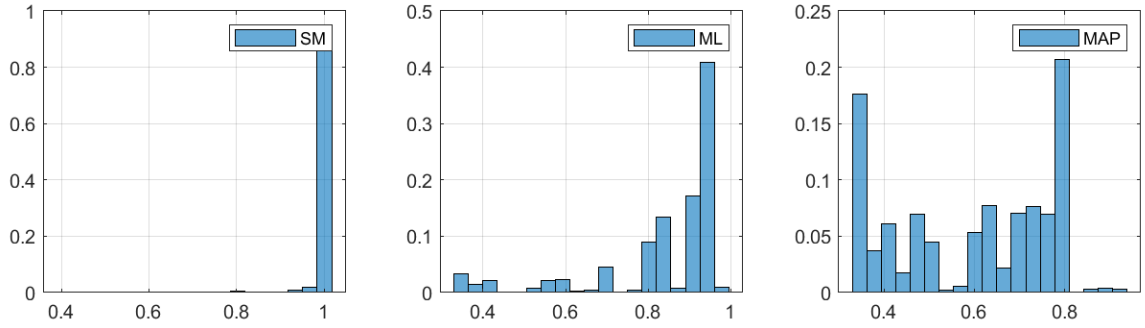
(a) $\lambda_{ML} = 7.91 \times 10^{-3}$ and $\lambda_{MAP} = 7.91 \times 10^{-3}$.



(b) $\lambda_{ML} = 9.91 \times 10^{-3}$ and $\lambda_{MAP} = 9.91 \times 10^{-3}$.

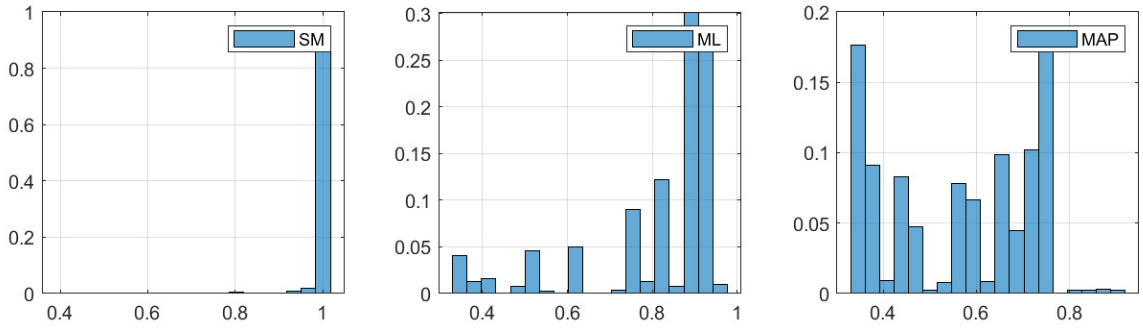


(c) $\lambda_{ML} = 2.991 \times 10^{-3}$ and $\lambda_{MAP} = 2.991 \times 10^{-3}$.

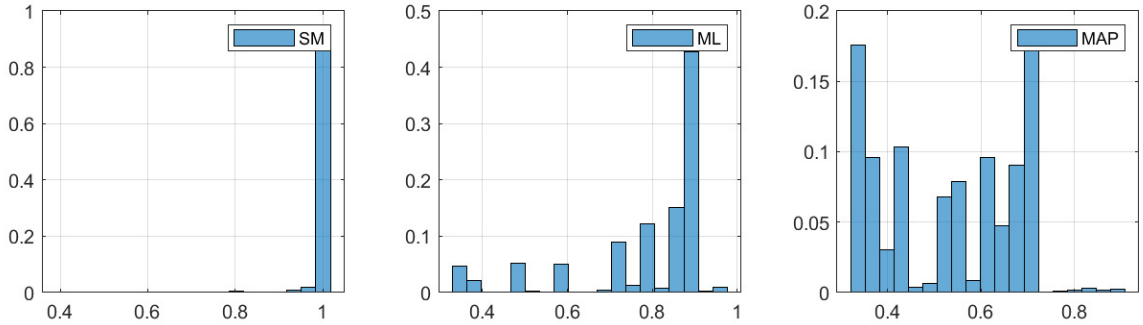


(d) $\lambda_{ML} = 3.991 \times 10^{-3}$ and $\lambda_{MAP} = 3.991 \times 10^{-3}$.

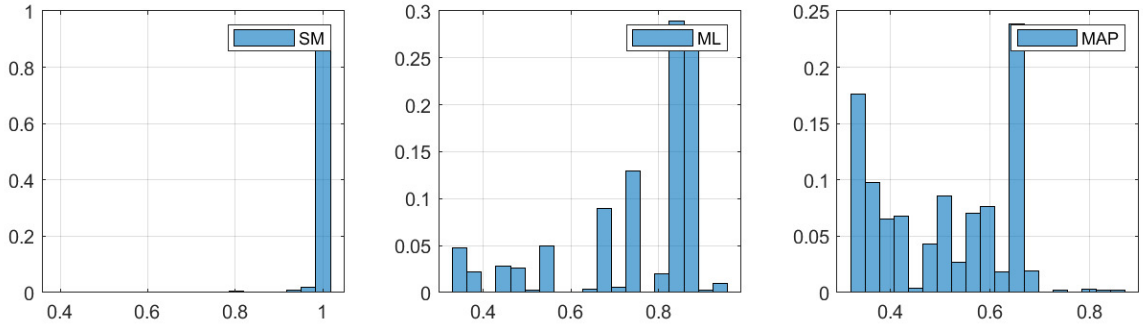
Fig. A.2 Prediction scores on the unseen data (RGB modality), for the *SM* layer (left side), and the variations in the *ML* (center) and *MAP* layers for different values of λ.



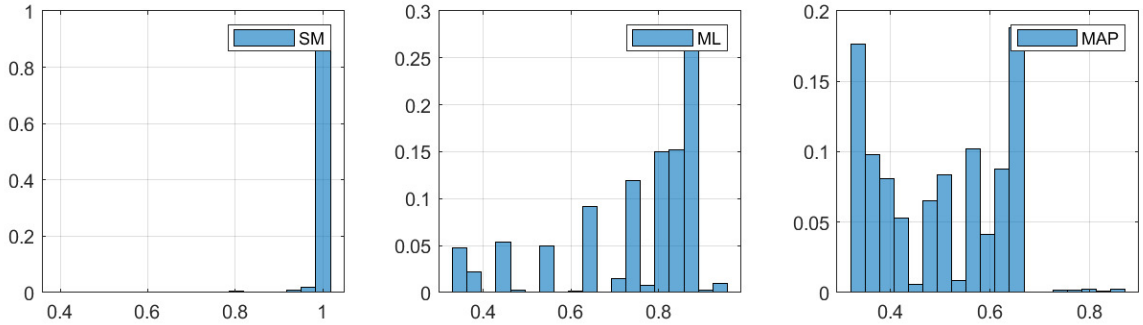
(a) $\lambda_{ML} = 5.491 \times 10^{-3}$ and $\lambda_{MAP} = 5.491 \times 10^{-3}$.



(b) $\lambda_{ML} = 6.991 \times 10^{-3}$ and $\lambda_{MAP} = 6.991 \times 10^{-3}$.



(c) $\lambda_{ML} 8.991 \times 10^{-3}$ and $\lambda_{MAP} = 8.991 \times 10^{-3}$.



(d) $\lambda_{ML} = 9.591 \times 10^{-3}$ and $\lambda_{MAP} = 9.591 \times 10^{-3}$.

Fig. A.3 Further results, in terms of the prediction scores (RGB modality), showing the influence of different values of λ on the *ML* (center) and the *MAP* (right side). The results using the *SM* layer in the left-hand side, serve as baseline for comparison.

A.3 Bayesian Inference As Late Fusion

Considering the classification problem from the different modalities (RGB images and LiDAR - $3D$ point clouds, RaV, and ReV maps), we defined such modalities as being an input vector $\mathbf{m} = \{m_1, m_2, \ldots, m_{n-1}, m_n\} \in \mathcal{M}$, where $\mathcal{M}$ is the modalities set for classes set $\mathbf{c} = \{c_1, \ldots, c_j\} \in C$, with j representing the number of classes and n being the number of modalities.

Since the problem is being considered as a classifier for three classes, the fusion between the modalities involves estimating the probabilities according to (A.1)-(A.3):

$$P(c = c_1 | m_1, m_2, \ldots, m_{n-1}, m_n) \tag{A.1}$$

$$P(c = c_2 | m_1, m_2, \ldots, m_{n-1}, m_n) \tag{A.2}$$

$$P(c = c_3 | m_1, m_2, \ldots, m_{n-1}, m_n), \tag{A.3}$$

at every point $(m_1, m_2, \ldots, m_{n-1}, m_n) \in \mathcal{M}$. Thus, the fusion strategy is developed based on the joint probability of the classes, also by applying the Product Rule (Chain Rule) as in:

$$P(m_1, m_2, \ldots, m_{n-1}, m_n, \mathbf{c}) = P(m_1) \Pi_{i=2}^{n} P(m_i | m_1, \ldots, m_{n-1}, \mathbf{c}), \tag{A.4}$$

$\forall\ m_{1,\ldots,n} \in \mathcal{M}$ and $c_{1,\ldots,j} \in C$.

Considering conditional independence, the conditional probability distribution over $m_{1\ldots n}$ factorized for every value of $\mathbf{c}$ is given by:

$$P(m_1, m_2, \ldots, m_{n-1}, m_n | \mathbf{c}) = P(m_1|\mathbf{c}) P(m_2|\mathbf{c}) \ldots P(m_{n-1}|\mathbf{c}) P(m_n|\mathbf{c}), \tag{A.5}$$

where m_1, m_2, ..., m_{n-1}, m_n are conditionally independent given $\mathbf{c}$. In this way, the conditional probability decomposition by means of the Product Rule is given as:

$$P(m_1, m_2, \ldots, m_n, \mathbf{c}) = P(m_1, m_2, \ldots, m_n | \mathbf{c}) P(\mathbf{c}) \tag{A.6}$$

$$P(\mathbf{c}, m_1, m_2, \ldots, m_n) = P(\mathbf{c} | m_1, m_2, \ldots, m_n) P(m_1, m_2, \ldots, m_n) \tag{A.7}$$

$$P(m_1, m_2, \ldots, m_n, \mathbf{c}) = P(\mathbf{c}, m_1, m_2, \ldots, m_n), \tag{A.8}$$

and the Bayes' Rule is derived as:

$$P(\mathbf{c}|m_1, m_2, \ldots, m_n) = \frac{P(m_1, m_2, \ldots, m_n|\mathbf{c})P(\mathbf{c})}{P(m_1, m_2, \ldots, m_n)}. \tag{A.9}$$

Considering that m_1, m_2, ..., m_n are independent, the model evidence is defined as $P(m_1, m_2, \ldots, m_n) = P(m_1)P(m_2), \ldots, P(m_n) \neq 0$. The term $P(\mathbf{c}|m_1, m_2, \ldots, m_n)$ is the posterior probability, $P(m_1, m_2, \ldots, m_n|\mathbf{c})$ is the likelihood function and, $P(\mathbf{c})$ is the prior probability.

As the classification consists of three classes, (A.9) is formulated for each class:

$$P(c_1|m_1, m_2, \ldots, m_n) = \frac{P(m_1, m_2, \ldots, m_n|c_1)P(c_1)}{P(m_1, m_2, \ldots, m_n)} \tag{A.10}$$

$$P(c_2|m_1, m_2, \ldots, m_n) = \frac{P(m_1, m_2, \ldots, m_n|c_2)P(c_2)}{P(m_1, m_2, \ldots, m_n)} \tag{A.11}$$

$$P(c_3|m_1, m_2, \ldots, m_n) = \frac{P(m_1, m_2, \ldots, m_n|c_3)P(c_3)}{P(m_1, m_2, \ldots, m_n)}, \tag{A.12}$$

where $P(m_1, m_2, \ldots, m_n)$ is often determined by the total probability law. In such manner, (A.10)-(A.12) can be rewritten using the *per-class* expression, with the added smoothing parameter (λ):

$$P(c_i|m_1, m_2, \ldots, m_n) = \frac{P(m_1, m_2, \ldots, m_n|c_i)P(c_i) + \lambda}{\sum_{i=1}^{nc} P(m_1, m_2, \ldots, m_n|c_i)P(c_i) + \lambda}. \tag{A.13}$$

Considering three modalities which are conditionally independent, the (A.13) can be written as:

$$P(c_i|m_1, m_2, \ldots, m_n) = \frac{P(m_1|c_i)P(m_2|c_i)P(m_3|c_i)P(c_i) + \lambda}{\sum_{i=1}^{nc} P(m_1, m_2, \ldots, m_n|c_i)P(c_i) + \lambda}. \tag{A.14}$$

The smoothing parameter has a significant influence on the result (average F-score) of the fusion strategy in A.13, as shown in Table A.4, where the prior in (A.13) was obtained by fitting a multi-Gaussian with the training data of the RGB modality, as

Table A.4 Influence of the λ parameter on the Bayesian inference late fusion strategy.

Parameter	Average F-score (%)			
λ	RaV_ReV	RGB_RaV	RGB_ReV	RGB_RaV_ReV
10^{-1}	94.66	96.63	97.06	97.24
10^{-2}	95.01	96.83	97.11	97.37
10^{-3}	95.51	97.11	97.20	97.41
10^{-4}	95.70	97.18	97.17	97.58
10^{-5}	95.72	97.18	97.17	97.58
10^{-6}	95.72	97.18	97.17	97.58
10^{-7}	95.72	97.18	97.17	97.58

defined in:

$$Prior_{Gaussian} = a_1\,\mathrm{e}^{-\frac{(b_1-x)^2}{{c_1}^2}} + a_2\,\mathrm{e}^{-\frac{(b_2-x)^2}{{c_2}^2}} + a_3\,\mathrm{e}^{-\frac{(b_3-x)^2}{{c_3}^2}}. \tag{A.15}$$

considering $a1 = -0.004092$, $b1 = 7.71$, $c1 = 0.06866$, $a2 = 0.4323$, $b2 = 7.346$, $c2 = 1.281$, $a3 = 0.23$, $b3 = 4.619$, and $c3 = 0.8774$.

A.4 Weighted Object Distance

As stated in Section 1.3, the disadvantage of LiDAR sensor is the limited range, which interferes in the number of the captured objects at a given distance, as shown in Fig. A.4 [69], that was computed from the vehicle (cars, vans, and trucks), cyclist and pedestrian classes, where the training dataset has 24484 objects (vehicles=20632, pedestrians=2827 and cyclists=1025), 2721 samples on the validation dataset (vehicles=2293, pedestrians=314 and cyclists=114), and 11659 samples on the test dataset (vehicles=9825, pedestrian=1346 and cyclists=488).

Thus, taking into account the relationship between the number of objects and the distance of each object from the LiDAR sensor, we propose a new late fusion strategy [152] taking the form of a weighted average (w) obtained from distances of the RaV maps, and PCs with the normalized classifiers F-scores on the training dataset, as in Fig. A.5 *i.e.*, Fig. A.5a shows the normalized average F-score with maximum value of 0.5, therefore we guarantee that the RGB model outputs (y_{RGB}) will have a maximum weight equal to 0.5, while the Fig. A.5b was normalized with a maximum value of 0.333.

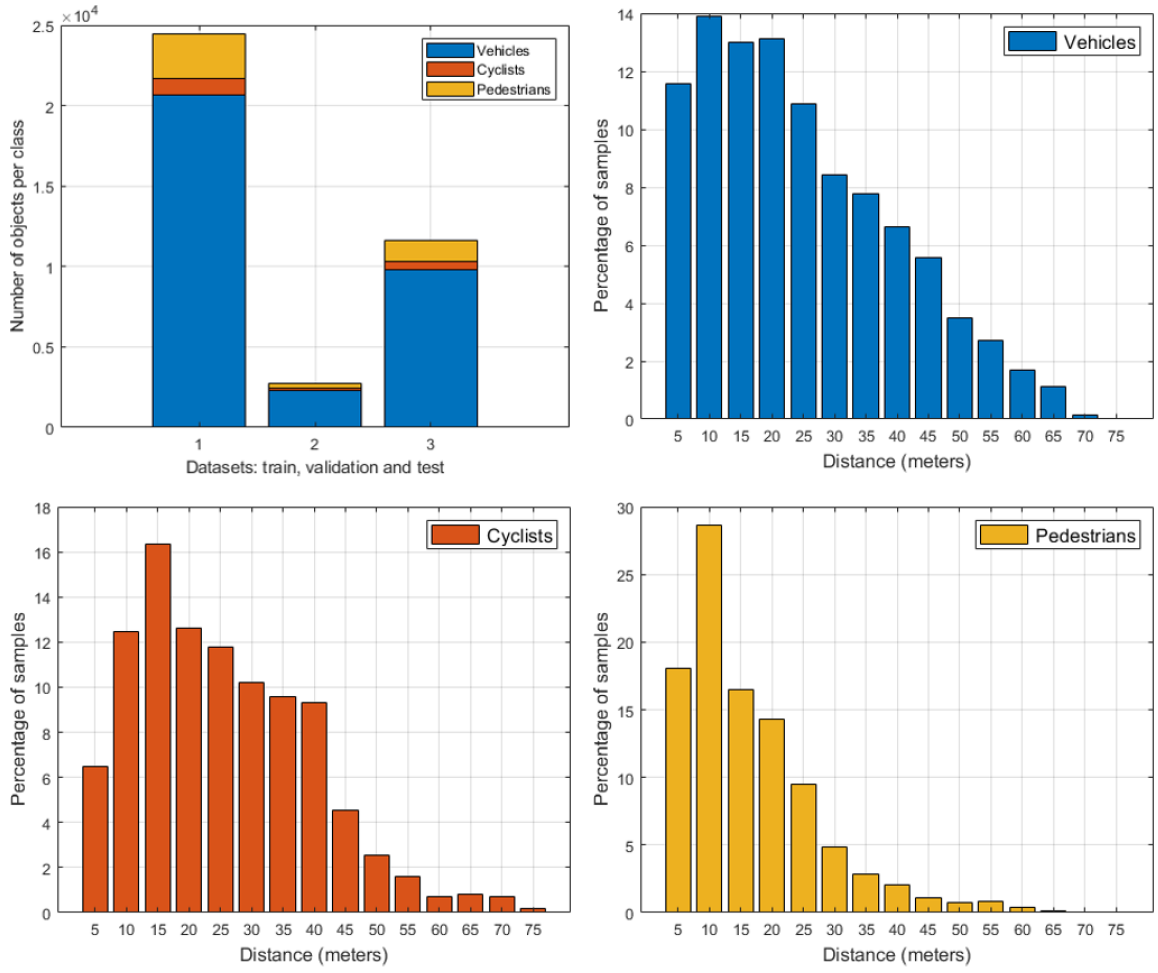
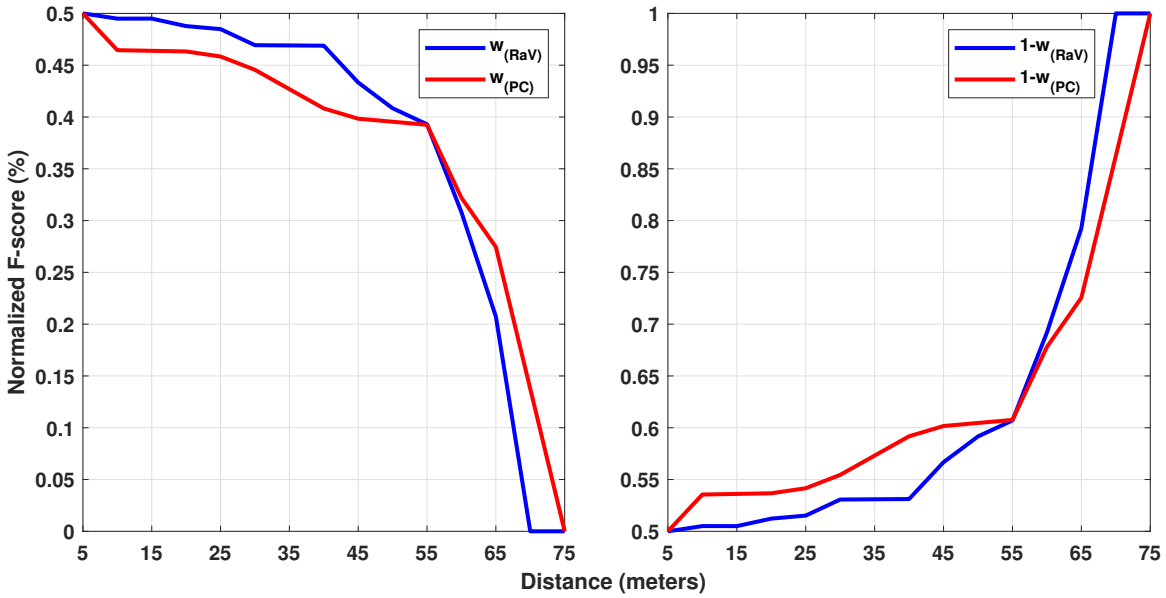
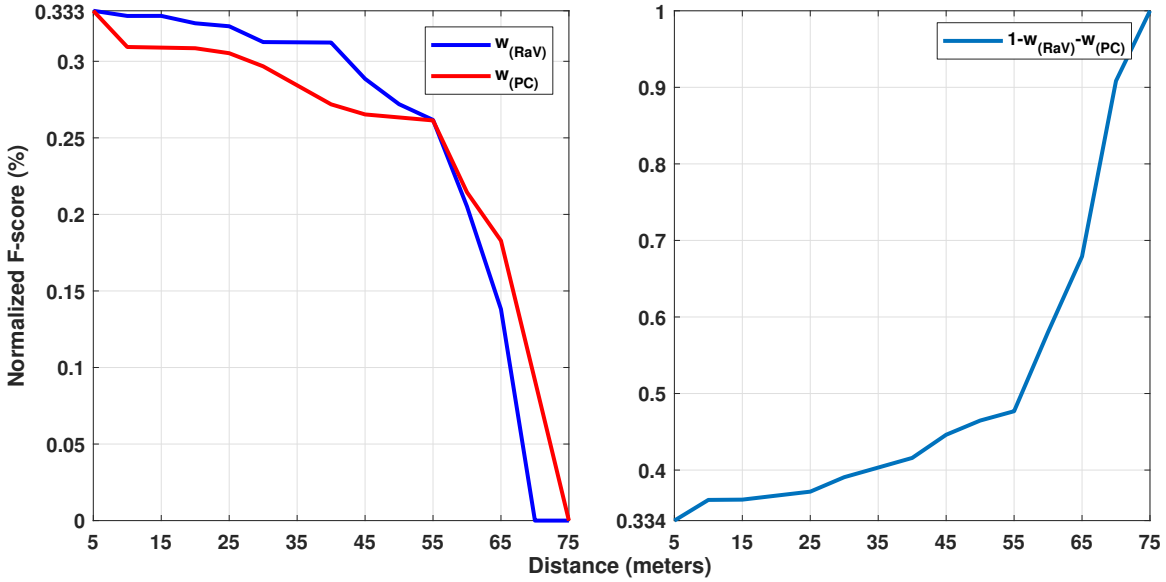


Fig. A.4 The bar-graph (top-left) shows the number of samples per class (vehicles, cyclists, and pedestrians) on the training, validation and testing datasets, respectively. The others graphs show the distribution of samples, separated by the categories and by the distance in meters.

The motivation comes from the fact that the LiDAR deep-models performance decrease as the distance to the objects increase. Unlike models that process data from the LiDAR sensor, models that process information from RGB modality maintain a relatively uniform performance with respect to the distance from objects.



(a) Curves for two modalities: $(1-w_{(PC)})y_{(RGB)}$ and $(1-w_{(RaV)})y_{(RGB)}$.



(b) Curves for three modalities: $(1-w_{(PC)}-w_{(RaV)})y_{(RGB)}$.

Fig. A.5 These curves show the normalized average F-scores obtained from the LIDAR-based model (RaV and PC) for increasing distance of objects. The curves on the left figure represent the weights for the RaV and PC modalities, while the curves on the right figure are the weights for the RGB modality *i.e.*, the weight w_i.

The new strategy takes into account the scores of the RGB, PC and RaV modalities, according to the equations (A.16)-(A.18):

$$Y_{(RGB,PC)} = (1-w_{(PC)})y_{(RGB)} + w_{(PC)}y_{(PC)} \tag{A.16}$$

$$Y_{(RGB,RaV)} = (1-w_{(RaV)})y_{(RGB)} + w_{(RaV)}y_{(RaV)} \tag{A.17}$$

$$Y_{(RGB,PC,RaV)} = (1-w_{(PC)}-w_{(RaV)})y_{(RGB)} + w_{(PC)}y_{(PC)} + w_{(RaV)}y_{(RaV)}, \tag{A.18}$$

Table A.5 F-score, for single modalities, on the test dataset.

Modalities	RGB	RaV	PC
F-score	96.24	89.55	88.46

Table A.6 Classification results on the training, in terms of F-score (in %), for the $3D$ point clouds using the PointNet model.

Classes	PC				
	64	128	256	512	1024
Ped.	75.65	86.85	**95.70**	91.93	86.10
Veh.	96.45	98.42	**99.41**	99.01	98.04
Cyc.	16.90	72.63	**91.74**	82.41	70.43
Ave.	63.00	85.97	**96.62**	91.12	84.86

where $y_{(RGB)}$, $y_{(RaV)}$, and $y_{(PC)}$ are the scores of each classified object, and $y_{(RGB,PC)}$, $y_{(RGB,RaV)}$, and $y_{(RGB,PC,RaV)}$ are the scores after the fusion, and the w is the relationship between normalized F-score and object distance obtained according to the curve shown in Fig. A.5 (F-score $\times$ distance).

The scores $y_{(RGB)}$, $y_{(RaV)}$ were obtained from the Inception V3 CNN, while $y_{(PC)}$ were obtained from the PointNet CNN. All networks were trained from the scratch. The training results are given through the F-score metric, considering the average of the F-scores of the classes, as presented by Table A.5. However, the classification result for $3D$ point clouds was computed for different amounts of points that integrate the $3D$ objects, according to Section 4.1.2, in order to define which point cloud dataset presents the best result, and thus, consider such dataset in the fusion strategies. In this case, the best result in the classification of $3D$ objects was achieved through the dataset of 256 points for each object, as showed in Table A.6.

The proposed approach, defined as Average Weighting Range (AWR), was compared with the fusion strategies defined in (4.16)-(4.19) *i.e.*, formulations of late fusion as maximum (Max), minimum (Min), average (Ave), normalized product (N-Prod), as well as learning strategies based on a SVM (support vector machine) [97] and a GA (genetic algorithm) [193, 13]. However, SVM received features from CNNs, while GA determined the parameters that maximized the F-score according to the fitness

Table A.7 Average F-score, using late fusion and multimodality representations on the testing set.

Late Fusion	Modalities		
	RGB_RaV	RGB_PC	RGB_RaV_PC
Max	96.88	96.91	96.75
Min	97.03	96.95	96.97
Ave	96.88	97.00	96.52
N-Prod	97.14	**97.10**	97.01
GA	97.05	97.03	**97.27**
SVM	96.46	96.59	96.24
AWR	**97.22**	96.98	96.26

functions (A.19)-(A.21):

$$Y_{(RGB,PC)} = I_1 y_{(RGB)} + I_2 y_{(PC)} \tag{A.19}$$

$$Y_{(RGB,RaV)} = I_1 y_{(RGB)} + I_2 y_{(RaV)} \tag{A.20}$$

$$Y_{(RGB,PC,RaV)} = I_1 y_{(RGB)} + I_2 y_{(PC)} + I_3 y_{(RaV)}, \tag{A.21}$$

where I_1, I_2, and I_3 are individuals ("chromosomes"), and y values are the modalities scores RGB, RaV, and PC.

The results using late fusion strategies are shown in Table A.7, where the overall classification performance surpassed the single modalities. The traditional methods of late fusion, as Max, Min, Ave, N-Prod, SVM and GA, have presented satisfactory performances, mainly for the fusion modalities $Y_{(RGB,PC)}$ and $Y_{(RGB,PC,RaV)}$ using N-Product and GA, respectively, which have represented the best overall performance for those two modalities. However, the result achieved by the proposed methodology was very close to that achieved by GA.

The present study is promising and it is worth to pay more attention, particularly on the idea of a performance measure, regarding object distances which can be taken into consideration in multi-classifiers combination.

References

[1] Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., and Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297.

[2] Ahmed, S., Huda, M. N., Rajbhandari, S., Saha, C., Elshaw, M., and Kanarachos, S. (2019). Pedestrian and cyclist detection and intent estimation for autonomous vehicles: A survey. *Applied Sciences*, 9(11).

[3] Akilan, T., Wu, Q. M. J., Safaei, A., and Jiang, W. (2017). A late fusion approach for Harnessing multi-CNN model high-level features. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 566–571.

[4] Aly, S. (2014). Partially occluded pedestrian classification using histogram of oriented gradients and local weighted linear kernel support vector machine. *IET Computer Vision*, 8(6):620–628.

[5] Apicella, A., Donnarumma, F., Isgrò, F., and Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, 138:14–32.

[6] Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D., and Mouzakitis, A. (2019). A survey on 3*d* object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10).

[7] Asvadi, A., Garrote, L., Premebida, C., Peixoto, P., and J. Nunes, U. (2018a). Multimodal vehicle detection: fusing 3D-LIDAR and color camera data. *Pattern Recognition Letters*, 115:20–29.

[8] Asvadi, A., Garrote, L., Premebida, C., Peixoto, P., and Nunes, U. J. (2017). DepthCN: Vehicle detection using 3D-LIDAR and convnet. In *IEEE International Conference on Intelligent Transportation Systems*, pages 1–6.

[9] Asvadi, A., Garrote, L., Premebida, C., Peixoto, P., and Nunes, U. J. (2018b). Real-time deep convnet-based vehicle detection using 3D-LIDAR reflection intensity data. In *ROBOT: Third Iberian Robotics Conference*, pages 475–486. Springer International Publishing.

[10] Atzmon, M., Maron, H., and Lipman, Y. (2018). Point convolutional neural networks by extension operators. *ACM Transactions on Graphics*, 37(4).

[11] Awad, A. I. and Hassaballah, M. (2016). *Image Feature Detectors and Descriptors: Foundations and Applications*. Springer Publishing Company, Incorporated, 1st edition.

[12] Baker, N., Lu, H., Erlikhman, G., and Kellman, P. J. (2020). Local features and global shape information in object classification by deep convolutional neural networks. *Vision Research*, 172:46–61.

[13] Bakırlı, G., Birant, D., and Kut, A. (2011). An incremental genetic algorithm for classification and sensitivity analysis of its parameters. *Expert Systems with Applications*, 38(3):2609 – 2620.

[14] Baltrušaitis, T., Ahuja, C., and Morency, L.-P. (2019). Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443.

[15] Bansal, M., Kumar, M., and Kumar, M. (2021). 2D object recognition techniques: State-of-the-art work. *Archives of Computational Methods in Engineering*, 28.

[16] Barrera, A., Guindel, C., Beltrán, J., and García, F. (2020). BirdNet+: End-to-end 3D object detection in lidar bird's eye view. In *IEEE 23rd International Conference on Intelligent Transportation Systems*, pages 1–6.

[17] Beltrán, J., Guindel, C., Moreno, F. M., Cruzado, D., García, F., and De La Escalera, A. (2018). BirdNet: A 3D object detection framework from lidar information. In *IEEE 21st International Conference on Intelligent Transportation Systems*, pages 3517–3523.

[18] Ben-Shabat, Y., Lindenbaum, M., and Fischer, A. (2018). 3DmFV: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters*, 3(4):3145–3152.

[19] Bertozzi, M., Broggi, A., Cellario, M., Fascioli, A., Lombardi, P., and Porta, M. (2002). Artificial vision in road vehicles. *Proceedings of the IEEE*, 90(7):1258–1271.

[20] Bertozzi, M., Broggi, A., and Fascioli, A. (2000). Vision-based intelligent vehicles: State of the art and perspectives. *Robotics and Autonomous Systems*, 32(1):1 – 16.

[21] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg.

[22] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *PMLR Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1613–1622.

[23] Bochkovskiy, A., Wang, C., and Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934.

[24] Boulahia, S. Y., Amamra, A., Madi, M. R., and Daikh, S. (2021). Early, intermediate and late fusion strategies for robust deep learning-based multimodal action recognition. *Machine Vision and Applications*, 32(121):1–18.

[25] Broggi, A., Bertozzi, M., Fascioli, A., and Conte, G. (1999). *Automatic Vehicle Guidance: The Experience of the ARGO Autonomous Vehicle*. World Scientific.

[26] Broggi, A., Cerri, P., Debattisti, S., Laghi, M. C., Medici, P., Molinari, D., Panciroli, M., and Prioletti, A. (2015). PROUD—Public road urban driverless-car test. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3508–3519.

[27] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). nuScenes: A multimodal dataset for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[28] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229.

[29] Chai, Y., Sun, P., Ngiam, J., Wang, W., Caine, B., Vasudevan, V., Zhang, X., and Anguelov, D. (2021). To the point: Efficient 3D object detection in the range image with graph convolution kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 16000–16009.

[30] Chen, H., Huang, Y., Tian, W., Gao, Z., and Xiong, L. (2021). MonoRUn: Monocular 3D object detection by reconstruction and uncertainty propagation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10379–10388.

[31] Chen, Q., Sun, L., Wang, Z., Jia, K., and Yuille, A. L. (2020). Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots. In *16th European Conference on Computer Vision*, volume 12366, pages 68–84.

[32] Chen, S. F. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Technical report, Harvard Computer Science Group Technical Report.

[33] Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017). Multi-view 3D object detection network for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition,*, pages 6526–6534.

[34] Chen, Y., Liu, S., Shen, X., and Jia, J. (2019). Fast point R-CNN. In *IEEE International Conference on Computer Vision*, pages 9774–9783.

[35] Chen, Z. and Huang, X. (2019). Pedestrian detection for autonomous vehicle using multi-spectral cameras. *IEEE Transactions on Intelligent Vehicles*, 4(2):211–219.

[36] Cheng, J. and Vasconcelos, N. (2022). Calibrating deep neural networks by pairwise constraints. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 13699–13708.

[37] Claussmann, L., Revilloud, M., Gruyer, D., and Glaser, S. (2020). A review of motion planning for highway autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):1826–1848.

[38] Corbière, C., THOME, N., Bar-Hen, A., Cord, M., and Pérez, P. (2019). Addressing failure prediction by learning model confidence. In *Advances in Neural Information Processing Systems*, volume 32.

[39] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223.

[40] Cui, Y., Chen, R., Chu, W., Chen, L., Tian, D., Li, Y., and Cao, D. (2021). Deep learning for image and point cloud fusion in autonomous driving: A review. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–18.

[41] Córdoba, A., Astrain, J. J., Villadangos, J., Azpilicueta, L., López-Iturri, P., Aguirre, E., and Falcone, F. (2017). SesToCross: Semantic expert system to manage single-lane road crossing. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1221–1233.

[42] Dai, J., Li, Y., He, K., and Sun, J. (2016). R-FCN: Object detection via region-based fully convolutional networks. In *30th International Conference on Neural Information Processing Systems*, page 379–387. Curran Associates Inc.

[43] Derbel, O. and Landry, R. J. (2018). Belief and fuzzy theories for driving behavior assessment in case of accident scenarios. *International Journal of Automotive Technology*, 19(1):167–177.

[44] DeVries, T. and Taylor, G. W. (2018). Learning confidence for out-of-distribution detection in neural networks. *CoRR, arXiv :1802.04865*.

[45] Dikmen, M. and Burns, C. (2017). Trust in autonomous vehicles: The case of tesla autopilot and summon. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 1093–1098.

[46] D'mello, S. K. and Kory, J. (2015). A review and meta-analysis of multimodal affect detection systems. *ACM Comput. Surv.*, 47(3).

[47] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

[48] Drews, F., Feng, D., Faion, F., Rosenbaum, L., Ulrich, M., and Gläser, C. (2022). Deepfusion: A robust and modular 3d object detector for lidars, cameras and radars. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 560–567.

[49] Drive.ai (2021). The self-driving car. https://cars.stanford.edu/blog/me302c-future-automobile-mobility-entrepreneurship-weekly-blog-post-carol-reiley-co-founder-and. Accessed on 07/22/2021.

[50] Duan, K., Xie, L., Qi, H., Bai, S., Huang, Q., and Tian, Q. (2020). Corner proposal network for anchor-free, two-stage object detection. In *European Conference on Computer Vision*, pages 399–416.

[51] Dudek, G. and Jenkin, M. (2010). *Computational Principles of Mobile Robotics*. Cambridge University Press, New York, NY, USA, 2nd edition.

[52] Endsley, M. R. (2017). Autonomous driving systems: A preliminary naturalistic study of the Tesla Model S. *Journal of Cognitive Engineering and Decision Making*, 11(3):225–238.

[53] Enzweiler, M. and Gavrila, D. (2011). A multilevel mixture-of-experts framework for pedestrian classification. *IEEE Transactions on Image Processing*, 20(10):296–2979.

[54] Everett, H. R. (1995). *Sensors for Mobile Robots: Theory and Application*. A. K. Peters, Ltd., Natick, MA, USA.

[55] Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Gläser, C., Timm, F., Wiesbeck, W., and Dietmayer, K. (2021). Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360.

[56] Feng, D., Rosenbaum, L., and Dietmayer, K. (2018). Towards safe autonomous driving: Capture uncertainty in the deep neural network for LiDAR 3D vehicle detection. In *21st International Conference on Intelligent Transportation Systems*, pages 3266–3273.

[57] Feng, D., Rosenbaum, L., Timm, F., and Dietmayer, K. (2019). Leveraging heteroscedastic aleatoric uncertainties for robust real-time lidar 3D object detection. In *IEEE Intelligent Vehicles Symposium*, pages 1280–1287.

[58] Fernandes, L. C., Souza, J. R., Pessin, G., Shinzato, P. Y., Sales, D., Mendes, C., Prado, M., Klaser, R., Magalhães, A. C., Hata, A., Pigatto, D., Branco, K. C., Grassi, V., Osorio, F. S., and Wolf, D. F. (2014). CaRINA intelligent robotic car: Architectural design and applications. *Journal of Systems Architecture*, 60(4):372–392.

[59] Filgueira, A., González-Jorge, H., Lagüela, S., Díaz-Vilariño, L., and Arias, P. (2017). Quantifying the influence of rain in LiDAR performance. *Measurement*, 95:143–148.

[60] Franke, U., Gavrila, D., Gorzig, S., Lindner, F., Puetzold, F., and Wohler, C. (1998). Autonomous driving goes downtown. *IEEE Intelligent Systems and their Applications*, 13(6):40–48.

[61] Frenkel, L. and Goldberger, J. (2022). Network calibration by temperature scaling based on the predicted confidence. In *30th European Signal Processing Conference (EUSIPCO)*, pages 1586–1590.

[62] Gadzicki, K., Khamsehashari, R., and Zetzsche, C. (2020). Early vs late fusion in multimodal convolutional neural networks. In *IEEE 23rd International Conference on Information Fusion (FUSION)*, pages 1–6.

[63] Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

[64] Gal, Y. and Ghahramani, Z. (2016a). Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *Workshop track, 4th International Conference on Learning Representations*.

[65] Gal, Y. and Ghahramani, Z. (2016b). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *PMLR Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 1050–1059.

[66] Gal, Y., Hron, J., and Kendall, A. (2017). Concrete dropout. In *31st Advances in Neural Information Processing Systems*, volume 30.

[67] Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J., and Li, D. (2018). Object classification using cnn-based fusion of vision and LIDAR in autonomous vehicle environment. *IEEE Transactions on Industrial Informatics*, 14(9):4224–4231.

[68] Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A. M., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., and Zhu, X. X. (2022). A survey of uncertainty in deep neural networks. *CoRR, arXiv*, abs/2107.03342.

[69] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 32(11):1231–1237.

[70] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361.

[71] Girshick, R. (2015). Fast R-CNN. In *IEEE International Conference on Computer Vision*, pages 1440–1448.

[72] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587.

[73] Gong, Y., Lin, X., Yao, Y., Dietterich, T. G., Divakaran, A., and Gervasio, M. (2021). Confidence calibration for domain generalization under covariate shift. In *IEEE International Conference on Computer Vision*, pages 8958–8967.

[74] Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. Adaptive Computation and Machine Learning. MIT Press.

[75] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. *CoRR, arXiv*, 1412.6572.

[76] Goshtasby, A. A. and Nikolov, S. (2007). Image fusion: Advances in the state of the art. *Information Fusion*, 8(2):114 – 118.

[77] Govada, K. A., Jonnalagadda, H. P., Kapavarapu, P., Alavala, S., and Vani, K. S. (2020). Road deformation detection. In *2020 7th International Conference on Smart Structures and Systems (ICSSS)*, pages 1–5.

[78] Graves, A. (2011). Practical variational inference for neural networks. In *24th Advances in Neural Information Processing Systems*, volume 24.

[79] Gu, S., Lu, T., Zhang, Y., Alvarez, J. M., Yang, J., and Kong, H. (2018). 3-D LiDAR + monocular camera: An inverse-depth-induced fusion framework for urban road detection. *IEEE Transactions on Intelligent Vehicles*, 3(3):351–360.

[80] Guizzo, E. (2021). How Google's self-driving car works. https://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works. Accessed on 27/01/2021.

[81] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *PMLR Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1321–1330.

[82] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. (2020). Deep learning for 3D point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–27.

[83] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Publishing Company, Incorporated, 2 edition.

[84] He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask R-CNN. In *IEEE International Conference on Computer Vision*.

[85] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916.

[86] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE.

[87] He, Q., Wang, Z., Zeng, H., Zeng, Y., Liu, Y., Liu, S., and Zeng, B. (2023). Stereo RGB and deeper lidar-based network for 3d object detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 24(1):152–162.

[88] Hebbalaguppe, R., Prakash, J., Madan, N., and Arora, C. (2022). A stitch in time saves nine: A train-time regularizing loss for improved neural network calibration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 16081–16090.

[89] Heinzler, R., Schindler, P., Seekircher, J., Ritter, W., and Stork, W. (2019). Weather influence and classification with automotive LiDAR sensors. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1527–1534.

[90] Hendrycks, D. and Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations*.

[91] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. In *Workshop on Deep Learning and Representation Learning, NIPS*.

[92] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR, arXiv*, abs/1207.0580.

[93] Hu, Q., Yang, B., Khalid, S., Xiao, W., Trigoni, N., and Markham, A. (2021). Towards semantic segmentation of urban-scale 3d point clouds: A dataset, benchmarks and challenges. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[94] Hur, Y., Yang, E., and Hwang, S. J. (2022). A simple framework for robust out-of-distribution detection. *IEEE Access*, 10:23086–23097.

[95] Hussain, S. M. F., Hamza, S. M., and Samad, A. (2022). Image segmentation for autonomous driving using u-net inception. In *7th International Conference on Signal and Image Processing (ICSIP)*, pages 426–429.

[96] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456.

[97] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

[98] Janai, J., Güney, F., Behl, A., and Geiger, A. (2020). Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends in Computer Graphics and Vision*, 12(1–3):1–308.

[99] Jang, C. H., Kim, C. S., Jo, K. C., and Sunwoo, M. (2017). Design factor optimization of 3D flash LiDAR sensor based on geometrical model for automated vehicle and advanced driver assistance system applications. *International Journal of Automotive Technology*, 18(1):147–156.

[100] Jingjing Liu, Shaoting Zhang, S. W. and Metaxas, D. (2016). Multispectral deep neural networks for pedestrian detection. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 73.1–73.13. BMVA Press.

[101] Joseph, K. J., Khan, S., Khan, F. S., and Balasubramanian, V. N. (2021). Towards open world object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5830–5840.

[102] Kang, Z., Yang, J., and Zhong, R. (2017). A bayesian-network-based classification method integrating airborne LiDAR data with optical images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(4):1651–1661.

[103] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *30th Advances in Neural Information Processing Systems*, volume 30, pages 5574–5584.

[104] Kesten, R. V., Usman, M., Houston, J., Pandya, T., Nadhamuni, K., Ferreira, A., Yuan, M., Low, B., Jain, A., Ondruska, P., Omari, S., Shah, S., Kulkarni, A., Kazakova, A., Tao, C., Platinsky, L., and Jiang, W. (2021). Lyft level 5 dataset. https://self-driving.lyft.com/level5/data/. Accessed on 05/19/2021.

[105] Kim, T. and Ghosh, J. (2016). Robust detection of non-motorized road users using deep learning on optical and LIDAR data. In *IEEE International Conference on Intelligent Transportation Systems*, pages 271–276.

[106] Kingma, D. and Welling, M. (2014). Auto-encoding variational Bayes. In *ICLR Proceedings 2nd International Conference on Learning Representations*.

[107] Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

[108] Kittler, J., Hatef, M., Duin, R. P., and Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239.

[109] Klein, L. A. (2012). *Sensor and Data Fusion: A Tool for Information Assessment and Decision Making*. Spie Press Monograph. SPIE Press, Bellingham, Washington, USA, 2nd edition.

[110] Klokov, R. and Lempitsky, V. S. (2017). Escape from cells: Deep kd-networks for the recognition of 3D point cloud models. In *IEEE International Conference on Computer Vision*, pages 863–872.

[111] Komarichev, A., Zhong, Z., and Hua, J. (2019). A-CNN: Annularly convolutional neural networks on point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7413–7422.

[112] Krishnan, R. and Tickoo, O. (2020). Improving model calibration with accuracy versus uncertainty optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 18237–18248.

[113] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *25th Advances in Neural Information Processing Systems*, pages 1097–1105.

[114] Kröger, F. (2016). *Automated Driving in Its Social, Historical and Cultural Contexts*, pages 41–68. Springer Berlin Heidelberg.

[115] Ku, J., Mozifian, M., Lee, J., Harakeh, A., and Waslander, S. L. (2018). Joint 3D proposal generation and object detection from view aggregation. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1–8.

[116] Kull, M., Perello Nieto, M., Kängsepp, M., Silva Filho, T., Song, H., and Flach, P. (2019). Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In *32nd Advances in Neural Information Processing Systems*, pages 12316–12326.

[117] Kull, M., Silva Filho, T., and Flach, P. (2017). Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *20th AISTATS.*, pages 623–631.

[118] Kumar, A., Sarawagi, S., and Jain, U. (2018). Trainable calibration measures for neural networks from kernel mean embeddings. In *PMLR Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 2805–2814.

[119] Kuncheva, L. (2002). A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286.

[120] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30.

[121] Lan, S., Yu, R., Yu, G., and Davis, L. S. (2019). Modeling local geometric structure of 3D point clouds using Geo-CNN. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 998–1008.

[122] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. (2019). PointPillars: Fast encoders for object detection from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12689–12697.

[123] Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[124] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

[125] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[126] Li, P. and Zhao, H. (2021). Monocular 3D detection with geometric constraint embedding and semi-supervised training. *IEEE Robotics and Automation Letters*, 6(3):5565–5572.

[127] Li, W., Dasarathy, G., and Berisha, V. (2020). Regularization via structural label smoothing. In *PMLR Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, volume 108, pages 1453–1463.

[128] Li, Y., Yao, T., Pan, Y., and Mei, T. (2023). Contextual transformer networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1489–1500.

[129] Liang, S., Li, Y., and Srikant, R. (2018). Enhancing the reliability of out-of-distribution image detection in neural networks. In *6th International Conference on Learning Representations*.

[130] Liao, Y., Xie, J., and Geiger, A. (2023). Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310.

[131] Lidstone, G. J. (1920). Note on the general case of the bayes-laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, 8:182–192.

[132] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2020). Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327.

[133] Liu, B., Ayed, I. B., Galdran, A., and Dolz, J. (2022a). The devil is in the margin: Margin-based label smoothing for network calibration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 80–88.

[134] Liu, S., Li, L., Tang, J., Wu, S., and Gaudiot, J.-L. (2017). *Creating Autonomous Vehicle Systems*. Morgan & Claypool Publishers.

[135] Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. (2018). Path aggregation network for instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768.

[136] Liu, T., Liu, Y., Hildebrandt, M., Joblin, M., Li, H., and Tresp, V. (2022b). On calibration of graph neural networks for node classification. In *International Joint Conference on Neural Networks*, pages 1–8.

[137] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *14th European Conference on Computer Vision*, pages 21–37.

[138] Liu, Z., Zhao, X., Huang, T., Hu, R., Zhou, Y., and Bai, X. (2020). Tanet: Robust 3d object detection from point clouds with triple attention. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence*, pages 11677–11684.

[139] Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., and Milford, M. J. (2016). Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19.

[140] Lu, J., Tang, S., Wang, J., Zhu, H., and Wang, Y. (2019). A review on object detection based on deep convolutional neural networks for autonomous driving. In *Chinese Control And Decision Conference (CCDC)*, pages 5301–5308.

[141] Lukasik, M., Bhojanapalli, S., Menon, A., and Kumar, S. (2020). Does label smoothing mitigate label noise? In *PMLR Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 6448–6458.

[142] Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2017). 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research*, 36(1):3–15.

[143] Mahoor, M. and Khodaei, A. (2018). Data fusion and machine learning integration for transformer loss of life estimation. In *IEEE Transmission and Distribution Conference and Exposition*, pages 1–9.

[144] Matsugatani, K. (2016). Sensing technologies required for ADAS/AD applications. In *Requirements Engineering Toward Sustainable World*, volume 1 of *671*. Third Asia-Pacific Symposium, APRES, Springer Singapore.

[145] Maurer, M., Gerdes, J. C., Lenz, B., and Winner, H. (2016). *Autonomous Driving: Technical, Legal and Social Aspects*. Springer-Verlag Berlin Heidelberg, 1st edition.

[146] McAllister, R., Gal, Y., Kendall, A., van der Wilk, M., Shah, A., Cipolla, R., and Weller, A. (2017). Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4745–4753.

[147] McCrae, S. and Zakhor, A. (2020). 3d object detection for autonomous driving using temporal lidar data. In *IEEE International Conference on Image Processing*, pages 2661–2665.

[148] Meister, C., SaleskyZ, E., and Cotterell, R. (2020). Generalized entropy regularization or: There's nothing special about label smoothing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 6870–6886.

[149] Melotti, G., Asvadi, A., and Premebida, C. (2018a). Cnn-lidar pedestrian classification: combining range and reflectance data. In *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 1–6.

[150] Melotti, G., Premebida, C., Bird, J. J., Faria, D. R., and Gonçalves, N. (2020a). Probabilistic object classification using CNN ML-MAP layers. In *Workshop on Perception for Autonomous Driving, European Conference on Computer Vision*.

[151] Melotti, G., Premebida, C., Goncalves, N. M. M. d. S., Nunes, U. J. C., and Faria, D. R. (2018b). Multimodal cnn pedestrian classification: A study on combining lidar and camera data. In *21st International Conference on Intelligent Transportation Systems*, pages 3138–3143.

[152] Melotti, G., Premebida, C., and Gonçalves, N. (2020b). Multimodal deep-learning for object recognition combining camera and lidar data. In *IEEE International Conference on Autonomous Robot Systems and Competitions*, pages 177–182.

[153] Mena, J., Pujol, O., and Vitrià, J. (2021). A survey on uncertainty estimation in deep learning classification systems from a bayesian perspective. *ACM Comput. Surv.*, 54(9).

[154] Mesquita, D. P. P., Freitas, L. A., Gomes, J. P. P., and Mattos, C. L. C. (2019). LS-SVR as a bayesian RBF network. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–5.

[155] Miron, A., Rogozan, A., Ainouz, S., Bensrhair, A., and Broggi, A. (2015). An evaluation of the pedestrian classification in a multi-domain multi-modality setup. *Sensors*, 15(6):13851–13873.

[156] Mitchell, H. (2010). *Multi-Sensor Data Fusion*. Springer-Verlag Berlin Heidelberg, Berlin, New York, USA, 1 edition.

[157] Molchanov, D., Ashukha, A., and Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. In *PMLR Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2498–2507.

[158] Morris, A. S. (2001). Measurement and instrumentation principles. In Morris, A. S., editor, *Butterworth-Heinemann*, Oxford.

[159] Mozaffari, S., Al-Jarrah, O. Y., Dianati, M., Jennings, P., and Mouzakitis, A. (2020). Deep learning-based vehicle behavior prediction for autonomous driving applications: A review. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15.

[160] Munder, S. and Gavrila, D. (2009). An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 28(11):1863–1868.

[161] Müller, R., Kornblith, S., and Hinton, G. E. (2019). When does label smoothing help? In *32nd Advances in Neural Information Processing Systems*, pages 4696–4705.

[162] Naeini, M. P., Cooper, G. F., and Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, page 2901–2907.

[163] Neuhold, G., Ollmann, T., Bulò, S. R., and Kontschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *IEEE International Conference on Computer Vision*, pages 5000–5009.

[164] Neumann, L., Zisserman, A., and Vedaldi, A. (2018). Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection. In *Workshop on Machine Learning for Intelligent Transportation System, NIPS*.

[165] Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*.

[166] Niculescu-Mizil, A. and Caruana, R. (2005). Predicting good probabilities with supervised learning. In *ICML Proceedings of the 22nd International Conference on Machine Learning*, pages 625–632.

[167] Nixon, J., Dusenberry, M. W., Zhang, L., Jerfel, G., and Tran, D. (2019). Measuring calibration in deep learning. In *Workshop on IEEE Conference on Computer Vision and Pattern Recognition*.

[168] Ouyang, W., Zhou, H., Li, H., Li, Q., Yan, J., and Wang, X. (2018). Jointly learning deep features, deformable parts, occlusion and classification for pedestrian detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

[169] Papoulis, A. and Pillai, S. U. (2002). *Probability, random variables and stochastic processes*. McGraw-Hill, 4th edition.

[170] Passalis, N., Tzelepi, M., and Tefas, A. (2020). Probabilistic knowledge transfer for lightweight deep representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–10.

[171] Patel, J., Shah, S., Thakkar, P., and Kotecha, K. (2015). Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4):2162 – 2172.

[172] Patil, A., Malla, S., Gang, H., and Chen, Y. (2019). The H3D dataset for full-surround 3D multi-object detection and tracking in crowded urban scenes. In *IEEE International Conference on Robotics and Automation*, pages 9552–9557.

[173] Patra, R., Hebbalaguppe, R., Dash, T., Shroff, G., and Vig, L. (2023). Calibrating deep neural networks using explicit regularisation and dynamic data pruning. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1541–1549.

[174] Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., and Hinton, G. E. (2017). Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548.

[175] Pham, Q. H., Sevestre, P., Pahwa, R. S., Zhan, H., Pang, C. H., Chen, Y., Mustafa, A., Chandrasekhar, V., and Lin, J. (2020). A*3D dataset: Towards autonomous driving in challenging environments. In *IEEE International Conference on Robotics and Automation*, pages 2267–2273.

[176] Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

[177] Pomerleau, D. (1989). ALVINN: An autonomous land vehicle in a neural network. In *1st Advances in Neural Information Processing Systems*. Morgan Kaufmann.

[178] Pop, D. O., Rogozan, A., Nashashibi, F., and Bensrhair, A. (2017). Incremental cross-modality deep learning for pedestrian recognition. In *IEEE Intelligent Vehicles Symposium*, pages 523–528.

[179] Posch, K. and Pilz, J. (2021). Correlated parameters to accurately measure uncertainty in deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(3):1037–1051.

[180] Premebida, C., Garrote, L., Asvadi, A., Ribeiro, A. P., and Nunes, U. (2016). High-resolution LIDAR-based depth mapping using bilateral filter. In *IEEE 19th International Conference on Intelligent Transportation Systems*, pages 2469–2474.

[181] Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. (2018). Frustum PointNets for 3D object detection from RGB-D data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927.

[182] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[183] Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J. (2016). Volumetric and multi-view cnns for object classification on 3D data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5648–5656.

[184] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *30th Advances in Neural Information Processing Systems*, pages 5099–5108.

[185] Qian, G., Abualshour, A., Li, G., Thabet, A., and Ghanem, B. (2021). PU-GCN: Point cloud upsampling using graph convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11683–11692.

[186] Qiu, H., Ma, Y., Li, Z., Liu, S., and Sun, J. (2020). Borderdet: Border feature for dense object detection. In *European Conference on Computer Vision*, pages 549–564.

[187] Rasshofer, R. H. and Gresser, K. (2005). Automotive radar and lidar systems for next generation driver assistance functions. *Advances in Radio Science*, 3:205–209.

[188] Redmon, J. (2013-2016). Darknet: Open source neural networks in C. http://pjreddie.com/darknet/.

[189] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788.

[190] Redmon, J. and Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6517–6525.

[191] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, abs/1804.02767.

[192] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

[193] Robu, R. and Stefan, H. (2011). A genetic algorithm for classification. In *Recent Researches in Computers and Computing - International Conference on Computers and Computing, ICCC'11*, pages 52–56.

[194] Roitberg, A., Peng, K., Schneider, D., Yang, K., Koulakis, M., Martinez, M., and Stiefelhagen, R. (2022). Is my driver observation model overconfident? input-guided calibration networks for reliable and interpretable confidence estimates. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):25271–25286.

[195] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA.

[196] Savelonas, M. A., Pratikakis, I., Theoharis, T., Thanellas, G., Abad, F., and Bendahan, R. (2018). Spatially sensitive statistical shape analysis for pedestrian recognition from lidar data. *Computer Vision and Image Understanding*.

[197] Schlosser, J., Chow, C. K., and Kira, Z. (2016). Fusing LIDAR and images for pedestrian detection using convolutional neural networks. In *IEEE International Conference on Robotics and Automation*, pages 2198–2205.

[198] Scott, D. W. (1992). *Multivariate density estimation : theory, practice, and visualization*. Wiley series in probability and mathematical statistics. Wiley.

[199] Sensoy, M., Kaplan, L., and Kandemir, M. (2018). Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems 31*, pages 3179–3189.

[200] Shen, Y., Feng, C., Yang, Y., and Tian, D. (2018). Mining point cloud local structures by kernel correlation and graph pooling. In *Conference on Computer Vision and Pattern Recognition*, pages 4548–4557.

[201] Shi, S., Wang, X., and Li, H. (2019). Pointrcnn: 3d object proposal generation and detection from point cloud. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779.

[202] Shridhar, K., Laumann, F., and Liwicki, M. (2019). A comprehensive guide to bayesian convolutional neural network with variational inference. *CoRR, arXiv*, abs/1901.02731.

[203] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR 3rd International Conference on Learning Representations*.

[204] Singh, G., Akrigg, S., Maio, M. D., Fontana, V., Alitappeh, R. J., Khan, S., Saha, S., Jeddisaravi, K., Yousefi, F., Culley, J., Nicholson, T., Omokeowa, J., Grazioso, S., Bradley, A., Gironimo, G. D., and Cuzzolin, F. (2023). Road: The road event awareness dataset for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1036–1054.

[205] Song, X., Wang, P., Zhou, D., Zhu, R., Guan, C., Dai, Y., Su, H., Li, H., and Yang, R. (2019). ApolloCar3D: A large 3D car instance understanding benchmark for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5447–5457.

[206] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

[207] Srivastava, R. (2013). *Research Developments in Computer Vision and Image Processing: Methodologies and Applications*. IGI Global, 1st edition.

[208] Stevens, W. B. (1996). The automated highway system program: A progress report. *13th World Congress of IFAC*, 29(1):8180 – 8188.

[209] Su, D., Zhang, H., Chen, H., Yi, J., Chen, P.-Y., and Gao, Y. (2018). Is robustness the cost of accuracy? A comprehensive study on the robustness of 18 deep image classification models. In *European Conference on Computer Vision*.

[210] Subrahmanya, N., Shin, Y. C., and Meckl, P. H. (2010). A bayesian machine learning method for sensor selection and fusion with application to on-board fault diagnostics. *Mechanical Systems and Signal Processing*, 24(1):182–192.

[211] Sun, P., Zhang, R., Jiang, Y., Kong, T., Xu, C., Zhan, W., Tomizuka, M., Li, L., Yuan, Z., Wang, C., and Luo, P. (2021). Sparse R-CNN: End-to-end object detection

with learnable proposals. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 14454–14463.

[212] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[213] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

[214] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations*.

[215] Tan, M. and Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *PMLR Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6105–6114.

[216] Te, G., Hu, W., Zheng, A., and Guo, Z. (2018). RGCNN: Regularized graph CNN for point cloud segmentation. In *ACM Multimedia Conference*, pages 746–754.

[217] Tenhundfeld, N. L., de Visser, E. J., Haring, K. S., Ries, A. J., Finomore, V. S., and Tossell, C. C. (2019). Calibrating trust in automation through familiarity with the autoparking feature of a tesla model x. *Journal of Cognitive Engineering and Decision Making*, 13(4):279–294.

[218] Thorpe, C., Hebert, M. H., Kanade, T., and Shafer, S. A. (1988). Vision and navigation for the carnegie-mellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373.

[219] Thulasidasan, S., Chennupati, G., Bilmes, J. A., Bhattacharya, T., and Michalak, S. (2019). On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems 32*, pages 13888–13899.

[220] Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., andXiaohua Zhai, L. B., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. (2021). Mlp-mixer: An all-mlp architecture for vision. *CoRR*, abs/2105.01601.

[221] Tran, T. H. . . P. and Jeon, J. W. (2020). Accurate real-time traffic light detection using YOLOv4. In *EEE International Conference on Consumer Electronics - Asia*, pages 1–4.

[222] Urmson, C., Anhalt, J., Bae, H., Bagnell, J. A. D., Baker, C. R., Bittner, R. E., Brown, T., Clark, M. N., Darms, M., Demitrish, D., Dolan, J. M., Duggins, D., Ferguson, D., Galatali, T., Geyer, C. M., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T., Kolski, S., Likhachev, M., Litkouhi, B., Kelly, A., McNaughton, M., Miller, N., Nickolaou, J., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Sadekar, V., Salesky, B., Seo, Y.-W., Singh, S., Snider, J. M., Struble, J. C., Stentz, A. T., Taylor, M., Whittaker, W. R. L., Wolkowicki, Z., Zhang, W., and Ziglar, J. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(8):425–466.

[223] Valcarce, D., Parapar, J., and Barreiro, Á. (2016). Additive smoothing for relevance-based language modelling of recommender systems. In *Proceedings of the 4th Spanish Conference on Information Retrieval*.

[224] Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). Regularization of neural networks using dropconnect. *Proceedings of the 30th International Conference on Machine Learning*, 28(3):1058–1066.

[225] Wang, H., Wang, Y., Zhao, X., Wang, G., Huang, H., and Zhang, J. (2019a). Lane detection of curving road for structural highway with straight-curve model on vision. *IEEE Transactions on Vehicular Technology*, 68(6):5321–5330.

[226] Wang, L., Du, L., Ye, X., Fu, Y., Guo, G., Xue, X., Feng, J., and Zhang, L. (2021a). Depth-conditioned dynamic message propagation for monocular 3D object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 454–463.

[227] Wang, Y., Li, B., Che, T., Zhou, K., Liu, Z., and Li, D. (2021b). Energy-based open-world uncertainty modeling for confidence calibration. In *IEEE International Conference on Computer Vision*, pages 9302–9311.

[228] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019b). Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics*, 38(5).

[229] Wang, Z. and Jia, K. (2019). Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1742–1749.

[230] Waymo (2020). Waymo's safety methodologies and safety readiness determinations. https://storage.googleapis.com/sdc-prod/v1/safety-report/Waymo-Safety-Methodologies-and-Readiness-Determinations.pdf. Accessed on 02/07/2022.

[231] Waymo (2021a). Waymo safety report. https://storage.googleapis.com/waymo-uploads/files/documents/safety/2021-12-waymo-safety-report.pdf. Accessed on 02/07/2022.

[232] Waymo (2021b). The waymo's self-dving car. https://blog.waymo.com/2020/05/resuming-our-driving-operations-in.html. Accessed on 01/27/2021.

[233] Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. (2018). Flipout: Efficient pseudo-independent weight perturbations on mini-batches. In *ICLR 6th International Conference on Learning Representations*.

[234] World Health Organization (2018). Global status report on road safety 2018: summary. Licence: CC BY-NC-SA 3.0 IGO. Geneva, Switzerland. Accessed on 02/01/2021.

[235] Wu, Y., Wang, Y., Zhang, S., and Ogai, H. (2021). Deep 3D object detection networks using lidar data: A review. *IEEE Sensors Journal*, 21(2):1152–1171.

[236] Xie, J., Xu, Y., Zheng, Z., Zhu, S.-C., and Wu, Y. N. (2021). Generative PointNet: Deep energy-based learning on unordered point sets for 3D generation, reconstruction and classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 14976–14985.

[237] Xu, D., Anguelov, D., and Jain, A. (2018). PointFusion: Deep sensor fusion for 3D bounding box estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253.

[238] Xu, Q., Sun, X., Wu, C., Wang, P., and Neumann, U. (2020a). Grid-GCN for fast and scalable point cloud learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5660–5669. IEEE.

[239] Xu, Z., Zhang, W., Tan, X., Yang, W., Huang, H., Wen, S., Ding, E., and Huang, L. (2020b). Segment as points for efficient online multi-object tracking and segmentation. In *European Conference on Computer Vision*, pages 264–281.

[240] Yang, G., Song, X., Huang, C., Deng, Z., Shi, J., and Zhou, B. (2019). DrivingStereo: A large-scale dataset for stereo matching in autonomous driving scenarios. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 899–908.

[241] Yang, J., Wang, C., Wang, H., and Li, Q. (2020). A RGB-D based real-time multiple object detection and ranging system for autonomous driving. *IEEE Sensors Journal*, 20(20):11959–11966.

[242] Yeung, M., Rundo, L., Nan, Y., Sala, E., Schönlieb, C.-B., and Yang, G. (2022). Calibrating the dice loss to handle neural network overconfidence for biomedical image segmentation. *Journal of Digital Imaging*.

[243] Yin, H. and Berger, C. (2017). When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets. In *IEEE 20th International Conference on Intelligent Transportation Systems*.

[244] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., and Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[245] Yu, H., Li, X., Murray, R. M., Ramesh, S., and Tomlin, C. (2019). *Safe, Autonomous and Intelligent Vehicles*. Springer International Publishing, 1st edition.

[246] Yu, L., Li, X., Fu, C.-W., Cohen-Or, D., and Heng, P.-A. (2018). PU-Net: Point cloud upsampling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[247] Yue, L., Abdel-Aty, M., Wu, Y., and Wang, L. (2018). Assessment of the safety benefits of vehicles' advanced driver assistance, connectivity and low level automation systems. *Accident Analysis & Prevention*, 117:55 – 64.

[248] Zadrozny, B. and Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 694– –699.

[249] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer International Publishing.

[250] Zhang, J., Kailkhura, B., and Han, T. (2020). Mix-n-Match: Ensemble and compositional methods for uncertainty calibration in deep learning. In *International Conference on Machine Learning*.

[251] Zhang, Q., Huang, N., Yao, L., Zhang, D., Shan, C., and Han, J. (2020). RGB-T salient object detection via fusing multi-level cnn features. *IEEE Transactions on Image Processing*, 29:3321–3335.

[252] Zhang, Y., Lu, J., and Zhou, J. (2021). Objects are different: Flexible monocular 3D object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3289–3298.

[253] Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. (2018). Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464.

[254] Zhou, D., Fang, J., Song, X., Guan, C., Yin, J., Dai, Y., and Yang, R. (2019). IoU Loss for 2D/3D object detection. In *IEEE International Conference on 3D Vision*, pages 85–94.

[255] Zhou, S., Xu, H., Zhang, G., Ma, T., and Yang, Y. (2022). Leveraging deep convolutional neural networks pre-trained on autonomous driving data for vehicle detection from roadside lidar data. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):22367–22377.

[256] Zhou, Y., He, Y., Zhu, H., Wang, C., Li, H., and Jiang, Q. (2021). Monocular 3D object detection: An extrinsic parameter free approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7556–7566.

[257] Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., Stiller, C., Dang, T., Franke, U., Appenrodt, N., Keller, C. G., Kaus, E., Herrtwich, R. G., Rabe, C., Pfeiffer, D., Lindner, F., Stein, F., Erbs, F., Enzweiler, M., Knöppel, C., Hipp, J., Haueis, M., Trepte, M., Brenk, C., Tamke, A., Ghanaat, M., Braun, M., Joos, A., Fritz, H., Mock, H., Hein, M., and Zeeb, E. (2014). Making bertha drive—an

autonomous journey on a historic route. *IEEE Intelligent Transportation Systems Magazine*, 6(2):8–20.

[258] Zou, Y., Yu, Z., Liu, X., Kumar, B. V. K. V., and Wang, J. (2019). Confidence regularized self-training. In *IEEE International Conference on Computer Vision*, pages 5981–5990.